
**UNIVERSITÀ DEGLI STUDI**
**DI MODENA E REGGIO EMILIA**

**Dottorato di ricerca in**
**Information and communication technologies (ICT)**

Ciclo XXXVIII

# Unified Transformer-based Architectures for Representing and Generating Heterogeneous Time-Dependent Tabular Data

Candidato: Fabrizio Garuti

Relatore (Tutor): Prof. Rita Cucchiara

Coordinatore del Corso di Dottorato: Prof. Luigi Rovati

*Advisor:*

Prof. Rita Cucchiara — University of Modena and Reggio Emilia

*Director of the School:*

Prof. Luigi Rovati — University of Modena and Reggio Emilia

*Review Committee:*

Prof. Paolo Giudici — University of Pavia
Prof. Petter Kolm — NYU Courant Institute of Mathematical Sciences

The work described in this thesis has been carried out within the International Doctorate in Information and Communication Technologies at the AImageLab research laboratory of the University of Modena and Reggio Emilia, thanks to a research collaboration with Prometeia Associazione and the Data Science team of Prometeia S.p.A.

This dissertation was typeset by the author using LaTeX $2_\varepsilon$, originally developed by Leslie Lamport and based on Donald Knuth's TeX. The body text is set in 11 point Egenolff-Berner Garamond, a revival of Claude Garamont's humanist typeface.

*To my family and those who love me,*

*To those who have sincerely believed in me,*

*You have created the conditions that made all of this possible.*

# Unified Transformer-based Architectures for Representing and Generating Heterogeneous Time-Dependent Tabular Data


## Abstract

The increasing adoption of Deep Learning models across diverse domains has recently stimulated research on extending such methods to structured tabular data, especially when these data exhibit temporal dependencies, as in transactional or financial time series. However, tabular datasets pose unique challenges due to their heterogeneous nature, combining numerical and categorical features, and the variable temporal structure of the data.

This thesis presents a unified Transformer-based framework for both representation and generation of time-dependent heterogeneous tabular data. We first introduce UniTTab, a Transformer architecture capable of representing transactional time series through a uniform embedding space that jointly encodes categorical and numerical attributes. Trained via a masked token pretext task, UniTTab effectively captures temporal dependencies and internal heterogeneity, outperforming traditional Machine Learning and Deep Learning methods across various prediction and classification tasks.

Building on this representation foundation, we explore large-scale training of Transformer models on real-world bank transaction datasets, establishing the first foundation model for transactional data. We then extend this framework to data generation, introducing two models—UniTTab-AR, an autoregressive Transformer, and BankDiT, a diffusion-based Transformer—for synthetic transaction generation. These models address privacy constraints in the financial domain by producing realistic yet non-sensitive synthetic data that preserve temporal dynamics and statistical properties of real transactions.

Finally, we generalize the diffusion-based approach to the broader task of tabular time series generation, introducing TabDiT, a Diffusion Transformer architecture specifically designed for heterogeneous and variable-length sequences, and demonstrating state-of-the-art performance across multiple datasets.

Overall, this work contributes a coherent framework that unifies the representation, learning, and generation of heterogeneous temporal tabular data through large-scale Transformer architectures, paving the way for foundation models in structured temporal domains such as finance, healthcare, and beyond.

# Architetture Transformer Unificate per la Rappresentazione e la Generazione di Serie Temporali di Dati Tabulari Eterogenei


## Sommario

La crescente adozione di modelli di Deep Learning in diversi ambiti applicativi ha recentemente stimolato la ricerca sull'estensione di tali metodologie anche ai dati tabulari strutturati, in particolare quando questi presentano dipendenze temporali, come nel caso delle serie transazionali o finanziarie. Tuttavia, i dataset tabulari pongono sfide peculiari dovute alla loro natura eterogenea — che combina attributi numerici e categoriali — e alla variabilità della loro struttura temporale.

Questa tesi presenta un framework unificato basato su architetture Transformer per la rappresentazione e la generazione di dati tabulari eterogenei dipendenti dal tempo. Inizialmente, viene introdotto UniTTab, un modello Transformer in grado di rappresentare serie temporali transazionali attraverso uno spazio di embedding uniforme che codifica congiuntamente caratteristiche numeriche e categoriche. Addestrato mediante un "pretext task" di tipo masked token, UniTTab cattura efficacemente le dipendenze temporali e l'eterogeneità interna dei dati, superando approcci di Machine Learning e Deep Learning tradizionali in diversi compiti di previsione e classificazione.

A partire da questa base di rappresentazione, il lavoro esplora l'addestramento su larga scala di modelli Transformer su dataset reali di transazioni bancarie, definendo il primo foundation model per dati transazionali. Successivamente, il framework viene esteso alla generazione di dati, introducendo due modelli: UniTTab-AR, un Transformer autoregressivo, e BankDiT, un Transformer

basato su modelli di tipo diffusion. Entrambi affrontano le problematiche di privacy tipiche del dominio finanziario, generando dati sintetici realistici ma non sensibili, che preservano le dinamiche temporali e le proprietà statistiche delle transazioni reali.

Infine, la tesi generalizza l'approccio basato su diffusion al compito più ampio della generazione di serie temporali tabulari, introducendo TabDiT, un'architettura Diffusion Transformer specificamente progettata per dati eterogenei e sequenze di lunghezza variabile, che dimostra prestazioni allo stato dell'arte su diversi dataset.

Nel complesso, questo lavoro propone un quadro coerente che unifica rappresentazione, apprendimento e generazione di dati tabulari temporali eterogenei tramite architetture Transformer su larga scala, aprendo la strada allo sviluppo di modelli fondamentali per domini strutturati e temporali, come quello finanziario, sanitario e oltre.

# Contents

# 1 Introduction

In recent years, Deep Learning has profoundly transformed a wide range of application domains, achieving remarkable success in areas such as Natural Language Processing, Computer Vision, Speech Recognition, and Generative Modeling [70, 109, 51]. These advances have been largely driven by the availability of large-scale datasets, the development of expressive neural architectures—most notably Transformers—and the widespread adoption of self-supervised learning paradigms [35, 24]. As a result, modern neural models are now able to learn rich semantic representations that can be effectively transferred across tasks, giving rise to the concept of foundation models [19].

Despite this progress, the impact of Deep Learning on **tabular data**—the dominant data format in many scientific, industrial, and institutional settings—has been comparatively limited [99]. Tabular datasets are ubiquitous in domains such as finance, healthcare, economics, and public administration, where information is naturally organized as collections of records described by heterogeneous attributes. In particular, **transactional and financial data** represent one of the most valuable and information-dense data sources for banks and financial institutions, as they encode fine-grained traces of user behavior

over time [32].

A distinctive characteristic of transactional data is their **temporal nature**. Rather than isolated samples, transactions form **time-dependent sequences**, where each element corresponds to a structured record composed of numerical, categorical, and often semi-structured attributes (e.g., timestamps, monetary amounts, transaction types, channels, or merchant information). Figure 1.1 illustrates this perspective, showing an example of a transactional time series represented as a sequence of tabular rows ordered in time. These sequences exhibit complex temporal patterns, including periodic behaviors, abrupt regime changes, and long-range dependencies, all of which are critical for downstream tasks such as fraud detection, credit risk assessment, customer profiling, and churn prediction [33, 45].

Historically, the analysis of transactional time series has relied on **rule-based systems** and **traditional Machine Learning models**, most notably tree-based ensembles such as Random Forests, Gradient Boosting, and their variants [23, 46]. While these methods remain strong baselines, they typically depend on extensive manual feature engineering and aggregation over fixed temporal windows, which limits their ability to capture fine-grained temporal dynamics and cross-feature interactions in an end-to-end manner. Moreover, these approaches do not naturally scale to representation learning or generative modeling paradigms [53].

The limited adoption of Deep Learning for time-dependent tabular data is not accidental. Unlike images or text, tabular data are inherently **heterogeneous**, combining attributes with different semantic meanings, value domains, and statistical properties [54]. Numerical and categorical features coexist within the same record, often with highly unbalanced distributions and complex interdependencies. Furthermore, real-world transactional datasets frequently exhibit **variable-length sequences** and **multi-structure rows**, where the set of available attributes may depend on the type of transaction itself. These characteristics pose fundamental challenges to standard neural architectures, which are typically designed for homogeneous and fixed-structure inputs [9].

An additional and crucial aspect is **data availability**. In the financial domain,

| Card | Timestamp | Amount | Use Chip | Merchant Name | Merchant City | Merchant State | Zip | MCC | Errors? |
|---|---|---|---|---|---|---|---|---|---|
| 6 | 2013-09-05 07:19 | $23.44 | Swipe Transaction | Applebees | Petersburg | VA | 23803 | 7832 | |
| 6 | 2013-09-05 16:08 | $84.80 | Online Transaction | Frontier Communications | ONLINE | | | 4784 | |
| 6 | 2013-09-06 10:33 | $15.66 | Swipe Transaction | Green Wholesale | Newport News | VA | 23605 | 5411 | Technical Glitch |
| 6 | 2013-09-06 14:45 | $42.38 | Online Transaction | Frontier Communications | ONLINE | | | 4784 | |
| 6 | 2013-09-07 10:44 | $15.82 | Swipe Transaction | Barnes & Noble | West Covina | CA | 23605 | 5541 | |
| 6 | 2013-09-08 14:03 | $36.75 | Online Transaction | Frontier Communications | ONLINE | | | 4784 | Bad CVV |
| 6 | 2013-09-08 09:56 | $80.05 | Swipe Transaction | Chevron | Rosemead | CA | 91772 | 4111 | |
| 6 | 2013-09-09 12:34 | $223.46 | Swipe Transaction | Kelly Auto Repair | Newport News | VA | 23606 | 3393 | |
| 6 | 2013-09-10 08:46 | $36.43 | Online Transaction | Frontier Communications | ONLINE | | | 4784 | |
| 6 | 2013-09-11 06:09 | $48.07 | Swipe Transaction | Anwar Grocery | Alhambra | CA | 91801 | 5411 | |

**Figure 1.1:** Example of a transactional time series represented as a tabular sequence. Each row corresponds to a single transaction event, ordered by time, while columns represent heterogeneous attributes (numerical, categorical, and temporal). The full table constitutes a variable-length multivariate time series associated with a single entity.

large-scale transactional datasets are mostly proprietary and subject to strict privacy and regulatory constraints, which severely limit data sharing and the creation of public benchmarks [43]. This has historically hindered the development of large Deep Learning models comparable to those trained on open corpora in NLP or Computer Vision. At the same time, the growing volume of digital transactions and the increasing computational capabilities of modern hardware have created unprecedented opportunities for training large models directly within private, industrial settings [27].

Within this context, there is a strong motivation to develop **unified neural architectures** capable of learning expressive representations of heterogeneous, time-dependent tabular data, leveraging self-supervised learning at scale and enabling both predictive and generative applications [118]. Such architectures would not only bridge the gap between structured data and modern Deep Learning techniques, but also pave the way for **foundation models in structured temporal domains**, analogous to those that have revolutionized unstructured data [19].

This thesis is motivated by the need to address these challenges through a coherent and scalable Transformer-based framework that unifies representation learning and data generation for heterogeneous tabular time series, with a particular focus on real-world financial and transactional data.

## 1.1 Definition of Foundation model in the Transactional Domain

In this thesis, we use the term foundation model to denote a model that satisfies the following criteria within the domain of transactional tabular time series:

1. Large-scale self-supervised pre-training on unlabeled raw data, without reliance on task-specific annotations;
2. Task-agnostic representations that can be transferred—with limited or no architectural modification—to multiple downstream tasks (e.g., fraud detection, churn prediction, credit risk);
3. Architectural generality, in the sense that a single pretrained backbone is reused across tasks and datasets;
4. Demonstrated transfer effectiveness, empirically validated through improvements over task-specific or non-pretrained baselines.

This definition mirrors core properties of foundation models established in NLP and vision, while adapting them to the constraints of structured, heterogeneous temporal data.

## 1.2 Challenges of Time-Dependent Tabular Data

Modeling time-dependent tabular data poses a set of challenges that are substantially different from those encountered in domains traditionally dominated by Deep Learning, such as text or images [70]. These challenges arise from the intrinsic structure of tabular datasets, the nature of temporal dependencies, and the constraints under which real-world data—especially financial data—are collected and used. Understanding these difficulties is essential to motivate the architectural and methodological choices explored throughout this thesis.

**Heterogeneity of features** is one of the most fundamental obstacles. A single tabular record typically consists of attributes with diverse semantic meanings

and data types. Numerical features, such as transaction amounts or balances, coexist with categorical attributes, such as transaction types, channels, or merchant categories, each characterized by different statistical properties and value distributions [54]. Unlike text, where tokens share a common representation space, or images, where pixels are homogeneous numerical values arranged on a grid, tabular features lack a natural, unified representation [99]. This heterogeneity complicates both the input encoding and the design of learning objectives, as different feature types are often treated with distinct preprocessing pipelines and loss functions [9].

A second major challenge is the **variable-length nature of temporal sequences**. Transactional time series are inherently irregular: different entities (e.g., bank customers or accounts) may generate vastly different numbers of events over the same time span, ranging from a handful of transactions per year to several per day [33]. Moreover, the temporal spacing between consecutive events is non-uniform, and meaningful patterns may emerge over both short and long horizons. Many standard neural architectures assume fixed-length inputs or rely on aggressive truncation and padding strategies, which can lead to information loss or inefficient representations when applied to real transactional data [13].

Closely related to sequence length is the problem of **temporal dependencies and long-range interactions**. Transactional behaviors often exhibit complex dynamics, including seasonality, recurring patterns, and abrupt changes driven by external factors such as economic conditions or life events [22]. Capturing these dependencies requires models capable of reasoning over long temporal contexts and learning interactions not only across time, but also across heterogeneous features within and across transactions. Traditional approaches based on feature aggregation or sliding windows struggle to model such interactions explicitly, while recurrent architectures may suffer from scalability limitations when sequences become long [60].

Another distinctive difficulty arises from the presence of **multi-structure rows**. In many real-world scenarios, especially in finance, not all transactions share the same internal structure. Different transaction types—such as card pay-

ments, bank transfers, or cash withdrawals—may be described by partially overlapping sets of attributes [20]. This variability violates the common assumption of a fixed schema underlying the entire dataset and makes it difficult to represent sequences as simple stacks of homogeneous vectors. Effectively handling such structural variability requires models that can flexibly adapt their representations while maintaining a coherent temporal embedding space [118].

Beyond modeling challenges, **privacy and data availability constraints** play a central role in shaping the landscape of research on time-dependent tabular data. Financial and transactional datasets are highly sensitive and subject to strict regulatory frameworks, which severely limit public access and data sharing [43]. As a consequence, most existing benchmarks are either small-scale, heavily anonymized, or synthetically generated, and they fail to reflect the complexity and scale of real industrial data [6]. This scarcity of large, open datasets has historically slowed progress in representation learning for tabular time series and has limited the applicability of data-hungry Deep Learning models [53].

Finally, these challenges collectively hinder the emergence of **general-purpose models** for structured temporal data. While foundation models have become a standard paradigm in unstructured domains, their counterparts for tabular time series remain largely unexplored [19]. Addressing heterogeneity, temporal structure, and scalability within a unified framework is therefore a prerequisite for developing models that can be pre-trained once and effectively transferred across multiple downstream tasks and domains [121].

The remainder of this thesis is structured around addressing these challenges through unified Transformer-based architectures, self-supervised learning strategies, and generative modeling techniques specifically designed for heterogeneous and variable-length tabular time series.

## 1.3 Research Questions and Objectives

The challenges outlined in Section 1.2 highlight a fundamental gap between the expressive power of modern Deep Learning models and the structural complexity of time-dependent tabular data. This gap motivates a set of core research ques-

tions that guide the contributions of this thesis and shape its methodological direction.

The first research question concerns **representation learning**:

*How can heterogeneous, time-dependent tabular data be represented in a unified and expressive latent space using Deep Learning architectures?*

Addressing this question requires designing representations that jointly encode numerical and categorical attributes, accommodate variable-length sequences, and preserve temporal dependencies without relying on handcrafted feature engineering. A key objective is to identify architectural principles that allow heterogeneous transactional records to be embedded into a common representation space that is both compact and semantically meaningful.

A second research question focuses on **self-supervised learning for structured temporal data**:

*How can large volumes of unlabeled transactional time series be exploited to learn transferable representations through self-supervision?*

Given the scarcity of labeled data and the abundance of raw transactional records, an essential objective is to define pre-training strategies that do not depend on task-specific annotations. This includes investigating masked prediction objectives and training protocols that are compatible with heterogeneous feature types and temporal structure, enabling the model to capture both intra-transaction and inter-transaction dependencies.

The third research question addresses **scalability and foundation models**:

*To what extent can Transformer-based architectures be scaled to millions of real-world transactions, and can such models act as foundation models for transactional data?*

This question is particularly relevant in industrial settings, where large private datasets are available but computational efficiency and robustness become critical. The corresponding objective is to evaluate whether large-scale pre-training leads to consistent performance improvements across diverse downstream tasks and whether a single pretrained model can be reused effectively in multiple financial applications.

Beyond representation learning, this thesis also investigates **data generation**

as a complementary and strategically important problem:

*How can realistic and temporally coherent tabular time series be generated while preserving privacy and statistical fidelity?*

This question is motivated by the need for privacy-preserving data sharing, data augmentation, and scenario simulation in sensitive domains such as finance. The objective is to explore both autoregressive and diffusion-based generative models capable of producing heterogeneous, variable-length sequences that closely resemble real transactional behaviors without exposing sensitive information.

Finally, the thesis seeks to generalize beyond a single application domain:

*Can the proposed architectures and methodologies be extended from financial transactions to general heterogeneous tabular time series?*

While finance serves as the primary use case, an overarching objective is to develop models and learning principles that are not domain-specific, but instead applicable to a broad class of structured temporal datasets, including healthcare records and industrial monitoring data.

In summary, the main objectives of this thesis are:

- To design unified Transformer-based architectures for representing heterogeneous time-dependent tabular data.
- To develop self-supervised learning strategies that leverage large-scale unlabeled transactional datasets.
- To establish large pretrained models as foundation models for structured temporal domains.
- To extend representation learning frameworks to generative modeling for privacy-preserving synthetic data generation.
- To demonstrate the generality and effectiveness of the proposed methods across multiple datasets and tasks.

These objectives collectively define a coherent research agenda aimed at bridging the gap between modern Deep Learning and real-world structured temporal data.

## 1.4 Main Contributions

This thesis makes a set of original contributions that jointly advance the state of the art in representation learning and generative modeling for heterogeneous time-dependent tabular data. The contributions are methodological, architectural, and empirical, and are unified by a common Transformer-based framework.

**Unified representation of heterogeneous tabular time series.** The first core contribution is the introduction of UniTTab, a Transformer architecture specifically designed to represent heterogeneous tabular time series. UniTTab addresses the intrinsic heterogeneity of tabular data by jointly modeling numerical and categorical attributes within a unified embedding space, while explicitly capturing temporal dependencies across transactions. Unlike prior approaches that rely on discretization or heavy feature engineering, UniTTab adopts dedicated representations for different feature types and integrates them within a hierarchical Transformer architecture, enabling end-to-end learning from raw transactional data.

**Self-supervised pre-training for structured temporal data.** A second major contribution is the formulation of a masked token pre-training strategy tailored to time-dependent tabular data. This strategy extends ideas from self-supervised learning in NLP and Computer Vision to the structured temporal domain, introducing masking mechanisms at the feature, row, and timestamp levels. The resulting pre-training objective allows UniTTab to leverage large volumes of unlabeled transactional data, learning transferable representations that can be fine-tuned for a wide range of downstream tasks.

**Large-scale foundation models for transactional data.** This thesis demonstrates, for the first time, that Transformer-based models can be effectively trained at scale on millions of real-world bank transactions, providing one of the earliest

systematic demonstrations that large-scale self-supervised Transformers can act as foundation models for transactional data. Extensive empirical evaluations show that large-scale pre-training consistently improves performance across multiple financial tasks, including fraud detection, credit risk assessment, and churn prediction, outperforming both traditional Machine Learning approaches and prior Deep Learning models.

**Autoregressive generation of transactional time series.** Building on the learned representations, the thesis introduces UniTTab-AR, an autoregressive Transformer model for generating transactional time series. UniTTab-AR is designed to generate transactions at the feature level while preserving internal consistency across attributes and temporal coherence across the sequence. This contribution enables realistic continuation and synthesis of transactional histories, providing a practical tool for data augmentation and behavioral simulation.

**Diffusion-based generation for heterogeneous time series.** To overcome the limitations of autoregressive generation in terms of diversity and unconditional sampling, the thesis further introduces BankDiT, a diffusion-based Transformer architecture for transactional data generation. By combining a variational latent space with a Diffusion Transformer backbone, BankDiT enables high-diversity, privacy-preserving generation of variable-length transactional sequences, conditioned or unconditioned on customer attributes.

**Generalization to tabular time series beyond finance.** Finally, the thesis generalizes the diffusion-based approach to broader structured temporal domains through TabDiT, a Diffusion Transformer architecture for general heterogeneous tabular time series. TabDiT introduces architectural and representational innovations that allow it to handle variable-length sequences and complex feature spaces, achieving state-of-the-art performance across multiple public datasets and establishing diffusion models as a viable paradigm for tabular time series generation.

Together, these contributions define a coherent framework that unifies representation learning, large-scale pre-training, and generative modeling for structured temporal data.

Evidence scope and reproducibility The empirical evidence supporting this thesis is drawn from a combination of publicly available benchmarks and large-scale proprietary transactional datasets. Public datasets (e.g., Pollution, Transaction, PKDD'99, Age2) are used to establish reproducibility and comparability with prior work. Proprietary banking datasets are used to evaluate scalability, long-horizon modeling, and industrial realism. Claims related to large-scale pre-training, model capacity, and real-world deployment should therefore be interpreted as demonstrated within private industrial settings, rather than as fully reproducible public benchmarks.

Positioning relative to closest prior work Prior Transformer-based approaches to tabular or transactional data (e.g., TabBERT, LUNA, REaLTabFormer) explore isolated aspects of representation learning or generation, typically under supervised, small-scale, or task-specific settings. In contrast, this thesis differs along three dimensions:

i. the consistent use of large-scale self-supervised pre-training as the primary learning signal;

ii. the reuse of a single pretrained backbone across multiple discriminative and generative tasks;

iii. empirical validation on long, heterogeneous, real-world transactional sequences exceeding the scale of prior published studies.

These differences motivate the interpretation of the proposed models as foundation-style architectures under the definition adopted in this work.

## 1.5 Thesis Organization

The remainder of this thesis is organized as follows.

**Chapter 2** reviews the background and related work on modeling tabular data as multivariate time series. It covers traditional Machine Learning approaches

for transactional data, early Deep Learning methods for tabular inputs, and recent Transformer-based architectures. Particular attention is devoted to tabular Transformers, autoregressive generative models, and the limitations of existing approaches, highlighting the research gaps that motivate this work.

**Chapter 3** introduces the formal problem formulation and the methodological framework adopted throughout the thesis. It defines heterogeneous tabular time series, distinguishes representation learning from generative modeling, discusses self-supervised learning strategies for structured temporal data, and presents the evaluation protocols and metrics used in subsequent chapters.

**Chapter 4** presents UniTTab, a unified Transformer-based architecture for representing heterogeneous, time-dependent tabular data. The chapter details the model design, including feature representations, hierarchical attention mechanisms, and the self-supervised masked pre-training strategy, and reports experimental results on representation learning tasks.

**Chapter 5** focuses on the application of Deep Learning models to real-world bank transactional data. Building on UniTTab, this chapter introduces a large-scale Transformer-based foundation model trained on extensive transactional datasets and evaluates its effectiveness across multiple downstream financial tasks.

**Chapter 6** further investigates large-scale Transformer models for transactional data, emphasizing scalability, long and variable-length sequences, and large-scale pre-training. The chapter analyzes the behavior of these models in realistic banking scenarios and reports extensive experimental results.

**Chapter 7** addresses the generative modeling of tabular time series. It introduces autoregressive Transformer-based generation for transactional data and explores diffusion-based generative models, comparing their strengths and limitations in terms of realism, diversity, and temporal coherence.

**Chapter 8** generalizes the diffusion-based approach to heterogeneous tabular time series beyond the financial domain. It introduces TabDiT, a Diffusion Transformer architecture designed for mixed numerical and categorical features and variable-length sequences, and presents state-of-the-art experimental results on multiple datasets.

**Chapter 9** discusses practical applications, limitations, ethical considerations,

and future research directions. It examines the implications of the proposed framework for financial applications, privacy-preserving analytics, and responsible deployment of large-scale models.

**Chapter 10** concludes the thesis by summarizing the main contributions and providing final remarks on the broader impact of unified Transformer-based models for structured temporal data.

# 2 Background

## 2.1 Tabular Data as Multivariate Time Series

Tabular data are traditionally modeled as collections of independent samples, where each row represents an observation described by a fixed set of attributes. This assumption underlies most classical Machine Learning methods for structured data, including linear models and tree-based ensembles. However, in many real-world applications, tabular records are not independent, but instead form **temporally ordered sequences** whose structure and dynamics carry essential information. In such cases, tabular datasets are more accurately described as **multivariate time series**.

A multivariate time series consists of an ordered sequence of observations indexed by time, where each observation is a vector of multiple variables. Transactional data naturally fit this definition: each transaction corresponds to a timestamped multivariate observation, and the full transaction history of an entity—such as a bank account or a customer—forms a time series. Unlike classical sensor-based time series, however, transactional data exhibit several distinctive characteristics that complicate their modeling.

First, transactional time series are typically **event-driven rather than regu-**

**larly sampled**. Transactions occur at irregular intervals, and the temporal distance between consecutive events can range from seconds to months. This irregularity makes standard time series models based on fixed sampling rates, such as autoregressive integrated moving average (ARIMA) models [22], less suitable without significant preprocessing or aggregation.

Second, each time step in a transactional time series is itself a **structured object** rather than a single numerical measurement. A transaction is composed of heterogeneous attributes, including numerical values (e.g., amounts), categorical variables (e.g., transaction type or channel), and composite fields (e.g., timestamps decomposed into calendar components). This contrasts with traditional multivariate time series, where each dimension typically corresponds to a homogeneous numerical signal.

Third, transactional time series are often **entity-specific and variable in length**. Different entities generate sequences of different lengths depending on their behavior, lifecycle, and usage patterns. As a result, datasets consist of collections of sequences with highly variable temporal horizons, which complicates batch processing and model training.

From a modeling perspective, treating tabular data as multivariate time series requires jointly capturing two levels of dependencies:

(i) **intra-record dependencies**, describing relationships among attributes within a single transaction, and

(ii) **inter-record dependencies**, describing temporal relationships across transactions in the sequence.

Many traditional approaches address these two aspects separately, for instance by summarizing temporal information through handcrafted aggregates before applying static models. While effective in practice, this strategy discards fine-grained temporal information and limits the expressiveness of the learned representations.

Recent advances in Deep Learning have renewed interest in modeling tabular data as sequences, leveraging architectures originally developed for sequential domains. Recurrent Neural Networks (RNNs) [39] and Long Short-Term Mem-

ory (LSTM) networks [60] have been applied to transactional sequences, but they often struggle with long-range dependencies and scalability when sequences become long or when the dimensionality of each time step increases. Moreover, these models typically assume homogeneous numerical inputs, making them less suited to heterogeneous tabular data without extensive preprocessing.

Transformer-based architectures [109] offer a promising alternative due to their ability to model long-range dependencies through attention mechanisms and their flexibility in handling variable-length sequences. However, directly applying Transformers to tabular time series is non-trivial, as it requires defining appropriate representations for heterogeneous features and adapting attention mechanisms to structured inputs. These challenges motivate the development of specialized Transformer architectures that explicitly treat tabular data as multivariate time series while preserving their internal structure.

This perspective—viewing tabular datasets as collections of heterogeneous multivariate time series—provides the conceptual foundation for the approaches reviewed in the remainder of this chapter and for the unified frameworks proposed in this thesis.

## 2.2 Traditional Approaches for Transactional Data

Before the recent interest in Deep Learning architectures, transactional and financial time series were predominantly analyzed using **traditional Machine Learning and statistical approaches**. These methods remain widely adopted in industry due to their robustness, interpretability, and relatively low computational requirements. However, they rely on strong modeling assumptions and extensive manual feature engineering, which limit their expressiveness when dealing with complex temporal and heterogeneous data.

Feature Engineering At the core of traditional approaches lies feature engineering, which aims to transform raw transactional data into a fixed-size vector representation suitable for static Machine Learning models. This process typi-

cally involves aggregating transactions over predefined temporal windows—such as days, weeks, or months—and computing summary statistics, including counts, averages, variances, extrema, and ratios. For example, common features include total spending in the last three months, average transaction amount, or frequency of specific transaction categories.

In this context, libraries such as **tsfresh** [31] have been proposed to partially automate the feature engineering process for time-series data. Tsfresh systematically extracts a large number of descriptive features from temporal signals by computing a comprehensive set of statistical, frequency-domain, and complexity-based characteristics, such as autocorrelations, Fourier coefficients, entropy measures, and distributional properties. The extracted features are computed over fixed windows and then optionally filtered through relevance tests to retain only those that are statistically significant for a given prediction task.

While aggregation-based and tsfresh-style features can capture coarse temporal patterns and summary characteristics of transaction histories, they inevitably lead to a loss of temporal resolution. Fine-grained ordering information and short-term behavioral changes are often smoothed out, making it difficult to detect subtle or abrupt pattern shifts. Moreover, despite the automation introduced by tsfresh, the resulting feature representations remain static and window-based, and the approach still relies on predefined feature calculators and assumptions about relevant temporal scales.

Another limitation of handcrafted and automatically extracted features is their limited adaptability. Once the feature set is defined or selected, it remains fixed regardless of changes in the downstream task or data distribution. As a result, traditional pipelines, even when supported by libraries such as tsfresh, struggle to flexibly adapt to new prediction objectives or evolving behavioral dynamics without significant reconfiguration.

Tree-Based Models Among traditional Machine Learning methods, **tree-based ensemble models** have become the dominant approach for transactional data analysis. Algorithms such as Random Forests [23], Gradient Boosted Decision Trees [47], XGBoost [26], LightGBM [65], and CatBoost [87] are partic-

ularly effective at handling heterogeneous feature types and non-linear interactions. Their ability to work directly with mixed numerical and categorical inputs, often with minimal preprocessing, makes them attractive for tabular datasets.

In financial applications, tree-based models have been successfully applied to tasks such as fraud detection, credit scoring, and customer segmentation. They offer strong predictive performance on small to medium-sized datasets and provide a degree of interpretability through feature importance measures and decision paths. For these reasons, they are often considered industry standards.

However, tree-based models exhibit several fundamental limitations when applied to time-dependent transactional data. First, they do not naturally model **sequential dependencies**. Temporal information must be encoded indirectly through engineered features, which restricts the model's ability to learn complex temporal dynamics. Second, these models operate on fixed-size inputs and therefore cannot natively handle variable-length sequences. Third, their representational capacity does not improve significantly with increasing dataset size, unlike Deep Learning models, whose performance often scales with data volume.

Limitations of Traditional Paradigms Overall, traditional approaches to transactional data analysis are characterized by a **two-stage pipeline**, where feature extraction and model learning are decoupled. While this paradigm has proven effective in practice, it limits end-to-end optimization and prevents the model from learning task-specific representations directly from raw data. Furthermore, these methods are not well suited to representation learning or generative modeling, both of which are increasingly important for tasks such as transfer learning, data augmentation, and privacy-preserving data synthesis.

These limitations have motivated the exploration of Deep Learning models for transactional data, with the goal of learning rich, task-agnostic representations directly from heterogeneous time-dependent tabular inputs. The following sections review these approaches, starting from early Deep Learning methods for tabular data and progressing toward Transformer-based architectures.

## 2.3 Deep Learning for Tabular Data

The limitations of traditional Machine Learning pipelines have motivated increasing interest in applying **Deep Learning models** to tabular data. The central promise of Deep Learning in this context is the ability to learn task-specific and transferable representations directly from raw inputs, reducing the reliance on manual feature engineering and enabling end-to-end optimization. However, extending Deep Learning to tabular data has proven significantly more challenging than in unstructured domains.

### 2.3.1 Numerical vs. Categorical Representations

One of the main difficulties lies in the **representation of heterogeneous features**. Categorical attributes are naturally compatible with neural architectures through embedding layers, which map discrete tokens into continuous vector spaces. This strategy, widely adopted in recommender systems and NLP-inspired models [54], allows the network to learn semantic relationships among categories and has become a standard practice for categorical fields.

In contrast, **numerical features** do not admit a universally optimal representation. A common approach consists in normalizing numerical values and feeding them directly as scalar inputs to neural networks. While simple and computationally efficient, this approach implicitly assumes that numerical variables are well-behaved, approximately symmetric, and linearly separable after normalization. In practice, these assumptions rarely hold. Real-world tabular data—especially in financial and transactional domains—are characterized by highly skewed, heavy-tailed, and non-stationary numerical distributions, as well as complex interactions with categorical attributes. Under these conditions, scalar representations tend to collapse most values into narrow regions of the input space, limiting the network's ability to discriminate between semantically distinct regimes and to model non-linear cross-feature dependencies. This representational weakness has been identified as a key factor behind the inferior performance of standard neural networks on tabular data compared to tree-based models ([21, 118]). Recent work such as [76] further shows that richer numerical

encodings are required to overcome the low-frequency bias of deep networks and to effectively capture the structure of heavy-tailed numerical features.

Alternative approaches **discretize numerical values** into bins and treat them as categorical tokens, enabling a uniform token-based representation [117]. However, discretization inevitably introduces quantization errors and may obscure fine-grained information that is crucial in domains such as finance.

Several works have proposed hybrid architectures that process numerical and categorical features through separate subnetworks before combining them [29]. While this mitigates some representational issues, it introduces architectural complexity and often requires careful tuning of loss functions and feature scaling strategies. As a result, no consensus has emerged on a principled and unified way to represent heterogeneous tabular features within Deep Learning models.

### 2.3.2 Limitations of Existing Deep Learning Approaches

Early Deep Learning models for tabular data typically rely on **Multilayer Perceptrons (MLPs)** augmented with embedding layers for categorical features. While these models can capture non-linear feature interactions, empirical studies have shown that they often underperform tree-based ensembles on small and medium-sized tabular datasets [97]. This observation has led to the widespread belief that Deep Learning is inherently less suited to tabular data.

More recent approaches attempt to close this gap by introducing architectural inductive biases tailored to tabular inputs. These include attention mechanisms over features [9], feature-wise gating, and hybrid models combining neural networks with decision tree structures [68]. Although such methods improve performance in some settings, they often remain focused on **static tabular data**, treating each row as an independent sample and ignoring temporal dependencies.

A further limitation is that many Deep Learning models for tabular data are trained in a fully supervised manner, which restricts their applicability when labeled data are scarce or expensive to obtain. In contrast to NLP and Computer Vision, where self-supervised pre-training has become a cornerstone of represen-

tation learning [35], analogous paradigms for tabular data are still emerging and lack standardization.

### 2.3.3 Toward Sequential and Self-Supervised Models

The recognition that tabular data often exhibit temporal structure has prompted efforts to extend Deep Learning models to **tabular time series**. Recurrent architectures and temporal convolutional networks have been applied with mixed success [13], but they struggle to scale to long sequences and to integrate heterogeneous features effectively.

Transformer-based models offer a compelling alternative due to their flexibility and scalability, but their success depends critically on the definition of suitable input representations and training objectives. In particular, adapting self-supervised learning techniques—such as masked prediction—to heterogeneous tabular data requires rethinking how tokens, masks, and targets are defined across numerical and categorical domains [62].

These challenges have given rise to a growing body of work on **tabular Transformers**, which aim to bridge the gap between structured data and modern sequence modeling architectures. The following section reviews these approaches with a focus on multivariate and time-dependent tabular data.

## 2.4 Tabular Transformers for Multivariate Time Series

The introduction of the Transformer architecture has marked a turning point in sequence modeling, enabling the effective learning of long-range dependencies through attention mechanisms. Originally developed for Natural Language Processing, Transformers have since been successfully adapted to other sequential domains, including vision, audio, and time series [109]. This success has naturally motivated their application to **tabular data**, particularly when such data exhibit temporal dependencies.

In the context of multivariate time series, Transformers offer two key advan-

tages over recurrent and convolutional architectures. First, self-attention allows direct interactions between any pair of time steps, making it easier to capture long-term dependencies without the vanishing gradient issues that affect recurrent models. Second, Transformers can process variable-length sequences efficiently and are well suited to large-scale parallel training.

However, applying Transformers to tabular time series is not straightforward. A naïve approach consists in encoding each tabular row into a single fixed-dimensional vector by concatenating the representations of all its features, and then modeling the resulting sequence as a standard multivariate time series. While simple, this flattening strategy discards the internal structure of each row and fails to distinguish between heterogeneous feature types. All attributes are treated as homogeneous dimensions of a vector, despite their different semantic roles and statistical properties. As a consequence, the model is not explicitly encouraged to capture intra-row dependencies among features, nor to separate them from temporal dependencies across rows.

To address this limitation, several works have proposed **feature-aware Transformer architectures**, in which attention is applied not only across time steps but also across features [102]. These models treat each feature as a token and allow the network to learn interactions between attributes within a single record. While effective for static tabular data, extending this paradigm to time-dependent scenarios leads to very long token sequences, as both the temporal and feature dimensions must be jointly represented. This raises scalability issues and increases the computational cost of self-attention.

An alternative and increasingly popular strategy is the adoption of **hierarchical Transformer architectures**. In this paradigm, each tabular row is first encoded into a compact representation using a local Transformer or feature-level encoder. These row-level embeddings are then fed into a second Transformer that models temporal dependencies across the sequence [72]. This two-level structure mirrors successful designs in other domains, such as video modeling, where spatial and temporal dimensions are processed separately. Hierarchical Transformers provide a natural way to balance expressiveness and scalability, while preserving the internal structure of tabular records.

Despite these advances, most existing tabular Transformer models make simplifying assumptions that limit their applicability to real-world transactional data. In particular, many approaches assume a fixed schema shared by all rows and rely on discretization of numerical features to define a common token vocabulary [62]. While this simplifies the modeling problem, it introduces information loss and restricts the ability of the model to handle **multi-structure rows**, where different time steps may be described by different sets of attributes.

Moreover, the majority of tabular Transformer models are designed for **discriminative tasks**, such as classification or regression, and are trained in a supervised or weakly supervised fashion. As a result, they do not fully exploit the potential of self-supervised pre-training on large unlabeled datasets, nor do they naturally extend to generative modeling tasks [9].

These limitations highlight the need for Transformer architectures that are explicitly designed for heterogeneous, time-dependent tabular data, capable of supporting both large-scale self-supervised learning and downstream discriminative or generative objectives. The next sections review complementary approaches that introduce architectural augmentations, language-inspired interfaces, and autoregressive modeling strategies for tabular data.

## 2.5 Attention-Augmented Convolutional Transformers for Tabular Time Series

In parallel with the development of pure Transformer-based architectures, a line of research has explored **hybrid models** that combine attention mechanisms with convolutional operations for modeling time-dependent tabular data. These approaches are motivated by the complementary strengths of convolutional and attention-based models: convolutions are effective at capturing local patterns and short-term dependencies, while attention mechanisms enable global context modeling and long-range interactions.

In the context of time series analysis, convolutional neural networks (CNNs) have been widely adopted due to their ability to extract temporal features through sliding kernels and hierarchical receptive fields. Temporal Convolutional Net-

works (TCNs), in particular, have demonstrated strong performance on a variety of sequence modeling tasks, benefiting from causal convolutions and dilation strategies to increase the temporal receptive field [13]. However, convolution-based models rely on fixed local neighborhoods and struggle to adaptively focus on non-local dependencies that may be critical in transactional data.

Attention-augmented convolutional models aim to address this limitation by integrating **self-attention layers** within or alongside convolutional blocks. In tabular time series settings, this typically involves using convolutions to process numerical signals or aggregated features over time, while attention mechanisms reweight temporal positions or feature channels based on their relevance to the task [72]. This hybrid design allows the model to capture both local temporal regularities and global behavioral patterns.

Several works adopt attention-augmented convolutions to improve interpretability and performance in multivariate time series forecasting and classification [88]. In these models, attention scores can highlight which time steps or features contribute most to the prediction, offering insights into temporal dynamics. However, such approaches often assume **homogeneous numerical inputs** and are primarily designed for regularly sampled signals, such as sensor measurements or energy consumption data.

When applied to heterogeneous tabular time series, attention-augmented convolutional models face significant challenges. Convolutions are not naturally suited to handle categorical variables or mixed feature types, and extending them to operate over complex tabular schemas typically requires substantial preprocessing and feature transformation. As a result, these models often rely on prior aggregation or embedding steps that reduce the original tabular structure to a homogeneous numerical representation.

Moreover, hybrid convolution-attention architectures are usually developed for **task-specific supervised learning**, such as forecasting or anomaly detection, and do not readily generalize to self-supervised or generative settings. Their architectural design is closely tied to the target task and input format, limiting their flexibility and reusability across domains.

In summary, attention-augmented convolutional Transformers represent an

important step toward richer temporal modeling, but their reliance on homogeneous inputs and task-specific design makes them less suitable for end-to-end representation learning on heterogeneous, variable-structure tabular time series. These limitations further motivate the exploration of fully attention-based architectures that can directly operate on structured tabular data without extensive manual preprocessing.

## 2.6 Language Understanding with Number Augmentations on Transformers

A complementary line of research relevant to tabular data modeling originates from the field of **Natural Language Processing**, where significant effort has been devoted to improving the numerical reasoning capabilities of Transformer-based language models. While these approaches are not designed specifically for tabular time series, they provide valuable insights into how numerical information can be integrated into token-based architectures originally conceived for categorical data.

Standard language models, such as BERT or GPT, treat numbers as atomic tokens or sequences of characters [35, 24]. This representation is often inadequate for tasks that require numerical understanding, as it fails to capture magnitude, ordering, and arithmetic relationships. To address this limitation, several works have proposed **number-aware or number-augmented Transformers**, introducing specialized embeddings or encoding schemes that enrich the representation of numerical values [110, 105].

Formally, given a base token embedding $e_t \in \mathbb{R}^d$ and a numerical value $v$, the augmented representation can be written as

$$\widetilde{e}_t = e_t \oplus \varphi(v),$$

where $\varphi(v) \in \mathbb{R}^k$ is a numerical feature mapping (e.g., raw value, log-magnitude, sign), and $\oplus$ denotes concatenation or linear fusion. This approach injects quantitative information directly into the embedding space while main-

taining compatibility with the Transformer architecture. Empirical results show that number-aware models improve performance on tasks involving numerical comparison, estimation, and reasoning [28].

Some works extend these ideas to **semi-structured inputs**, such as tables or key–value pairs, by linearizing tabular data into textual sequences and feeding them to pretrained language models [57]. In this setting, each row or cell is converted into a textual prompt that includes both attribute names and values. While this strategy enables the reuse of large pretrained language models, it introduces several limitations when applied to time-dependent tabular data.

First, textual linearization does not scale well to large datasets or long sequences. Transactional time series may consist of hundreds of records, each with multiple attributes, quickly exceeding the maximum input length of language models. Second, domain-specific tabular attributes and categorical values often contain abbreviations or codes that are poorly represented in the pre-training corpora of language models, leading to out-of-distribution inputs and degraded performance. Third, textual representations obscure the inherent structure of tabular data, forcing the model to implicitly rediscover relationships that are explicit in the original schema.

Moreover, number-augmented language models are primarily designed for **discriminative reasoning tasks** and are rarely evaluated in large-scale self-supervised or generative settings involving structured temporal data. While they demonstrate that Transformers can be extended to handle numerical information more effectively, they do not provide a complete solution for heterogeneous tabular time series, where numerical and categorical features coexist at scale and temporal coherence is essential.

Overall, research on numerical augmentation in language models highlights both the potential and the limitations of adapting NLP-centric architectures to structured data. These insights reinforce the need for domain-specific Transformer designs that treat tabular data as first-class structured objects, rather than as approximations of natural language.

## 2.7 Autoregressive Transformer Models for Tabular Data Generation

Beyond discriminative tasks, an important research direction concerns the **generative modeling of tabular data**, with the goal of synthesizing realistic samples that preserve the statistical properties of the original dataset. In sensitive domains such as finance, data generation plays a crucial role in privacy preservation, data augmentation, and stress testing. Among generative paradigms, **autoregressive (AR) Transformer models** have emerged as a natural extension of language modeling techniques to structured data.

Autoregressive models factorize the joint probability distribution of a data sample into a product of conditional distributions, generating one token at a time conditioned on previously generated tokens. When applied to tabular data, this approach typically requires defining a linear ordering over fields and converting each row into a sequence of tokens. Numerical features are often discretized or transformed into categorical representations to fit this framework, allowing the model to treat all attributes uniformly [111].

Early autoregressive approaches demonstrated that Transformers can learn complex inter-feature dependencies and generate coherent tabular rows [41]. When extended to time-dependent scenarios, autoregressive models can generate entire sequences by conditioning each new row on the previously generated ones, thus capturing temporal dynamics. This paradigm is particularly appealing because it naturally supports variable-length sequences and explicit conditioning.

Despite these advantages, autoregressive generation exhibits several well-known limitations when applied to tabular and time-dependent data. First, the generation process is inherently sequential, as each token depends on all previously generated tokens. This leads to **high inference latency** and poor scalability when generating long sequences or large synthetic datasets, which is especially problematic in industrial settings.

Second, autoregressive models often suffer from **limited diversity** in unconditional generation scenarios, a limitation that is closely related to inference-time

decoding strategies rather than to the learned model alone. In autoregressive generation, each token is produced by sampling from the conditional distribution $p(x_t \mid x_{<t})$, and the overall diversity of the generated samples depends not only on the learned model but also on how this distribution is decoded at inference time. When deterministic decoding strategies such as *greedy decoding* are adopted—i.e., selecting the token with maximum probability at each step—the generation process becomes fully deterministic [106]. As a result, given the same initial conditions, the model will always produce the same output sequence, leading to zero sample diversity.

To mitigate this issue, stochastic decoding methods based on sampling (e.g., multinomial sampling, temperature scaling, top-$k$, or nucleus sampling) are commonly employed [44, 61]. These techniques introduce randomness into the generation process and can substantially increase output variability. However, in the absence of an external conditioning signal—such as an input sequence, context, or control variables—the diversity achieved by autoregressive models remains intrinsically constrained. Even under stochastic sampling, the model tends to concentrate probability mass around high-frequency patterns observed during training, repeatedly generating variations of dominant modes of the data distribution [42].

In contrast, alternative generative paradigms such as diffusion models naturally promote higher diversity in unconditional generation [58]. Diffusion-based models generate samples by starting from an initial latent variable $X_T$ drawn from a fully random noise distribution and progressively transforming it into a data sample through a stochastic denoising process [104]. Because each generation begins from an independent random initialization, diffusion models can explore a wider range of modes in the data distribution, leading to increased sample diversity. Overall, this highlights that generative diversity is not solely a property of the model architecture, but also a consequence of the underlying generative process and, crucially, of the sampling strategy employed at inference time [66].

A further limitation concerns the treatment of numerical attributes. Representing numerical values through discretization or digit-level tokenization can lead to long token sequences and cumulative error propagation during genera-

tion. Small errors in early digits or bins may amplify across time steps, degrading the realism and stability of generated sequences, particularly in long-horizon generation tasks.

Overall, autoregressive Transformer models constitute a strong and well-established baseline for tabular data generation, offering explicit control over the generation process and strong guarantees in terms of feature-level consistency and conditional modeling. However, they also present inherent limitations related to inference efficiency, numerical robustness, and sample diversity, particularly in unconditional generation settings. These limitations arise from the sequential nature of autoregressive decoding, the sensitivity to tokenization choices for numerical attributes, and the dependence of generation diversity on inference-time sampling strategies. As a result, while autoregressive models are effective in scenarios with strong conditioning signals or well-defined generation constraints, their ability to produce diverse and scalable synthetic tabular sequences remains intrinsically bounded in the absence of explicit conditioning information.

### 2.7.1 REaLTabFormer: Generating Realistic Relational and Tabular Data using Transformers

REaLTabFormer is one of the most influential autoregressive Transformer models for tabular data generation [101]. The model is designed to generate both single-table and relational datasets by factorizing the joint distribution of rows and attributes in an autoregressive manner. Each row is represented as a sequence of tokens corresponding to individual fields, and numerical values are typically encoded as sequences of digits to enable lossless reconstruction.

A distinctive aspect of REaLTabFormer is its ability to perform **conditional generation**. In relational settings, a parent table—describing entity-level attributes—is used to condition the generation of child tables, such as transaction histories associated with a specific customer. This design makes the model particularly suitable for structured data generation scenarios where contextual information is available.

When extended to time series, REaLTabFormer can generate transactional sequences conditioned on entity attributes, capturing both inter-row and inter-feature dependencies. However, its autoregressive nature imposes practical constraints. The digit-based representation of numerical features leads to long token sequences, increasing computational complexity and amplifying error propagation during generation. Moreover, the model is primarily evaluated on relatively short sequences and moderate-sized datasets, limiting its scalability to industrial-scale transactional data.

From a methodological perspective, REaLTabFormer highlights both the strengths and the limitations of autoregressive Transformers for tabular data. While it demonstrates that coherent and conditionally consistent tabular sequences can be generated, it also exposes challenges related to efficiency, diversity, and numerical robustness that motivate alternative generative paradigms.

## 2.8 Summary of Related Work and Research Gaps

This chapter has reviewed the main strands of research related to the modeling of heterogeneous tabular data and time-dependent transactional sequences, highlighting both the progress achieved and the limitations of existing approaches. Taken together, the literature reveals a fragmented landscape in which different methods address isolated aspects of the problem, but few provide a unified and scalable solution.

Traditional Machine Learning approaches, particularly feature engineering pipelines combined with tree-based models, remain strong baselines for transactional data analysis. Their robustness and interpretability make them appealing in practice, but their reliance on handcrafted features and static representations limits their ability to capture fine-grained temporal dynamics and complex cross-feature interactions. Moreover, these methods do not naturally extend to representation learning or generative modeling.

Early Deep Learning approaches for tabular data improve representational flexibility but struggle to consistently outperform tree-based models, especially

on small and medium-sized datasets [97]. A central challenge lies in the lack of a principled and unified representation for numerical and categorical features. Many models either treat numerical features as raw scalars—losing expressive power—or discretize them, introducing quantization artifacts and information loss.

Transformer-based architectures offer a promising direction for modeling multivariate time series, thanks to their ability to capture long-range dependencies and handle variable-length sequences [109]. However, most existing tabular Transformers are designed for static data or assume a fixed schema shared by all rows. Feature-wise attention models face scalability issues, while hierarchical Transformers often rely on simplifying assumptions, such as homogeneous row structures or full discretization of numerical values [62].

Hybrid architectures combining attention and convolution improve local temporal modeling but are generally tailored to homogeneous numerical signals and supervised tasks [13]. Similarly, language-inspired approaches that linearize tabular data into textual prompts benefit from pretrained language models but do not scale to long sequences, domain-specific vocabularies, or large datasets, and they obscure the inherent structure of tabular data [57].

In the generative domain, autoregressive Transformer models demonstrate that it is possible to synthesize realistic tabular rows and sequences while preserving feature-level consistency [101]. Nevertheless, autoregressive generation suffers from efficiency limitations, reduced diversity in unconditional settings, and numerical robustness issues, particularly when digit-based representations are used. These limitations become increasingly severe when scaling to long, heterogeneous time series.

Across all these approaches, several **research gaps** emerge clearly:

- **Lack of a unified representation for heterogeneous time series**, capable of jointly encoding numerical and categorical features without heavy discretization or handcrafted preprocessing.
- **Insufficient handling of multi-structure rows**, where the set of available attributes may vary across time steps within the same sequence.

- **Limited exploitation of self-supervised learning**, despite the abundance of unlabeled transactional data and the success of pre-training paradigms in unstructured domains [35].
- **Scalability constraints**, which hinder the development of large pre-trained models for tabular time series comparable to foundation models in NLP or Computer Vision.
- **Limitations of autoregressive generation**, including inefficiency and lack of diversity, and the near absence of diffusion-based approaches for heterogeneous tabular time series.

In light of the reviewed literature, none of the existing approaches fully satisfy all criteria of a foundation model as defined in Section 1.1 (large-scale self-supervised pre-training, task-agnostic transfer, architectural reuse, and empirically demonstrated cross-task effectiveness).

These gaps motivate the research agenda pursued in the remainder of this thesis. In the next chapter, we formally define the problem of heterogeneous tabular time series and introduce the methodological framework adopted to address both representation learning and generative modeling within a unified Transformer-based paradigm.

# 3

# Problem Formulation and Methodological Framework

## 3.1 Formal Definition of Heterogeneous Tabular Time Series

This thesis focuses on the modeling, representation, and generation of **heterogeneous time-dependent tabular data**, with particular emphasis on transactional and financial datasets. To establish a precise and general framework, we begin by introducing a formal definition of the data structures considered throughout this work.

Let $\mathcal{A} = \{a_1, a_2, \ldots, a_K\}$ denote a finite set of attributes describing a tabular record. Each attribute $a_k$ is associated with a domain $\mathcal{D}_k$, which may be **categorical** (finite and discrete) or **numerical** (continuous or discrete-valued). A **tabular row** $r$ is defined as an ordered tuple

$$r = (v_1, v_2, \ldots, v_K), v_k \in \mathcal{D}_k,$$

where each value $v_k$ corresponds to the attribute $a_k$.

A **heterogeneous tabular time series** is a temporally ordered sequence of tabular rows:

$$x = (r_1, r_2, \ldots, r_T),$$

where $T$ denotes the sequence length and each row $r_t$ is associated with a timestamp $t_t$. In general, $T$ is variable across sequences, reflecting the irregular and entity-specific nature of real-world transactional data.

Unlike classical multivariate time series, where each time step is represented by a homogeneous numerical vector of fixed dimensionality, heterogeneous tabular time series exhibit several distinctive properties:

1. **Feature heterogeneity**: Each row combines numerical and categorical attributes with different statistical and semantic properties.

2. **Irregular temporal structure**: Observations are event-driven rather than regularly sampled, and temporal gaps between consecutive rows are non-uniform.

3. **Variable-length sequences**: Different entities generate sequences of different lengths, often spanning multiple temporal scales.

4. **Multi-structure rows**: In some datasets, the effective schema of a row may depend on its semantic type, leading to partially observed or structurally variable records.

We assume access to a dataset $\mathcal{X} = \{x^{(1)}, x^{(2)}, \ldots, x^{(N)}\}$, where each element corresponds to the time series associated with a distinct entity, such as a customer, account, or patient. The dataset is assumed to be sampled from an unknown data-generating distribution $p(x)$.

In addition to the primary time series, some applications involve an associated **parent table** or **contextual table** $\mathcal{P}$, where each entity is described by a static tabular row

$$u = (w_1, w_2, \ldots, w_M),$$

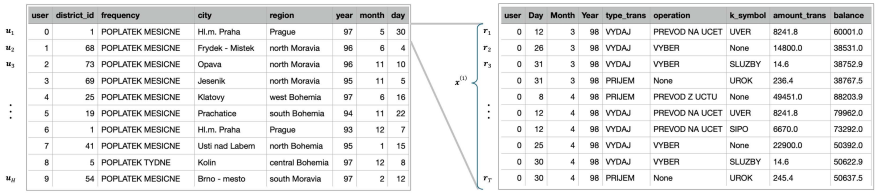
containing attributes that influence the generation or interpretation of the corresponding time series. Examples include demographic information for customers or metadata describing an industrial system. This structure naturally gives rise to **conditional modeling tasks**, where the goal is to model $p(x \mid u)$ rather than the marginal distribution $p(x)$. Figure 3.1 provides a schematic representation of this relational setting, illustrating the association between entities in the parent table and their corresponding heterogeneous tabular time series.

Within this formal setting, the problems addressed in this thesis can be broadly categorized into two complementary tasks:

- **Representation learning**, where the objective is to learn a function $f_\theta$ that maps a heterogeneous tabular time series $x$ (or its components) into a latent representation suitable for downstream tasks. In this thesis, we mainly focus on discriminative downstream tasks, where the learned representations are evaluated through supervised objectives such as classification and regression. These tasks serve as a proxy to assess the quality, expressiveness, and transferability of the learned latent representations across different application scenarios.

- **Generative modeling**, where the objective is to learn a model capable of sampling synthetic sequences $\widetilde{x}$ from $p(x)$ or $p(x \mid u)$ that preserve the statistical, structural, and temporal properties of the real data. A typical example of this generative task consists in using the trained model to synthesize entirely new tabular time series, which can be collected into large synthetic datasets. Such datasets can then be employed for downstream purposes such as data augmentation, benchmarking, or privacy-preserving data sharing, while retaining the key characteristics of the original data distribution.

The following sections build on this formalization to distinguish representation learning from data generation, introduce self-supervised learning paradigms for tabular data, and define the evaluation protocols adopted throughout the thesis.



| | user | district_id | frequency | city | region | year | month | day |
|---|---|---|---|---|---|---|---|---|
| $u_1$ | 0 | 1 | POPLATEK MESICNE | Hl.m. Praha | Prague | 97 | 5 | 30 |
| $u_2$ | 1 | 68 | POPLATEK MESICNE | Frydek - Mistek | north Moravia | 96 | 6 | 4 |
| $u_3$ | 2 | 73 | POPLATEK MESICNE | Opava | north Moravia | 96 | 11 | 10 |
| | 3 | 69 | POPLATEK MESICNE | Jesenik | north Moravia | 95 | 11 | 5 |
| | 4 | 25 | POPLATEK MESICNE | Klatovy | west Bohemia | 97 | 6 | 16 |
| ⋮ | 5 | 19 | POPLATEK MESICNE | Prachatice | south Bohemia | 94 | 11 | 22 |
| | 6 | 1 | POPLATEK MESICNE | Hl.m. Praha | Prague | 93 | 12 | 7 |
| | 7 | 41 | POPLATEK MESICNE | Usti nad Labem | north Bohemia | 95 | 1 | 15 |
| | 8 | 5 | POPLATEK TYDNE | Kolin | central Bohemia | 97 | 12 | 8 |
| $u_H$ | 9 | 54 | POPLATEK MESICNE | Brno - mesto | south Moravia | 97 | 2 | 12 |

$x^{(1)}$

| | user | Day | Month | Year | type_trans | operation | k_symbol | amount_trans | balance |
|---|---|---|---|---|---|---|---|---|---|
| $r_1$ | 0 | 12 | 3 | 98 | VYDAJ | PREVOD NA UCET | UVER | 8241.8 | 60001.0 |
| $r_2$ | 0 | 26 | 3 | 98 | VYDAJ | VYBER | None | 14800.0 | 38531.0 |
| $r_3$ | 0 | 31 | 3 | 98 | VYDAJ | VYBER | SLUZBY | 14.6 | 38752.9 |
| | 0 | 31 | 3 | 98 | PRIJEM | None | UROK | 236.4 | 38767.5 |
| | 0 | 8 | 4 | 98 | PRIJEM | PREVOD Z UCTU | None | 49451.0 | 88203.9 |
| ⋮ | 0 | 12 | 4 | 98 | VYDAJ | PREVOD NA UCET | UVER | 8241.8 | 79962.0 |
| | 0 | 12 | 4 | 98 | VYDAJ | PREVOD NA UCET | SIPO | 6670.0 | 73292.0 |
| | 0 | 25 | 4 | 98 | VYDAJ | VYBER | None | 22900.0 | 50392.0 |
| | 0 | 30 | 4 | 98 | VYDAJ | VYBER | SLUZBY | 14.6 | 50622.9 |
| $r_T$ | 0 | 30 | 4 | 98 | PRIJEM | None | UROK | 245.4 | 50637.5 |

**Figure 3.1:** Representation of a heterogeneous relational time series. The left panel shows the entities $u_1, u_2, \ldots, u_H$ of the parent table $P$, where each entity $u_h$ is described by a static tabular row. The right panel shows a tabular time series $x^{(1)} = (r_1, r_2, \ldots, r_T)$ that is a subsequence of rows drawn from the child table, temporally ordered and associated with the specific parent entity $u_1$.

## 3.2 Representation Learning vs. Data Generation

Within the formal framework introduced in the previous section, two closely related but conceptually distinct learning paradigms play a central role in this thesis: **representation learning** and **data generation**. While both aim to capture the underlying structure of heterogeneous tabular time series, they differ in objectives, modeling assumptions, and evaluation criteria.

Representation Learning Representation learning focuses on learning a mapping from raw data to a latent space that captures the salient properties of the input while discarding irrelevant variability [16]. Formally, given a time series $x$, the goal is to learn a function

$$f_\theta : x \to z,$$

where $z \in \mathcal{Z}$ is a latent representation that can be used as input for downstream tasks such as classification, regression, clustering, or anomaly detection.

In the context of heterogeneous tabular time series, effective representations must satisfy several requirements. First, they should encode **intra-row relationships**, capturing dependencies among heterogeneous attributes within individual records. Second, they must preserve **temporal structure**, allowing downstream models to exploit sequential dependencies and long-range pattern

[60, 109]. Third, representations should be **task-agnostic**, enabling transfer across different prediction problems without requiring task-specific retraining from scratch [91].

A key design choice concerns the **granularity of representations**. Some approaches learn representations at the level of individual rows, which can then be aggregated or further processed for sequence-level tasks. Others aim to directly produce sequence-level embeddings that summarize an entire time series. This thesis explores both perspectives, emphasizing architectures that can flexibly operate at multiple levels of abstraction.

Representation learning is particularly well suited to **self-supervised training**, where learning signals are derived from the structure of the data itself rather than from external labels [35]. In domains such as finance, where labeled data are scarce or costly, self-supervised representations provide a powerful foundation for scalable learning.

Data Generation Data generation, by contrast, aims to model the full data-generating distribution and to synthesize new samples that resemble those observed in the training set [51]. Formally, the objective is to learn a generative model $g_\varphi$ such that

$$\widetilde{x} \sim g_\varphi \approx p(x),$$

or, in the conditional case,

$$\widetilde{x} \sim g_\varphi(\cdot \mid u) \approx p(x \mid u).$$

In heterogeneous tabular time series, generative modeling is particularly challenging due to the need to jointly generate:

1. Heterogeneous feature values with complex dependencies,
2. Temporally coherent sequences,
3. Variable-length outputs,

4. Structurally valid records that respect domain-specific constraints.

Unlike representation learning, where small distortions in the latent space may be acceptable, generative modeling requires **high-fidelity reconstruction** of the data distribution. Errors in feature dependencies or temporal ordering can lead to unrealistic or inconsistent synthetic sequences, which limits the usefulness of generated data for downstream applications.

Nevertheless, data generation plays a crucial role in several practical scenarios. In privacy-sensitive domains, synthetic data can enable data sharing and model development without exposing sensitive information [63]. Generative models can also be used for data augmentation, scenario simulation, and stress testing, complementing discriminative models.

Complementarity of the Two Paradigms Although representation learning and data generation are often treated as separate problems, they are deeply interconnected. High-quality representations can facilitate generative modeling by providing structured latent spaces [67], while generative objectives can regularize representation learning by forcing models to capture the full data distribution [7].

This thesis adopts a **unified perspective**, in which representation learning serves as the foundation for both discriminative and generative tasks. Transformer-based architectures are first employed to learn expressive representations of heterogeneous tabular time series, and these representations are subsequently leveraged or extended to autoregressive [24] and diffusion-based generative models [58].

The next section introduces the self-supervised learning strategies that enable this unification, focusing on how meaningful learning signals can be extracted from unlabeled tabular time series.

## 3.3 Self-Supervised Learning for Tabular Data

Self-supervised learning has emerged as a central paradigm in modern Deep Learning, enabling models to learn meaningful representations from large

amounts of unlabeled data by exploiting intrinsic structural regularities [16]. In domains such as Natural Language Processing and Computer Vision, self-supervised objectives—such as masked language modeling or contrastive learning—have been instrumental in the success of large pretrained models [35, 56]. Extending these ideas to **tabular data**, and in particular to **time-dependent heterogeneous tabular data**, presents both opportunities and challenges.

The core principle of self-supervised learning is to define a **pretext task** that can be derived automatically from the data itself, without requiring manual annotations. In the context of tabular time series, the data contain multiple sources of structure that can be exploited: feature co-occurrence within a row, temporal dependencies across rows, and regularities in numerical and categorical distributions. Designing effective pretext tasks requires identifying which aspects of this structure should be predicted or reconstructed to induce useful representations.

A natural approach is **masked prediction**, inspired by masked language modeling [35]. In this setting, parts of the input are deliberately hidden or corrupted, and the model is trained to predict the missing components based on the remaining context. For heterogeneous tabular data, masking can be applied at different granularities:

- **Feature-level masking**, where individual attributes within a row are masked and must be inferred from other features and temporal context.
- **Row-level masking**, where entire rows are hidden, forcing the model to rely on surrounding transactions to reconstruct the missing record.
- **Temporal masking**, where timestamp-related information is partially masked, encouraging the model to learn temporal regularities and ordering [72].

These masking strategies allow the model to jointly capture intra-row dependencies among heterogeneous features and inter-row dependencies across time, aligning closely with the structural properties of transactional data.

An important consideration in self-supervised learning for tabular data is the treatment of **numerical attributes**. Unlike categorical features, numerical values do not naturally correspond to discrete prediction targets. As a result, masked prediction objectives must either rely on regression losses, discretization schemes, or specialized representations that enable numerical reconstruction [52]. The choice of numerical representation therefore has a direct impact on the effectiveness of self-supervised learning and on the quality of the learned latent space.

Another challenge arises from the **non-stationary and entity-specific nature** of transactional time series. Behavioral patterns may evolve over time, and different entities may exhibit fundamentally different dynamics. Self-supervised objectives must be robust to such variability and avoid learning trivial shortcuts, such as memorizing marginal distributions. This motivates the use of large and diverse datasets, as well as masking strategies that require genuine contextual understanding rather than local reconstruction [113].

From a methodological perspective, self-supervised learning enables a clear separation between **pre-training** and **fine-tuning**. During pre-training, the model learns general-purpose representations by solving the pretext task on unlabeled data. During fine-tuning, these representations are adapted to specific downstream tasks using supervised objectives. This separation is particularly valuable in domains like finance, where labeled data for certain tasks may be scarce, imbalanced, or sensitive.

In this thesis, self-supervised learning serves as a cornerstone for building scalable and transferable models for heterogeneous tabular time series. The following section introduces the evaluation protocols and metrics used to assess both representation learning and generative modeling performance within this framework.

## 3.4 Evaluation Protocols and Metrics

Evaluating models for heterogeneous tabular time series presents unique challenges that differ substantially from those encountered in unstructured domains. The diversity of learning objectives—ranging from representation learning to generative modeling—requires a careful definition of evaluation protocols and

metrics that accurately reflect model performance while remaining comparable across approaches [108].

Evaluation of Representation Learning For representation learning, the quality of a learned representation is not measured directly, but rather through its effectiveness on **downstream tasks** [16]. In this thesis, representations are primarily evaluated using supervised tasks relevant to transactional and financial data, such as classification and regression problems. Typical examples include fraud detection, credit risk assessment, and churn prediction.

The standard evaluation protocol follows a **pre-training–fine-tuning paradigm** [91]. Models are first pretrained in a self-supervised manner on large unlabeled datasets. The learned representations are then either frozen or fine-tuned using labeled data for a specific downstream task. Performance is compared against strong baselines, including traditional Machine Learning models and Deep Learning architectures trained from scratch.

To ensure fair comparisons, downstream evaluations control for factors such as training data size, feature availability, and model capacity. Performance metrics commonly include accuracy, area under the ROC curve (AUC), precision–recall metrics, and task-specific loss functions. Improvements in downstream performance are interpreted as evidence that the pretrained representations capture meaningful and transferable information about the underlying data [40].

Evaluation of Generative Models Evaluating generative models for tabular time series is inherently more complex, as there is no single metric that fully captures the quality of generated data [108]. Instead, evaluation must balance multiple criteria, including **fidelity**, **diversity**, and **utility**.

- **Fidelity** refers to how closely the generated data resemble the real data distribution. This includes matching marginal feature distributions, preserving inter-feature dependencies, and maintaining temporal coherence across sequences [114].
- **Diversity** measures the variability of generated samples and the model's

ability to cover multiple modes of the data distribution, rather than producing near-duplicates of a few frequent patterns [10].

- **Utility** assesses whether synthetic data can effectively replace or augment real data in downstream tasks, such as training classifiers or performing statistical analyses [41].

To address these aspects, this thesis adopts a combination of statistical, discriminative, and task-based evaluation strategies. Statistical metrics compare feature distributions and summary statistics between real and synthetic data. Discriminative metrics train a classifier to distinguish real from generated samples, using the classifier's accuracy as an indicator of distributional similarity [48]. Finally, utility-based metrics evaluate the performance of models trained on synthetic data when applied to real-world tasks [63].

For time-dependent tabular data, particular emphasis is placed on metrics that account for **temporal structure**. This includes evaluating sequence-level features, such as trends and periodicity, and using discriminators that operate on entire sequences rather than on individual rows [119]. Such metrics provide a more faithful assessment of temporal coherence than row-wise evaluations alone.

Reproducibility and Fair Comparison Given the lack of standardized benchmarks for heterogeneous tabular time series, reproducibility and transparency are critical [86]. Throughout this thesis, evaluation protocols are designed to minimize confounding factors, and results are reported across multiple random splits and experimental runs whenever possible. Comparisons with prior work are conducted using publicly available implementations or faithfully reimplemented baselines, ensuring that observed performance differences reflect genuine methodological advances.

By adopting a comprehensive and principled evaluation framework, this thesis aims to provide a reliable assessment of both representation learning and generative modeling approaches, laying a solid foundation for the empirical analyses presented in the subsequent chapters.

## 3.5 Unified Design Choices for Representations Across Discriminative and Generative Models

A central challenge in modeling heterogeneous tabular time series is that different tasks—most notably discriminative prediction and generative modeling—impose different constraints on how numerical, categorical, and temporal attributes should be represented. While the architectural foundations introduced in the previous sections deliberately aim for a unified framework, several representational decisions necessarily diverge across chapters due to the fundamentally different objectives of classification/regression vs. structured sequence generation. This section consolidates these design choices, explains the rationale behind them, and highlights both their commonalities and task specific differences, thereby improving cross chapter coherence.

### 3.5.1 Common Architectural Principles

Despite the differences introduced later in this section, all models in this thesis are built upon a shared set of architectural principles:

- **Hierarchical modeling of tabular time series**. Every architecture (UniTTab, UniTTab AR, BankDiT, TabDiT) decomposes each time step into field level embeddings (Field Transformer or VAE encoder) and processes the resulting transaction level embeddings with a Sequence Transformer. This hierarchical decomposition is common across all chapters because it cleanly separates intra transaction structure from temporal structure, and it scales well to variable length sequences.

- **Shared sequence level architecture**. Both discriminative and generative models use the same type of sequence encoder—a multi layer Transformer—to capture long range temporal dependencies. Although the downstream heads differ (classification head vs. AR field decoder), the core temporal modeling component remains architecturally identical.

- **Consistent treatment of categorical features**. Across all tasks, categorical fields (e.g., transaction type, merchant category, region code) are encoded through standard learnable embeddings. This design choice is shared because categorical attributes are naturally discrete, and embedding layers provide a unified semantic space regardless of whether the model is predicting labels or generating sequences.

- **Shared temporal decomposition strategy**. For timestamps, all models decompose time into year/month/day/hour categorical fields. This choice improves periodicity modeling (e.g., salaries, holidays, seasonality) and is equally beneficial for classification and generation tasks.

These commonalities form the backbone of a unified modeling paradigm and ensure that differences introduced in the following subsections emerge from task specific requirements—not architectural inconsistency.

### 3.5.2 Task Specific Divergence in Numerical Feature Representation

Where discriminative and generative models necessarily diverge is the representation of numerical fields, such as amounts, balances, frequencies, and measurements. Numerical values play different roles in prediction and generation, and each task demands a tailored encoding strategy.

#### Numerical Encoding in Discriminative Models

For discriminative tasks—such as pollution regression, fraud detection, churn prediction, or credit scoring—numerical attributes must be represented with high precision and smoothness to maximize predictive sensitivity. To this end, models in Chapters 4 to 6 rely on a frequency based encoding of numerical values (sinusoidal features), which provides:

- High resolution encoding across local neighborhoods

- Smooth variation that preserves gradients

- Effective discrimination even for small perturbations

In the Pollution dataset, the regression target is sensitive to variations in pollutant measurements at the fourth or fifth decimal place. The frequency based representation allows the model to distinguish values like 0.00341 vs. 0.00346 more effectively than discretization would, because sinusoidal encodings preserve fine scale differences in a continuous embedding space. This is essential to achieve competitive RMSE performance.

### Numerical Encoding in Generative Models

For generative models—autoregressive (UniTTab AR) and diffusion based (BankDiT, TabDiT)—the model must produce numerical attributes that are:

- exact, not approximate
- consistent with semantic constraints
- reconstructable without ambiguity

This makes frequency based encodings unsuitable, because frequency embeddings cannot be inverted exactly. For generation, the model must emit precise numerical values, so the thesis adopts digit based encodings, where numerical values are represented as sequences of digits (for high precision values). This ensures that generative decoding yields exact values without regression drift.

When the model needs to generate a transaction such as a cash withdrawal of exactly €50.00, digit based encoding guarantees exact reconstruction:

- “5”, “0”, “.”, “0”, “0” are represented as discrete tokens;
- the decoder generates those tokens deterministically;
- the output amount is precisely €50.00.

A frequency based representation would blur this information and could reconstruct €49.98 or €50.02 instead, which is unacceptable for generative tasks.

In autoregressive generation, numerical drift compounds over time. Digit based representations eliminate drift entirely: if "€1,200.75" must appear monthly (e.g., salary), the model can reproduce this exact amount every month without cumulative rounding errors.

### Why These Differences Are Necessary

The choice of numerical representation is driven by the fundamental objectives of each task category:

**Discriminative tasks**:

- focus on patterns, not exact reconstruction;
- require continuous sensitivity to small changes;
- benefit from smooth encoding spaces;
- do not require outputting numerical values.

**Generative tasks**:

- must output exact values;
- must guarantee internal consistency (e.g., merchant category ↔ amount);
- must reproduce financial constraints (no negative ATM withdrawals, integer-cent currency amounts);
- cannot rely on regression losses, which accumulate error.

Thus, although the models share a common representation pipeline for categorical, temporal, and architectural components, numerical features necessarily diverge because the reconstructability and sampling requirements of generative modeling fundamentally differ from the discriminative objective.

### Trade-offs and Cross-Chapter Implications

These design choices introduce several trade-offs that are explicitly addressed across the thesis:

- Expressiveness vs. fidelity: smooth encodings enhance discriminative expressiveness, while discrete encodings maximize generative fidelity.
- Invertibility vs. continuity: discriminative tasks optimize on continuous values; generative tasks require exact reversibility.
- Sampling diversity vs. token precision: autoregressive models benefit from variable-length tokenization but must balance diversity with precision; diffusion models benefit from continuous latents but require discretization steps for numerical decoding.
- Unified architecture, task specific heads: although the models share the same Sequence Transformer, discriminative models use a classification/regression head, while generative models use an autoregressive field decoder or a diffusion denoiser.

These distinctions explain why Chapters 4 to 6 (representation learning and prediction) employ continuous frequency-based embeddings, while Chapters 7 and 8 (transaction and time-series generation) rely on discrete tokenizations.

# 4

# One Transformer for All Time Series: Representing and Training with Time-Dependent Heterogeneous Tabular Data

**Publication:**

"S. Luetto *et al*., One transformer for all time series: Representing and training with time-dependent heterogeneous tabular data [5]".

The terms '*our*' and '*we*' refer to this thesis's author and the publication's authors.

This chapter introduces UniTTab, a unified Transformer-based architecture designed to learn expressive representations of heterogeneous, time-dependent tabular data. UniTTab is motivated by the limitations of existing approaches reviewed in Chapter 2 and by the methodological framework outlined in Chapter 3.

This Chapter is related to the publication "S. Luetto *et al*., One transformer for all time series: Representing and training with time-dependent heterogeneous tabular data" [5]. See the list of Publications on page 175 for more details.

Its design explicitly addresses feature heterogeneity, temporal dependencies, and scalability, while remaining compatible with large-scale self-supervised learning.

## 4.1 Preliminaries

**Problem Statement.** Tabular data are represented as a set of attributes (field names) $A = \{a_1, ..., a_k\}$, where each $a_j \in A$ can be either categorical or numerical, and a set of table rows $\boldsymbol{r}_1, ..., \boldsymbol{r}_N$, which specify a value for each field: $\boldsymbol{r}_i = [v_1, ..., v_k]$. If $a_j$ is numerical, then $v_j \in \mathbb{R}$, otherwise $v_j \in V_j$, where $V_j$ is an unordered attribute-specific vocabulary of categories. A time series is a (variable length) sequence of rows $\boldsymbol{s} = [\boldsymbol{r}_1, ..., \boldsymbol{r}_t]$ which are related to each other by a temporal dynamics. For instance, in financial transactional data, $\boldsymbol{s}$ can represent the last $t$ bank account transactions of a given client. When adopting a Deep Learning method, a common paradigm is to use a large dataset of time series to pretrain a network with self-supervised learning, and then fine-tune the model for a specific downstream task using task-specific labeled data and a possible smaller (supervised) dataset.

In this paper, we further generalize the previous scenario introducing time series composed of different row types. This generalization is particularly useful in real life datasets, in which, for instance, a transaction time series is composed of different transaction types (e.g., POS type, ATM type, etc.). Formally, we describe this situation using a function which associates each row in $\boldsymbol{s}$ to a predefined set of row types: $type(\boldsymbol{r}_i) = h \in T = \{1, ..., n\}$ and using a type-dependent set of attributes $A_h = \{a_1, ..., a_{k_h}\}$ to specify the fields of $\boldsymbol{r}_i$. Note that the cardinality of the attributes ($k_h$) varies depending on $h$. Moreover, we assume that $type(\boldsymbol{r}_i)$ is always defined for each $\boldsymbol{r}_i \in \boldsymbol{s}$: for example, each transaction in a time series of a bank account can be of only one type (e.g., POS, or ATM, etc.).

**TabBERT architecture.** TabBERT is a hierarchical architecture composed of two different Transformers, trained end-to-end (Figure 4.1 (a)). The first Transformer (called "Field Transformer") takes as input the $k$ field values of a single table row $\boldsymbol{r}_i$. Note that $k$ is constant for all the rows, as TabBERT implic-

itly assumes that there is only one row type (in our notation: $|T| = n = 1$). Numerical features are discretized using an attribute specific set of bins. In this way, both numerical and categorical features can be associated to a specific discrete token. The tokens are transformed in embedding vectors using a standard look-up table of learned embeddings [109]. In Figure 4.1 (a), this is indicated as a set of field embeddings $\boldsymbol{f}_1, ..., \boldsymbol{f}_k$. The Field Transformer transforms these vectors in $k$ final embeddings of dimension $d$, which are concatenated in a single vector $\boldsymbol{g}$ of dimensions $d \cdot k$. Then, $\boldsymbol{g}$ is fed to the second Transformer ("Sequence Transformer"), jointly with the representations of all the other rows in the input time series $\boldsymbol{s}$. Note that the dimension of each $\boldsymbol{g}$ should be constant because $\boldsymbol{g}$ is a fixed-size initial embedding vector for the second Transformer. The Sequence Transformer outputs a sequence of $t$ final embedding vectors $\boldsymbol{z}_1, ..., \boldsymbol{z}_t$. Finally, each $\boldsymbol{z}_i$ is split in $k$ vectors, on top of which a shallow MLP is used to output a posterior distribution over the attribute-specific vocabulary. This makes it possible to apply a Masked Token pretext task [35] during pre-training, in which a few tokens are randomly masked and the network is asked to predict the masked tokens.

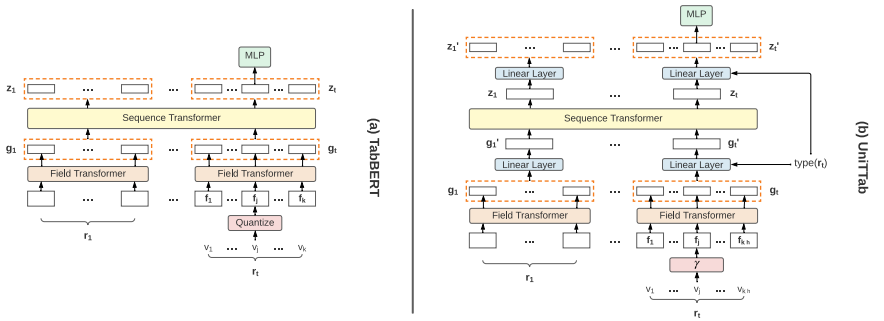



**Figure 4.1:** A schematic comparison between the architectures of TabBERT (a) and UniTTab (b). In both figures, $v_j$ is a numerical value. Note that in (b) the number of attributes of each row ($k_b$) is variable.

## 4.2 Method

In this section we present UniTTab, showing the architecture of the network, the way in which heterogeneous features are represented and the uniform pre-training strategy.

**Row-type dependent embedding.** We first extend the hierarchical architecture of TabBERT to deal with a variable number of row types ($n > 1$). The main problem we need to solve is that $k_h$ depends on $type(\boldsymbol{r}_i)$ for each $\boldsymbol{r}_i \in \boldsymbol{s}$, while the dimension of $\boldsymbol{g}$ should be fixed (Section 4.1). We solve this problem using a linear projection layer (Figure 4.1 (b)) which takes $h = type(\boldsymbol{r}_i)$ as input and transforms $\boldsymbol{g} \in \mathbb{R}^{d \cdot k_h}$ in $\boldsymbol{g}' \in \mathbb{R}^m$, where $m$ is fixed and the transformation depends on a row-type specific linear matrix $W_h$:

$$\boldsymbol{g}' = W_h \boldsymbol{g} \quad (W_h \in \mathbb{R}^{m \times (d \cdot k_h)}). \tag{4.1}$$

The set of learnable projection matrices $W_1, ..., W_n$, one for each row type, constitutes a look-up table of embeddings for the initial layer of the second Transformer, and naturally extends the common initial embedding look-up table used in Transformer networks. Analogously, each final row embedding $\boldsymbol{z}$ (Section 4.1) is transformed in $\boldsymbol{z}' \in \mathbb{R}^{d \cdot k_h}$ using a specific weight matrix $S_h$ ($S_h \in \mathbb{R}^{(d \cdot k_h) \times m}$) before being fed to the final MLP (Figure 4.1 (b)).

**Feature representation.** We represent each categorical feature using a standard linear embedding based on its attribute-specific vocabulary. However, for numerical features, we extract a frequency-based representation as follows. Let $v$ be a scalar value corresponding to a numerical attribute. Similarly to [79], we transform $v$ using:

$$\gamma(v) = (\sin(2^0 \pi v), \cos(2^0 \pi v), ..., \sin(2^{L-1} \pi v), \cos(2^{L-1} \pi v)), \tag{4.2}$$

where $L = 8$ is the number of sine/cosine pairs used (see Section 4.3.1). The vector $\gamma(v)$ so obtained is then fed to a linear (learnable) layer whose output is the initial embedding vector for $v$. Finally, similarly to [96], we represent the numerical

value of a time stamp attribute (e.g, measured in minutes) using a combination of different time fields (i.e, the year, the month, the day, and, if necessary, the hour). Each such basic value is represented as a categorical feature (e.g., with 12 elements for the month, etc.). In preliminary experiments, we also tried to represent each basic time field of a date as a numerical attribute (i.e. by means of our frequency-based representation), but this led to slightly worse results, presumably because the periodicity of some time series (e.g., in transactional data) can better be represented by the network as a categorical value. Note, that, differently from [96], we do not add the corresponding field embeddings but treat them as additional fields simply by increasing the number of attributes of each single row.

**Training.** During the unsupervised pre-train stage, we train UniTTab using *only* a Masked Token pretext task. Specifically, for an input sample $\boldsymbol{s} = [\boldsymbol{r}_1, ..., \boldsymbol{r}_t]$, we randomly replace a field value $v \in \boldsymbol{r}_i$ with the special symbol `[MASK]`. We use a standard replacement probability value $p_f = 0.15$ [35, 83]. Moreover, with probability $p_r = 0.1$, we also mask *all* the values in a row $\boldsymbol{r}_i$, while, for the fields representing the time stamp, they are always either jointly masked or jointly unmasked. We call these additional masking strategies "Row masking" and "Time stamp masking", inspired by the block masking of adjacent image patches used in BEiT [14], and we use them to make the pretext task more challenging for the network.

Note that we can replace (the initial embedding vector of) $v$ with `[MASK]` independently on whether $v$ is numerical or categorical. However, the problem is what the network should predict in correspondence of $v$ if this is not an element of a discrete vocabulary. A possible solution could be to adopt a regression loss function and directly ask to the network to reconstruct the original numerical value $v$. The disadvantage of this hybrid solution is that we would need two different loss functions: one for the categorical features (e.g., the Cross Entropy), and another for the numerical values (e.g., an MSE loss), which then need to be suitably weighted. Conversely, we propose a different solution: inspired by BEiT, we quantize $v$ and we use its categorical representation as the target label. Specifically, if $a_j$ is a numerical attribute, we define a vocabulary of bin values $V_j = \{b_1, ..., b_q\}$ spanning the whole range of possible values for $a_j$. Then, when

a given value $v$ for this attribute is masked, we quantize $v$ ($b = quantize(v)$) and we use $b \in V_j$ as a pseudo label for $v$, which is used as the token to be predicted. Note that $b$ is *not* input to the network.

When we compute the loss function, we use Label Smoothing [107], which replaces the one-hot vector representation of a categorical label $v \in V_j$ with a vector of smoothed probability values $\boldsymbol{p}(v)$. In more detail, if $V_j = \{v_1, ..., v_{q_j}\}$ is the vocabulary corresponding to $a_j$, and $q_j$ is its cardinality, then a ground truth value $v \in V_j$ is associated with the vector $\boldsymbol{p}(v)$ defined as follows:

$$[\boldsymbol{p}(v)]_l = \begin{cases} 1-\varepsilon, & \text{if } l = v \\ \frac{\varepsilon}{q_j - 1} & \text{otherwise,} \end{cases} \tag{4.3}$$

where $[\boldsymbol{p}(v)]_l$ is the $l$-th element of $\boldsymbol{p}(v)$ and $\varepsilon$ a small constant. Furthermore, in case $a_j$ is a numerical attribute, we propose Neighborhood Label Smoothing, in which $b = quantize(v)$ is smoothed using only a small neighborhood centered in $b$. Specifically, we use the range $R = \{b-5, ..., b+5\}$, and we replace Equation (4.3) with:

$$[\boldsymbol{p}(v)]_l = \begin{cases} 1-\varepsilon, & \text{if } l = b \\ \frac{\varepsilon}{10} & \text{if } l \in R, l \neq b \\ 0 & \text{otherwise.} \end{cases} \tag{4.4}$$

Note that this is possible because, if $a_j$ is numerical, then $V_j$, obtained with quantization, is an ordered set.

Finally, if $\boldsymbol{s}'$ is the perturbed version of $\boldsymbol{s}$, in which some random field values have been masked as explained above, then our Masked Token pretext task can be formulated as minimizing the following Cross Entropy loss:

$$\min_{\theta} - \sum_{v \in \boldsymbol{s}} 1_{Masked(v)} \boldsymbol{p}(v) \log p_\theta(quantize(v)|\boldsymbol{s}'), \tag{4.5}$$

where: $\theta$ are the parameters of the network, $p_\theta(y|\boldsymbol{x})$ is the probability of the network to predict $y$ given the sequence $\boldsymbol{x}$ as input, $1_{Masked(v)}$ is 1 only when $v$ was masked (otherwise is 0), $quantize(v) = v$ if $v$ is categorical, and, with a slight

abuse of notation, $v \in \boldsymbol{s}$ indicates a generic field value of one of the rows in $\boldsymbol{s}$.

## 4.3 Experiments

In this section, we evaluate UniTTab using different datasets and downstream tasks. We use the same architecture for all the datasets, with the only difference being the number of fields ($k$) and the length ($t$) of the time series, which depend on the specific dataset and the adopted evaluation protocol. Specifically, independently of the dataset, we always use only one self-attention layer with 8 heads in the Field Transformer, and 12 self-attention layers with 12 heads each in the Sequence Transformer. For a fair comparison, the total number of parameters of UniTTab was kept approximately the same as in TabBERT. We use five different-size datasets of time series of heterogeneous tabular data: the Pollution Dataset [73], the Transaction Dataset [83] (both adopted by TabBERT), the PKDD'99 Financial Dataset [18], the Age2 dataset [49], and our Real Bank Account Transaction Dataset (in short, RBAT Dataset). In the latter, each time series is composed of three different row types (i.e., $n = |T| = 3$, see Section 4.1). The other four datasets have only one row type ($n = |T| = 1$), thus, in all the experiments but those on the RBAT Dataset, we use only one projection matrix in Equation (4.1). All the five datasets are composed of tabular data with both numerical and categorical fields. In all the experiments, the Deep Learning based models are first pre-trained using self-supervision (Section 4.2) and then evaluated using a dataset specific (supervised) downstream task, while standard Machine Learning methods are directly trained on the downstream task training set. Moreover, to enable a fair comparison with other methods tested on the Pollution Dataset and the Transaction Dataset, we follow the protocol defined in [83] for the downstream task definition and the training/testing partitions. However, since the stride used in [83] to extract the time series from the Pollution Dataset leads to a partial information leak between the training and the testing data of the downstream task, we additionally use a different training/testing partition for this dataset, which we call "our partition", while we use the term "original partition" to refer to the initial split adopted in [83] and in other works.

### 4.3.1 Ablation study

In this section, we analyze the contribution of each component of our method. We use the Pollution Dataset [73] and a random subset of the Transaction Dataset [83] (400K time series). The former dataset is composed of time series where each row contains $k = 11$ fields, 10 of which are numerical attributes and one is categorical. Hence, this dataset is particularly suitable for investigating the influence of different numerical feature representations (Section 4.2). The downstream task is a regression task, and the performance is measured using RMSE (the lower the better). In the Transaction Dataset, the rows of the time series are composed of 2 numerical and 8 categorical attributes, and its associated downstream task is a binary classification task. In this case, we use the F1 score as the performance metric (the higher the better). Both datasets contain a time stamp attribute, which we represent using 4 time attributes as explained in Section 4.2. For both datasets, following [83], after self-supervised pre-training, the final embeddings of the Sequence Transformer $(\boldsymbol{z}_1, \ldots, \boldsymbol{z}_t)$ are used as input to an LSTM which is (separately) trained to solve a supervised regression or binary classification task using the corresponding task labels. Specifically, we first train UniTTab using Equation (4.5) and then we train an LSTM (using the same LSTM architecture adopted in [83]) on top of the final embeddings of our Sequence Transformer. In Section 4.3.2 we also show the results obtained by directly fine-tuning our model, without using an LSTM, as well as comparative results obtained using the full Transaction Dataset and both the original and our partition of the Pollution Dataset.

Table 4.1 analyses the impact of the main components of our method. The first row is the baseline, which corresponds to our implementation of TabBERT, starting from its publicly available code and keeping fixed all the main hyper-parameters (e.g., the number of layers of the two Transformers, the number of heads, the embedding size, etc.). In this baseline, the numerical features are quantized into discrete bins and treated as categorical *when input to the network*. Note that there is no Row or Time stamp masking nor Label Smoothing. The the Cross Entropy loss is the only loss used because the tokens of the quantized nu-

**Table 4.1:** Ablation study using the Pollution Dataset (our partition) and a subset of the Transaction Dataset.

| | Quantized numerical features | Combina. time stamp | Frequency based num. features | Regression loss | CE loss (only) | Standard Label Smoothing | Neighbor Label Smoothing | **Pollution Dataset** (RMSE ↓) | **Transaction Dataset** (F1-score ↑) |
|---|---|---|---|---|---|---|---|---|---|
| 1 | ✓ | | | | ✓ | | | 34.30 | 0.829 |
| 2 | ✓ | ✓ | | | ✓ | | | 32.23 | 0.840 |
| 3 | | | ✓ | ✓ | | | | 32.47 | 0.844 |
| 4 | | | ✓ | | ✓ | | | 31.52 | 0.846 |
| 5 | | ✓ | ✓ | | ✓ | | | 29.63 | 0.847 |
| 6 | | ✓ | ✓ | | ✓ | ✓ | | 29.47 | 0.848 |
| 7 | | ✓ | ✓ | | ✓ | ✓ | ✓ | **29.05** | **0.850** |

merical features can be masked and predicted just like categorical tokens. The second row of the table shows the improvement obtained when the time stamp is split in 4 different fields, which corresponds to a significant performance improvement in both datasets and tasks. In the third row, we replace the discrete representation of the numerical features with our frequency-based representation (Section 4.2). In this case, for each masked numerical field value $v$, we use a regression function, which consists in predicting the original scalar value of $v$ (before the frequency-based embedding). We use the squared difference between the predicted and the ground truth value as the loss function (MSE loss) for these numerical features, which is summed to the Cross Entropy loss computed with the categorical features. Note that, as mentioned in Section 4.2, one of the problems when using two different loss functions is the necessity to weight their relative importance. In this ablation experiment, we set the MSE loss weight using a simple heuristic in which we compute the average MSE loss value and the average Cross Entropy loss value and we impose these two values to be equivalent using a relative weight. Comparing this row with the baseline (first row), the improvement obtained when using a frequency-based numerical representation jointly with a regression loss is comparable with the introduction of the combinatorial time stamp. In row 4, the regression loss is replaced by the Cross Entropy loss, which is applied to numerical features using pseudo target labels (see Equation (4.5)). Comparing this row with the baseline, we obtain the largest relative improvement, which shows that using frequency-based numerical representations jointly

**Table 4.2:** An analysis of different masking strategies using the Pollution Dataset (our partition) and a subset of the Transaction Dataset.

| | Row masking $p_r = 0.05$ | Row masking $p_r = 0.1$ | Time stamp masking | **Pollution Dataset** (RMSE ↓) | **Transaction Dataset** (F1-score ↑) |
|---|---|---|---|---|---|
| 1 | | | | 29.05 | 0.850 |
| 2 | ✓ | | | 29.01 | 0.851 |
| 3 | | ✓ | | 28.91 | 0.854 |
| 4 | | | ✓ | 28.82 | 0.855 |
| 5 | | ✓ | ✓ | **27.99** | **0.858** |

with pseudo labels for the Cross Entropy can significantly boost the network performance. In row 5, the combinatorial time stamp is added to the setting of row 4, bringing an additional benefit. Finally, row 6 and row 7 show the results corresponding to the introduction of standard Label Smoothing and Neighborhood Label Smoothing, respectively.

In Table 4.2, we analyze the impact of using Row masking and Time stamp masking, where the first strategy is evaluated with two different selection probability values $p_r$. The best result corresponds to using both Row masking (with probability $p_r = 0.1$) and Time stamp masking. In Table 4.3 we empirically evaluate the influence of the number of frequency function pairs $L$ (Section 4.2) used for our frequency-based numerical feature representation. In the rest of this paper, we use the best value ($L = 8$) for all the other datasets and tasks.

Finally, we postpone the ablation of the Row-type dependent embedding (Section 4.2) to Section 4.3.2, where we introduce our RBAT Dataset composed of variable row types. In the same section, we sow the impact of different amount of pre-training data on the downstream task performance.

**Table 4.3:** The influence of the number of frequency function pairs ($L$) using the Pollution Dataset (our partition) and a subset of the Transaction Dataset.

| $L$ | **Pollution Dataset** (RMSE ↓) | **Transaction Dataset** (F1-score ↑) |
|---|---|---|
| 4 | 28.63 | 0.828 |
| 6 | 28.38 | 0.830 |
| 8 | **27.99** | **0.858** |
| 10 | 28.11 | 0.808 |
| 12 | 28.20 | 0.804 |

### 4.3.2 Main results

In this section, we compare UniTTab with different state-of-the-art approaches using different datasets and downstream tasks.

**Pollution prediction task.** This is the regression task used in Section 4.3.1. We show results based on two main paradigms: based on LSTM training and on directly fine-tuning the pre-trained model. The former is based on the protocol proposed by Padhi et al. [83], and trains a separate LSTM using the vectors $\boldsymbol{z}_1, \ldots, \boldsymbol{z}_t$ as input and the labels of the downstream task as the supervision (see Section 4.3.1). However, the latter, proposed here, is likely a much more natural choice, and it is coherent with most of the recent AI literature, where the backbone network, after pre-training, is directly fine-tuned for a specific downstream task without training a separate network. Specifically, when we fine-tune either UniTTab or TabBERT, we include a `[CLS]` token in the input sequence of the Sequence Transformer and we use the corresponding $\boldsymbol{z}_{\texttt{[MASK]}}$ final embedding vector as input to a final linear layer dedicated to the specific task (and trained from scratch). In case of LUNA [55], we report the results we obtained using its public code and only the LSTM training paradigm (adopted also in [55]), because fine-tuning this method requires non-trivial modifications of its architecture.

To enable a comparison with results published in previous work, in the experiments of Table 4.4 we strictly follow [83] and we use the original partition of the Pollution Dataset (see Section 4.3) jointly with a time series length $t = 10$. Conversely, in Table 4.5 we use our partition (where $t = 10$ but also the time

**Table 4.4:** Pollution prediction task (original partition). † Our reproduction. ‡ Results reported in the corresponding paper.

| Downstream task training | Model | RMSE |
|---|---|---|
| Fine-tuning | TabBERT † | 24.10 |
| | UniTTab (ours) | **20.05** |
| LSTM | TabBERT [83] ‡ | 32.80 |
| | LUNA † | 37.73 |
| | UniTTab (ours) | **25.42** |
| Training from scratch | XGBoost † | 34.14 |
| | CatBoost † | 34.15 |
| | VAR † | 83.83 |

series extraction stride is 10) and we repeat all the experiments 5 times with different random seeds. Note that the random seed is used both to initialize the models' parameters and to randomly split the time series in training and testing splits. Finally, in Table 4.6 we extend these experiments to longer time series with $t = 50$ . In all cases (Tables 4.4 to 4.6), UniTTab outperforms both TabBERT and LUNA by a large margin, and, as expected, direct fine-tuning significantly improves over the separate LSTM training.

In the same tables, we also report the results obtained using both XGBoost [26] and CatBoost [87], which are the state-of-the-art non Deep Learning based methods for tabular data. Specifically, since both XGBoost and CatBoost cannot directly work on (variable-length) time series, we used a standard library [31] to extract features from a time series. These "engineered" features include field-specific statistics (e.g., the mean or the variance for numerical features and the mode for the categorical ones, etc.), autocorrelation with different "lags", the number of local minima and maxima, etc. Moreover, the features are automatically pre-selected using a Filter feature selection method on a validation set extracted and separated from the training data. Finally, we used grid search on the validation set to set the optimal values of the XGBoost and the CatBoost hyperparameters (e.g., the number of trees, the max depth of each tree, etc.). After feature selection and hyperparameter tuning, the validation set is merged with the

**Table 4.5:** Pollution prediction task (our partition): average and standard deviation results obtained with 5 random seeds. † Our reproduction using the official code.

| Downstream task training | Model | RMSE |
|---|---|---|
| Fine-tuning | TabBERT † | 31.41 (±1.74) |
| | UniTTab (ours) | **25.37** (±1.59) |
| LSTM | TabBERT † | 37.20 (±1.67) |
| | LUNA † | 45.17 (±0.62) |
| | UniTTab (ours) | **30.88** (±1.70) |
| Training from scratch | XGBoost † | 48.89 (±0.73) |
| | CatBoost † | 47.09 (±0.89) |
| | VAR † | 84.05 (±1.71) |

training set and XGBoost/CatBoost are re-trained using the same data adopted for training the LSTMs or fine-tuning the UniTTab/TabBERT models on the supervised downstream task. We will use this procedure (task and dataset dependent feature selection and hyperparameter tuning) for XGBoost CatBoost in all the other downstream tasks of this paper. Additionally, we use Vector AutoRegression (VAR) [77], which is a common (non Deep Learning based) autoregressive model for multivariate time series, where we transform categorical features in numerical by assigning a numerical value to each element in the vocabulary $V_j$. In case of VAR, the field values of all the rows of the time series are directly input to the model without intermediate engineered features. The results in Tables 4.4 to 4.6 show that UniTTab largely outperforms "standard" Machine Learning approaches, and the gap is particularly significant with longer sequences (Table 4.6), where the hierarchical Transformer architecture of both UniTTab and TabBERT can likely better represent long inter-row dependencies.

**Table 4.6:** Pollution prediction task (our partition) with longer time series ($t = 50$). Average and standard deviation results obtained with 5 random seeds. † Our reproduction using the official code.

| Downstream task training | Model | RMSE |
|---|---|---|
| Fine-tuning | TabBERT † | 30.95 ($\pm$0.53) |
| | UniTTab (ours) | **25.11** ($\pm$0.48) |
| LSTM | TabBERT [83] ‡ | 36.67 ($\pm$0.81) |
| | LUNA † | 43.93 ($\pm$2.83) |
| | UniTTab (ours) | **28.17** ($\pm$0.51) |
| Training from scratch | XGBoost † | 56.37 ($\pm$0.92) |
| | CatBoost † | 56.26 ($\pm$0.99) |
| | VAR † | 83.37 ($\pm$0.95) |

**Fraud detection task.** This is the classification task used in Section 4.3.1. Differently from Section 4.3.1, in the experiments of this section we use the full dataset for pre-training ($\sim$ 4, 9M time series), which leads to a higher absolute performance in the results reported in Table 4.7. Specifically, we use the original time series length $t = 10$ in Table 4.7 and we extend the experiments to $t = 50$ in Table 4.8. These tables shows that UniTTab outperforms all the compared methods, independently on whether an LSTM is used or not and the gain is particularly significant with longer time series (Table 4.8). Similarly to the Pollution prediction task, in Tables 4.7 and 4.8 we also report the results obtained using VAR, XGBoost and CatBoost, with task and dataset specific features and hyperparameters for XGBoost and CatBoost (see above). Also in the Fraud detection task, UniTTab significantly outperforms all the non Deep Learning based methods, and the improvement is particularly significative with longer time series (Table 4.8).

**Table 4.7:** Fraud detection task ($t = 10$). † Our reproduction. ‡ Results reported in the corresponding paper.

| Downstream task training | Model | F1 score |
|---|---|---|
| Fine-tuning | TabBERT † | 0.910 |
| | TabAConvBERT [96] ‡ | 0.896 |
| | UniTTab (ours) | **0.915** |
| LSTM | TabBERT ‡[83] | 0.860 |
| | LUNA ‡[55] | 0.862 |
| | UniTTab (ours) | **0.914** |
| Training from scratch | XGBoost † | 0.779 |
| | CatBoost † | 0.499 |
| | VAR † | 0.495 |

**Table 4.8:** Fraud detection task ($t = 50$). † Our reproduction.

| Downstream task training | Model | F1 score |
|---|---|---|
| Fine-tuning | TabBERT † | 0.777 |
| | UniTTab (ours) | **0.812** |
| LSTM | TabBERT †[83] | 0.883 |
| | LUNA †[55] | 0.884 |
| | UniTTab (ours) | **0.931** |
| Training from scratch | XGBoost † | 0.349 |
| | CatBoost † | 0.337 |
| | VAR † | 0.497 |

**Loan default prediction task.** For this task, we use the PKDD'99 Financial Dataset [18], which is smaller than the other datasets, thus we report the average results obtained with 5 random splits. The dataset consists of the complete (real) transaction history of several bank customers, the average length of which is 232. To increase the number of time series available for pre-training, we set a maximum length value $t_{max}$, and we use time series with variable length $t$, where $t \leq t_{max}$. Specifically, for each client, assuming that $t_{all}$ is her/his total number of transactions, if $t_{all} < t_{max}$, then we use $t = t_{all}$. Otherwise, *at each pre-training iteration* we randomly select a time series $\boldsymbol{s}$ of $t = t_{max}$ consecutive transactions

**Table 4.9:** Loan default prediction task: average and standard deviation results obtained with 5 random seeds. † Our reproduction. ‡ Results reported in the corresponding paper.

| Pre-training ($t_{max}$) | Model | F1 score |
|---|---|---|
| 50 | TabBERT † | 0.611(±0.032) |
| | LUNA † | 0.604(±0.048) |
| | UniTTab (ours) | 0.619(±0.011) |
| 100 | TabBERT † | 0.636(±0.024) |
| | LUNA † | 0.624(±0.075) |
| | UniTTab (ours) | 0.654(±0.032) |
| 150 | TabBERT † | 0.620(±0.024) |
| | LUNA † | 0.637(±0.043) |
| | UniTTab (ours) | 0.673(±0.038) |
| | Random Forest [115] ‡ | 0.2667 |
| | XGBoost † | 0.608(±0.079) |
| | CatBoost † | 0.527(±0.065) |
| | VAR † | 0.474(±0.007) |

over the sequence of all possible $t_{all}$ rows of that client. In Table 4.9 we present results with $t_{max}$ ranging from 50 to 150. This experiment is important to test whether a model dealing with time series can operate in real life scenarios where the actual row sequence length is variable and it can be very long ($t \geq 50$).

We fine-tune the models similarly to the Fraud detection task (i.e., using a `[CLS]` token, etc.). Both in the fine-tuning and in the testing stage, the length $t$ of a transaction is variable. However, differently from the pre-training stage, if $t_{all} > t_{max}$, then we select the *last* $t_{max}$ transactions.

Table 4.9 shows that UniTTab outperforms both TabBERT and LUNA, especially when using very long sequences ($t_{max} = 150$). The bottom of that table shows the results of VAR, XGBoost and CatBoost, with the usual dataset and task specific features and hyperparameters selection for XGBoost and CatBoost. Additionally, we also report the result obtained by Xu [115] using Random Forests with 18 features, some of which are computed extracting aggregated information from the time series, while others are socio-demographic information about the client which all the other methods do *not* use. The results in Table 4.9 show that also in this small dataset UniTTab outperforms the standard

**Table 4.10:** Age prediction task on the Age2 dataset. † Our reproduction.

| Downstream task training | Model | F1 score |
|---|---|---|
| Fine-tuning | TabBERT † | 0.662 |
| | UniTTab (ours) | **0.678** |
| LSTM | TabBERT ‡[83] | 0.645 |
| | LUNA ‡[55] | 0.648 |
| | UniTTab (ours) | **0.664** |
| Training from scratch | XGBoost † | 0.542 |
| | CatBoost † | 0.622 |
| | VAR † | 0.354 |

Machine Learning methods and, similarly to the other tasks, the benefit is larger with longer time series.

**Age prediction task.** These experiments are based on the Age2 Dataset and its associated Age prediction downstream task. Similarly to the PKDD'99 Financial Dataset, Age2 is composed of the bank transaction history of different bank customers. Each row (transaction) is composed of $k = 3$ fields (5 after splitting the time stamp in 3 fields). For data augmentation, we follow the same protocol adopted when pre-training on the PKDD'99 Financial Dataset, i.e., we use time series of variable length $t \leq t_{max}$, extracted at random at each iteration from the whole history of each bank account. We use $t_{max} = 50$. Similarly to the Loan default prediction task, during both fine-tuning and testing, we use time series with variable length $t$ and, if $t_{all} > t_{max}$, then we select the *last* $t_{max}$ transactions.

**Churn prediction task.** In this last battery of experiments, we use the large RBAT Dataset and its associated downstream task (Churn prediction). Similarly to the PKDD'99 Financial Dataset and the Age2 Dataset, also this dataset is composed of the bank transaction history of different real users. However, RBAT is much more complex than most public datasets, and each transaction (row) of its time series is associated with a specific type. To be precise, there are $n = 3$ types of mutually exclusive transactions (see Section 4.1): (1) generic transactions, with $k_g = 5$ fields, (2) POS transactions, with $k_p = 8$ fields and (3) ATM transactions, with $k_a = 7$ fields. Since the generic transaction fields are shared also by the other

two types of transactions, the total number of different fields is 10, which become 12 when the time stamp is split in 3 different fields (day, month, year).

For data augmentation, we follow the same protocol adopted when pre-training on both the PKDD'99 Financial Dataset and the Age2 Dataset, i.e., we use time series of variable length $t \leq t_{max}$, extracted at random at each iteration from the whole history of each account. In this case, we use $t_{max} = 150$. Since the baseline TabBERT cannot deal with different types of transactions and needs a fixed number of fields for each row, when training TabBERT we concatenate all the $k = 10$ fields of all the row types. In a given row $\boldsymbol{r}$, the field values of those attributes which are not included in the type of $\boldsymbol{r}$, are represented using a `[MISSING]` token. Note that this solution is computationally more expensive, especially in the Field Transformer, where the computation costs grow quadratically with $k$, and the computational benefit of our proposal is larger when $n$ is bigger. For ablation reasons (see Section 4.3.1), we also train a modified version of TabBERT in which we introduce the Row-type dependent embedding (Equation (4.1)) *without any other changes*. This baseline corresponds to the baseline used in Table 4.1 (first row), on top of which we add the Row-type dependent embedding described in Section 4.2, thus we call it "TabBERT + Variable Row Types" (TabBERT + VRT).

Similarly to the Loan default prediction task and the Age prediction task, during both fine-tuning and testing, we use time series with variable length $t$ and, if $t_{all} > t_{max}$, then we select the *last* $t_{max}$ transactions (which are the closest to a possible bank account closure). Since this dataset is much larger than the others, for computational reasons we have not included LUNA in this comparison. However, similarly to the other tasks, we also used VAR, XGBoost and CatBoost with the usual dataset and task specific features and hyperparameter selection for the last two.

Table 4.11 shows that TabBERT + VRT significantly improves the baseline TabBERT, showing that the Row-type dependent embedding has an accuracy benefit on the downstream task which goes beyond its computational advantages. Moreover, also in this task our full method (UniTTab) outperforms TabBERT, VAR, XGBoost and CatBoost with a large margin.

**Table 4.11:** Churn prediction task on the RBAT Dataset. † Our reproduction.

| Pre-training ($t_{max}$) | Model | F1 score |
|---|---|---|
| 150 | TabBERT † | 0.526 |
| | TabBERT + VRT (ours) | 0.536 |
| | UniTTab (ours) | **0.604** |
| | XGBoost † | 0.485 |
| | CatBoost † | 0.483 |
| | VAR † | 0.472 |

### 4.3.3 Effect of pre-training

One of the main advantages of using Deep Learning methods over more traditional Machine Learning approaches, is the possibility to pretrain a large network using self-supervision and a large unsupervised dataset, and then fine-tune the same network on the available supervised data of a downstream task. For instance, in case of UniTTab, we first pre-train the network using a Masked Token pretext task (Section 4.2) using all the available training data. These data do not need to be annotated, since predicting a masked token is a self-supervised task. Then, we use downstream task-specific annotated training data (which are usually much sparser than the unsupervised data) to fine-tune the network. This is not possible with techniques like VAR, Gradient Boosted Decision Trees or Random Forests, which need labeled data and thus cannot exploit the knowledge contained in the unlabeled samples.

In order to quantify the contribution of the pre-training phase, and to show that this is useful also when the unlabeled dataset is not huge, we use the two smallest datasets, i.e., the Pollution Dataset and the PKDD'99 Financial Dataset, and we pre-train the models with different portions of the pre-training dataset. Specifically, in both Table 4.12 and Table 4.13 we indicate the fraction of the pre-training dataset used for each experiment, where zero corresponds to training the models from scratch directly on the (labeled) downstream task data. The results in these tables show that both TabBERT and UniTTab significanltly ben-

**Table 4.12:** Pollution prediction task (our partition): impact of different portions of the pre-training dataset. † Our reproduction.

| Pre-training portion | Model | RMSE |
|---|---|---|
| 0 | TabBERT † | 33.37 |
| | UniTTab (ours) | **27.10** |
| | XGBoost † | 48.89 |
| | CatBoost † | 47.09 |
| | VAR † | 84.05 |
| 0.25 | TabBERT † | 30.78 |
| | UniTTab (ours) | **24.55** |
| 0.5 | TabBERT † | 30.11 |
| | UniTTab (ours) | **23.73** |
| 0.75 | TabBERT † | 29.51 |
| | UniTTab (ours) | **23.32** |
| 1 | TabBERT † | 29.13 |
| | UniTTab (ours) | **23.29** |

efit from the pre-training phase, despite both datasets have a small-medium size, and even when only a small portion of the unlabeled data (e.g., 0.25) is used for pre-training. Moreover, Table 4.12 shows that UniTTab can drastically outperform the non Deep Learning based methods (VAR, XGBoost and CatBoost) even with no pre-training. Conversely, in the smallest dataset (PKDD'99), XGBoost beats all the other methods with a large margin when no pre-training is used (Table 4.13). Note that this dataset is very small, with only 478 labeled samples, and training from scratch networks with 100M parameters with these scarce data is very difficult. However, when pre-training is used, UniTTab gets a significantly higher F1 score than XGBoost.

**Table 4.13:** Loan default prediction task: impact of different portions of the pre-training dataset. We report average and standard deviation results obtained with 5 random seeds. For both TabBERT and UniTTab, we keep fixed $t_{max} = 150$. † Our reproduction. ‡ Results reported in the corresponding paper.

| Pre-training portion | Model | F1 score |
|---|---|---|
| 0 | TabBERT † | 0.526 (±0.009) |
| | UniTTab (ours) | 0.548 (±0.015) |
| | Random Forest [115] ‡ | 0.2667 |
| | XGBoost † | 0.608(±0.079) |
| | CatBoost † | 0.527(±0.065) |
| | VAR † | 0.474(±0.007) |
| 0.25 | TabBERT † | 0.586 (±0.023) |
| | UniTTab (ours) | **0.593** (±0.022) |
| 0.5 | TabBERT † | 0.577 (±0.022) |
| | UniTTab (ours) | **0.607** (±0.029) |
| 0.75 | TabBERT † | **0.628** (±0.019) |
| | UniTTab (ours) | 0.620 (±0.027) |
| 1 | TabBERT † | 0.620(±0.024) |
| | UniTTab (ours) | 0.673(±0.038) |

## 4.4 Experimental setting

In this section, we present in more detail the datasets and the associated downstream tasks we used for our evaluations. Generally speaking, the five datasets we used are very different from each other in terms of size, attributes, complexity and downstream tasks. Moreover, we use two main protocols to extract the time series and split the data in training and testing partitions, the first proposed in [83], and the second proposed in this paper. Specifically, for both the Pollution Dataset and the Transaction Dataset we follow the protocol proposed in [83], in which time series are extracted from longer sequences using a temporal sliding window (whose length corresponds to the time series length $t$) and a stride. These time series are then randomly split in training and testing sequences using a fixed training/testing ratio and used for the corresponding downstream task. Note that the *pre-training* data include the downstream task testing samples, however, *no downstream task label is used in pre-training*, since this phase is completely

unsupervised. Even if a clearer separation between the pre-training dataset and the downstream task testing dataset would be desirable, we followed the protocol proposed in [83] to make possible a comparison with the methods that have been evaluated with those benchmarks using the same approach [83, 55, 96]. On the other hand, for the other three datasets adopted in this paper, the PKDD'99 Financial Dataset, the Age2 Dataset and the RBAT Dataset (all bank transaction datasets), we split the time series based on the bank client, which leads to a more realistic scenario and a sharper training-testing separation. Specifically, we create two data partitions (training and testing) based on the bank customer identity, and in each partition we collect the entire transaction history of the corresponding clients. Therefore, the history of a given customer can either belong (entirely) to the training data or (entirely) to the testing data. The training split is used both in the pre-training phase and in the downstream task training phase, while testing data are observed by the models only at the time of evaluating the downstream task. Finally, as explained in Section 4.3.2, for each dataset we define a $t_{max}$ length and we extract time series with variable length $t$, where $t \leq t_{max}$. We provide below more details on each dataset/task, while in Section 4.5 we show the statistics of each dataset.

**The Pollution Dataset** [73] is a public UCI dataset based on pollution data collected from 12 monitoring sites. Every row is composed of $k = 11$ fields ($k = 14$ using our time stamp representation with 4 fields) and it was adopted in [83] to extract time series using the concatenation of $t$ time-dependent, consecutive rows, obtained with a $t$ long sliding window and a stride of 5 (see below). Following [83], in Tables 4.4 and 4.5 we *pre-train* all the models using 76K time series with length $t = 10$.

For evaluation, we adopt the Pollution prediction (supervised) downstream task [83], consisting in predicting the air pollution concentration. For this task, Padhi et al. [83] use 45K (supervised) time series samples for training and 15K (supervised) samples for testing, in which the time series are obtained using a sliding window and a stride of 5. However, using this stride, there is an overlapping between adjacent sequences, and, since the training-testing splits of the downstream task are obtained by random sampling of the time series, the two splits

may have a partial information leak. To avoid this, we use a stride of 10 (with no overlapping), which leads to a downstream task training-testing partition different from the one used in [83], with 23K training and 7.6K testing sequences. As mentioned in Section 4.3, we call "our partition" the training-testing random splits obtained using a stride of 10, and "original partition" the training-testing random splits used in [83] and based on a stride of 5 (see Section 4.5 for more details).

Finally, in Table 4.6 we use time series with length $t = 50$ and a stride of 50, which leads to ∼7.6K unsupervised time series used for pre-training, ∼6.1K used for the supervised downstream task training and ∼1.5K testing sequences.

**The Transaction Dataset** [83] is a synthetic dataset created with heuristic rules to generate realistic credit card transactions (see Figure 1.1). The dataset is composed of 24M transactions from 20,000 users. There are $k = 10$ attributes, which become $k = 13$ when the time stamp is split in 4 different fields (Section 4.2). Following [83], a time series is composed of $t = 10$ transactions (i.e., rows), and, for pre-training, the stride of the sliding window is 5, which leads to a total number of 4.87M pre-training time series.

The associated Fraud detection downstream task is a binary classification task based on the prediction of the fraudulent label, associated with a subset of samples. Following [83], we use 1.9M labeled samples (with $t = 10$) for the supervised training stage. Note that, for the downstream task, Padhi et al. [83] extract time series using a stride of 10, thus, differently from the Pollution prediction downstream task, there is no overlapping between two adjacent time series and no information leak in the splits defined in [83]. However, since the positive and the negative classes are highly unbalanced, the positive samples are upsampled (for more details, we refer to [83]). The testing set is composed of 487K time series.

For computational reasons, in the ablation experiments presented in Section 4.3.1 (Tables 4.1 to 4.3) we use only 400K randomly selected time-series for pre-training, of which 360K are used for the downstream task training and 40K for testing. Finally, in Table 4.8, we use $t = 50$ and a stride of 50, which leads to ∼475K unsupervised time series for pre-training, ∼428K supervised time series

for the downstream task training and ∼47K supervised downstream task testing time series.

**The PKDD'99 Financial Dataset** [18] is a collection of real anonymized financial data of a Czech bank. Besides other client specific information (which we do not use), this dataset contains bank account transactions of 4,500 clients. We use $k = 6$ fields to represent each transaction (row), which become $k = 8$ after splitting the time stamp in 3 different fields. On average, the total length of transaction history for a given customer is 232.

The associated Loan default prediction downstream task is a binary classification task consisting in predicting how likely a client is going to default the loan. For this task, we use 682 clients, jointly with their loan ground truth label, which, following [115], are split in 478 clients for training and 204 for testing, and we report the average results obtained with 5 random splits. Note that the testing clients are not used for pre-training and we removed (from the pre-training, the fine-tuning and the testing data) all those transactions directly related to the loan payment. Moreover, both at fine-tuning and at testing time, we cut the transaction history of each client before the loan started.

**The Age2 Dataset** [49] is a public dataset composed of real bank transactions of different clients. Specifically, there are 43K clients, which we split in 39K for training (used both in the pre-training stage and in the downstream task training phase) and 4K for testing (used only for evaluation). The average length of the transaction history of each client is ∼84 rows. As shown in Section 4.3.2, time series of variable length are extracted from each client with a maximum length of $t_{max} = 50$.

The downstream task requires the model to predict the age of the client given a time series. Specifically, we formulate this task as a binary classification task, where the models should predict whether the customer is over 30 years old.

**The RBAT Dataset** is a proprietary dataset provided by an international bank*, which is composed of several hundred thousands real bank account transactions of private clients. From this datasets, we have randomly selected 100K bank accounts, corresponding to about 32.5M transactions (i.e., rows), which

*For both privacy and commercial reasons, this dataset cannot be released.

we used for both pre-training and fine-tuning on the donwstream task, and another 20K accounts used only for the downstream task testing. Overall, the time series transactions span about 2 years, from 2021 to 2022. For a given account, the average transaction history length is 325. There are $n = 3$ types of transactions (see Section 4.3.2), and, in the future, we plan to extend our experiments with larger portions of this dataset, including other types of transactions ($n > 3$).

The Churn prediction downstream task consists in predicting a bank account closure after a period of one month from the last transaction considered. In more detail, at inference time, given a time series $\boldsymbol{s}$, extracted from a given bank account, the model should predict a possible closure of that account which can happen any day of the month following the last transaction contained in $\boldsymbol{s}$ (which is a supervised information provided at fine-tuning time).

## 4.5 Dataset statistics

In Tables 4.14 and 4.15 we report the main characteristics of the datasets used in our experiments, including our RBAT Dataset. Specifically, both "Pre-training samples" and "Downstream training samples" refer to the original number of time series samples *before* any upsampling or data-augmentation process. The reported number of attributes include the time stamp counted as a single field. For the Pollution Dataset, we also report the statistics of our partition, obtained using a non-overlapping stride which guarantees that there is no information leak between the training and the testing split (Section 4.4). For both the Pollution and Transaction datasets, we also report the statistics corresponding to time series of length $t = 50$, used in Table 4.6 and Table 4.8, respectively. Finally, the last column of Table 4.14 refers to the random subset of the Transaction Dataset used in the ablation experiments of Section 4.3.1.

**Table 4.14:** Dataset statistics: Pollution Dataset and Transaction Dataset.

| | Pollution Dataset | | | Transaction Dataset | | |
|---|---|---|---|---|---|---|
| | our partition | original partition | $t = 50$ | original | $t = 50$ | ablation |
| Total dataset rows | 382,168 | 382,168 | 382,168 | 24,386,900 | 24,386,900 | 4,000,000 |
| Number of attributes ($k$) | 11 | 11 | 11 | 10 | 10 | 10 |
| Number of categorical fields | 1 | 1 | 1 | 8 | 8 | 8 |
| Number of numerical fields | 10 | 10 | 10 | 2 | 2 | 2 |
| Time series length ($t$) | 10 | 10 | 50 | 10 | 50 | 10 |
| Pre-training samples | 76,414 | 76,414 | 7,638 | 4,874,597 | 475,066 | 400,000 |
| Downstream training samples | 22,927 | 45,850 | 6,114 | 1,950,224 | 427,559 | 360,000 |
| Testing samples | 7,642 | 15,282 | 1,524 | 487,556 | 47,507 | 40,000 |
| Data availability | public | public | public | public | public | public |

**Table 4.15:** Dataset statistics: PKDD'99 Financial Dataset, Age2 Dataset and RBAT Dataset.

| | PKDD'99 Financial Dataset | Age2 Dataset | RBAT Dataset |
|---|---|---|---|
| Total dataset rows | 1,042,740 | 3,652,757 | 39,040,010 |
| Number of attributes ($k$) | 6 | 11 | 10 |
| Number of categorical fields | 3 | 1 | 7 |
| Number of numerical fields | 3 | 10 | 3 |
| Time series length ($t$) | variable | variable | variable |
| Pre-training samples | 4,500 | 38,961 | 100,000 |
| Downstream training samples | 478 | 38,961 | 100,000 |
| Testing samples | 204 | 4,328 | 20,000 |
| Data availability | public | public | private |

## 4.6 Implementation details

Training UniTTab—both during large-scale self-supervised pre-training and downstream fine-tuning—requires substantial computational resources. Across all experiments, models were implemented in PyTorch 1.11 and trained on four NVIDIA RTX A6000 GPUs (48GB each). While the core architecture is shared across datasets, the total number of parameters varies depending on the dataset-specific categorical vocabularies (an unavoidable source of parameter growth in tabular Transformers). In all cases, model sizes range from 90M

to over 300M parameters, with sequence lengths up to 150 transactions and hierarchical attention applied at both field and sequence level.

The experiments reported in this chapter rely on extensive self-supervised pre-training, which is by far the most computationally intensive stage. For the largest datasets, pre-training spans several days of continuous multi-GPU computation, even with highly optimized data pipelines. As a consequence:

- Pre-training runs cannot realistically be repeated for the sole purpose of measuring variance or tuning hyperparameters. Each pre-training run represents a multi-day, high-cost computation, and repeating it multiple times would require resources well beyond typical academic availability.

- Ablation experiments cannot be executed with multiple random seeds. While multi-seed ablations are standard in small-scale settings, they would here multiply the cost of pre-training by the number of seeds, making them computationally prohibitive. Instead, ablations isolate architectural or methodological components while keeping all other factors fixed, following common practice in large-model research.

- Downstream fine-tuning is inexpensive, but strongly depends on the pretrained checkpoint. Since fine-tuning depends on the underlying pretrained representation, repeating fine-tuning with multiple seeds would still not provide meaningful insight into the variability of the pre-training process, which dominates model behavior.

Given these constraints, our evaluation follows the same principles adopted in large-scale language and vision models: we report results based on a single pre-trained model per configuration, perform ablations by modifying components while keeping the pretrained backbone fixed, and run multi-seed repetitions only for pre-training and fine-tuning on small datasets, where computational cost is manageable.

A detailed summary of dataset-specific parameters and training time is provided in Table 4.16.

**Table 4.16:** UniTTab hyperparameter values.

| | Pollution Dataset | Transaction Dataset | PKDD'99 Financial Dataset | RBAT Dataset | Age2 Dataset |
|---|---|---|---|---|---|
| Optimizer | AdamW | AdamW | AdamW | AdamW | AdamW |
| Learning rate | 5e-05 | 5e-05 | 5e-05 | 5e-05 | 5e-05 |
| Dropout | 0.1 | 0.1 | 0.1 | 0.1 | 0.1 |
| Label Smoothing ($\varepsilon$) | 0.1 | 0.1 | 0.1 | 0.1 | 0.1 |
| Batch size | 120 | 120 | 120 | 120 | 120 |
| Model size (parameters) | 137M | 300M | 100M | 90M | 44M |
| Field Transformer layers | 1 | 1 | 1 | 1 | 1 |
| Field Transformer heads | 8 | 8 | 8 | 8 | 8 |
| Field Transformer embedding size ($d$) | 72 | 72 | 72 | 72 | 72 |
| Sequence Transformer layers | 12 | 12 | 12 | 12 | 12 |
| Sequence Transformer heads | 12 | 12 | 12 | 12 | 12 |
| Sequence Transformer embedding size ($m$) | 1080 | 1080 | 1080 | 1080 | 1080 |
| Pre-training epochs | 12 | 5 | 20 | 20 | 20 |
| Pre-training iterations | 7.6k | 203k | 0.76k | 17k | 21k |
| Fine-tuning epochs | 10 | 10 | 30 | 30 | 20 |
| Fine-tuning iterations | 5.7k | 485k | 0.21k | 30k | 21k |
| Total training time | 1.5 hours | 4 days | 15 minutes | 8 hours | 2 hours |

# 5

# Deep Learning and Large-Scale Models for Bank Transactions

**Publication:**

"F. Garuti *et al*., Deep Learning and Large Scale Models for Bank Transactions [1]".

The terms '*our*' and '*we*' refer to this thesis's author and the publication's authors.

This chapter presents the second contribution of the thesis, which focuses on the application of deep learning and large-scale models to bank transactional data. Building upon the representation framework introduced in Chapter 4, this work addresses the unique challenges posed by real-world banking transactions, including extreme data heterogeneity, irregular temporal dynamics, and the need for scalability to tens of millions of events. The chapter introduces a Transformer-based foundation model trained on large-scale real transactional datasets, with the goal of learning general-purpose representations that can be effectively trans-

This Chapter is related to the publication "F. Garuti *et al*., Deep Learning and Large Scale Models for Bank Transactions" [1]. See the list of Publications on page 175 for more details.

ferred across multiple downstream tasks such as fraud detection, churn prediction, and credit risk assessment. By leveraging self-supervised learning at unprecedented scale, this study advances the use of deep learning in financial transaction modeling and positions large-scale transactional models as a foundational component for modern banking analytics.

## 5.1 The challenges of transactional data and Deep Learning

Dealing with transactional data is more complex than working with music, images or text because of the heterogeneity of the input. It is also more difficult than working with multimodal data; in addition the public transactional datasets are often too small and limited in diversity.

Indeed, the impactful results of UniTTab (Unified Transformer model for Tabular data) were driven by the availability of large transactional datasets as well as the availability of NVIDIA GPUs, which facilitated the definition of new neural architectural models, specifically designed for banking transactional data.

The main challenges addressed by UniTTab depend on the aspects of these data, briefly outlined in the following:

- Tabular data, usually collected from different sources (e.g., separate databases) thus they require an intensive pre-processing for data cleaning and interoperability.

- Time dependence. Transactional data represent a special case of time-series with non-regular frequency: bank customers carry out a variable number of transactions per year, ranging from very few transactions up to several thousand transactions per year.

- Heterogeneous data. Transactions have not homogeneous fields: some of them are numerical (e.g. the amount), some categorical (e.g. the type of transaction), some textual (e.g. the bank transfer description) or with a specific structure (e.g. the date).

- Multiple-structure data. The transactions of a client account have a different field-structure according with the type of transaction (e.g. a POS, a credit card, an ATM or a bank transfer).

- Correlated data. The transaction fields are often correlated to each other in the same time series (e.g. in periodical payments) and among different time series: each client can own different bank products, different accounts, and some accounts have different owners (join accounts). Finally, some transactions are correlated with external but unknown conditions (e.g. holiday times or the lockdown in the pandemic period).

## 5.2 The architecture

Given the previously discussed challenges of data variability, quantity and heterogeneity, deep learning for transactional data has been largely underexplored, with only a few public experiments and a small number of private institutions and banking research centers. One interesting recent position paper (J.P Morgan 2021 [12]) concerns synthetic data generation, even if it actually defines the problems but does not offer any solutions.

In non-financial AI, state-of-the-art models are usually based on Transformer architectures, usually trained using self-supervised learning (e.g., using "masked word" prediction tasks or through generative or "contrastive learning"). Initially defined as language models in the field of NLP, they are now common also in other AI areas such as Computer Vision, making Transformer networks the basic paradigm for contemporary AI. Specifically, "Foundational Models" are typically large Transformers pre-trained on huge datasets (e.g., the Wikipedia documents, or billions of Web-collected images). Their goal is to create compact, intermediate representations of the input in a latent space, useful for different tasks, such as classification, generation, recognition, image segmentation, anomaly detection, etc. Examples are BERT [35], GPT2/3/3.5 [24], CLIP [90], etc. They can be used in a simple way, see the worldwide success of Chat-GPT3, and "fine-tuned" to be adapted to specific tasks. On the other hand, using Transformers to create

Foundation models for new types of data -such as the transactional data- is more complex, both because training and testing requires days of GPU computation, jointly with very large datasets, and because the Transformer architecture should be re-defined for the specific domain.

In the UniTTAB project we explore this trend by redesigning attentive and generative models, i.e. Transformers. This allowed us to deal with the heterogeneous nature of transactional data: tabular time series with multiple-structure and multimodal fields. The designed architecture is resumed in Figure 5.1.

Borrowing the techniques used in text analysis in BERT or GPT models, we used input time series with variable length. We varied the sequence length from 50 to 150 transactions, where each transaction is composed of a fixed number of 10 fields. As a result, each time series can vary in length from 500 to 1500 items, a challenging length to be managed even for text sentences.

Given the data structure we use the hierarchical architecture shown in Figure 5.1. First, we endow a field-level transformer, which encode individual transactions into embeddings. Then these embeddings are fed into the second-level transformer, that processes the time-series to encode them as a single element in the latent space. This is the foundation latent space where the representation can be potentially exploited for many tasks, such as Classification (e.g., to classify the client behavior), Detection (e.g., to detect anomalies, frauds, etc.), Prediction (e.g., to predict product churn in next few months). As shown in Figure 5.1, this model can also be used for a generative task. For instance generating time series data has the advantage of preserving the content but also the privacy of the client.

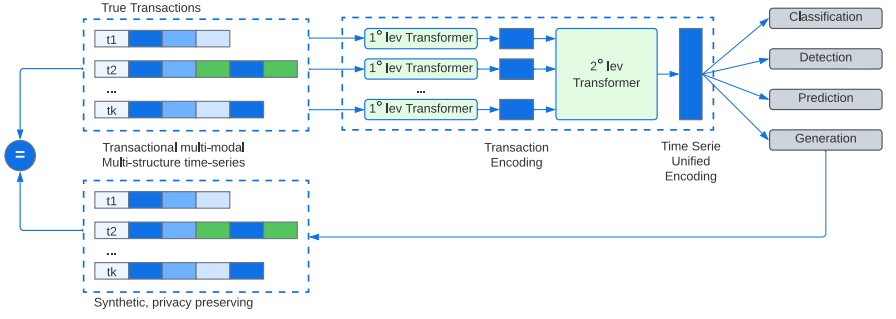



**Figure 5.1:** A schematic illustration of the UniTTAB architecture for financial data.

## 5.3 Experiments on available datasets

Details of the architecture, at least for some tasks of detection and prediction, are described in UniTTaB proposed by Simone Luetto et al. [76]. The effectiveness of the model has been tested over various datasets, used as benchmarks for different tasks, according with the provided manual annotation.

- A Fraud Detection task has been tested on the Transactions Dataset [8] proposed in 2021 with synthetic credit card transactions, composed of multimodal data (some fields are categorical, some are numerical). Trained on about 2 Million samples, tests have been provided on a sets of about 450K sequence of transactions. As reported in Table 5.1 UniTTab strongly outperform any competitor, with an accuracy 93.5 and an F1 measure of 0.915.

- A Loan Default Prediction task is evaluated on the PKDD'99 Financial Dataset [18]: it is a relatively small dataset with "only" 45K clients, each performing 200 transactions in average. Although the dataset is very unbalanced, loan Default is correctly predicted with a F1 measure of 0.673 and an accuracy of 92.3 (Table 5.1). Also in this case results are the state-of-the art.

- A Churn Rate Prediction task is finally evaluated on the RBAT Dataset. A subset of approximately 100K bank accounts with about 50 Million of transactions have been adopted for training the complete architecture of Figure 5.1. As reported in Table 5.1, prediction is very precise – absolutely better than other competitors - with an accuracy of 90.8 and an F1 measure of 0.604.

A brief comparison of the results of our UniTTab model against the competitors on the previously mentioned tasks is provided in Table 5.1.

**Table 5.1:** A quantitative comparison between UniTTAB and the state-of-the-art Deep Learning methods for time series data on three tasks: (1) a Fraud detection task on the (synthetic) Transactions Dataset, (2) a Loan default prediction task on the (real) PKDD'99 Financial Dataset, (3) a Churn prediction task on our (real) RBAT Dataset.

| | Transactions Dataset | PKDD'99 Financial Dataset | | RBAT Dataset | |
|---|---|---|---|---|---|
| Model | F1 score | F1 score | Accuracy | F1 score | Accuracy |
| TabBERT [83] | 0.860 | 0.620 | 91.6 | 0.526 | 86.5 |
| LUNA [55] | 0.862 | - | - | - | - |
| TabAConvBERT [96] | 0.896 | - | - | - | - |
| UniTTab (ours) | **0.915** | **0.673** | **92.3** | **0.604** | **90.8** |

## 5.4 Dataset Statistics

The experiments presented in section 5.3 rely on three datasets previously introduced in chapter 4: the Transactions Dataset, the PKDD'99 Financial Dataset, and the RBAT Dataset. Their structure, feature composition, and overall characteristics have already been described in section 4.4 (Experimental Setup) and section 4.5 (Dataset Statistics). The procedures used to extract time-series samples—such as temporal windowing, variable-length sequence handling, and the construction of training and testing splits—are exactly the same protocols defined in chapter 4, and in particular follow the original extraction and partitioning strategy proposed in [83] for the datasets where that protocol applies.

In this chapter, we exclusively adopt the same original splits defined in chapter 4 and derived from the [83] protocol, without introducing alternative windowing strategies, different sampling configurations, or additional partitioning schemes. This ensures methodological continuity across chapters and allows us to directly compare the results obtained here with those of chapter 4, both in terms of model behaviour and dataset usage.

To support reproducibility, we report at the end of this section a summary table detailing the exact dataset partitions used in the experiments of section 5.3. For each dataset, the table includes:

- the number of time-series samples used for self-supervised pre-training,

- the number of labeled samples used for downstream training and testing,
- the specific time-series length configuration (fixed or variable, including the value of $t_{max}$), and
- the number and type of fields composing each transactional record.

**Table 5.2:** Dataset statistics: Transactions Dataset, PKDD'99 Financial Dataset and RBAT Dataset.

| | Transactions Dataset | PKDD'99 Financial Dataset | RBAT Dataset |
|---|---|---|---|
| Total dataset rows | 24,386,900 | 1,042,740 | 39,040,010 |
| Number of attributes ($k$) | 10 | 6 | 10 |
| Number of categorical fields | 8 | 3 | 7 |
| Number of numerical fields | 2 | 3 | 3 |
| Time series length ($t$) | 10 | variable | variable |
| Pre-training samples | 4,874,597 | 4,500 | 100,000 |
| Downstream training samples | 1,950,224 | 478 | 100,000 |
| Testing samples | 487,556 | 204 | 20,000 |
| Data availability | public | public | private |

## 5.5 Implementation details

The experiments presented in this chapter build directly upon the UniTTab framework and training methodology introduced in Chapter 4. For consistency and comparability, all hyperparameters used in Chapter 5 exactly match those reported in Chapter 4 for the corresponding datasets. No dataset-specific modifications or retuning were introduced for the experiments in this chapter.

All models were implemented in PyTorch and trained on four NVIDIA RTX A6000 GPUs (48GB), using the same computational setup adopted in Chapter 4. This ensures that differences in performance across chapters arise solely from data scale, model configuration, or task definition—not from changes in the underlying training environment.

Because this chapter introduces large-scale experiments on real banking data,

the computational footprint is dominated by self-supervised pre-training. Nevertheless:

- No hyperparameter search was performed.
- All models and tasks inherit exactly the same architectural and optimization settings defined in Chapter 4.
- Any performance differences stem from dataset scale and task definition, not from implementation choices.

To maintain consistency across the thesis, pretrained checkpoints used in Chapter 5 were produced using the same pre-training setup outlined in Chapter 4. Moreover, the fine-tuning procedures follow the same strategy described in Section 4.6, including the use of a [CLS] token and a task-specific linear classifier.

**Table 5.3:** UniTTab hyperparameter values.

| | Transaction Dataset | PKDD'99 Financial Dataset | RBAT Dataset |
|---|---|---|---|
| Optimizer | AdamW | AdamW | AdamW |
| Learning rate | 5e-05 | 5e-05 | 5e-05 |
| Dropout | 0.1 | 0.1 | 0.1 |
| Label Smoothing ($\varepsilon$) | 0.1 | 0.1 | 0.1 |
| Batch size | 120 | 120 | 120 |
| Model size (parameters) | 300M | 100M | 90M |
| Field Transformer layers | 1 | 1 | 1 |
| Field Transformer heads | 8 | 8 | 8 |
| Field Transformer embedding size ($d$) | 72 | 72 | 72 |
| Sequence Transformer layers | 12 | 12 | 12 |
| Sequence Transformer heads | 12 | 12 | 12 |
| Sequence Transformer embedding size ($m$) | 1080 | 1080 | 1080 |
| Pre-training epochs | 5 | 20 | 20 |
| Pre-training iterations | 203k | 0.76k | 17k |
| Fine-tuning epochs | 10 | 30 | 30 |
| Fine-tuning iterations | 485k | 0.21k | 30k |
| Total training time | 4 days | 15 minutes | 8 hours |

# 6

# Large-Scale Transformer Models for Transactional Data

**Publication:**

"F. Garuti *et al*., Large-Scale Transformer models for Transactional Data [2]".

The terms '*our*' and '*we*' refer to this thesis's author and the publication's authors.

This chapter introduces the third contribution of the thesis, which investigates large-scale Transformer models for transactional data. Building on the architectural foundations and representation learning principles developed in the previous chapters, this work focuses on scaling Transformer-based models to massive, real-world transaction datasets and on analyzing their behavior in realistic banking scenarios. The chapter explores how self-supervised pre-training on long, heterogeneous transactional time series enables the learning of robust, transferable representations that effectively capture temporal dependencies, feature interactions, and long-range patterns. Particular emphasis is placed

This Chapter is related to the publication "F. Garuti *et al*., Large-Scale Transformer models for Transactional Data" [2]. See the list of Publications on page 175 for more details.

on scalability, variable-length sequences, and the transition from traditional tree-based pipelines to end-to-end deep learning systems, positioning large-scale Transformers as a viable foundation for predictive, detection, and risk-related tasks in modern financial applications.

## 6.1 Method

### 6.1.1 Pre-training and Fine-tuning

The current great availability of data together with the advancement of AI research have opened the possibility of developing larger models with more general purposes. This is already a reality in almost every application regarding unstructured data, like text, images and video. Many general-purpose models have gained fame in recent years: GPT-3 [24], BERT [35], CLIP [90], DALL-E [92]. All these models have been **pre-trained** using large datasets jointly with a self-supervised approach.

The goal of the pre-training phase is to learn a good representation of the input data. As a result, models trained within this scheme show great generalization capabilities even without further training, like the generation of text of GPT models. These capabilities can further improve after a second training phase, called **fine-tuning** over a specific task, for example, ChatGPT's stunning ability to conversate. Taking inspiration from these models, we use a large dataset of transactions to pre-train a Transformer network using self-supervised learning, and then we use a (smaller) labeled dataset to fine-tune the network for a specific task.

During the pre-training stage, we train our UniTTab model using the **Masked Token** pretext task [35]. Some of the input features are masked to the model, and the model is trained to predict the masked features as a function of the "visible" ones. This method makes it possible to train a model to automatically learn the semantics and to extract relevant information contained in the sequence of transactions. Specifically, for an input sequence, we randomly replace a field value with the special symbol [MASK]. We use a standard replacement probability value of 0.15 [35, 83]. Moreover, with probability 0.1, we also mask all the fields in

a transaction, while, for the fields representing the time stamp, they are always either jointly masked or jointly unmasked. These additional masking strategies, inspired by the block masking of adjacent image patches used in BEiT [14], make the pretext task more challenging for the network.

It is important to remark that, within the pre-training phase, the training of the model solely relies on input features of transactions, eliminating the need for labels. This way we can use the entire transactional dataset, even if it is not fully labeled for the specific downstream task. This is the case of the Czech dataset [18] used for loan default prediction, described in Section 6.2.1.

### 6.1.2 The architecture

We develop a custom model called UniTTab, designed to be suited for sequences of heterogeneous transactional data. Borrowing the techniques used in text analysis in BERT or GPT models, we use input time series with variable length. We vary the sequence length from 10 to 150 transactions, where each transaction is composed of a fixed number of 10 or 6 fields. As a result, each time series can vary in length from 100 to 900 items, a challenging length to manage even for text sentences.

Given the data structure, we propose the hierarchical architecture shown in Figure 6.1. The architecture is composed of two different Transformers, and it is trained end-to-end. The first Transformer ("**Field Transformer**") takes as input the $k$ features describing a single transaction, like transaction amount, merchant information, transaction date and time. The features are transformed in $k$ final embeddings, which are concatenated in a single embedding vector. This embedding is the representation of a single transaction. Then a sequence of these embeddings, each representing a transaction, is fed to the second Transformer ("**Sequence Transformer**"). The Sequence Transformer models the statistic dependencies between different transactions in the sequence and outputs embedding elements in the latent space. This is the general-purpose latent space where the representation can be potentially exploited for many tasks, such as classification (e.g., to classify the client behavior), detection (e.g., to detect anomalies,

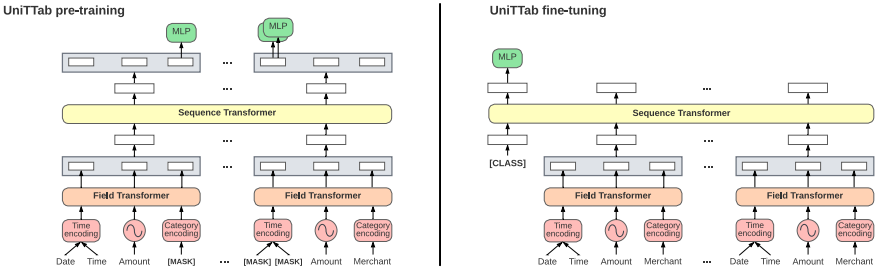



**Figure 6.1:** A schematic illustration of the UniTTab architecture for financial data.

frauds), and prediction (e.g., to predict product churn in next few months).

As depicted in Figure 6.1, during pre-training, for each masked field we use the corresponding output embedding to predict the field value. Instead, during fine-tuning, we add a class token [CLASS] at the beginning of the transaction embedding sequence, and we use the corresponding output embedding to solve the financial tasks. This embedding can attend to all transaction embeddings in the sequence, allowing it to exploit all field information of all transactions.

### 6.1.3 Feature representation

Our model can effectively represent heterogeneous data, encoding both numerical fields (e.g., the amount), categorical fields (e.g., the type of transaction), and fields with a specific structure (e.g., the date).

The most common approach to tackle this challenge is to reduce all the features to a common representation: usually numerical for ensemble of trees or categorical for deep learning architectures like TabBERT [83]. However, discretizing numerical features into a finite set of values results in a loss of information. For example, it could be important to know if an amount is precisely 20 euros or 20.50 to distinguish between a withdrawal and a grocery expense. For this reason we develop a custom representation to transform numerical values in the input vector. In particular, we represent each numerical value as a feature vector obtained by the concatenation of a battery of different frequency functions (depicted with a sine wave symbol in Figure 6.1). Similar representations are used in NeRFs [79] for 3D synthesis. Conversely, we adopt a traditional "category en-

coding" to represent the categorical features, by using simple embedding neural networks (as used in [83]). Finally, we use a custom "time encoding" method for the timestamp attributes. The value of a timestamp is split using a combination of different field values: the year, the month, the day and, if necessary, the hour. Then each such value is represented as a categorical feature (e.g., with 12 elements for the month).

## 6.2 Experimental results

The effectiveness of the model has been tested over two **large size datasets** of transactions: the PKDD'99 Financial Dataset [18] and our Real Bank Account Transaction Dataset (in short, RBAT Dataset). The first dataset is public and is used as a benchmark for predicting loan default. Instead, the second dataset is private and is used to assess how well our model predicts customer churn in comparison to standard industry models. The chosen experiments are binary classification tasks with a large level of unbalance in the statistics of the two target classes.

In all the experiments, the model is first pre-trained using self-supervision (Section 6.1.1) and then fine-tuned on the classification task, in a standard supervised way, using the labeled data.

### 6.2.1 Loan default prediction

The loan default prediction is a classification task defined on the PKDD'99 Financial Dataset, which is a public dataset of real transactions from a Czech bank [18]. This dataset is composed of 1M of transactions from 4500 clients. It also includes customer information, but we use only the transactions, each composed of 6 fields (timestamp, amount, type and channel of the transaction).

The dataset presents a large fraction of unlabeled data, in fact most of the accounts don't have any loans, and they cannot be used for the classification task. This is the perfect example of the potentiality of our model: we perform the pre-training on all the accounts present in the dataset (4500) and then we fine-tune the model only on the labeled ones (478 for training and 204 in test). With such

**Table 6.1:** Loan default prediction task: average and standard deviation results obtained with 5 random seeds.

| $t_{max}$ | Model | F1 score | Average Precision | ROC AUC | Accuracy |
|---|---|---|---|---|---|
| 50 | TabBERT [83] | 0.611(±0.032) | 0.594(±0.031) | 0.827(±0.048) | 90.7(±1.6) |
| | LUNA [55] | 0.604(±0.048) | 0.613(±0.048) | 0.869(±0.030) | 92.5(±1.7) |
| | UniTTab (ours) | 0.619(±0.011) | 0.574(±0.017) | 0.882(±0.021) | 90.2(±1.5) |
| 100 | TabBERT [83] | 0.636(±0.024) | 0.625(±0.036) | 0.874(±0.019) | 91.6(±0.9) |
| | LUNA [55] | 0.624(±0.075) | 0.601(±0.018) | 0.846(±0.025) | 92.5(±1.7) |
| | UniTTab (ours) | 0.654(±0.032) | 0.653(±0.033) | 0.903(±0.006) | 91.4(±1.2) |
| 150 | TabBERT [83] | 0.620(±0.024) | 0.603(±0.016) | 0.857(±0.026) | 91.6(±1.1) |
| | LUNA [55] | 0.637(±0.043) | 0.589(±0.017) | 0.851(±0.030) | 92.6(±1.2) |
| | UniTTab (ours) | 0.673(±0.038) | 0.690(±0.030) | 0.912(±0.018) | 92.3(±1.1) |
| - | Random Forest [115] | 0.2667 | - | 0.6957 | 89.27 |
| | XGBoost | 0.608(±0.079) | 0.700(±0.040) | 0.894(±0.019) | 92.8(±1.8) |
| | CatBoost | 0.527(±0.065) | 0.617(±0.079) | 0.866(±0.043) | 92.0(±1.1) |

a small number of samples UniTTab has been able to obtain good results, way higher than ensemble of trees, and the possibility to exploit all the data in pre-training is its main advantage.

To evaluate our model's ability to deal with longer sequences and variable lengths, we test **different sequence lengths** of transactions. We define a maximum length value $t_{max}$ (ranging from 50 to 150), and for each account we include the entire sequence of transactions if it is shorter than the maximum. Instead, if the sequence exceeds the maximum length, we only consider the most recent $t_{max}$ transactions. It's important to note that, during pre-training the average sequence length is 232 transactions, whereas during fine-tuning the average sequence length is 80 transactions. This happens because, for fine-tuning, we only take transactions made before the loan begins. For this reason, if we set $t_{max}$ to 150, during fine-tuning almost all transaction sequences are of variable length. It's also interesting to observe that, increasing the length of the sequence, the result of the model improves. This is likely due to the information increase in the input sequence, but it demonstrates that the model is able to deal with long sequences of transactions.

### 6.2.2 Effect of Pre-training

One of the main advantages of using Deep Learning methods over traditional Machine Learning approaches is the possibility to pre-train a large network using a large unsupervised dataset, and then fine-tune the same network on the (usually scarcer) available annotated data of a downstream task. In order to quantify the contribution of the pre-training phase, and to show that this is useful also when the unlabeled dataset is not huge, we use the PKDD'99 Financial Dataset, and we pre-train the models with different portions of the pre-training dataset. Specifically, in Figure 6.2 we indicate the fraction of the pre-training dataset used for each experiment, where zero corresponds to training the models from scratch directly on the (labeled) downstream task data. The results in the figure show that both Deep Learning methods (i.e., TabBERT and UniTTab) significantly benefit from the pre-training phase, even using only a small portion of the unlabeled data (e.g., 0.25). Furthermore, when pre-training is performed, our UniTTab gets a significantly higher F1 score than traditional tree-based models.

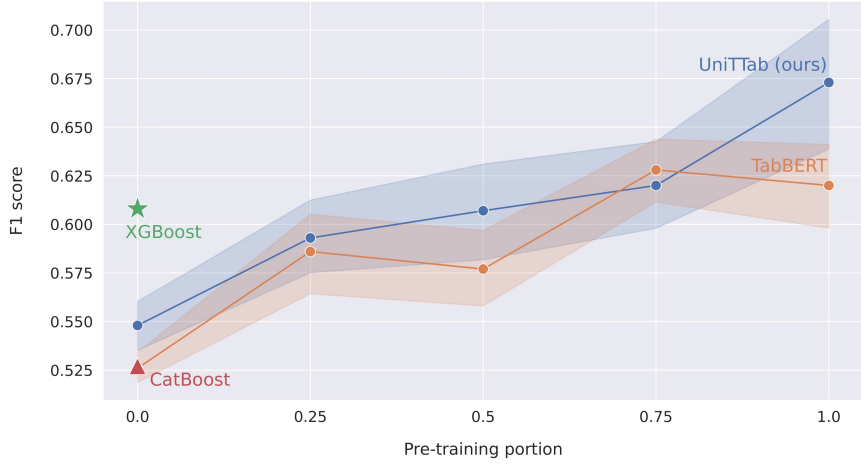



**Figure 6.2:** Loan default prediction task: impact of different portions of the pre-training dataset.

### 6.2.3 Churn prediction: comparison with industry standards

We also compare our model with a custom transactional tree-based pipeline on a churn rate prediction task. The task is defined on the private RBAT dataset, which is provided by an international bank and is composed of several hundred million real transactions of bank customers.

The churn prediction task is defined as whether or not a customer churns in the next 3 months, given a 6-month history sequence of transactions performed by that customer. The history sequences provided to the models are of variable length, with an average of 192 transactions and up to a maximum of 500 transactions. Each sequence is associated with a binary target that represents the presence of the churn event for that customer in the following 3 months. Initially, our model has been pre-trained on a random sample of 1M untargeted accounts, corresponding to approximately 300 million transactions. Then, we evaluate the performance of our model and industry standards using fine-tuning datasets of different sizes, ranging from 50K transaction sequences up to 1 million sequences.

Figure 6.3 shows that our UniTTab model significantly outperforms industry standards for every training dataset size. It also demonstrates the scalability of our model through an increased number of fine-tuning samples: increasing the number of training accounts yields considerably improved AUC on the churn prediction task.

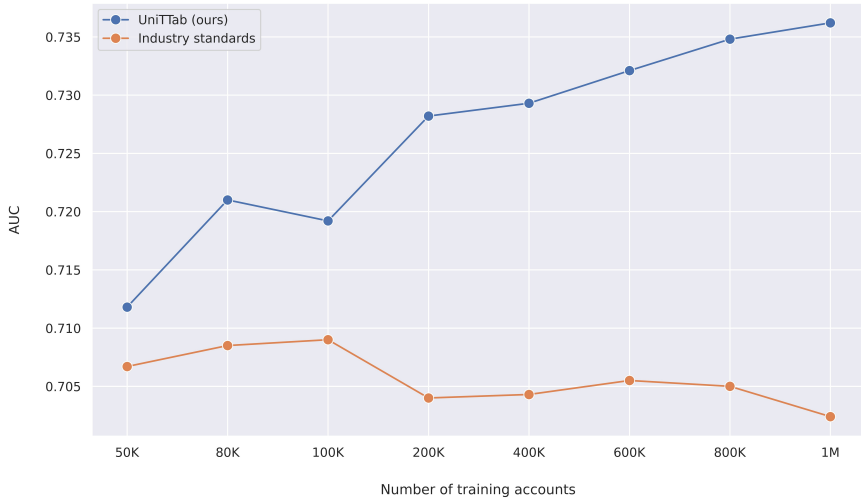



**Figure 6.3:** Customer churn rate prediction task: comparison with industry standard on different portions of the fine-tuning dataset.

## 6.3 Dataset Statistics

The experiments presented in Section 6.2 rely exclusively on two datasets: the PKDD'99 Financial Dataset and the RBAT Dataset. Both datasets were already used in Chapter 5, and the preprocessing and sampling protocols adopted here follow exactly the same procedures described in Section 5.4. In particular, no new extraction strategies, resampling procedures, or modified train/test partitions were introduced for the experiments in this chapter.

For the PKDD'99 Financial Dataset, time-series samples are constructed using variable-length sequences capped by a maximum allowed horizon, and the training and testing partitions correspond precisely to the original splits defined in Chapter 4 and reused in Chapter 5. Similarly, the RBAT Dataset is processed using the same variable-length extraction method and the same customer-level partitioning protocol employed in the Churn Prediction experiments of Chapter 5. In both cases, the model is first pretrained using the original unsupervised split and then fine-tuned using the same labeled downstream-task split previously adopted.

Since the dataset partitions used here are identical to those already documented in Chapter 5, and no additional extraction strategies or dataset transformations were applied, we refer the reader to Table 5.2 for the complete dataset statistics relevant to the experiments reported in this chapter.

## 6.4 Implementation Details

The experiments presented in this chapter assess the behaviour of UniTTab under alternative supervision conditions and modified evaluation scenarios. All experiments were carried out within the same software and hardware environment used in the previous chapters, and no new hyperparameters or architectural modifications were introduced. To ensure strict comparability across the thesis, all hyperparameters used in Chapter 6 are exactly the same as those defined in Chapter 4 and Chapter 5 for the respective datasets. This applies to the model architecture, masking strategy, optimizer settings, learning rate schedule, batching configuration, and preprocessing procedures.

All models were trained using PyTorch on a multi-GPU setup comprising four NVIDIA RTX A6000 GPUs (48GB each). This is the same computational environment used for the experiments in Chapter 4 and Chapter 5. By maintaining an identical hardware and software setup, we isolate the effects of the experimental conditions explored in Chapter 6 from implementation-related variability.

### 6.4.1 Dataset-Specific Configuration

- **PKDD'99 Financial Dataset**. All experiments involving PKDD'99 reuse the complete hyperparameter configuration defined in Chapter 4 and Chapter 5. This includes the Transformer depth, hidden dimensions, embedding sizes, masking probabilities, optimization settings, and data preprocessing steps. The only differences introduced in this chapter concern the experimental protocol (e.g., modified supervision settings), not the model configuration itself.

- **RBAT Dataset**. The experiments conducted on the RBAT dataset also

rely entirely on the hyperparameters previously defined in Chapter 4 and Chapter 5. This includes the hierarchical transaction encoder, feature normalization, field-type embeddings, masking parameters, and optimizer configuration. Sequence handling procedures such as padding, bucketing, and maximum length constraints are unchanged.

# 7

# Generative Modeling of Tabular Time Series

Generative modeling of tabular time series addresses a problem that is fundamentally different from representation learning for discriminative tasks. While Chapters 4 to 6 focused on learning transferable representations optimized for prediction and classification, this chapter deliberately shifts the attention to data synthesis, with distinct objectives, assumptions, and evaluation criteria.

Many practical applications—particularly in privacy-sensitive settings—require the ability to synthesize realistic transactional sequences that resemble real data while not disclosing sensitive information. In such scenarios, the goal is not predictive performance, but the faithful approximation of the underlying data-generating process, enabling data sharing, simulation, and privacy-preserving analytics.

---

This Section is related to the publication "F. Garuti *et al*., Diffusion and Autoregressive Deep Learning Models for Transactional Data Generation" [3]. See the list of Publications on page 175 for more details.

Despite this conceptual shift, the generative models introduced in this chapter build upon the representational and architectural foundations developed in the previous chapters. In particular, the embedding spaces, feature encodings, and hierarchical Transformer designs originally introduced for discriminative learning provide a suitable substrate for modeling the full distribution of heterogeneous transactional time series.

In the context of tabular data, generative modeling aims to approximate the joint distribution of heterogeneous attributes over time. Formally, given a transactional time series $x = (r_1, r_2, \ldots, r_T)$, the objective is to learn a generative model $p_\theta(x)$ capable of sampling new sequences $\widetilde{x}$ that preserve the statistical, structural, and temporal properties of the original data [41]. This includes modeling dependencies both **within individual transactions** and **across transactions in time** [119].

## 7.1 Autoregressive Generation of Tabular Time Series

Among generative paradigms, **autoregressive (AR) models** provide a principled and conceptually simple approach. Autoregressive models factorize the joint probability of a sequence as a product of conditional distributions:

$$p(x) = \prod_{t=1}^{T} p(r_t \mid r_1, \ldots, r_{t-1}),$$

where each transaction is generated conditioned on the entire past history [17, 109]. This formulation naturally aligns with the sequential nature of transactional data and supports variable-length sequence generation.

When applied to tabular time series, autoregressive modeling must address an additional layer of complexity. Each transaction $r_t$is itself a structured object composed of multiple heterogeneous attributes. As a result, the conditional distribution $p(r_t \mid r_{<t})$ must capture both:

1. **Temporal dependencies**, determining when and how a new transaction

occurs given past activity, and

2. **Feature-level dependencies**, ensuring internal consistency among the attributes of the generated transaction citeguo2021towards.

A common strategy is to further factorize the conditional distribution of each transaction into a sequence of attribute-level predictions:

$$p(r_t \mid r_{<t}) = \prod_{k=1}^{K} p(v_{t,k} \mid r_{<t}, v_{t,<k}),$$

where attributes are generated one at a time in a predefined order [82]. This hierarchical autoregressive formulation enables fine-grained control over feature generation and allows the model to enforce consistency across attributes.

Autoregressive models offer several advantages in the tabular domain. They provide **explicit likelihoods**, making training and evaluation straightforward. They naturally support **conditional generation**, allowing sequences to be generated based on contextual information such as entity-level attributes [101]. Moreover, they integrate seamlessly with Transformer architectures, which excel at modeling long-range dependencies [109, 24].

However, autoregressive generation also presents notable limitations. The sequential nature of generation leads to **slow sampling**, especially for long sequences and high-dimensional records [30]. Error propagation across time steps and attributes can degrade generation quality, and unconditional sampling often suffers from limited diversity [61]. These drawbacks highlight the need for generative approaches that can mitigate sequential dependencies while improving sample quality and diversity. The next section introduces diffusion-based generative models as an alternative paradigm for tabular time series generation [100, 58].

## 7.2 Diffusion Models for Tabular Time Series

In recent years, **diffusion models** have emerged as a powerful generative paradigm, achieving state-of-the-art performance in domains such as image synthesis, audio generation, and, more recently, text and video modeling

[100, 58, 104]. Diffusion models differ fundamentally from autoregressive approaches in that they generate samples through an **iterative denoising process**, rather than by sequentially sampling individual tokens or time steps. This distinction has important implications for both generation quality and diversity [36].

At a high level, diffusion models define a forward process that gradually corrupts data by adding noise, and a learned reverse process that progressively removes noise to recover samples from the data distribution [58]. Because generation proceeds through a sequence of refinement steps rather than strict conditional factorization, diffusion models tend to produce samples that better cover the support of the data distribution and exhibit higher diversity [81].

Applying diffusion models to **tabular time series** is particularly appealing for several reasons. First, diffusion-based generation is inherently **parallel across dimensions**, avoiding the token-by-token bottleneck of autoregressive decoding [103]. Second, the denoising objective encourages the model to capture global structure and correlations, rather than relying exclusively on local conditional dependencies [66]. Third, diffusion models have been shown to be robust to mode collapse and to perform well in high-dimensional generative tasks [36].

However, extending diffusion models to heterogeneous tabular time series introduces new challenges. Classical diffusion models are defined over continuous data spaces, whereas tabular datasets combine continuous numerical attributes with discrete categorical variables and exhibit complex structural constraints [114]. Moreover, time-dependent tabular data introduce an additional sequential dimension, requiring the model to generate not only realistic individual records but also coherent temporal dynamics [93].

These challenges necessitate careful adaptations of the diffusion framework, including the definition of suitable latent spaces, noise models, and conditioning mechanisms [66, 59]. In the following sections, we introduce **BankDiT**, a diffusion-based Transformer architecture specifically designed for transactional data generation. BankDiT combines a VAE that compresses individual transactions into a variational latent space with a Diffusion Transformer (DiT) denoising backbone [85]. We also introduce UniTTab an autoregressive Transformer

model, discussing its architecture and differences with respect to BankDiT. We present extensive experiments on a large-scale public dataset, comparing the autoregressive model and the diffusion model with the state-of-the-art [101, 58].

## 7.3 Diffusion and Autoregressive Deep Learning Models for Transactional Data Generation

**Publication:**

"F. Garuti *et al*., Diffusion and Autoregressive Deep Learning Models for Transactional Data Generation [3]".

The terms '*our*' and '*we*' refer to this thesis's author and the publication's authors.

### 7.3.1 Architectures

In this section we present the Deep Learning architectures that have been used to generate transactional data time series.

#### UniTTab Architecture

We modify the hierarchical discriminative architecture proposed in [83] and adopted also in [76] to create an AR generative model.

The complete architecture is shown in Figure 7.1. The architecture is composed of three different Transformers, and it is trained end-to-end. The first Transformer ("Field Transformer") takes as input the $k$ features describing a single transaction $(f_1, ..., f_k)$, like transaction amount, merchant information, transaction date and time. The features are transformed into $k$ final embeddings, which are concatenated in a single embedding vector. This embedding is the representation of a single transaction. Then a sequence of these embeddings $(g_1, ..., g_t)$, each representing a transaction, is fed to the second Transformer ("Sequence Transformer"). The Sequence Transformer models the dependencies between different transactions in the sequence and outputs embedding elements in the latent space. These embeddings contain valuable information that can be used to generate subsequent transactions in the sequence. Finally, we use a

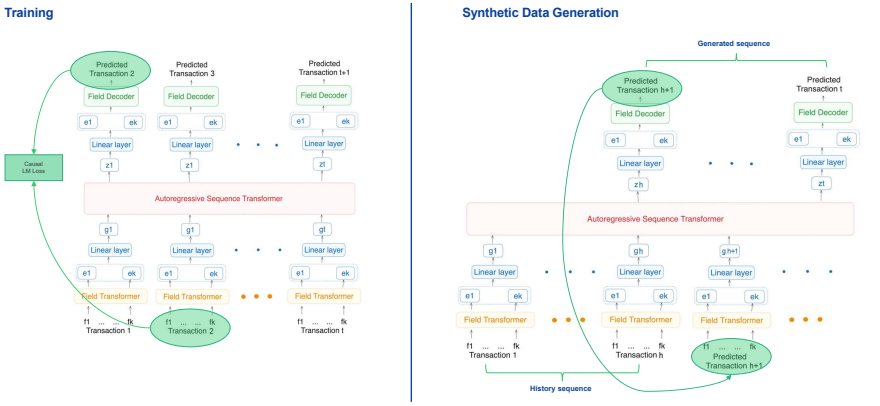



**Figure 7.1:** A schematic illustration of the UniTTab architecture.

third autoregressive transformer ("Field Decoder") to generate all the fields of the next transaction, one at a time. In particular, each of the fields is generated conditioned on the already generated ones, achieving consistency across all the different fields. For example, if the Field Decoder generates a salary category, we condition on that information to ensure that the generated amount of the transaction is coherent with that category.

As depicted in Figure 7.1, during self-supervised training, for each transaction we use the corresponding output embedding to predict the next transaction in the sequence. Instead, during generation, we predict the next transaction and feed it as input, concatenating it with the previous one, to continue the generation process.

### BankDiT Architecture

The UniTTab architecture, presented in Section 7.3.1, generates synthetic transactions as a continuation of initail real transaction time series, fed as input, and it can also generate sequences from scratch. However, the diversity of the sequences generated in this way is lower (as reported in Table 7.1). For this reason, we propose BankDiT, a diffusion-based architecture to unconditionally generate transaction sequences, which does not need a real transaction history sequence as input.

In the fields of image and video generation, many diffusion models have been effectively proposed, such as DALL·E, StableDiffusion and Sora. Given noisy input patches (and conditioning information such as text messages), they are trained to predict the original "clean" patches removing the noise one step at a time. In the same way, given input noisy transaction embeddings (and customer conditioning information), BankDiT is trained to predict the original "clean" transactions.

As shown in Figure 7.2, BankDiT is composed of two different networks: a VAE network and a denoising network.

We use our VAE to separately compress only individual *transactions*, and then we combine multiple, independent latent representations of transactions using the DiT-based DM. In more detail, our VAE encoder $\mathcal{E}_{\boldsymbol{\varphi}}$ represents a transaction $\boldsymbol{r}$ with a latent vector $\boldsymbol{z} = \mathcal{E}_{\boldsymbol{\varphi}}(\boldsymbol{r}), \boldsymbol{z} \in \mathcal{Z}$, which is decoded using the VAE decoder $\mathcal{D}_{\boldsymbol{\phi}}$.

Figure 7.2 shows the proposed framework, which includes $\mathcal{E}_{\boldsymbol{\varphi}}$, $\mathcal{D}_{\boldsymbol{\phi}}$ and our DiT-based denoising network $\mathcal{F}_{\boldsymbol{\theta}}$. At training time, the latter takes as input a sequence of latent row representations of a specific time series. In more detail, given $\boldsymbol{x}_i \in X$, $\boldsymbol{x}_i = [\boldsymbol{r}_1, ..., \boldsymbol{r}_\tau]$, we compute $\boldsymbol{s}_0 = [\boldsymbol{z}_1, ..., \boldsymbol{z}_\tau] = [\mathcal{E}_{\boldsymbol{\varphi}}(\boldsymbol{r}_1), ..., \mathcal{E}_{\boldsymbol{\varphi}}(\boldsymbol{r}_\tau)]$ and we use $\boldsymbol{s}_0$ to train $\mathcal{F}_{\boldsymbol{\theta}}$ via diffusion denoising. The timestep $t$ is first encoded using a small MLP and then its embedding vector is used to condition $\mathcal{F}_{\boldsymbol{\theta}}$. We also condition $\mathcal{F}_{\boldsymbol{\theta}}$ using a set of customer information $\boldsymbol{u} = [\boldsymbol{w}_1, ..., \boldsymbol{w}_b]$. To do so, we encode $\boldsymbol{u}$ using a specific Transformer encoder $\mathcal{C}_{\boldsymbol{\psi}}$. We also refer to the customer information $\boldsymbol{u}$ as the parent table.

At inference time, a latent $s_T$ is sampled and fed to the DiT network, which follows the reverse process sampling chain for $T$ steps until $s_0$ is generated. Finally $s_0$ is split in $\tau$ embeddings $[\boldsymbol{z}_1, ..., \boldsymbol{z}_\tau]$ which are fed separately to the VAE decoder $\mathcal{D}_{\boldsymbol{\phi}}$ which generates the transactions. More details are explained in the paper [50].

In this way, BankDiT is able to generate transaction time series coherent with a specific set of customer information, e.g., age, gender, marital status, account balance, account opening and termination dates.

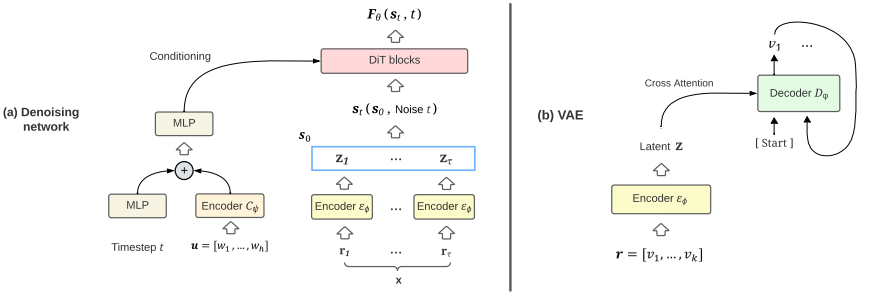



**Figure 7.2:** A schematic illustration of the denoising (a) and the VAE (b) network of BankDiT.

### 7.3.2 Feature representation

Our model is able to effectively represent heterogeneous data, encoding both numerical fields (e.g., the amount), categorical fields (e.g., the type of transaction), and fields with a specific structure (e.g., the date).

Transformers are particularly suited to work with categorical features, but discretizing numerical features in a discrete set of values results in a loss of information. For example, it could be important to know if an amount is precisely 20 euros or 20.50 to distinguish between a withdrawal and a grocery expense. For this reason, we developed a custom representation to transform numerical values in the input vector. In particular, we propose a variable-range decimal representation of the numerical field values, based on a sequence of digits preceded by a magnitude order prefix. This representation is a trade-off between a lossless coding and the length of the sequence.

Conversely, we adopt a traditional "category encoding" to represent the categorical features, by using simple embedding neural networks (as used in [83] and [101]).

Finally, we use a custom "time encoding" method for the timestamp attributes. The value of a timestamp is split using a combination of different field values: the year, the month, the day and, if necessary, the hour. Then each such value is represented as a categorical feature (e.g., with 12 elements for the month, etc.).

### 7.3.3 Validation framework

The effectiveness of the models has been tested over a large-size datasets of transactions, the PKDD'99 Financial Dataset [18].
The PKDD'99 Financial Dataset is a public dataset of real transactions from a Czech bank. This dataset is composed of 1M of transactions from 4500 clients. Each transaction is composed of 6 fields, including timestamp, amount, and information on the type and channel of the transaction. The dataset also includes information about the bank clients, and we use that information to condition the generation. Specifically, we use the location of the client (district name, region), the frequency of issuance of statements, the age and the gender.
Due to the lack of a common evaluation protocol for transactional time series generation, we propose a unified framework for evaluating the conditional generation tasks using different metrics.

### 7.3.4 Metrics

We use a collection of different metrics, in which we include common metrics adopted in single-row tabular data generation [120], as well as metrics used in previous work on time series [101], possibly adapted for our scenarios.
Specifically, first of all, we use a set of **discriminative metrics**, where a discriminator is trained to distinguish between real and generated data. As a discriminator model, we propose to use CatBoost [87], jointly with a standard library [31] to extract features from a time series. The discriminator takes an entire time series as input, which below is underlined by the suffix "TS". The lower the accuracy, the better the generator (indicated with $\downarrow$), because it means that the discriminator struggles in separating the two distributions.
Moreover, we use the *Logistic Detection* adopted in [101], which is based on the ROC-AUC scores of a Random Forest discriminator (the higher the better $\uparrow$). Since this Random Forest takes as input a single tabular row, we use the suffix "SR" (LD-SR).
We also adopt the **high-order metrics** used in [120]. Specifically, *$\beta$-Recall-SR* evaluates how much the generated data covers the entire distribution of the real

data, while *α-Precision-SR* measures how much the synthetic data is close to the real one [120]. They are both based on single row (SR) distribution only comparisons. Note that the "TS" metrics lead to a much stricter criterion for judging the quality of the generator because "SR" metrics cannot evaluate the time dependency among different rows of the same time series.
Finally, we evaluate qualitatively the monovariate distribution of each transaction field (e.g., date, time, amount, categories) and the consistency of the recurrent transactions, like incomes, grocery expenses, investments, travel.

### 7.3.5 Comparison with the state of the art

We compare our BankDiT with state-of-the-art deep learning models for tabular data time series generation, based on GPT-like autoregressive architectures. We also compare our BankDiT with SDV, a traditional non-deep-learning method based on Gaussian Copulas to model inter-field dependencies in tabular data.
We evaluate the models by generating transaction sequences conditioned on the ground-truth parent table ("child-GT") and conditioned on the generated parent table ("child"). In the latter case, we also evaluate the joint probability ("merged"). We perform three generations, and we report in Table 7.1 the mean and the standard deviation of the evaluated metrics.
The results in Table 7.1 show that across all tasks and metrics, BankDiT outperforms all the baselines by a large margin, and approaches the lower bound of 50% CatBoost-TS accuracy.

**Table 7.1:** Conditional generation using the *PKDD'99 Financial Dataset*. Sequences include up to 50 transactions.

| Method | Task | CatBoost-TS ↓ | LD-SR ↑ | $\beta$-Recall-SR ↑ | $\alpha$-Precision-SR ↑ |
|---|---|---|---|---|---|
| SDV | child | 97.95 ±1.42 | 6.53 ±0.58 | 12.8 ±0.42 | 49.9 ±0.37 |
| | merged | 98.12 ±1.17 | 8.77 ±0.59 | - | - |
| REaLTabFormer | child gt-cond | 97.87 ±0.59 | 21.97 ±0.55 | 22.30 ±1.25 | 91.83 ±2.58 |
| | child | 59.33 ±3.82 | 21.50 ±0.72 | 22.57 ±0.85 | 91.50 ±2.33 |
| | merged | 58.77 ±3.05 | 26.00 ±1.61 | - | - |
| UniTTab (ours) | child gt-cond | 68.33 ±4.37 | 81.13 ±1.51 | 57.87 ±0.55 | 98.27 ±0.77 |
| | child | 79.07 ±6.23 | 67.60 ±4.36 | 54.10 ±0.70 | 94.90 ±3.91 |
| | merged | 83.33 ±5.75 | 38.73 ±4.22 | - | - |
| BankDiT (ours) | child gt-cond | **59.50** ±10.53 | **81.20** ±2.71 | **59.03** ±3.72 | **98.63** ±1.33 |
| | child | **51.80** ±6.44 | **79.13** ±3.04 | **56.93** ±1.10 | **97.83** ±0.40 |
| | merged | **54.03** ±3.95 | **53.20** ±0.66 | - | - |

### 7.3.6 Model scalability

We use the PKDD'99 Financial Dataset and we test the models' scalability by varying the sequence length of the generated time series. The sequences have been generated conditioned on the ground-truth customer information. We evaluate the CatBoost-TS metric using sequences of 4 months of transactions, 2 years and 4 years of transactions.

Table 7.2 shows that our BankDiT can accurately generate a few months of transactions, conditioned by exogenous variables. On the other hand, the autoregressive model achieves the highest performance in forecasting and continuation of real transaction time series. In fact, it is possible to observe that by increasing the sequence, the accuracy of our UniTTab model remains almost unchanged. For this reason, we also evaluate our UniTTab in the task of generating synthetic transactions as a continuation of a time series generated by the diffusion model ("BankDiT + UniTTab" in Table 7.2). In this case, we conditioned the generation on the 4 months generated by our BankDiT.

Table 7.2 shows that "BankDiT + UniTTab" outperforms the UniTTab baseline by a large margin.

**Table 7.2:** Model scalability using the *PKDD'99 Financial Dataset*. Sequences include up to 600 transactions.

| Method | Sequence length | CatBoost-TS ↓ |
|---|---|---|
| BankDiT (ours) | 4 months | **50.70** |
| UniTTab (ours) | 4 months | 58.33 |
| UniTTab (ours) | 24 months | 62.50 |
| BankDiT + UniTTab (ours) | 24 months | **53.84** |
| UniTTab (ours) | 48 months | 64.72 |
| BankDiT + UniTTab (ours) | 48 months | **56.46** |

### 7.3.7 Qualitative results

In this section, we show qualitative results using some numerical and categorical fields of the PKDD'99 Financial Dataset.

In Fig. 8.3, 8.4, 8.6 we compare the distributions of the generated transactions and the real ones. In the numerical plots, the x-axis is based on a logarithmic scale and a fixed number of 50 bins for both the real and the generated numerical values. For each bin, in the y-axis, we show the percentage of tabular rows that contain numerical values that belong to that bin.

The results show that the generated distributions are very similar to the real distributions for both numerical and categorical variables. This demonstrates our models' ability to generate heterogeneous fields with high variability in data values.

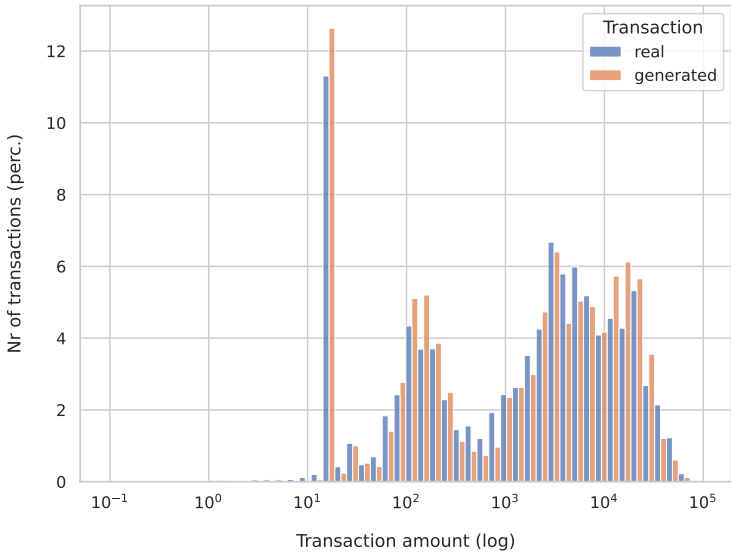



**Figure 7.3:** Real and generated distributions of the Transaction amount numerical field.

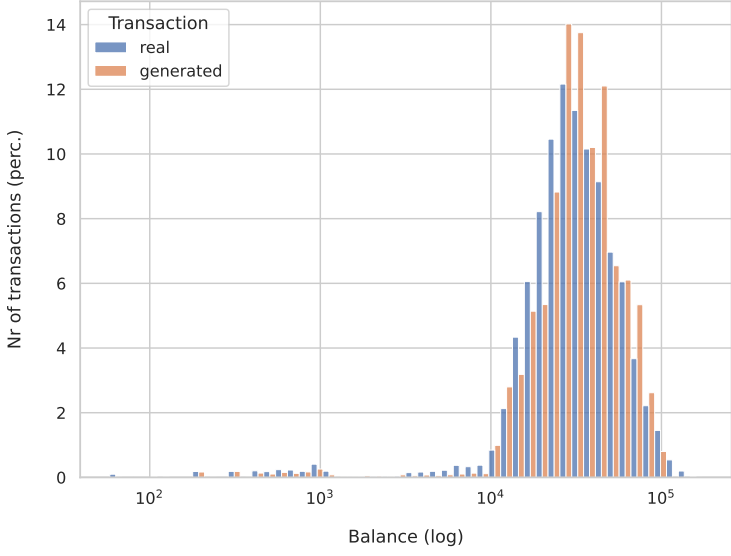



**Figure 7.4:** Real and generated distributions of the Balance numerical fields.

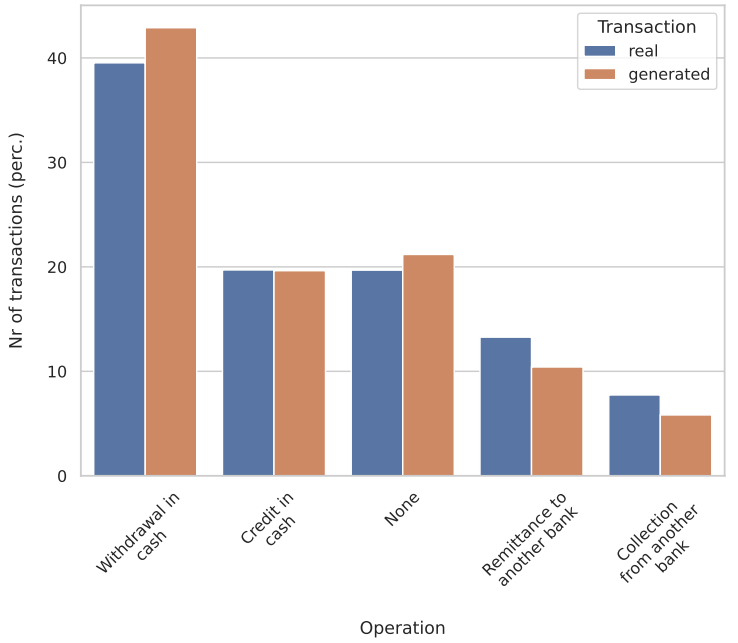



**Figure 7.5:** Real and generated distributions of the Operation categorical fields.

### 7.3.8 Dataset statistics

The experiments on conditional generative modeling presented in this section rely exclusively on the *PKDD'99 Financial* Dataset*, a public real-world banking dataset containing anonymized financial activities of Czech bank customers. The dataset provides two complementary components:

- A parent table, describing static client- or account-level attributes.
- A time-series table, containing the complete chronological sequence of transactions associated with each account.

For the parent table, we limit the conditioning features to a selected subset of fields that provide meaningful contextual information. Specifically, the following attributes are used as conditioning variables in our experiments: district_id, frequency, city, region, account creation date. All other parent-table variables remain unused during conditioning but are still part of the dataset's original structure.

The main dataset statistics relevant for the generative modeling experiments are summarized below.

- Total number of transactional rows: 1,042,740
- Number of attributes per transaction (after timestamp decomposition): 6
- Number of categorical fields: 3
- Number of numerical fields: 3
- Time-series length: variable across clients
- Number of time-series samples (i.e., number of clients): 4,500

*PKDD'99 Financial dataset [18]

### 7.3.9 Implementation Details

The experiments conducted in this chapter are implemented within the same modelling and training framework described in earlier chapters. All methods, architectural components, and optimization procedures follow the design principles introduced in Chapter 4. Moreover, the hyperparameters used for the PKDD'99 dataset in this chapter correspond exactly to those reported in Section 4.6, except for the rows referring to downstream fine-tuning, which are not relevant here. For convenience, Table 7.3 summarises these hyperparameters and reproduces the values from the PKDD'99 column of Table 4.16.

All models were trained using PyTorch on the same multi-GPU environment employed throughout the thesis, consisting of four NVIDIA RTX A6000 GPUs (48 GB each). This ensures that performance differences stem solely from the experimental settings explored in this chapter —not from changes in computational resources or implementation infrastructure.

The PKDD'99 dataset is sufficiently small to make all experimental components computationally lightweight, including self-supervised pre-training, sequence generation using the autoregressive decoder, and downstream evaluation procedures. As a result, every stage of the experimental pipeline—pre-training, generation, and evaluation—was repeated across multiple random seeds. This allows us to quantify variability and report stable, seed-robust results. The aggregated results reported in this chapter thus reflect averaged performance across these repetitions.

**Table 7.3:** UniTTab and BankDiT hyperparameter values.

| | PKDD'99 Financial Dataset |
|---|---|
| Optimizer | AdamW |
| Learning rate | 5e-05 |
| Dropout | 0.1 |
| Batch size | 120 |
| UniTTab model size (parameters) | 100M |
| Field Transformer layers | 1 |
| Field Transformer heads | 8 |
| Field Transformer embedding size | 72 |
| Sequence Transformer layers | 12 |
| Sequence Transformer heads | 12 |
| Sequence Transformer embedding size | 1,080 |
| BankDiT model size (parameters) | 140M |
| DiT depth | 12 |
| DiT num heads | 12 |
| Hidden size (d) | 792 |
| VAE Encoder Transformer layers | 3 |
| VAE Encoder Transformer heads | 8 |
| VAE Encoder hidden size | 72 |
| VAE Decoder Transformer layers | 3 |
| VAE Decoder Transformer heads | 8 |
| VAE Decoder hidden size | 72 |
| Pre-training epochs | 2,000 |
| Pre-training iterations | 58,000 |
| Total training time | 5 hours |

# 8

# Diffusion Transformers for Tabular Data Time Series Generation

**Publication:**

"F. Garuti *et al*., Diffusion Transformers for Tabular Data Time Series Generation [4]".

The terms '*our*' and '*we*' refer to this thesis's author and the publication's authors.

This chapter (1) introduces the fourth contribution of the thesis, which explores generative modeling for tabular time series through Diffusion Transformers. Moving beyond representation learning and predictive modeling, this work addresses the problem of synthesizing realistic, heterogeneous, and variable-length tabular time series, a task that is critical for data augmentation, privacy preservation, and simulation in data-scarce domains. The chapter presents a diffusion-based generative framework that combines the expressive power of Transformers with the diversity and stability of latent diffusion

(1) This Chapter is related to the publication "F. Garuti *et al*., Diffusion Transformers for Tabular Data Time Series Generation" [4]. See the list of Publications on page 175 for more details.

models, explicitly designed to handle mixed numerical and categorical features as well as complex temporal dependencies. By reformulating tabular time series generation as a denoising process in a structured latent space, this contribution advances the state of the art in generative modeling for structured temporal data and opens new directions for scalable and controllable synthetic data generation.

## 8.1 Preliminaries

**Diffusion Transformer.** The LDM paradigm [95] is based on two separated training stages: the first using a VAE and the second using a Gaussian DM. The main goal of the VAE is to compress the initial input space. Specifically, in DiT [85], an image $x$ is compressed into a smaller spatial representation using the VAE encoder $z = E(x)$. The VAE latent representation $z$ is then "patchified" into a sequence $\boldsymbol{s} = [\boldsymbol{z}_1, ..., \boldsymbol{z}_k]$. Note that grouping "pixels" of $z$ into patches is a procedure *external* to the VAE, which holistically compresses the entire image. Then, random noise is added to $\boldsymbol{s}$ and its corrupted version is fed to a Transformer-based denoising network (DiT) which is trained to reverse the diffusion process. More specifically, given a sample $\boldsymbol{s}_0$ extracted from the real data distribution (defined on the VAE latent space, $\boldsymbol{s}_0 \sim q(\boldsymbol{s}_0)$), and a prefixed noise schedule $(\bar{\alpha}_1, ..., \bar{\alpha}_T)$, the DM iteratively adds Gaussian noise for $T$ diffusion steps: $q(\boldsymbol{s}_t|\boldsymbol{s}_0) = \mathcal{N}(\boldsymbol{s}_t; \sqrt{\bar{\alpha}_t}\boldsymbol{s}_0, (1-\bar{\alpha}_t)\boldsymbol{I})$, $t = 1, ..., T$. Using the reparametrization trick, $\boldsymbol{s}_t$ can be obtained by: $\boldsymbol{s}_t = \sqrt{\bar{\alpha}_t}\boldsymbol{s}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\varepsilon}_t$, where $\boldsymbol{\varepsilon}_t \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$ is a noise vector. A denoising network is trained to invert this process by learning the reverse process: $p_{\boldsymbol{\theta}}(\boldsymbol{s}_{t-1}|\boldsymbol{s}_t) = \mathcal{N}(\boldsymbol{s}_{t-1}; \mu_{\boldsymbol{\theta}}(\boldsymbol{s}_t, t), \sigma_{\boldsymbol{\theta}}(\boldsymbol{s}_t, t)\boldsymbol{I})$. Training is based on minimizing the variational lower bound (VLB) [67], which can be simplified to the following loss function [58]:

$$L(\boldsymbol{\theta})^{Simple} = \mathbb{E}_{\boldsymbol{s}_0 \sim q(\boldsymbol{s}_0), \boldsymbol{\varepsilon} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}), t \sim \mathbb{U}(\{1,...,T\})} \left[ ||\boldsymbol{\varepsilon}_t - \varepsilon_{\boldsymbol{\theta}}(\boldsymbol{s}_t, t)||_2^2 \right], \tag{8.1}$$

where $\mu_{\boldsymbol{\theta}}(\cdot)$ is reparametrized into a noise prediction network $\varepsilon_{\boldsymbol{\theta}}(\cdot)$. Following [81], DiT predicts also the noise variance ($\sigma_{\boldsymbol{\theta}}(\cdot)$), which is used to compute the KL-divergence of the VLB in closed form. However, for simplicity, we do not

predict the noise variance.

At inference time, a latent $z_T$ is sampled and fed to the DiT network, which follows the reverse-process sampling chain for $T$ steps until $z_0$ is generated. Finally, the VAE decoder $D(\cdot)$ decodes $z_0$ into a synthetic image. For *condional* generation, DiT encodes the conditional information (e.g., the desired class label of the image) using a small MLP. The latter regresses, for each DiT block, the parameters of the adaptive layer norm ($\boldsymbol{\beta}, \boldsymbol{\gamma}$) and the scaling parameters ($\boldsymbol{\alpha}$) used in the residual connections. In [85], this is called *adaLN-Zero block* conditioning and it is used also to represent the timestep $t$.

**Problem formulation.** Tabular data are characterized by a set of *field names* (or *attributes*) $A = \{a_1, ..., a_k\}$, where each $a_j \in A$ is either a categorical or a numerical attribute. A tabular row $\boldsymbol{r} = [v_1, ..., v_k]$ is a sequence of *k field values*, one per attribute. If $a_j$ is a numerical attribute, then $v_j \in \mathbb{R}$, otherwise $v_j \in V_j$, where $V_j$ is an attribute-specific vocabulary. A time series is a variable-length, time-dependent sequence of rows $\boldsymbol{x} = [\boldsymbol{r}_1, ..., \boldsymbol{r}_\tau]$. Given a training set $X = \{\boldsymbol{x}_1, ..., \boldsymbol{x}_N\}$ empirically representing the real data distribution $q(\boldsymbol{x})$, in *unconditional* tabular data time series generation the goal is to train a generator which can synthesize time series following $q(\boldsymbol{x})$. Moreover, inspired by [101], for the *conditional* generation case, we assume to have a "parent" table $P$ associated with the elements in $X$. A tabular row $\boldsymbol{u} = [w_1, ..., w_b]$ in $P$ is associated with a sequence $\boldsymbol{x}_i \in X$ and it describes some characteristics that affect the nature of $\boldsymbol{x}_i$. For instance, if $\boldsymbol{x}_i \in X$ is the history of the transactions of the i-th bank client, the corresponding row in $P$ could describe the attributes (e.g., the age, the gender, etc.) of this client. In a conditional generation task, given $\boldsymbol{u}$, the goal is to generate a synthetic time series according to $q(\boldsymbol{x}|\boldsymbol{u})$.

## 8.2 Method

In this section we present our approach, starting from an unconditional generation scenario, which we will later extend to the conditional case. Our first goal is to simplify the VAE latent space: since the time series have a variable length and a complex dynamics, rather than representing their distribution using a variational

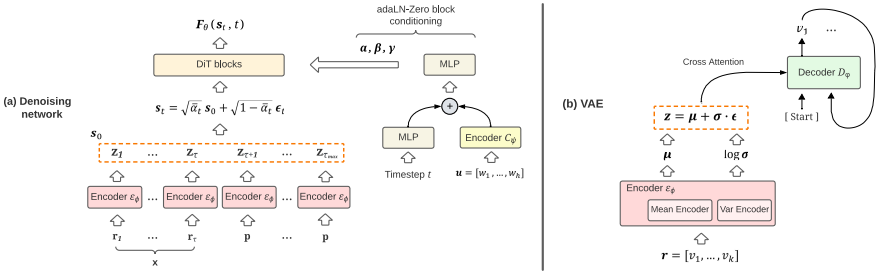



**Figure 8.1:** A schematic illustration of the denoising (a) and the VAE (b) network of TabDiT.

approach [67], we use our VAE to separately compress only individual *tabular rows*, and then we combine multiple, independent latent representations of rows using the DiT-based DM. In more detail, our VAE encoder $\mathcal{E}_{\boldsymbol{\varphi}}$ represents a tabular row $\boldsymbol{r}$ with a latent vector $\boldsymbol{z} = \mathcal{E}_{\boldsymbol{\varphi}}(\boldsymbol{r}), \boldsymbol{z} \in \mathcal{Z}$, which is decoded using the VAE decoder $\mathcal{D}_{\boldsymbol{\phi}}$. The parameters $\boldsymbol{\Phi} = [\boldsymbol{\varphi}, \boldsymbol{\phi}]$ of $\mathcal{E}_{\boldsymbol{\varphi}}$ and $\mathcal{D}_{\boldsymbol{\phi}}$ are trained with a convex combination of a reconstruction loss and a KL-divergence [67]. The training set is $R = \{\boldsymbol{r}_1, ..., \boldsymbol{r}_M\}$, where each $\boldsymbol{r}_j \in R$ is a row extracted from one of the time series $\boldsymbol{x}_i \in X$. Note that $M >> N$ and that the rows in $R$ are supposed to be i.i.d., i.e., we treat the samples in $R$ as independent from each other. Hence, the semantic space of our VAE ($\mathcal{Z} = \mathbb{R}^d$) will not embed any time-dependency among the rows of the same time series, since $\mathcal{E}_{\boldsymbol{\varphi}}$ and $\mathcal{D}_{\boldsymbol{\phi}}$ cannot observe this relation.

Figure 8.1 shows the proposed framework, which includes $\mathcal{E}_{\boldsymbol{\varphi}}$, $\mathcal{D}_{\boldsymbol{\phi}}$ and our DiT-based denoising network $\mathcal{F}_{\boldsymbol{\theta}}$. At training time, the latter takes as input a sequence of latent row representations *of a specific time series*. In more detail, given $\boldsymbol{x}_i \in X, \boldsymbol{x}_i = [\boldsymbol{r}_1, ..., \boldsymbol{r}_\tau]$, we compute $\boldsymbol{s}_0 = [\boldsymbol{z}_1, ..., \boldsymbol{z}_\tau] = [\mathcal{E}_{\boldsymbol{\varphi}}(\boldsymbol{r}_1), ..., \mathcal{E}_{\boldsymbol{\varphi}}(\boldsymbol{r}_\tau)]$ and we use $\boldsymbol{s}_0$ as explained in Section 8.1 to train $\mathcal{F}_{\boldsymbol{\theta}}$, where $\mathcal{F}_{\boldsymbol{\theta}}(\boldsymbol{s}_t, t)$ implements $\varepsilon_{\boldsymbol{\theta}}(\boldsymbol{s}_t, t)$. The timestep $t$ is first encoded using a small MLP and then its embedding vector is used to condition $\mathcal{F}_{\boldsymbol{\theta}}$ (see later). Note that $\boldsymbol{s}_0$ is a *time-dependent* sequence of row embeddings, thus the real data distribution $q(\boldsymbol{s}_0)$ we use to train $\mathcal{F}_{\boldsymbol{\theta}}$ *does* include the statistical dependencies among tabular rows of the same time series. In other words, while $\mathcal{Z}$ represents only individual rows, we use $\mathcal{F}_{\boldsymbol{\theta}}$ to combine independent vectors lying in this space into a sequence of time-dependent final-embedding vectors representing an entire time series.

**Conditional generation.** In a conditional generation task, we want to condition $\mathcal{F}_{\theta}$ using a row $\boldsymbol{u}$ of a parent table $P$ (Section 8.1). To do so, we first encode $\boldsymbol{u}$ using a specific encoder $\mathcal{C}_{\psi}$. The architecture and the field value representations of $\mathcal{C}_{\psi}$ are the same as $\mathcal{E}_{\varphi}$ (Section 8.2.1). However, $\mathcal{C}_{\psi}$ is disjoint from the VAE and it is trained jointly with $\mathcal{F}_{\theta}$. The output vector $\boldsymbol{c} = \mathcal{C}_{\psi}(\boldsymbol{u})$ is summed with the embedding vector of the timestep $t$ and fed to the MLP of the adaLN-Zero block conditioning mechanism (Section 8.1). Moreover, following most of the image generation DM literature, in the conditional generation scenario we also use Classifier-Free Guidance (CFG). Specifically, if the denoising network output is interpreted as the score function [104], then the DM conditional sampling procedure can be formulated as [85]:

$$\hat{\mathcal{F}}_{\theta}(\boldsymbol{s}_t, t, \boldsymbol{u}) = \mathcal{F}_{\theta}(\boldsymbol{s}_t, t, \emptyset) + s \cdot (\mathcal{F}_{\theta}(\boldsymbol{s}_t, t, \boldsymbol{u}) - \mathcal{F}_{\theta}(\boldsymbol{s}_t, t, \emptyset)), \tag{8.2}$$

where $\mathcal{F}_{\theta}(\boldsymbol{s}_t, t, \emptyset)$ is the unconditional prediction of the network and $s > 1$ indicates the scale of the guidance ($s = 1$ corresponds to no CFG). At training time, for each sample $\boldsymbol{x}_i$, with probability $p_d \in [0, 1]$ the condition $\boldsymbol{u}$ is dropped and the network is trained unconditionally.

### 8.2.1 Encoding and decoding in the VAE latent space

**Field value representations.** For categorical attributes $a_j$, we use a widely adopted tokenization approach [120, 76, 83], in which each possible value $v_j \in V_j$ is associated with a token and a lookup table of token embeddings transforms these tokens into the initial embedding vectors of $\mathcal{E}_{\varphi}$. However, how to represent numerical values ($v_j \in \mathbb{R}$) in a way which is coherent with categorical feature tokens is still an open problem and each method adopts a specific solution. For instance, REaLTabFormer [101] converts $v_j$ into a sequence of digits and then treats each digit as a categorical feature. If the maximum possible value in $X$ for the attribute $a_j$ is $v_{max_j}$, and assuming, for simplicity, that $a_j$ can only take on positive integer values, then $v_j$ is converted into a sequence $L = [D_1, ..., D_p]$, where each $D_k \in \{\text{'0'}, ..., \text{'9'}\}$ corresponds to a digit in the decimal representation of $v_j$ and $p$ is a fixed sequence length corresponding to

the number of digits necessary to represent $v_{max_j}$. Importantly, if the decimal representation of $v_j$ requires less than $p$ digits, the sequence is left-padded with zeros [101]. This representation is lossless, but it leads to very long sequences which are frequently full of zeros. Indeed, the value distribution for $a_j$ is usually a Gaussian with very long tails. For instance, the "amount" field of a bank transaction can range from tens of millions to a few cents, but most values are smaller than 1,000 (e.g., 35$).

To solve this problem, we propose a *variable-range* representation using a small, fixed number of digits preceded by a magnitude order. For simplicity, let us assume that $v_j$ is a positive integer, thus:

$$v_j = \sum_{k=0}^{m} b_k 10^k, \quad DR(v_j) = [b_m b_{m-1} ... b_0], \tag{8.3}$$

where $m$ is the largest exponent and $DR(v_j)$ is the representation of $v_j$ by means of a sequence of digits (e.g., [35967]). Then, we represent $v_j$ using a sequence $Q$ defined as follows:

$$Q = [O, D_m, D_{m-1}, ..., D_{m-n+1}]. \tag{8.4}$$

In Equation (8.4), $O$ is the *magnitude order prefix* and it corresponds to $m$ in Equation (8.3). Specifically, in the tabular data domain we can assume that $v_j < 10^{10}$, hence, $m \in \{0, ..., 9\}$ and $O \in \{$'0', ..., '9'$\}$ is the token corresponding to $m$. The value of $O$ is the first one that will be generated by $\mathcal{D}_\phi$ when decoding the sequence representing a numerical value, which corresponds to predict its magnitude order ($m$). We then encode the $n$ most significant digits of $v_j$ using $D_m, D_{m-1}, ..., D_{m-n+1}$, where $D_k \in \{$'0', ..., '9'$\}$ ($m \leq k \leq m - n + 1$) is the token corresponding to the digit $b_k$, and $[b_m b_{m-1} ... b_{m-n+1}] \subseteq DR(v_j)$ is the ($m$-depending) range we represent. For instance, if $v_j = 35967$ and $n = 4$, then $Q =$ ['4', '3', '5', '9', '6']. Once decoded, $Q$ can be used to compute the (possibly truncated) value of $v_j$. For instance, using $Q =$ ['4', '3', '5', '9', '6'], we get: $\hat{v}_j = 3 * 10^4 + 5 * 10^3 + 9 * 10^2 + 6 * 10 = 35960$. We use $n = 4$ *regardless of attribute or dataset*, and this value was chosen in preliminary studies

as a trade-off between the length of the resulting sequences $Q$ and the amount of truncated information. On the other hand, if $m < n$ (the most frequent case), no truncation is necessary and we right-pad $Q$ with zeros. For instance, if $v_j = 35$, then we use $Q = [\text{'1'}, \text{'3'}, \text{'5'}, \text{'0'}, \text{'0'}]$. At decoding time, $O$ is the first token generated by $\mathcal{D}_{\boldsymbol{\phi}}$, which is converted into $m$. If $m < n$, the last $n - m - 1$ tokens in $Q$ are ignored because they are zero-padding digits (e.g., in $Q = [\text{'1'}, \text{'3'}, \text{'5'}, \text{'0'}, \text{'0'}]$, this corresponds to the last 2 elements of $Q$). This implies that possible generation errors in the last $n - m - 1$ tokens are ignored. More formally, using our representation, the joint distribution over the tokens that the decoder should model is restricted to $min(n+1, m+2)$ variables, as opposed to a joint distribution over $p$ variables ($p > m + 2$) as in the case of the fixed digit sequence proposed in [101], reducing the overall error probability.

**Encoder and Decoder Architectures.** Given a tabular row $\boldsymbol{r}$, each categorical value is converted into a token and each numerical value is converted in a sequence $Q$ of $n + 1$ tokens (see above). After that, we use attribute-specific lookup tables to represent all the tokens as embedding vectors: $\boldsymbol{e}_1, ...\boldsymbol{e}_\nu$, where $\nu$ is the sum of the number of categorical attributes plus the number of the numerical attributes multiplied by $(n+1)$. The whole sequence $\boldsymbol{e}_1, ...\boldsymbol{e}_\nu$ is fed to $\mathcal{E}_{\boldsymbol{\varphi}}$, which, following [120], is composed of two separated towers, respectively computing the mean ($\boldsymbol{\mu}$) and log variance ($\log \boldsymbol{\sigma}$) of the latent representation $\boldsymbol{z}$ of $\boldsymbol{r}$ in $\mathcal{Z}$. However, differently from [120], we use multi-head self-attention in $\mathcal{E}_{\boldsymbol{\varphi}}$ because, in preliminary experiments, we found that this is more effective than single-head attention.

Using the standard VAE reparameterization trick, we obtain: $\boldsymbol{z} = \boldsymbol{\mu} + \boldsymbol{\sigma} \cdot \boldsymbol{\varepsilon}$, $\boldsymbol{\varepsilon} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$, and $\boldsymbol{z}$ is fed to the decoder $\mathcal{D}_{\boldsymbol{\phi}}$. Moreover, differently from most DM-based tabular data generation approaches, we propose an AR decoder which is conditioned on $\boldsymbol{z}$. Specifically, decoding a tabular row $\boldsymbol{r}$ starts with a special token `[Start]`. Every time the next-token is predicted, it is coded back and fed to $\mathcal{D}_{\boldsymbol{\phi}}$, which is a Transformer with 3 blocks. Each block alternates multi-head causal attention layers with respect to previously predicted tokens with (multi-head) cross attention layers to the $\nu$ embedding vectors of $\boldsymbol{z}$. For each categorical attribute $a_j$, a final linear layer followed by `softmax` computes a posterior over

the specific vocabulary $V_j$. Similarly, if $a_j$ is numerical, it computes a posterior for each of the $n+1$ tokens in its variable-range representation $Q$. At inference time, we *deterministically* select a token from each of these posteriors using `argmax`. Finally, the predicted value $v_j$ is autoregressively encoded into $\mathcal{D}_{\boldsymbol{\phi}}$ using the same field value representation mechanism used for the encoder.

### 8.2.2 Variable-length time series

Given two sequences $\boldsymbol{x}_i = [\boldsymbol{r}_1, ..., \boldsymbol{r}_{\tau_i}]$ and $\boldsymbol{x}_j = [\boldsymbol{r}_1, ..., \boldsymbol{r}_{\tau_j}]$, $\boldsymbol{x}_i, \boldsymbol{x}_j \in X$, their length is usually different ($\tau_i \neq \tau_j$). Following FiT [75], we use a maximum length $\tau_{max}$ and, for each $\boldsymbol{x}_i$, we append $\tau_{max} - \tau_i$ *padding rows* $\boldsymbol{p}$ (see below) to the right side of $\boldsymbol{x}_i$. However, in FiT the padding tokens are used only to pack data into batches of uniform shape for parallel processing, and are ignored during the forward pass using a masked attention (basically, they are not used). Conversely, we use our padding rows to let the network decide the length of the time series it is generating. To do so, we add an `[EoS]` token to each categorical vocabulary $V_j$, included the vocabulary representing the digits, and we form the special row $\boldsymbol{p} = [\texttt{[EoS]}, ..., \texttt{[EoS]}]$. During VAE training, with probability 0.05 we sample $\boldsymbol{p}$ and with 0.95 we sample a real row from $R$. During the denoising network training, if $\tau_i < \tau_{max}$, $\boldsymbol{x}_i$ is padded with $\boldsymbol{p}$, which results in $\boldsymbol{s}_0$ being a sequence of $\tau_{max}$ latent vectors, where the last vectors correspond to the representation of $\boldsymbol{p}$ in $Z$ and contribute to compute the loss. Finally, at inference time, we randomly sample $\tau_{max}$ vectors $\boldsymbol{z}_1, ..., \boldsymbol{z}_{\tau_{max}}$ in $\mathcal{Z}$, which form $\boldsymbol{s}_T = [\boldsymbol{z}_1, ..., \boldsymbol{z}_{\tau_{max}}]$, the starting point of the DM sampling chain. The ending point of the sampling chain $\boldsymbol{s}_0$ is split in $\tau_{max}$ final vectors $\boldsymbol{z}'_1, ..., \boldsymbol{z}'_{\tau_{max}}$ which are individually decoded using $\mathcal{D}_{\boldsymbol{\phi}}$. We stop decoding as soon as we meet the first padding row $\boldsymbol{p}$. More precisely, we stop decoding when we meet the first row containing at least one token `[EoS]`. In this way, we use $\mathcal{F}_{\boldsymbol{\theta}}$ to *predict the end of the time series jointly with its content*.

## 8.3 Experiments

### 8.3.1 Evaluation Protocol

Due to the lack of a shared evaluation protocol for tabular data time series generation, we propose a unified framework which is composed of different public datasets, conditional and unconditional generation tasks and different metrics.

**Datasets.** We use six public datasets. *Age1*, *Age2*, *Leaving*, taken from [49], and *PKDD'99 Financial Dataset*, taken from [18], are composed of bank transaction time series of different real banks with different attributes. Each time series is the temporally ordered sequence of bank transactions of a given bank client. On the other hand, the *Rossmann* and the *Airbnb* datasets, used in [84, 101], are composed of, respectively, historic sales data for different stores and access log data from Airbnb users. These six datasets are widely different from each other both in terms of their attributes and their sizes, thus representing very different application scenarios. Two of these datasets, *Age1* and *Leaving*, do not have an associated parent table, so they are used only for unconditional generation.

**Metrics.** Since heterogeneous time series generation is a relatively unexplored domain, there is also a lack of consolidated metrics. For instance, Solatorio and Dupriez [101] use the *Logistic Detection*, which is a discriminative metric based on training a binary classifier to distinguish between real and generated data, and then using the ROC-AUC scores of the discriminator (the higher the better ↑) on an held-out set of real and synthetic data. We also adopt this metric in Section 8.3.3 to compare our results with REaLTabFormer. However, the classifier used in the Logistic Detection (a Random Forest) takes as input only an *individual row* of the real/generated time series, thus this criterion is insufficient to assess, e.g., the temporal coherence of a time series composed of different rows. For this reason, we extend this metric using a classifier which takes as input the *entire time series* instead of individual rows. Inspired by similar metrics commonly adopted in single-row tabular data generation [74], we call our metric Machine Learning Detection (*MLD*) and we measure the discriminator accuracy (the lower the better ↓, because it means that the discriminator struggles in separating the real from

the synthetic distribution). To emphasize the difference with respect to the Logistic Detection used in Solatorio and Dupriez [101], use the suffix "SR" (single row) for the latter (*LD-SR*) and the suffix "TS" (*MLD-TS*) to indicate that our MLD metric depends on the entire time series. More specifically, in *MLD-TS* we use CatBoost [87] as the classifier, because it is one of the most common non-deep learning based methods for heterogeneous tabular data discriminative tasks [76, 52, 26], jointly with a standard library [31] extracting a fixed-size feature vector from a time series [76].

### 8.3.2 Ablation

In Table 8.1 we use *Age2*, which, among the six datasets, is both one of the largest in terms of number of time series and one of the most complex for different types of attributes. The results in Table 8.1 are based on an unconditional generation task and evaluate the contribution of each component of our method. The last row, *TabDiT*, refers to the full method as described in Section 8.2, while all the other rows refer to our full-model with one missing component. For instance, *Parallel VAE* refers to a standard, non-AR VAE decoder, in which all the fields of a tabular row are predicted in parallel. *No cross-att VAE* differs from the VAE introduced in Section 8.2.1 because in $\mathcal{D}_{\phi}$ we remove the cross attention layers to $\boldsymbol{z}$, and $\boldsymbol{z}$ is directly fed to $\mathcal{D}_{\phi}$ in place of the `[Start]` token. In all the other entries of the table we use our AR VAE as described in Section 8.2.1. *Fixed digit seq* corresponds to the numerical value representation proposed in REaLTabFormer [101], in which we use $p = 7$ digits (Section 8.2.1), which are enough to represent all the numerical values in *Age2*. In *Quantize*, we follow TabGPT [83] and we convert each numerical feature into a categorical one, using quantization and a field specific vocabulary. In *Linear transf*, we follow TabSyn [120], where numerical features are predicted using a linear transformation of the corresponding last-layer token embedding of the VAE decoder. In all the variants, we always use a coherent numerical value representation in the corresponding VAE encoder. We also evaluated hybrid solutions, where VAE encoders and decoders have different representations of the numerical values, but we always obtained

worse results than in cases of coherent representations. Moreover, in *W/o length pred* we follow FiT [75] and we use a masked attention which completely ignores the padding rows. In this case, at testing time the sequence length is randomly sampled using a mono-modal Gaussian distribution fitted on the length of the training time series. Finally, *AR Baseline* is a (strong) baseline based on a purely AR Transformer which we use to validate the effectiveness of the DM paradigm (see below).

In our **AR Baseline**, we modify the *hierarchical discriminative* architecture proposed in [83] and adopted also in [76] to create an AR generative model. Specifically, we use a causal attention in the "Sequence Transformer" and we replace the "Field Transformer" with one of the towers of our VAE encoder architecture. Moreover, we use our AR VAE decoder architecture to predict the output sequence. Note that we use the VAE architectural components but we *do not* use variational training and the entire network is trained end-to-end using only a next-token prediction task, following [83, 101]. The numerical features are represented using our variable-range decimal representation. In short, this baseline is a purely AR Transformer where we merged the hierarchical architecture used in [83, 76] with our decoding scheme and our feature value representation.

Table 8.1 shows that all the components of the proposed method are important, since their individual removal always leads to a significant decrement of the MLD. Specifically, the numerical value representation has a high impact on the results. For instance, both *Linear transf*, adopted in [120], and *Quantize*, used in [83], lead to a drastic worsening of results. On the other hand, our *AR Baseline*, which is based on the same input representation and shares important architectural details with TabDiT, is largely outperformed by the latter, showing the advantage of our LDM-based approach. Finally, the difference between *No cross-att VAE* and the full method is subtle, showing that the cross attention layers in $\mathcal{D}_{\phi}$ can be replaced by directly feeding $\boldsymbol{z}$ to the decoder, as long as the latter has an AR architecture.

**Table 8.1:** Ablation study on the *Age2* dataset.

| **Method** | Parallel VAE | No cross-att VAE | Fixed digit seq | Quantize | Linear transf | W/o length pred | AR Baseline | TabDiT |
|---|---|---|---|---|---|---|---|---|
| **MLD-TS** ↓ | 62.2 | 51.2 | 64.0 | 83.8 | 83.7 | 61.0 | 68.3 | **50.8** |

**Table 8.2:** Unconditional generation results on *Rossmann*, *Airbnb* and *PKDD'99*.

| | **Rossmann** | | **Airbnb** | | **PKDD'99 Financial** | |
|---|---|---|---|---|---|---|
| **Method** | **MLD-TS** ↓ | **LD-SR** ↑ | **MLD-TS** ↓ | **LD-SR** ↑ | **MLD-TS** ↓ | **LD-SR** ↑ |
| AR Baseline (ours) | 97.80$_{\pm 2.20}$ | 49.97$_{\pm 3.26}$ | 77.23$_{\pm 1.46}$ | 56.43$_{\pm 2.80}$ | 92.87$_{\pm 1.68}$ | 71.93$_{\pm 1.34}$ |
| TabDiT (ours) | **82.60**$_{\pm 3.92}$ | **77.07**$_{\pm 5.37}$ | **55.07**$_{\pm 3.52}$ | **78.07**$_{\pm 2.77}$ | **85.53**$_{\pm 4.18}$ | **79.10**$_{\pm 6.09}$ |

### 8.3.3 Main experiments

Following the protocol adopted in [101], all the experiments of this section have been repeated three times with different random splits of the samples between the generator training data, the discriminators' training data and the testing data. For each experiment, we report the means and the standard deviations of the *MLD-TS* and the *LD-SR* metrics.

**Unconditional generation.** We are not aware of any unconditional generative model for time series of heterogeneous tabular data with public code or published results. Indeed, REaLTabFormer is a *conditional* method, while TabGPT is a *forecasting* model, both lacking of an unconditional sampling mechanism. For this reason, we can only compare TabDiT with our AR Baseline (Section 8.3.2). The unconditional results in Tables 8.2 and 8.3 show that TabDiT outperforms the AR Baseline in all the datasets by a *large margin*. For instance, in the largest dataset (*Age1*), TabDiT improves the *MLD-TS* score by more than 27 points compared to the AR baseline. On *Age2*, TabDiT outperforms the AR Baseline by more than 17 points, achieving an almost ideal situation in which the real and the generated distributions cannot be distinguished by each other (discrimination accuracy = 50.43%, very close to the chance level). Note that the *Age2* results are slightly different from those reported in Table 8.1 because they were obtained averaging 3 different runs.

**Conditional generation.** We indicate with "child gt-cond" the conditional generation task in which the parent table row $\boldsymbol{u}$, used for conditioning, is a

**Table 8.3:** Unconditional generation results on *Age2*, *Age1* and *Leaving*.

| | Age2 | | Age1 | | Leaving | |
|---|---|---|---|---|---|---|
| **Method** | **MLD-TS** ↓ | **LD-SR** ↑ | **MLD-TS** ↓ | **LD-SR** ↑ | **MLD-TS** ↓ | **LD-SR** ↑ |
| AR Baseline (ours) | 67.53$_{\pm 0.75}$ | 83.47$_{\pm 0.38}$ | 91.20$_{\pm 0.46}$ | 74.23$_{\pm 1.06}$ | 69.43$_{\pm 4.02}$ | 75.33$_{\pm 2.86}$ |
| TabDiT (ours) | **50.43**$_{\pm 1.85}$ | **87.00**$_{\pm 1.54}$ | **63.93**$_{\pm 3.20}$ | **76.00**$_{\pm 4.25}$ | **62.33**$_{\pm 0.99}$ | **75.63**$_{\pm 4.20}$ |

ground truth, real row extracted from the testing dataset (Section 8.1). Specifically, we use all the elements $\boldsymbol{u} \in P_{test}$ (Section 8.1) to condition the generator networks. Moreover, to make a comparison with Solatorio and Dupriez [101] possible, we also follow their protocol and we indicate with "child" the conditional generation task on the time series ($\boldsymbol{x}$) where also the conditioning information ($\boldsymbol{u}$) is automatically generated. Specifically, in our case, we generate $\boldsymbol{u}$ by training a dedicated DiT-based denoising network and a corresponding AR VAE on $P$. These networks have the same structure and are trained using the same approach described in Section 8.2 but using $P$ instead of $X$. Finally, in [101] "merged" indicates the evaluation of the joint probability of generating both $\boldsymbol{x}$ and $\boldsymbol{u}$. Using *MLD-TS* and *LD-SR*, this is obtained by concatenating $\boldsymbol{u}$ with either $\boldsymbol{x}$ or with its individual rows $\boldsymbol{r}$, respectively, and then feeding the result to the corresponding discriminator. In Table 8.4, we indicate with * the results of REaLTabFormer and SDV which we report from [101], while in all the other cases they have been reproduced by us using the corresponding public available code. Specifically, while using REaLTabFormer with a real data-conditioning task ("child gt-cond") is easy, that was not possible for SDV. The choice of REaLTabFormer and SDV follows [101], where the selected baselines are those which have open-sourced models.

The results in Table 8.4 confirm the results in Tables 8.2 and 8.3. Across all datasets and tasks, and with both metrics, TabDiT outperforms all the baselines by a large margin, often approaching the lower bound of 50% *MLD-TS* accuracy. In the "child gt-cond" task, AR Baseline is the second best most of the time, while in "child" and "merged" the second best is REaLTabFormer. We believe that the reason of this discrepancy is most likely due to the fact that both the "child" and the "merged" task evaluation depend on the quality of the *single-row*

**Table 8.4:** Conditional generation results. * Values reported from [101].

| Method | Task | Rossmann | | Airbnb | | Age2 | | PKDD'99 Financial | |
|---|---|---|---|---|---|---|---|---|---|
| | | MLD-TS ↓ | LD-SR ↑ | MLD-TS ↓ | LD-SR ↑ | MLD-TS ↓ | LD-SR ↑ | MLD-TS ↓ | LD-SR ↑ |
| SDV | child | 99.63 ±0.64 | 6.53* ±0.39 | 93.30 ±0.61 | 0.00* ±0.00 | 96.03 ±0.11 | 44.80 ±1.73 | 97.95 ±1.42 | 6.53 ±0.58 |
| | merged | 100.00 ±0.00 | 2.80* ±0.25 | 94.40 ±1.65 | 0.00* ±0.00 | 96.27 ±0.06 | 37.63 ±1.47 | 98.12 ±1.17 | 8.77 ±0.59 |
| REaLTabFormer | child gt-cond | 98.90 ±1.10 | 60.63 ±2.65 | 63.63 ±1.20 | 86.17 ±1.29 | 66.77 ±0.42 | 77.90 ±0.85 | 97.87 ±0.59 | 21.97 ±0.55 |
| | child | 64.83 ±1.33 | 52.08 * ±0.89 | 57.77 ±0.67 | 30.48* ±0.79 | 52.97 ±2.32 | 77.30 ±0.92 | 59.33 ±3.82 | 21.50 ±0.72 |
| | merged | 74.43 ±8.85 | 28.33 * ±2.31 | 76.97 ±2.04 | 21.43 * ±1.10 | 52.10 ±2.17 | 75.53 ±0.65 | 58.77 ±3.05 | 26.00 ±1.61 |
| AR Baseline (ours) | child gt-cond | 95.57 ±1.96 | 71.60 ±2.42 | 57.97 ±1.72 | 82.77 ±0.49 | 69.97 ±0.90 | 80.73 ±0.59 | 68.33 ±4.37 | 81.13 ±1.51 |
| | child | 99.63 ±0.64 | 36.03 ±8.79 | 82.33 ±1.53 | 62.53 ±3.93 | 79.03 ±1.62 | 65.83 ±3.95 | 79.07 ±6.23 | 67.60 ±4.36 |
| | merged | 99.63 ±0.64 | 19.70 ±6.80 | 93.50 ±1.30 | 8.53 ±2.49 | 81.30 ±0.78 | 48.03 ±3.61 | 83.33 ±5.75 | 38.73 ±4.22 |
| TabDiT (ours) | child gt-cond | **72.20** ±1.10 | **82.90** ±1.32 | **51.10** ±2.60 | **98.07** ±0.25 | **51.40** ±2.95 | **84.60** ±1.87 | **59.50** ±10.53 | **81.20** ±2.71 |
| | child | **64.03** ±0.64 | **80.13** ±3.02 | **49.33** ±1.18 | **81.10** ±0.98 | **50.47** ±1.71 | **84.70** ±1.21 | **51.80** ±6.44 | **79.13** ±3.04 |
| | merged | **71.83** ±2.77 | **38.63** ±1.04 | **54.63** ±0.85 | **47.37** ±2.68 | **51.53** ±3.04 | **78.93** ±0.64 | **54.03** ±3.95 | **53.20** ±0.66 |

parent generation, in which REaLTabFormer gets better results.

## 8.4 Numerical value representations

In this section, we provide more details on the numerical value representation approaches used by previous work and in our ablation analysis in Section 4.3.1, as well as on our variable-range representation. For clarity, we adopt the same terminology and numerical examples used in Section 8.2.1.

TabGPT [83] applies a quantization which associates $v_j$ with a bin value $B$, which is then treated as a categorical feature. The disadvantage of this representation is an information loss due to the difference between the decoded bin $B$ and the actual value $v_j$. Indeed, $B$ corresponds to the center of the numerical interval assigned to the $B$-th bin during the quantization phase. This coding-decoding scheme corresponds to the entry *Quantize* in Table 4.1.

TabSyn [120] applies a linear transformation when coding $v_j$ and another linear transformation (i.e., a linear regression) when decoding back the embedding vector of the last-layer decoder to $v_j$. However, also this solution is sub-optimal, because linear regression struggles in respecting some implicit value distribution constraints. For instance, if the admissible values for $a_j$ are only integers, a linear regression layer may predict a number with a decimal point. This coding-decoding scheme corresponds to the entry *Linear transf* in Table 4.1.

Finally, *Fixed digit seq* in Table 4.1 corresponds to the sequence of digits $L$ (see

Section 8.2.1) used in REaLTabFormer [101]. We believe that one of the reasons why this representation is sub-optimal with respect to our variable-range decimal representation $Q$ (Sections 4.3.1 and 8.2.1) is that its longer length leads, on average, to a more difficult decoding problem. For instance, in the *Age2* dataset, the "amount" attribute needs $p = 7$ digits to represent $v_{max_j}$. Thus, the length of $L$ is $p = 7$. When the VAE decoder should decode a numerical value, the probability of error is given by the (complement of the) joint distribution of all the digits of its representation $L$. For instance, if the target value is $v_j = 35$, then, the decoder should generate this sequence: $L =$ ['0', '0', '0', '0', '0', '3', '5']. The probability of error when generating $L$ is:

$$\mathcal{P}_L = 1 - P_L(L) = 1 - \prod_{k=1}^{p} P_L(D_k|D_1, ..., D_{k-1}), \tag{8.5}$$

which, in case of $v_j = 35$, is:

$$\begin{aligned} 1 - [&P_L(\text{'0'})P_L(\text{'0'}|\text{'0'})P_L(\text{'0'}|\text{'0', '0'})P_L(\text{'0'}|\text{'0', '0', '0'}) \\ &P_L(\text{'0'}|\text{'0', '0', '0', '0'})P_L(\text{'3'}|\text{'0', '0', '0', '0', '0'}) \\ &P_L(\text{'5'}|\text{'3', '0', '0', '0', '0', '0'})]. \end{aligned} \tag{8.6}$$

On the other hand, in case of $Q$ (Section 8.2.1), since we use an AR decoding, once the magnitude order prefix $O$ has been generated, we can convert $O$ in its corresponding value $m$ (Equation (8.3)) and use $m$ to *ignore* possible zero-padding tokens on the right side of the sequence. Specifically, if $m < n$, the probability of error is given by:

$$\mathcal{P}_Q = 1 - P_Q(Q) = 1 - [P_Q(O) \prod_{k=0}^{m} P_Q(D_{m-k}|O, D_m, ..., D_{m-k+1})], \tag{8.7}$$

which, in case of $v_j = 35$, $Q =$ ['1', '3', '5', '0', '0'], and $m = 1$, is:

$$1 - [P_Q(\text{'1'})P_Q(\text{'3'}|\text{'1'})P_Q(\text{'5'}|\text{'1', '3'})]. \tag{8.8}$$

The comparison between Equation (8.6) and Equation (8.8) intuitively shows that $Q$ is much easier to generate than $L$ if $v_j$ is small. More formally, if we assume that all the digit generations have, on average, the same probability to be correct, i.e., that, on average: $P_Q(O) = P_L(D_1)$ and $P_L(D_k|D_1, ..., D_{k-1}) = P_Q(D_{m-k}|O, D_m, ..., D_{m-k+1})$, then, if $m < p - 2$, from Equations (8.5) and (8.7) follows that $\mathcal{P}_L > \mathcal{P}_Q$.

The proposed representation can be easily extended to negative numbers and non-integer values. For instance, if the admissible values for $a_j$ include negative numbers, then we prepend in $Q$ a token $S$ corresponding to the sign of $v_j$: $Q = [S, O, D_m, D_{m-1}, ..., D_{m-n+1}]$, where $S \in \{\text{'-'}, \text{'+'}\}$. Note that, in case of negative numbers, also $L$ should be extended to include $S$ [101]. On the other hand, in case of rational numbers, $L$ needs to be extended to include a token representing the decimal point [101], while our variable-range representation $Q$ remains unchanged. For instance, if $v_j = 3.5$, then we have $Q = [\text{'0', '3', '5', '0', '0'}]$.

Finally, we provide below more details on how $n$ was chosen. We used the *Age2* dataset [49], adopted for most of our ablation studies. For each numerical values $v_j$ in *Age2*, we compute $DR(v_j)$ (Equation (8.3)) and we remove from $DR(v_j)$ the possible subsequence of only zeros on its right (e.g., from [87600] we remove [00]). Note that these all-zero subsequences do not lead to any truncation error. Let $Significant(v_j)$ be the part of $DR(v_j)$ that remains after this cut (e.g., in the previous example, $Significant(v_j) = [876]$). Finally, we computed the average ($\mu_S$) and the standard deviation ($\sigma_S$) of the lengths of all the sequences $Significant(v_j)$ in the training dataset, getting $\mu_S = 2.26$ and $\sigma_S = 0.47$, respectively. The value $n = 4$ was chosen as the first integer greater than $\mu_S + 2\sigma_S$. In this way, most of the training data in *Age2* do not need any truncation when represented using $Q$. The value $n = 4$ was used in all the other datasets. In Section 8.7.1 we show some qualitative results which compare to each other the distributions of some numerical values generated using the representations pre-

sented in this section.

## 8.5 Metrics

Besides the discriminative metrics described in Section 8.3.1, in the single-row tabular data generation literature there are many other evaluation metrics, which however we do not believe suitable for the time series domain. For instance, *low-order statistics* include statistics computed either using the values of an individual tabular attribute or statistics such as the pair-wise correlation between the values of two numerical attributes [120]. However, in a time series composed of dozens of rows, each row composed of different fields, statistics on a single field or pairs of fields are not very informative. Similarly, we do not use *high-order statistics* (e.g., *$\alpha$-precision* and *$\beta$-recall*) [120], because they are single-row criteria and they usually use a fragile nearest-neighbor like approach in the data space to estimate the distribution coverage.

On the other hand, we believe that the most useful metric is the *MLD-TS* proposed in Section 8.3.1, for which we provide below additional details. The CatBoost [87] discriminator is trained using a balanced dataset composed of both real and synthetic data (separately generated by each compared generative method). Then, a *separate* testing set, also composed of 50% real and 50% generated data, is used to assess the discriminator accuracy. Real training and testing data *do not* include data used to train the generator. Thus, basically the real data are split in: samples used to train the generator, samples used (jointly with synthetic data) to train the discriminator, and samples used (jointly with other synthetic data) to test the discriminator. We randomly change these three splits in each of the run used to compute the results in Section 8.3.3.*

We use the same training-testing protocol for the *Logistic Detection*, which, following [101], is defined as: $LD\text{-}SR = 100 \times (1 - \mu_{RA})$, where:

*The evaluation protocol code is available at: https://github.com/fabriziogaruti/TabDiT

$$\mu_{RA} = \frac{1}{F}\sum_{i=1}^{F} \max(0.5, ROC - AUC) \times 2 - 1. \tag{8.9}$$

In Equation (8.9), ROC and AUC indicate the ROC-AUC scores, computed using a Random Forest trained and tested using single tabular rows. $F = 3$ is the number of cross-validation folds, in which training and testing of the discriminator is repeated $F$ times, keeping fixed the generator weights. We use this metric with its corresponding publicly available code [101] for a fair comparison with REaLTabFormer [101] and SDV [84]. Specifically, in the conditional generation scenario, we follow [101] and we evaluate the *LD-SR* for the "merged" task by concatenating $\boldsymbol{u}$ with all the rows $\boldsymbol{r}$ extracted from a generated time series $\boldsymbol{x}$. The Random Forest is then trained and tested on this "augmented" rows. In case of *MLD-TS*, we concatenate $\boldsymbol{u}$ with the fixed-dimension feature vector extracted from $\boldsymbol{x}$ using the feature extraction library [31] (Section 8.3.1), and we use the "augmented" feature vector to train and test CatBoost.

Finally, we introduce an additional metric based on the Machine Learning Efficiency (MLE) [120], [69], which is based on the accuracy of a classifier trained on generated data and evaluated on a real data testing set. MLE can also be used to simulate an application scenario in which, for instance, the generated data are used to replace real data (e.g., because protected by privacy or legal constraints) in training and testing public machine learning methods. In this case, the classifier accuracy, when trained with synthetic data (only) is usually upper bounded by the accuracy of the same classifier trained on the real data. Similarly to *MLD-TS*, we adapt this metric (which we refer to as *MLE-TS*) to our time series domain using CatBoost as the classifier, fed using a fixed-size feature vector extracted from a given time series [31] (Section 8.3.1). For simplicity, we always use binary classification tasks, whose lower bound is 50% (chance level), and in Section 8.6.1 we show how these tasks can be formulated selecting specific attributes of the parent table to be used as target labels.

## 8.6 Additional experiments

### 8.6.1 Machine Learning Efficiency

In this section, we show additional experiments using the *MLE-TS* metric introduced in Section 8.5. Since this metric is based on training a classifier to predict a target label, we can use *MLE-TS* only to evaluate conditional generations, where these labels can be extracted from the parent table of the corresponding dataset (individual time series or rows are not labeled). Moreover, the "merged" task cannot be used in this case because, as defined in [101], at inference time, it involves the concatenation of the generated data with the conditioning parent table (Section 8.5), the latter used to extract the ground truth target labels. Specifically:

- In Rossmann, we use the binary attribute "Promo2", indicating the presence of a promo in that store.
- In Airbnb, we use the attribute "n_sessions" which indicates the number of sessions opened by a user. We binarize this attribute predicting whether the user has opened more than 20 sessions ($n\_sessions >= 20$).
- In Age2, we use the attribute "age", indicating the age of a customer. In particular, we predict whether the customer is over 30 years old ($age > 30$).
- In PKDD'99 Financial, we use the attribute "region" which indicates the region a customer belongs to. We predict whether the customer is located in Moravia or in Bohemia (including Prague).

In Table 8.5, "Original" indicates that the classifier has been trained on *real* data and tested on real data, and it is an ideal value of the expected accuracy when the same classifier is trained on synthetic data. In the same table, "child" and "child gt-cond" refer to the tasks used in Table 8.4. Specifically, in both cases the classifier is trained with the *generated* data and tested on real data. However, in case of "child gt-cond", we use the *real* parent table row values as the classification

**Table 8.5:** Conditional generation results using the *MLE-TS* ↑ metric (mean and standard deviation over three runs).

| Method | Task | Rossmann | Airbnb | Age2 | PKDD'99 Financial |
|---|---|---|---|---|---|
| Original | - | 66.70 ±8.90 | 100.00 ±0.00 | 60.07 ±1.74 | 61.67 ±2.95 |
| SDV | child | 54.10 ±2.60 | 93.97 ±1.37 | 54.43 ±1.95 | 61.30 ±4.81 |
| REaLTabFormer | child gt-cond | 53.33±2.25 | 60.87±3.10 | 54.53±1.42 | 61.10±2.44 |
| | child | 53.37±5.88 | 61.97±1.57 | 54.97±1.80 | 61.47±8.29 |
| AR Baseline (ours) | child gt-cond | 59.27 ±1.27 | **100.00** ±0.00 | 58.23±1.96 | **62.03** ±5.41 |
| | child | 50.37±9.26 | 59.90±2.46 | 55.77±4.11 | 60.73±3.54 |
| TabDiT (ours) | child gt-cond | **64.43**±10.19 | **100.00**±0.00 | **61.57**±3.05 | 61.30±1.76 |
| | child | **65.93**±9.22 | **99.63**±0.40 | **61.23**±2.55 | **62.60**±3.82 |

target labels (see above). Conversely, in case of "child", we use the *generated* parent table rows. This is because, in a realistic scenario, the "child" task corresponds to a situation in which the parent table data are missing and they are generated as well (Section 8.3.3), thus we coherently use the synthetic parent table values to label the corresponding generated time series. The real time series are always labeled with their corresponding real parent table values.

The results in Table 8.5 confirm those reported in Table 8.4, showing that TabDiT significantly outperforms all the other baselines in all the datasets. Specifically, in Age2, TabDiT even surpasses the ideal "Original" accuracy (61.57 versus 60.07), being very close to the ideal case in all the other datasets. We believe that these results show the effectiveness of the synthetic time series generated by our method, which can potentially be used to replace real data in machine learning tasks when, e.g., the real data cannot be made public.

### 8.6.2 Larger scale experiments

In this section, we use a much larger dataset to show the potentialities of our method to be scaled. Since, as far as we know, heterogeneous time series datasets considerably larger than those used in Section 8.3 are not publicly available, we used a private dataset which we call *Large Scale Bank Data*, and which was provided by an international bank[†]. Large Scale Bank Data consists of several

[†]For both privacy and commercial reasons, this dataset cannot be released.

**Table 8.6:** Large scale conditional experiments using the "child gt-cond" task and Large Scale Bank Data.

| Method | MLD-TS ↓ | LD-SR ↑ | MLE-TS ↑ |
|---|---|---|---|
| Original | - | - | 73.22 |
| AR Baseline (ours) | 58.03 | 83.33 | 70.41 |
| TabDiT (ours) | **56.77** | **83.72** | **71.38** |

hundred million real bank account transactions of private customers. From this dataset, we randomly selected 100K client bank accounts, corresponding to approximately 87.3M transactions (i.e., rows), which we use to train TabDiT and the AR Baseline. Moreover, we selected another set of 10K bank accounts (not included in the training set) for evaluation. Furthermore, with Large Scale Bank Data we use longer time series, setting $\tau_{max} = 100$, which corresponds to an average of one month of bank transactions of a given customer (see Section 8.8 for more details).

In Table 8.6, we compare TabDiT with AR Baseline using the "child gt-cond" task. Note that we were not able to use REaLTabFormer on this large-scale datasets for computational reasons. Indeed the long length of the time series ($\tau_{max} = 100$), combined with the larger number of time series fields (9, as reported in Table 8.8), led to memory usage and time complexity problems during training with REaLTabFormer. Conversely, both the hierarchical architecture of AR Baseline (Section 4.3.1) and our LDM approach allow a *much faster and lower memory consumption* training, which made possible to use a huge dataset like Large Scale Bank Data.

In Table 8.6 we report the *MLD-TS*, the *LD-SR*, and the *MLE-TS* metric values. These results show that TabDiT significantly outperforms AR Baseline. Moreover, even when using a large-scale dataset, TabDiT approaches the lower bound of 50% *MLD-TS* accuracy, achieves a score higher than 80% *LD-SR*, and is able to approach the upper bound of *MLE-TS* obtained using the real data.

### 8.6.3 Constraint satisfaction checks on private data

In addition to the quantitative evaluations reported in the previous sections, we also performed a set of constraint satisfaction checks aimed at validating the structural and semantic correctness of the generated sequences. These diagnostics go beyond discriminative metrics and distributional evaluations by explicitly verifying domain-specific rules that must be respected by realistic transactional data.

The analyses were conducted exclusively on a large proprietary dataset, which is also the largest dataset used in this work, as these constraints are directly derived from real operational rules of that specific transactional system.

In particular, we verified that: (i) transaction timestamps are always monotonically increasing within each generated sequence; (ii) transaction category hierarchies are respected, meaning that generated fine-grained categories are always compatible with their corresponding higher-level categories; (iii) only admissible transaction categories are produced for a given transaction channel (e.g., ATM, POS, transfer); and (iv) withdrawal amounts are generated as round numbers, consistently with real-world ATM constraints.

Across all generated sequences evaluated on this private dataset, these constraints were always satisfied.

While these results cannot be reported quantitatively on public benchmarks, they provide complementary evidence that the proposed generative models capture not only global statistical properties, but also hard structural constraints and temporal consistency. This supports the robustness of the models beyond discriminator-based and distribution-centric evaluations.

### 8.6.4 Additional ablations

In this section we present additional ablation studies. In Table 8.7, we use the "child gt-cond" task to investigate the influence of the CFG and its hyperparameter values $s$ and $p_d$ (Section 4.2). For these experiments we use *PKDD'99 Financial* dataset instead of *Age2* (used in all other ablations) because, on *Age2*, TabDiT achieves a nearly ideal *MLD-TS* score ($\sim 50\%$) *even without CFG*, so there is no room for improvement. In Table 8.7, the scale value $s = 1$ corre-

sponds to non CFG (see Equation (8.2)), and the results reported in the table clearly shows that CFG is beneficial for conditional generation of tabular data time series, a finding which is aligned with the empirical importance of CFG in the image generation literature [85]. Using these results, we selected the values $p_d = 0.005$ and $s = 4$ and we used these hyperparameter values *in all the datasets and conditional tasks*. Although a dataset-dependent hyperparameter tuning may likely lead to even better results, we opted for a simpler and computationally less expensive solution based on dataset-agnostic CFG hyperparameters.

For training our VAE we adopt the scheduling proposed in [120], where the reconstruction loss and the KL-divergence loss are balanced using a coefficient $\beta$ which weights the importance of the latter. The value of $\beta$ starts from an initial $\beta_{max}$ and, during training, it is progressively and adaptively reduced. We refer to [120] for more details. In Figure 8.2 we use *Age2* to show the impact of $\beta_{max}$, which is evaluated jointly with the number of diffusion steps $T$ of the denoising network (Section 4.1). We evaluate the values of $\beta_{max}$ and $T$ jointly because they are strongly related to each other. This ablation shows that using a relatively small number of diffusion steps (e.g., greater than 100) is sufficient to get good results, and that there is no significant improvement with very long trajectories. Conversely, a higher value of $\beta_{max}$, corresponding to a higher regularization of the latent space, improves the results. In all the datasets and tasks, we use 200 diffusion steps and $\beta_{max} = 5$.

**Table 8.7:** PKDD'99 Financial dataset. Analysis of the influence of the CFG hyperparameter values using the **MLD-TS** $\downarrow$ metric (mean and standard deviation over three runs).

| $p_d$ | $s = 1$ | $s = 2$ | $s = 3$ | $s = 4$ | $s = 5$ |
|---|---|---|---|---|---|
| 0.001 | 83.33 ±3.61 | 72.67 ±3.52 | 78.33 ±7.86 | 71.13 ±9.38 | 69.33 ±7.92 |
| 0.005 | 80.47 ±8.02 | 73.00 ±8.84 | 72.47 ±14.95 | **59.50** ±10.53 | 75.00 ±12.30 |
| 0.010 | 77.87 ±10.40 | 70.80 ±10.83 | 69.50 ±13.95 | 76.83 ±6.40 | 73.97 ±9.51 |
| 0.100 | 80.67 ±4.74 | 71.10 ±19.11 | 69.13 ±12.90 | 66.77 ±12.79 | 69.63 ±16.61 |

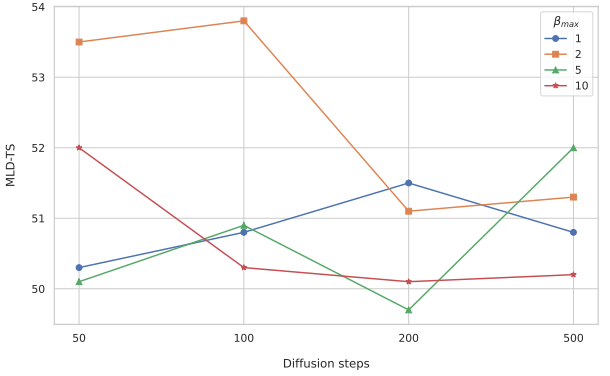



**Figure 8.2:** Age2 dataset. Analysis of the influence of the number of diffusion time steps $T$ jointly with the $\beta_{max}$ value using the **MLD-TS** $\downarrow$ metric.

## 8.7 Qualitative results

### 8.7.1 Numerical field representations

In this section, we show some qualitative results using the numerical representation methods evaluated in Table 8.1. Specifically, we use the *Airbnb* dataset and we select the "secs_elapsed" attribute. This is the field with the highest variability, with values ranging from 0 to 1.8M, a mean of 3.3K and a standard deviation of approximately 13K.

In Figures 8.3 to 8.6 we compare to each other the distributions of all evaluated numerical representations. In every plot, the $x$ axis is based on a logarithmic scale and a fixed number of 50 bins for both the real and the generated numerical values. For each bin, in the $y$ axis we show the percentage of tabular rows that contain numerical values that belong to that bin.

In *Quantize* (Figure 8.3), the "secs_elapsed" field values are quantized using equal-size bins of 360 seconds (1 hour) and we generate the bin center. As depicted in the figure, this method struggles in representing small numerical values, which are all grouped in a few bins. On the other hand, the *Linear transf* representation (Figure 8.4) tends to under-sample the tails of the distribution. In the last two figures, we qualitatively evaluate the *Fixed digit seq* (Figure 8.5) and our *variable-range decimal representation* (Figure 8.6). The corresponding distributions show that our variable-range representation is more accurate in reproducing numerical values. We believe that one of the reasons why the *Fixed digit seq* representation is sub-optimal is that its longer length leads, on average, to a more difficult decoding problem.

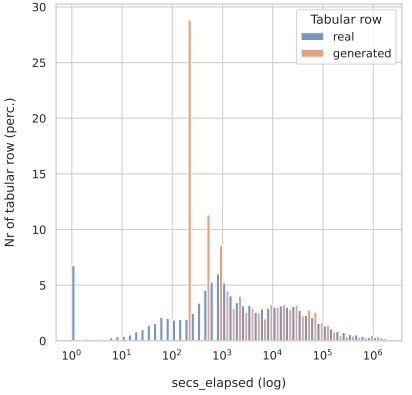



**Figure 8.3:** Real and generated distributions of the values of the "secs_elapsed" attribute using *Quantize* as the numerical field value representation.

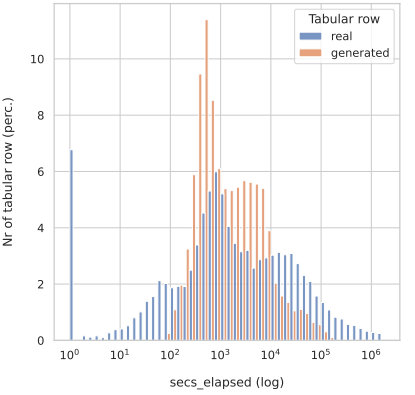



**Figure 8.4:** Real and generated distributions of "secs_elapsed" using *Linear transf*.

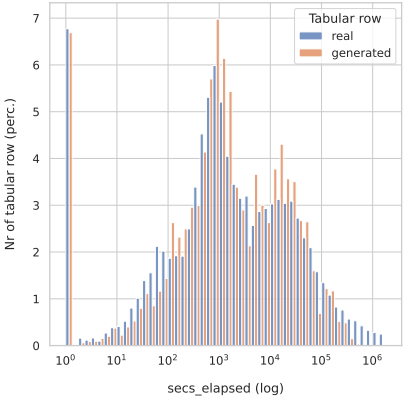



**Figure 8.5:** Real and generated distributions of "secs_elapsed" using *Fixed digit seq*.

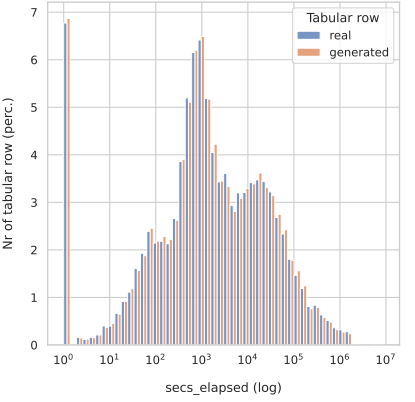



**Figure 8.6:** Real and generated distributions of "secs_elapsed" using *variable-range decimal representation*.

### 8.7.2 Time series length

In Figures 8.7 and 8.8 we show the distribution of the real and the generated time series lengths on *Airbnb*. In the first plot, we use the *W/o length pred* (Section 8.3.2) method: at inference time the sequence length is randomly sampled using a mono-modal Gaussian distribution fitted on the length of the training time series. On the other hand, in Figure 8.8 we use our padding rows (Section 8.2.2) to predict the time series length. Figures 8.7 and 8.8 show that, in the latter case, the time series length distribution is more accurately reproduced.

Figures 8.9 and 8.10 show the distributions of the difference between the real and the generated time series length using a "child gt-cond" task. Specifically, given a real parent table $\boldsymbol{u}$, representing a client of the *Airbnb* dataset, we generate a time series $\hat{\boldsymbol{x}}$ using both our full method (Figure 8.10) and *W/o length pred* (Figure 8.9). In both cases we compute the difference between the length of the generated series $\hat{\boldsymbol{x}}$ ($\tau_{\hat{\boldsymbol{x}}}$) and the length of the real time series $\boldsymbol{x}$ ($\tau_{\boldsymbol{x}}$) associated with $\boldsymbol{u}$. The shorter the difference, the better the method in predicting the real length. Figures 8.9 and 8.10 show these differences for the two methods. Specifically, in Figure 8.9, since $\tau_{\hat{\boldsymbol{x}}}$ is sampled from a mono-modal Gaussian fitted on the training set (Section 8.2.2), it is independent of $\boldsymbol{u}$. As a result, the denoising network cannot predict the correct time series length. On the other hand, Figure 8.10 shows that, in case of our full-method, the distribution of the difference between $\tau_{\hat{\boldsymbol{x}}}$ and $\tau_{\boldsymbol{x}}$ is very close to zero, showing the effectiveness of predicting the series length jointly with its content.

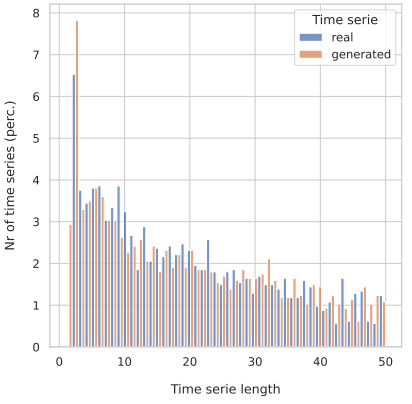



**Figure 8.7:** Distribution of the lengths of the real and the generated time series, using *W/o length pred* method.

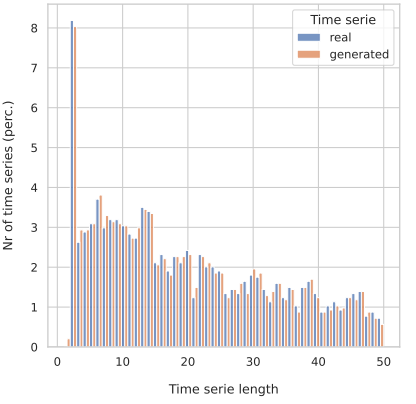



**Figure 8.8:** Distribution of the lengths of the real and the generated time series, using our padding row generation (section 8.2.2).

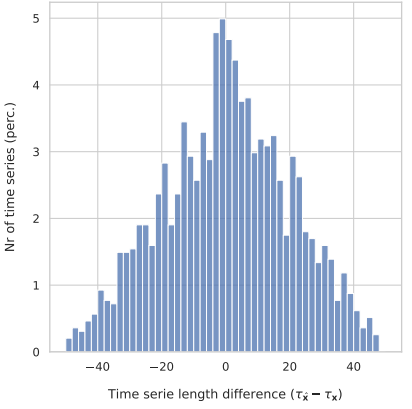



**Figure 8.9:** Distribution of the differences between the real and the generated time series lengths when using *W/o length pred*.

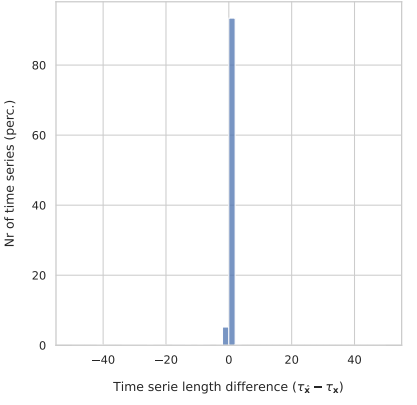



**Figure 8.10:** Distribution of the differences between the real and the generated time series lengths when using padding row prediction.

### 8.7.3 Time series examples

In this section, we show some examples of real and generated time series to qualitatively evaluate the generation results. We use the "child gt-cond" task on the *PKDD'99 Financial* dataset and on the *Airbnb* dataset. We also use the unconditional generation task on the *Leaving* dataset.

In the conditional task (Figure 8.11-Figure 8.16), we first provide an example of a real parent table row, used as the conditional ground truth information, then we show the first ten rows of the corresponding real time series, and finally the generated one. In these examples, we compare TabDiT with REaLTabFormer. In the unconditional task, we provide the first ten rows of a real time series example (Figure 8.17), a time series example generated using TabDiT (Figure 8.18), and a time series example generated using AR baseline (Figure 8.19).

The results in Figure 8.13 and Figure 8.19 show that both REaLTabFormer and the AR baseline make some errors in generating correctly time-ordered dates in the time series. This may be an important error in real-world applications.

**Real parent table row**

| district_id | frequency | city | region | year | month | day |
|---|---|---|---|---|---|---|
| 61 | POPLATEK MESICNE | Trebic | south Moravia | 94 | 6 | 17 |

**Real time series**

| Year | Month | Day | type_trans | operation | k_symbol | amount_trans | balance |
|---|---|---|---|---|---|---|---|
| 96 | 10 | 8 | PRIJEM | VKLAD | None | 13818.0 | 48891.9 |
| 96 | 10 | 12 | VYDAJ | PREVOD NA UCET | UVER | 3757.0 | 45134.9 |
| 96 | 10 | 16 | VYDAJ | VYBER | None | 4440.0 | 40694.9 |
| 96 | 10 | 20 | VYDAJ | VYBER | None | 3600.0 | 37094.9 |
| 96 | 10 | 27 | VYDAJ | VYBER | None | 5280.0 | 31814.9 |
| 96 | 10 | 31 | VYDAJ | VYBER | SLUZBY | 14.6 | 31970.6 |
| 96 | 10 | 31 | PRIJEM | None | UROK | 170.3 | 31985.2 |
| 96 | 11 | 3 | VYDAJ | VYBER | None | 5600.0 | 26370.6 |
| 96 | 11 | 8 | PRIJEM | VKLAD | None | 13818.0 | 40188.6 |
| 96 | 11 | 12 | VYDAJ | PREVOD NA UCET | UVER | 3757.0 | 36431.6 |

**Figure 8.11:** *PKDD'99 Financial* dataset: an example of a **real time series** and its corresponding real parent table row.

**Real parent table row**

| district_id | frequency | city | region | year | month | day |
|---|---|---|---|---|---|---|
| 61 | POPLATEK MESICNE | Trebic | south Moravia | 94 | 6 | 17 |

**Generated time series**

| Year | Month | Day | type_trans | operation | k_symbol | amount_trans | balance |
|---|---|---|---|---|---|---|---|
| 96 | 2 | 18 | VYDAJ | VYBER | None | 720.0 | 19090.0 |
| 96 | 2 | 28 | VYDAJ | VYBER | None | 2800.0 | 16690.0 |
| 96 | 2 | 29 | VYDAJ | VYBER | SLUZBY | 14.6 | 16750.0 |
| 96 | 2 | 29 | PRIJEM | None | UROK | 80.5 | 16750.0 |
| 96 | 3 | 5 | VYDAJ | PREVOD NA UCET | | 3146.0 | 14410.0 |
| 96 | 3 | 8 | PRIJEM | VKLAD | None | 6914.0 | 20790.0 |
| 96 | 3 | 12 | VYDAJ | VYBER | None | 1500.0 | 20420.0 |
| 96 | 3 | 13 | PRIJEM | VKLAD | None | 5718.0 | 26530.0 |
| 96 | 3 | 31 | VYDAJ | VYBER | SLUZBY | 14.6 | 26690.0 |
| 96 | 3 | 31 | PRIJEM | None | UROK | 87.5 | 26700.0 |

**Figure 8.12:** *PKDD'99 Financial* dataset: an example of a **generated time series** conditioned on a ground truth parent table row using TabDiT.

**Real parent table row**

| district_id | frequency | city | region | year | month | day |
|---|---|---|---|---|---|---|
| 61 | POPLATEK MESICNE | Trebic | south Moravia | 94 | 6 | 17 |

**Generated time series**

| Year | Month | Day | type_trans | operation | k_symbol | amount_trans | balance |
|---|---|---|---|---|---|---|---|
| 96 | 11 | 7 | VYDAJ | PREVOD NA UCET | | 76.8 | 13675.2 |
| 96 | 12 | 16 | VYDAJ | PREVOD NA UCET | SIPO | 4508.0 | 19801.2 |
| 96 | 12 | 2 | VYDAJ | VYBER | SLUZBY | 1372.0 | 14904.4 |
| 96 | 11 | 14 | PRIJEM | PREVOD Z UCTU | DUCHOD | 180.0 | 16491.9 |
| 97 | 1 | 6 | VYDAJ | PREVOD NA UCET | None | 1823.0 | 23939.3 |
| 96 | 12 | 7 | VYDAJ | VYBER | None | 4527.0 | 15727.0 |
| 96 | 12 | 6 | VYDAJ | VYBER | SLUZBY | 1771.0 | 11381.1 |
| 97 | 4 | 5 | VYDAJ | PREVOD NA UCET | SIPO | 71.6 | 66265.1 |
| 97 | 1 | 11 | VYDAJ | VYBER | None | 114.3 | 19241.6 |
| 96 | 10 | 3 | VYDAJ | PREVOD NA UCET | SIPO | 14.0 | 15248.9 |

**Figure 8.13:** *PKDD'99 Financial* dataset: an example of a **generated time series** conditioned on a ground truth parent table row using REaLTabFormer. The sequence of the dates is not chronologically ordered.

**Real parent table row**

| CompetitionDistance | Promo2SinceWeek | CompetitionOpenSinceYear | CompetitionOpenSinceMonth | Promo2SinceYear | StoreType | Assortment | PromoInterval | Promo2 |
|---|---|---|---|---|---|---|---|---|
| 1420.0 | 40.0 | 2012.0 | 10.0 | 2014.0 | a | a | Jan,Apr,Jul,Oct | 1 |

**Real time series**

| Open | Promo | StateHoliday | SchoolHoliday | DayOfWeek | Customers | Sales | Date_month | Date_day |
|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 0 | 0 | 5 | 743.0 | 7509.0 | 7 | 31 |
| 1 | 1 | 0 | 0 | 4 | 687.0 | 7171.0 | 7 | 30 |
| 1 | 1 | 0 | 0 | 3 | 647.0 | 6926.0 | 7 | 29 |
| 1 | 1 | 0 | 0 | 2 | 696.0 | 7432.0 | 7 | 28 |
| 1 | 1 | 0 | 0 | 1 | 753.0 | 8528.0 | 7 | 27 |
| 0 | 0 | 0 | 0 | 7 | 0.0 | 0.0 | 7 | 26 |
| 1 | 0 | 0 | 0 | 6 | 710.0 | 6887.0 | 7 | 25 |
| 1 | 0 | 0 | 0 | 5 | 593.0 | 5056.0 | 7 | 24 |
| 1 | 0 | 0 | 0 | 4 | 586.0 | 5557.0 | 7 | 23 |
| 1 | 0 | 0 | 0 | 3 | 491.0 | 4603.0 | 7 | 22 |

**Figure 8.14:** *Airbnb* dataset: an example of a **real time series** and a real parent table row.

**Real parent table row**

| CompetitionDistance | Promo2SinceWeek | CompetitionOpenSinceYear | CompetitionOpenSinceMonth | Promo2SinceYear | StoreType | Assortment | PromoInterval | Promo2 |
|---|---|---|---|---|---|---|---|---|
| 1420.0 | 40.0 | 2012.0 | 10.0 | 2014.0 | a | a | Jan,Apr,Jul,Oct | 1 |

**Generated time series**

| Open | Promo | StateHoliday | SchoolHoliday | DayOfWeek | Customers | Sales | Date_month | Date_day |
|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 0 | 0 | 5 | 593.0 | 6882.0 | 7 | 31 |
| 1 | 1 | 0 | 0 | 4 | 585.0 | 6349.0 | 7 | 30 |
| 1 | 1 | 0 | 0 | 3 | 505.0 | 5094.0 | 7 | 29 |
| 1 | 1 | 0 | 0 | 2 | 571.0 | 6752.0 | 7 | 28 |
| 1 | 1 | 0 | 0 | 1 | 596.0 | 7502.0 | 7 | 27 |
| 0 | 0 | 0 | 0 | 7 | 0.0 | 0.0 | 7 | 26 |
| 1 | 0 | 0 | 0 | 6 | 471.0 | 5136.0 | 7 | 25 |
| 1 | 0 | 0 | 0 | 5 | 462.0 | 4352.0 | 7 | 24 |
| 1 | 0 | 0 | 0 | 4 | 470.0 | 4249.0 | 7 | 23 |
| 1 | 0 | 0 | 0 | 3 | 371.0 | 3722.0 | 7 | 22 |

**Figure 8.15:** *Airbnb* dataset: an example of a **generated time series** conditioned on a ground truth parent table row using TabDiT.

**Real parent table row**

| CompetitionDistance | Promo2SinceWeek | CompetitionOpenSinceYear | CompetitionOpenSinceMonth | Promo2SinceYear | StoreType | Assortment | PromoInterval | Promo2 |
|---|---|---|---|---|---|---|---|---|
| 1420.0 | 40.0 | 2012.0 | 10.0 | 2014.0 | a | a | Jan,Apr,Jul,Oct | 1 |

**Generated time series**

| Open | Promo | StateHoliday | SchoolHoliday | DayOfWeek | Customers | Sales | Date_month | Date_day |
|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 0 | 1 | 5 | 732 | 10200 | 7 | 31 |
| 1 | 1 | 0 | 1 | 4 | 494 | 4633 | 7 | 30 |
| 1 | 1 | 0 | 1 | 3 | 595 | 8110 | 7 | 29 |
| 1 | 1 | 0 | 1 | 2 | 791 | 5896 | 7 | 28 |
| 1 | 1 | 0 | 1 | 1 | 1134 | 7871 | 7 | 27 |
| 0 | 0 | 0 | 0 | 7 | 0 | 0 | 7 | 26 |
| 1 | 0 | 0 | 0 | 6 | 407 | 5000 | 7 | 25 |
| 1 | 0 | 0 | 1 | 5 | 785 | 5642 | 7 | 24 |
| 1 | 0 | 0 | 1 | 4 | 984 | 4870 | 7 | 23 |
| 1 | 0 | 0 | 1 | 3 | 688 | 4380 | 7 | 22 |

**Figure 8.16:** *Airbnb* dataset: an example of a **generated time series** conditioned on a ground truth parent table row using REaLTabFormer.

| year | month | day | hour | minute | second | channel_type | trx_category | currency | MCC | amount |
|---|---|---|---|---|---|---|---|---|---|---|
| 2017 | 7 | 10 | 0 | 0 | 0 | type1 | POS | 810 | 5211 | 1471.49 |
| 2017 | 7 | 10 | 4 | 59 | 42 | type1 | POS | 810 | 5999 | 1300.00 |
| 2017 | 7 | 11 | 0 | 0 | 0 | type1 | POS | 810 | 5411 | 102.00 |
| 2017 | 7 | 12 | 0 | 0 | 0 | type1 | POS | 810 | 5411 | 312.00 |
| 2017 | 7 | 12 | 0 | 0 | 0 | type1 | POS | 810 | 5211 | 1822.63 |
| 2017 | 8 | 1 | 0 | 0 | 0 | type1 | POS | 810 | 5411 | 184.00 |
| 2017 | 8 | 11 | 0 | 0 | 0 | type1 | WD_ATM_PARTNER | 810 | 6011 | 7000.00 |
| 2017 | 8 | 11 | 0 | 0 | 0 | type1 | POS | 810 | 5691 | 4027.00 |
| 2017 | 8 | 11 | 0 | 0 | 0 | type1 | POS | 810 | 5691 | 499.00 |
| 2017 | 8 | 11 | 0 | 0 | 0 | type1 | POS | 810 | 7922 | 980.00 |

**Figure 8.17:** *Leaving* (unconditional task): an example of a **real time series**.

| year | month | day | hour | minute | second | channel_type | trx_category | currency | MCC | amount |
|---|---|---|---|---|---|---|---|---|---|---|
| 2017 | 4 | 20 | 0 | 0 | 0 | type2 | POS | 810 | 5411 | 756.0 |
| 2017 | 4 | 22 | 0 | 0 | 0 | type2 | POS | 810 | 8999 | 1600.0 |
| 2017 | 4 | 22 | 0 | 0 | 0 | type2 | POS | 810 | 5912 | 500.0 |
| 2017 | 4 | 28 | 0 | 0 | 0 | type2 | POS | 810 | 5977 | 3028.0 |
| 2017 | 5 | 1 | 0 | 0 | 0 | type2 | POS | 810 | 5411 | 749.0 |
| 2017 | 5 | 12 | 14 | 41 | 43 | type2 | WD_ATM_ROS | 810 | 6011 | 1000.0 |
| 2017 | 5 | 21 | 0 | 0 | 0 | type2 | POS | 810 | 5411 | 1068.0 |
| 2017 | 5 | 21 | 19 | 8 | 6 | type2 | WD_ATM_ROS | 810 | 6011 | 2100.0 |
| 2017 | 5 | 22 | 0 | 0 | 0 | type2 | WD_ATM_OTHER | 810 | 6011 | 2400.0 |
| 2017 | 5 | 22 | 0 | 0 | 0 | type2 | POS | 810 | 5411 | 857.1 |

**Figure 8.18:** *Leaving* (unconditional task): an example of a **generated time series** using TabDiT.

| year | month | day | hour | minute | second | channel_type | trx_category | currency | MCC | amount |
|---|---|---|---|---|---|---|---|---|---|---|
| 2017 | 3 | 2 | 10 | 4 | 15 | type1 | WD_ATM_ROS | 810 | 6011 | 40000.00 |
| 2017 | 3 | 28 | 0 | 0 | 0 | type1 | POS | 810 | 5331 | 17660.00 |
| 2017 | 4 | 26 | 0 | 0 | 0 | type1 | POS | 810 | 5964 | 33640.00 |
| 2017 | 4 | 8 | 0 | 0 | 0 | type1 | POS | 810 | 5411 | 3775.00 |
| 2017 | 4 | 9 | 0 | 41 | 36 | type1 | POS | 810 | 5814 | 490.00 |
| 2017 | 5 | 3 | 0 | 0 | 0 | type1 | POS | 810 | 5942 | 102.00 |
| 2017 | 5 | 22 | 0 | 0 | 0 | type1 | POS | 810 | 5411 | 4998.00 |
| 2017 | 5 | 18 | 0 | 0 | 0 | type1 | POS | 810 | 5681 | 3240.00 |
| 2017 | 5 | 18 | 10 | 23 | 6 | type1 | WD_ATM_ROS | 810 | 5912 | 4564.00 |
| 2017 | 6 | 15 | 12 | 29 | 1 | type1 | WD_ATM_ROS | 810 | 3381 | 0.35 |

**Figure 8.19:** *Leaving* (unconditional task): an example of a **generated time series** using AR baseline. The sequence of the dates is not chronologically ordered.

## 8.8 Dataset statistics

In Table 8.8 we report the main characteristics of the datasets used in our experiments. We use six real-world public datasets: *Age2*[‡], *Age1*[§], *Leaving*[¶], the *PKDD'99 Financial* dataset[‖], the *Rossmann* store sales dataset[**] and the *Airbnb* new user bookings dataset[††]. The first three datasets have been previously used in [49], while the last two datasets have been used in [101]. *Age2*, *PKDD'99 Financial*, *Rossmann* and *Airbnb* include both parent and time series data, and they can be used for the conditional generation tasks. Conversely, *Age1* and *Leaving* do not include the parent table, thus we used them only for the unconditional generation task. Original source, copyright, and license information are available in the links in the footnotes.

In *Rossmann* and *Airbnb*, we use the same training and testing splits created in [101]. Specifically, in *Rossmann*, we use 80% of the store data and their associated sales records for training the generator. We use the remaining stores as the testing data. Again following [101], we limit the data used in the experiments from the years 2015-2016 onwards, spanning 2 months of sales data per store. Moreover, in the *Airbnb* dataset, we consider a random sample of 10,000 users for the experiment. We take 8,000 as part of our training data, and we assess the metrics using the 2,000 users in the testing data. We also limit the users considered to those having at most 50 sessions in the data. Regarding *Age2*, *Age1*, *Leaving* and *PKDD'99 Financial*, we use the entire datasets, without any data filtering. The only exception is for the *PKDD'99 Financial* parent table where we used the following fields: district_id, frequency, city, region, and the account creation date.

Table 8.8 also includes the statistics of *Large Scale Bank Data*. In our experiments with this dataset, we used both its parent table and its time series, without any data filtering.

[‡] Age2 dataset [49]
[§] Age1 dataset [49]
[¶] Leaving dataset [49]
[‖] PKDD'99 Financial dataset [18]
[**] Rossmann store sales dataset [101]
[††] Airbnb new user bookings dataset [101]

**Table 8.8:** Dataset statistics.

| | Age2 | Age1 | Leaving | PKDD'99 Financial | Rossmann | Airbnb | Large Scale Bank Data |
|---|---|---|---|---|---|---|---|
| Total dataset rows | 3,652,757 | 44,117,905 | 490,513 | 1,056,320 | 68,015 | 192,596 | 96,040,010 |
| Total time series | 43,289 | 50,000 | 5,000 | 4,500 | 1,115 | 10,000 | 110,000 |
| Training samples | 38,961 | 45,000 | 4,000 | 3,600 | 892 | 8,000 | 100,000 |
| Testing samples | 4,328 | 5,000 | 1,000 | 900 | 223 | 2,000 | 10,000 |
| $\tau_{max}$ | 50 | 50 | 50 | 50 | 61 | 50 | 100 |
| Time series fields | 3 | 3 | 6 | 6 | 8 | 5 | 9 |
| Time series numerical fields | 1 | 1 | 1 | 2 | 2 | 1 | 1 |
| Time series categorical fields | 1 | 2 | 4 | 3 | 5 | 4 | 7 |
| Time series date/time fields | 1 | 0 | 1 | 1 | 1 | 0 | 1 |
| Parent fields | 4 | 0 | 0 | 5 | 9 | 16 | 8 |
| Parent numerical fields | 2 | 0 | 0 | 0 | 1 | 2 | 0 |
| Parent categorical fields | 2 | 0 | 0 | 4 | 8 | 11 | 8 |
| Parent date/time fields | 0 | 0 | 0 | 1 | 0 | 3 | 0 |
| Data availability | public | public | public | public | public | public | private |

## 8.9 Implementation Details

In this section, we provide additional implementation details jointly with the values of the hyperparameters used in our experiments. Table 8.9 shows the number of training epochs and iterations for both the VAE and the denoising network (TabDiT and SR-TabDiT). Since Large Scale Bank Data was used only for a "child gt-cond" task, we did not train a SR-TabDiT with this dataset.

The model hyperparameter values in Tables 8.10 and 8.11 are shared by all the datasets. Moreover, both TabDiT and SR-TabDiT share the same hyperparameter values, both for the VAE and the denoising network.

Table 8.10 shows the VAE hyperparameters. The encoder $\mathcal{E}_{\phi}$ and decoder $\mathcal{D}_{\phi}$ of the VAE architecture have the same number of layers and the same number of heads, and the same hidden size. The latent space size of the VAE is the same as the DiT-based denoising network $\mathcal{F}_{\phi}$ hidden size $d$, in order to have the denoising network work directly in the latent space of the VAE model.

The DiT-based denoising network $\mathcal{F}_{\phi}$ hyperparameters are detailed in Table 8.11. Most of the hyperparameter values are borrowed by the Dit-B model presented in [85]. We use a standard frequency-based positional embedding, the same as DiT, the only difference being that we have a single dimension input (the time series length) rather than a 2D image. The main hyperparameter of the

**Table 8.9:** Dataset-specific hyperparameter values.

| | | Age2 | Age1 | Leaving | PKDD'99 Financial | Rossmann | Airbnb | Large Scale Bank Data |
|---|---|---|---|---|---|---|---|---|
| TabDiT VAE | Training epochs | 50 | 5 | 100 | 50 | 2,000 | 300 | 3 |
| | Training iterations | 175,950 | 215,100 | 43,200 | 49,150 | 120,000 | 40,800 | 378,800 |
| TabDiT | Training epochs | 150 | 150 | 2,000 | 2,000 | 4,000 | 800 | 180 |
| Denoising network | Training iterations | 45,750 | 52,800 | 64,000 | 58,000 | 28,000 | 50,400 | 118,400 |
| SR-TabDiT VAE | Training epochs | 1,000 | - | - | 5,000 | 6,000 | 20,000 | - |
| | Training iterations | 39,000 | - | - | 20,000 | 6,000 | 180,000 | - |
| SR-TabDiT Denoising | Training epochs | 1,000 | - | - | 3,000 | 6,000 | 3,000 | - |
| network | Training iterations | 170,000 | - | - | 54,000 | 54,000 | 120,000 | - |

**Table 8.10:** Dataset-independent hyperparameter values for the VAE.

| Hyperparameter | Value |
|---|---|
| Optimizer | AdamW |
| Learning rate | 5e-05 |
| Training dropout | 0.1 |
| Batch size | 1,024 |
| Model size (parameters) | 2M |
| VAE Encoder Transformer layers | 3 |
| VAE Encoder Transformer heads | 8 |
| VAE Encoder hidden size | 72 |
| VAE Decoder Transformer layers | 3 |
| VAE Decoder Transformer heads | 8 |
| VAE Encoder hidden size | 72 |
| VAE latent space size ($d$) | 792 |
| $\beta_{max}$ | 5 |
| $\beta_{min}$ | 0.05 |
| $\lambda$ | 0.7 |
| *patience* | 5 |

denoising network that we change is the number of diffusion steps ($T$), which needs to be adapted to our tabular data time series domain.

**Table 8.11:** Dataset-independent hyperparameter values of the denoising network.

| Hyperparameter | Value |
|---|---|
| Optimizer | AdamW |
| Positional encoding | frequency-based |
| Learning rate | 1e-04 |
| Dropout | 0.1 |
| Batch size | 128 |
| Model size (parameters) | 140M |
| DiT depth | 12 |
| DiT num heads | 12 |
| Hidden size ($d$) | 792 |
| Diffusion steps | 200 |
| $p_d$ | 0.005 |
| $s$ | 4 |

## 8.10 Computing resources

All the experiments presented in this chapter have been performed on an internal compute node composed of:

- 2 CPUs AMD EPYC 7282 16-Core, for a total of 32 physical and 64 logical cores,
- 256 Gb RAM,
- 4 GPUs Nvidia RTX A6000, each with 48 Gb of memory, for a total of 192 Gb.

Table 8.12 shows the training time for the VAE and the denoising network on each dataset.

Due to the very high computational cost of pre-training, it was not feasible to repeat the full pre-training process with different random seeds. Each pre-training run requires significant multi-GPU time and memory, making multiple repetitions impractical within the available compute budget. However, to still capture variability in the generation process, we repeated the decoding (sampling) phase multiple times with different seeds. Decoding is comparatively

**Table 8.12:** Dataset-specific total training time (measured in hours).

| | **Age2** | **Age1** | **Leaving** | **PKDD'99 Financial** | **Rossmann** | **Airbnb** | **Large Scale Bank Data** |
|---|---|---|---|---|---|---|---|
| VAE | 3h | 3h | 1h | 3h | 2h | 2h | 8h |
| Denoising network | 5h | 5h | 5h | 5h | 3h | 3h | 26h |

lightweight, and repeating it allows us to report metrics that accurately reflect the stochastic nature of diffusion-based generation, even though the underlying pre-trained model remains fixed.

# 9 Applications, Limitations, and Future Directions

## 9.1 Applications in Finance

The unified Transformer-based framework presented in this thesis is particularly well suited for financial applications, where data are inherently structured, heterogeneous, and strongly time-dependent [71]. Financial institutions generate massive volumes of transactional data that encode rich information about customer behavior, risk exposure, and economic activity. Effectively leveraging these data requires models capable of capturing complex temporal dependencies, heterogeneous feature interactions, and long-range behavioral patterns, while also complying with strict privacy and regulatory constraints [11]. The methods proposed in this work address these requirements and enable a broad spectrum of practical applications in finance.

Fraud Detection and Anomaly Detection One of the most established and impactful applications of transactional modeling is fraud detection. Fraudulent behaviors are typically rare, highly heterogeneous, and adaptive,

making them difficult to detect with rule-based systems or static Machine Learning models [34]. Traditional approaches rely heavily on handcrafted features aggregated over fixed time windows, which may fail to capture subtle temporal signals or evolving fraud patterns.

The representation learning framework introduced in this thesis enables a fundamentally different approach. By encoding entire transaction histories into dense latent representations, models such as UniTTab can learn behavioral profiles that reflect both short-term dynamics and long-term regularities [116]. These representations can be used directly for supervised fraud classification, or indirectly for anomaly detection by identifying deviations from learned normal behavior. In practice, this allows for earlier detection of fraudulent activities, improved generalization to unseen fraud patterns, and reduced dependence on manually engineered features.

Moreover, the generative models proposed in later chapters offer additional benefits in this context. Synthetic transaction sequences generated by UniTTab-AR or BankDiT can be used to augment training data, particularly for rare fraud classes, improving class balance and robustness without exposing sensitive customer information [114, 11].

Credit Risk and Default Prediction Credit risk assessment is another core financial application where transactional time series play a crucial role. While traditional credit scoring models primarily rely on static customer attributes and aggregated financial indicators, transactional data provide a much more granular and dynamic view of a customer's financial behavior, including income stability, spending habits, and liquidity management [71].

The Transformer-based representations learned through self-supervised pre-training enable models to capture early signals of financial distress, such as changes in spending regularity, increasing reliance on credit, or irregular income patterns [116]. These representations can be fine-tuned for downstream tasks such as loan default prediction, early warning systems, and dynamic credit limit adjustment. Importantly, the ability to pretrain on large volumes of unlabeled data makes it possible to leverage the full transactional history of customers,

even when labeled default events are scarce.

In this setting, foundation models for transactional data act as reusable backbones that can be adapted across multiple credit-related tasks, reducing development time and improving consistency across risk models within an institution [19].

Customer Profiling, Segmentation, and Churn Prediction Understanding customer behavior is central to customer relationship management, marketing strategies, and retention policies. Transactional time series provide a detailed behavioral fingerprint of each customer, but their complexity has historically limited their use to coarse aggregations and manually defined segments.

The methods proposed in this thesis allow customer profiles to be learned directly from raw transactional sequences, capturing both the diversity and the temporal evolution of behaviors [109]. Learned embeddings can be used for clustering customers into behaviorally meaningful segments, enabling more personalized offers, targeted interventions, and tailored communication strategies.

In the context of churn prediction, temporal representations are particularly valuable. Changes in transaction frequency, spending categories, or account usage patterns often precede churn events. By modeling these dynamics explicitly, Transformer-based architectures can identify early churn signals more reliably than static models, enabling proactive retention actions [98].

Synthetic Data Generation and Privacy-Preserving Analytics Privacy and regulatory constraints severely limit the sharing and reuse of real transactional data, both within and across financial institutions [11]. Synthetic data generation therefore represents a critical enabling technology for collaborative research, model development, and scenario analysis.

The generative models introduced in this work—both autoregressive and diffusion-based—enable the creation of realistic synthetic transactional time series that preserve key statistical and temporal properties of real data while mitigating privacy risks [114, 58]. These synthetic datasets can be used for

model training, stress testing, and benchmarking, as well as for sharing data with external partners or research institutions without exposing sensitive customer information.

Diffusion-based models, in particular, provide strong advantages in terms of diversity and unconditional generation, making them suitable for simulating a wide range of plausible customer behaviors and economic scenarios [64]. This opens the door to applications such as what-if analyses, synthetic population modeling, and robustness testing of financial systems under hypothetical conditions.

Toward Foundation Models for Financial Transactions Taken together, these applications illustrate how the proposed framework contributes to the emergence of foundation models for financial transactional data. By unifying representation learning, large-scale self-supervised training, and generative modeling within a single architectural paradigm, the methods presented in this thesis enable a shift from task-specific, feature-engineered pipelines to more general, reusable, and scalable solutions [19].

Under the definition of foundation models adopted in this thesis (Section 1.1), the presented architectures demonstrate the feasibility of foundation-style modeling for transactional data. Such models have the potential to become core infrastructure components within financial institutions, supporting a wide range of analytical tasks while continuously improving as more data become available. At the same time, their deployment raises important questions related to limitations, ethical considerations, and future research directions, which are discussed in the following sections.

## 9.2 Limitations of Current Approaches

Despite the promising results and broad applicability of the Transformer-based frameworks presented in this thesis, several limitations remain [53]. These limitations are not specific to a single model, but rather reflect more general challenges associated with modeling heterogeneous, time-dependent tabular data at scale,

particularly in sensitive and highly regulated domains such as finance [71]. A critical discussion of these aspects is essential to properly contextualize the contributions of this work and to identify directions for further improvement.

Computational Cost and Scalability One of the primary limitations of large Transformer-based models is their computational cost [109]. Although hierarchical architectures such as UniTTab significantly reduce the effective sequence length processed by the sequence-level Transformer, the overall training and inference pipelines remain computationally intensive. Pre-training on millions of transaction sequences requires substantial GPU resources, long training times, and careful engineering of data pipelines [19].

This cost can limit the accessibility of such models, particularly for smaller institutions or research groups without access to large-scale computational infrastructure. While tree-based models and simpler neural architectures may underperform in terms of expressiveness, they remain attractive in production environments due to their lower computational footprint and faster deployment cycles [23, 47]. Bridging this gap between performance and efficiency remains an open challenge.

Model Interpretability and Explainability Interpretability is a longstanding concern in financial applications, where regulatory requirements and risk management practices demand transparent and explainable decision-making processes [37]. Transformer-based models, despite their strong empirical performance, are often perceived as black boxes, especially when compared to rule-based systems or tree-based models that provide more explicit decision logic [71].

Although attention mechanisms offer some degree of interpretability, their outputs are not always straightforward to translate into actionable explanations for end users or regulators [112]. Moreover, the hierarchical and multi-component nature of the architectures proposed in this thesis further complicates interpretability, as decisions emerge from interactions across multiple layers and time steps.

As a result, the adoption of these models in high-stakes financial settings

may require complementary explainability techniques, post-hoc analysis tools, or hybrid modeling strategies that combine deep representations with more interpretable decision layers [94].

Data Quality, Bias, and Representativeness The effectiveness of large-scale self-supervised learning critically depends on the quality and representativeness of the training data [19]. Transactional datasets, even when large, may reflect historical biases, incomplete coverage of certain customer segments, or institution-specific behaviors that limit generalization.

For instance, customers with limited transaction histories or atypical usage patterns may be underrepresented in the data, leading to less reliable representations and predictions for these groups. Similarly, changes in economic conditions, regulatory frameworks, or customer behavior over time may introduce distribution shifts that degrade model performance if not properly addressed [89].

While pre-training on large datasets improves robustness, it does not eliminate the need for continuous monitoring, periodic retraining, and careful validation across different populations and time periods.

Limitations of Generative Models Although the generative models introduced in this thesis represent a significant advance in synthetic transactional data generation, they also exhibit inherent limitations [114]. Autoregressive models, such as UniTTab-AR, tend to accumulate errors over long generation horizons and may exhibit reduced diversity in unconditional generation scenarios. Diffusion-based models, while more robust in terms of diversity, introduce additional complexity in training and inference and require careful tuning of diffusion schedules and architectural parameters [58, 64].

Moreover, evaluating the quality of generated tabular time series remains an open problem [11]. Unlike images or text, where perceptual quality can be assessed qualitatively, synthetic transactional data must be evaluated through indirect metrics, statistical tests, or downstream task performance. These evaluation procedures are often task-dependent and may fail to capture subtle but important discrepancies between real and generated data.

Dependence on Domain-Specific Design Choices Finally, despite the goal of generality, several components of the proposed framework rely on domain-specific design choices [52]. Examples include the representation of timestamps, the handling of transaction types with different schemas, and the discretization or encoding strategies adopted for numerical values. While these choices are well motivated for financial data, transferring the models to other domains may require careful adaptation and re-engineering.

This highlights a broader tension between model generality and domain specificity: achieving strong performance on complex real-world data often necessitates tailored architectural and representational decisions, which may limit out-of-the-box applicability to new domains.

In summary, while the approaches presented in this thesis demonstrate strong potential for modeling heterogeneous tabular time series, their practical deployment must account for computational constraints, interpretability requirements, data quality issues, and the intrinsic challenges of generative modeling. Addressing these limitations is a key motivation for the ethical considerations and future research directions discussed in the following sections.

## 9.3 Ethical and Privacy Considerations

The application of advanced Deep Learning models to financial transactional data raises important ethical and privacy considerations that must be carefully addressed [11]. Transactional datasets contain highly sensitive information about individuals' behaviors, preferences, and economic conditions, and improper use or disclosure of such data can lead to significant harm, including discrimination, financial loss, and violations of fundamental privacy rights [43].

One of the primary ethical concerns relates to data privacy and confidentiality. Even when explicit identifiers are removed, transactional data may remain re-identifiable through linkage attacks or inference based on unique behavioral patterns [80]. As a result, strict controls on data access, storage, and processing are required throughout the model development lifecycle. The use of large-scale self-supervised learning amplifies these concerns, as models are trained on vast

amounts of raw data that may encode sensitive information implicitly [19].

Synthetic data generation represents a promising mitigation strategy, but it also introduces new risks. While synthetic datasets are often perceived as privacy-preserving by design, generative models may inadvertently memorize and reproduce rare or unique patterns from the training data, potentially leading to information leakage [25]. Ensuring that synthetic transactional data do not expose sensitive information therefore requires careful model design, regularization, and rigorous evaluation of privacy guarantees [114, 11].

Another critical ethical aspect concerns fairness and bias. Transactional data reflect historical behaviors and institutional practices, which may embed societal biases related to income, geography, gender, or access to financial services [15]. Models trained on such data risk perpetuating or amplifying these biases, particularly when deployed in high-stakes decision-making processes such as credit approval, fraud investigation, or customer segmentation [78]. Addressing these risks requires explicit fairness assessments, bias monitoring, and, where appropriate, the incorporation of fairness-aware learning objectives.

Transparency and accountability also play a central role in the ethical deployment of Deep Learning models in finance. Regulatory frameworks increasingly require institutions to provide explanations for automated decisions that affect individuals [43]. While Transformer-based architectures offer strong predictive performance, their complexity poses challenges for explainability and human oversight [37]. Ethical deployment therefore necessitates complementary tools and processes that enable model inspection, auditability, and meaningful human intervention when needed.

Finally, the long-term societal impact of large foundation models for financial data must be considered. As these models become more widely adopted, they may influence market dynamics, customer behavior, and institutional practices in ways that are difficult to anticipate [19]. Responsible research and deployment require ongoing evaluation of these broader effects, as well as adherence to principles of responsible AI, including robustness, transparency, and respect for individual rights.

In summary, while the methods proposed in this thesis offer powerful tools

for modeling and generating transactional data, their use must be guided by careful consideration of ethical and privacy implications. Addressing these concerns is not only a technical challenge but also an organizational and societal responsibility, and it is essential for ensuring that advances in Deep Learning for finance are aligned with legal requirements and ethical standards.

## 9.4 Future Research Directions

The work presented in this thesis opens several promising directions for future research in the modeling of heterogeneous, time-dependent tabular data. While the proposed architectures and methodologies address many of the limitations of existing approaches, they also highlight new challenges and opportunities that merit further investigation [19].

A first important direction concerns scalability and efficiency. Although Transformer-based architectures have demonstrated strong performance when trained at scale, their computational and memory requirements remain substantial [109]. Future research could explore more efficient attention mechanisms, sparse or low-rank approximations, and hybrid architectures that balance expressiveness and efficiency, enabling the deployment of large pretrained models in resource-constrained environments [30]. Advances in this direction would be particularly valuable for extending foundation models for tabular data beyond large institutions with access to extensive computational infrastructure.

A second research avenue involves improving interpretability and explainability. While representation learning and generative modeling benefit from expressive latent spaces, understanding how these representations encode behavioral patterns and drive predictions remains an open problem [37]. Developing interpretability techniques specifically tailored to heterogeneous tabular time series—such as temporally aware attribution methods or feature-level explanation frameworks—could significantly enhance trust, transparency, and regulatory compliance in high-stakes applications [94].

Another promising direction relates to robustness and distribution shift. Transactional data are subject to non-stationarity driven by changes in economic

conditions, user behavior, regulatory environments, and technological adoption [89]. Future work could investigate continual learning, domain adaptation, and online learning strategies that allow models to adapt to evolving data distributions without catastrophic forgetting. This is particularly relevant for long-lived foundation models intended to operate over extended time horizons.

Generative modeling also offers substantial opportunities for further research. While diffusion-based models address several limitations of autoregressive generation, their application to heterogeneous tabular time series is still in its early stages [58, 64]. Future work could explore improved latent representations, hybrid generative paradigms, and conditional generation mechanisms that allow finer control over generated sequences. Additionally, integrating formal privacy guarantees—such as differential privacy—into generative training pipelines remains an open and important challenge [38].

A further research direction concerns cross-domain generalization. Although this thesis focuses primarily on financial transactional data, many other domains—such as healthcare, mobility, and industrial monitoring—share similar characteristics, including heterogeneity, temporal structure, and privacy sensitivity [98]. Extending and validating the proposed models across these domains would help clarify which architectural components are truly domain-agnostic and which require domain-specific adaptation.

Finally, the emergence of foundation models for structured temporal data raises broader questions about responsible deployment and governance. As such models become more widely adopted, future research must consider not only technical performance but also societal impact, ethical alignment, and long-term sustainability [19]. Addressing these questions will require interdisciplinary collaboration, combining advances in machine learning with insights from law, economics, and the social sciences.

In summary, while this thesis establishes a unified framework for representation learning and generative modeling of heterogeneous tabular time series, it also points toward a rich and evolving research landscape. Continued progress in this area has the potential to fundamentally reshape how structured temporal data are analyzed, shared, and leveraged across a wide range of applications.

### 9.4.1 Future Directions on Explainability, Robustness, and Fairness

A promising and still largely unexplored direction concerns the development of methodologies that enhance the trustworthiness, transparency, and ethical reliability of Transformer-based models applied to heterogeneous tabular time series.

On the discriminative side, future research should investigate explainability and robustness, two aspects that become particularly relevant when deploying large-scale models in high-stakes financial environments. Although the unified representation framework introduced in this thesis effectively captures complex temporal and cross-feature relationships, current methods offer limited insight into why a certain prediction is made or how sensitive the model is to perturbations in the input sequence. Understanding which transactions, temporal segments, or feature interactions are most influential for a given decision would enable practitioners to interpret model outputs more reliably, facilitate regulatory compliance, and support domain experts in diagnosing unexpected behaviours. Likewise, robustness analyses—such as evaluating the stability of predictions under noisy inputs, missing events, adversarial manipulations, or distributional shifts—would provide a principled way to assess the resilience of Transformer-based architectures in realistic operational settings, where data irregularities are common and potentially harmful if unaccounted for.

Conversely, in the context of generative modeling, a natural extension of the present work relates to the study of fairness in synthetic data generation. As shown by the experiments on autoregressive and diffusion-based models, generative architectures are capable of producing realistic transactional sequences that preserve complex statistical and temporal patterns. However, the relationship between generation quality and demographic or behavioural subgroups remains underexplored. Future investigations should examine whether generative models inadvertently reproduce or amplify biases present in historical datasets, and whether fairness constraints or debiasing mechanisms can be integrated into autoregressive and diffusion-based pipelines. Ensuring equitable treatment across user groups is particularly critical when synthetic data are used for downstream

tasks such as credit scoring, risk modelling, or customer analytics, where biased synthetic samples may propagate undesirable disparities.

Overall, integrating explainability, robustness, and fairness into the next generation of Transformer-based models would contribute to a more holistic view of reliability in both discriminative and generative settings. These research directions complement the technical contributions of this thesis and represent essential steps toward the responsible adoption of large-scale foundation models in domains—such as finance—where trust, transparency, and equitable decision-making are indispensable.

# 10 Conclusions

## 10.1 Summary of Contributions

This thesis has addressed the problem of modeling heterogeneous, time-dependent tabular data through a unified Deep Learning framework based on Transformer architectures. Motivated by the limited impact of modern representation learning techniques on structured temporal data—despite their ubiquity in real-world domains such as finance—the work set out to bridge the gap between recent advances in foundation models and the specific challenges posed by transactional time series.

The first major contribution of this thesis is the formulation of a unified representation learning framework for heterogeneous tabular time series. By explicitly modeling numerical and categorical attributes within a shared embedding space and by leveraging hierarchical Transformer architectures, the proposed UniTTab model overcomes key limitations of prior approaches that relied on discretization, handcrafted feature engineering, or static representations. Through self-supervised masked token pre-training, UniTTab learns rich temporal representations directly from raw transactional data, capturing both intra-transaction structure and inter-transaction dependencies.

A second contribution lies in demonstrating that Transformer-based models can be effectively trained at scale on real-world financial transactional datasets. By exploiting large volumes of unlabeled data, this thesis establishes, to the best of our knowledge, the first publicly documented foundation-style model for transactional data, showing that large-scale pre-training leads to consistent performance improvements across a variety of downstream financial tasks, including fraud detection, credit risk assessment, churn prediction, and customer profiling. These results highlight the importance of scale and self-supervision in structured temporal domains, mirroring trends previously observed in unstructured data.

The thesis further extends the representation learning framework to the task of generative modeling. Two complementary approaches are explored. UniTTab-AR introduces an autoregressive Transformer architecture capable of generating temporally coherent transactional sequences while preserving internal feature consistency. Building on this, diffusion-based models—BankDiT and the more general TabDiT—are proposed to address the limitations of autoregressive generation, particularly in terms of diversity and unconditional sampling. These models demonstrate that diffusion paradigms, when combined with Transformers and appropriate representations for heterogeneous data, can achieve state-of-the-art results in tabular time series generation.

Another important contribution is the explicit focus on privacy-preserving data generation. By enabling the synthesis of realistic yet non-identifiable transactional data, the generative models proposed in this work provide practical tools for data sharing, augmentation, and experimentation in privacy-sensitive environments. This aspect is particularly relevant for finance, where regulatory constraints severely limit the availability of open datasets and hinder collaborative research.

Finally, the thesis contributes to the broader research agenda of foundation models for structured temporal data. By unifying representation learning, large-scale pre-training, and generative modeling within a single architectural paradigm, the work demonstrates that many of the principles underlying recent successes in Natural Language Processing and Computer Vision can

be adapted—albeit with significant modifications—to heterogeneous tabular time series. This opens the door to more general, reusable models that can be transferred across tasks and domains.

## 10.2 Final Remarks

The rapid evolution of Deep Learning over the past decade has fundamentally reshaped the way complex data are modeled and understood. While much of this progress has been driven by unstructured domains such as text and images, structured temporal data—despite their central role in many scientific and industrial applications—have remained comparatively underexplored. This thesis has argued that this gap is not due to a lack of relevance, but rather to the intrinsic challenges posed by heterogeneity, temporal structure, and data accessibility.

By focusing on transactional data as a representative and demanding case study, this work has shown that many of the conceptual tools underlying modern foundation models can be successfully adapted to structured time-dependent data. Transformer architectures, when carefully redesigned to respect the semantics of tabular features and temporal dependencies, provide a powerful and flexible modeling paradigm. Self-supervised learning emerges as a key enabler, allowing models to leverage vast amounts of unlabeled data and to learn representations that generalize across tasks.

Beyond predictive performance, this thesis emphasizes the importance of generative modeling as a first-class component of structured data analytics. The ability to generate realistic, temporally coherent, and privacy-preserving synthetic data is not merely a technical achievement, but a practical necessity in domains where data sharing is constrained. In this respect, the diffusion-based approaches explored in this work represent a significant step forward, offering improved diversity, robustness, and flexibility compared to earlier generative models.

At the same time, the results presented here highlight that technical advances alone are not sufficient. The deployment of large-scale models in finance requires careful consideration of interpretability, fairness, privacy, and governance. These aspects must be treated as integral components of model design rather than as af-

terthoughts. As Artificial Intelligence systems increasingly influence high-stakes financial decisions, maintaining human oversight and institutional accountability will remain essential.

In conclusion, this thesis contributes to the growing body of research that seeks to extend the reach of Deep Learning beyond unstructured data, toward the rich and complex world of structured temporal information. By unifying representation learning and generative modeling for heterogeneous tabular time series, it lays the foundation for future work on general-purpose models in finance and related domains. It is the author's hope that these contributions will not only advance academic research, but also support the development of more robust, responsible, and data-efficient AI systems in real-world applications.

## 10.3 Ph.D. Activities

This final section presents a list of the main activities carried out by the candidate during the Ph.D. program in Information and Communication Technologies (ICT).

### Conference Attendances

**April 2025**: ICLR 2025, Poster presentation, Singapore;

**November 2024**: ICAIF 2024, Workshop presentation, New York, USA;

**May 2024**: Ital-IA 2024, Paper presentation, Napoli, Italy;

**November 2023**: ICAIF 2023, Attended as a visitor, New York, USA;

**May 2023**: Ital-IA 2023, Paper presentation, Pisa, Italy.

### Seminars and Workshops

**June 2025:** AI & ML for Systematic Investing, Speaker: Prof. Petter Kolm;

**January 2025:** Responsibility in the Times of Generative AI, Speaker: Dr. Cristian Canton Ferrer;

**January 2025:** YOUZ AI – FairER: Artificial Intelligence and the Future of Young People in Work and Society, Speaker: Prof. Rita Cucchiara;

**September 2024:** Training – Generative AI: An Overview on Regulation, Speaker: Elena Ferretti;

**June 2024:** Deep Learning on Bank Transactions, Prometeia's Foundation and Generative Model, Speaker: Valerio Consorti;

**June 2024:** Training – Generative AI: LLM Validation, Speaker: Irem Demirtas;

**May 2024:** Training GenAI: Generative AI – 101 for Developers, Speaker: Lorenzo Balzani;

**May 2024:** GenAI Working Group: Presentation and Demo of the Intelligent Document Assistant (IDA), Speaker: Federico Casciari;

**April 2024:** Cantiere GenAI: Generative AI – 101 for Business Users, Speaker: Michele Filannino;

**March 2024:** The Impact of AI on the Labour Market in OECD Countries, Speaker: Alexandre Georgieff;

**January 2024:** Quant Tales from the Prom Side: AI-Powered Option Pricing Model Calibration, Speaker: Nicola Meccheri;

**October 2023:** ONNX and Mobile Models, Speaker: Simone Luetto;

**September 2023:** NeRF and 3D Models, Speaker: Davide Di Nucci;

**July 2023:** AI for CLO, Speakers: Matteo Cataldo and Alessandro Benetti;

**May 2023:** Experimental Machine Learning Methods for Market Abuse Detection, Speaker: Prof. Francesca Medda;

**December 2022:** 3D Computer Vision for Animals, Speaker: Prof. Silvia Zuffi;

**December 2022:** From Handcrafted to End-to-End Learning, and Back: A Journey for Multi-Object Tracking, Speaker: Prof. Laura Leal-Taixé;

**November 2022:** Graph Signal Processing for Machine Learning: Challenges and Use-Cases, Speaker: Prof. Laura Toni;

**November 2022:** Digital Humanities and Artificial Intelligence for Humans in Today's Society, Speaker: Prof. Rita Cucchiara;

### Summer Schools

**July 2024**: International School on Deep Learning and the Future of Artificial Intelligence (DeepLearn), University of Maia, Porto, Portugal;

**September 2023**: ELLIS Summer School on Large-Scale AI for Research and Industry, Modena, Italy.

# List of Publications

[1] Garuti, F., Luetto, S., Cucchiara, R., and Sangineto, E. (2023). Deep learning and large scale models for bank transactions. In *Proceedings of the Italia Intelligenza Artificiale - Thematic Workshops co-located with the 3rd CINI National Lab AIIS Conference on Artificial Intelligence (Ital-IA 2023)*, volume 3486 of *CEUR Workshop Proceedings*, pages 512–516, Pisa, Italy. CEUR-WS.org. Presented at Ital-IA 2023, May 29-30, 2023.

[2] Garuti, F., Luetto, S., Sangineto, E., and Cucchiara, R. (2024a). Large-scale transformer models for transactional data. In *Proceedings of the Ital-IA Intelligenza Artificiale - Thematic Workshops (Ital-IA 2024)*, volume 3762 of *CEUR Workshop Proceedings*, pages 242–247, Napoli, Italy. CEUR-WS.org.

[3] Garuti, F., Luetto, S., Sangineto, E., Forni, L., and Cucchiara, R. (2024b). Diffusion and autoregressive deep learning models for transactional data generation. In *3rd Workshop on Synthetic Data and GenAI in Finance (ICAIF-24)*, New York, USA. Presented at the 5th ACM International Conference on AI in Finance (ICAIF-24).

[4] Garuti, F., Sangineto, E., Luetto, S., Forni, L., and Cucchiara, R. (2025). Diffusion transformers for tabular data time series generation. In Yue, Y., Garg, A., Peng, N., Sha, F., and Yu, R., editors, *International Conference on Representation Learning*, volume 2025, pages 93127–93152.

[5] Luetto, S., Garuti, F., Sangineto, E., Forni, L., and Cucchiara, R. (2025). One transformer for all time series: representing and training with time-dependent heterogeneous tabular data. *Machine Learning*, 114(6):145.

# Bibliography

[6] Abay, N., Zhou, Y., et al. (2018). Privacy preserving synthetic data release using deep learning. *Machine Learning and Knowledge Discovery in Databases*.

[7] Alain, G. and Bengio, Y. (2014). What regularized auto-encoders learn from the data generating distribution. *Journal of Machine Learning Research*, 15:3563–3593.

[8] Altman, E. R. (2019). Synthesizing Credit Card Transactions.

[9] Arik, S. and Pfister, T. (2021). Tabnet: Attentive interpretable tabular learning. *Proceedings of AAAI*, 35(8):6679–6687.

[10] Arjovsky, M., Chintala, S., and Bottou, L. (2017). Wasserstein gan. *Proceedings of the 34th International Conference on Machine Learning*.

[11] Arora, S. and Zhang, Y. (2022). Synthetic data generation for tabular data: A survey. *arXiv preprint arXiv:2208.03231*.

[12] Assefa, S., Dervovic, D., Mahfouz, M., Balch, T., Reddy, P., and Veloso, M. (2021). Generating synthetic data in finance: opportunities, challenges and pitfalls. *Workshop on AI in Finance Neurips*.

[13] Bai, S., Kolter, J. Z., and Koltun, V. (2018). An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *arXiv preprint arXiv:1803.01271*.

[14] Bao, H., Dong, L., and Wei, F. (2022). BEiT: BERT pre-training of image transformers. *ICLR*.

[15] Barocas, S. and Selbst, A. D. (2016). Big data's disparate impact. *California Law Review*, 104(3):671–732.

[16] Bengio, Y., Courville, A., and Vincent, P. (2013). Representation learning: A review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8):1798–1828.

[17] Bengio, Y., Ducharme, R., Vincent, P., and Jauvin, C. (2003). A neural probabilistic language model. In *Journal of Machine Learning Research*, volume 3, pages 1137–1155.

[18] Berka, P. (1999). Workshop notes on Discovery Challenge PKDD'99. https://sorry.vse.cz/~berka/challenge/pkdd1999/berka.htm.

[19] Bommasani, R. et al. (2021). On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258*.

[20] Borisov, V. et al. (2022). Deep learning for transactional data. *ACM Computing Surveys*.

[21] Borisov, V. e. (2021). Deep neural networks and tabular data: A survey. *IEEE transactions on neural networks and learning systems*.

[22] Box, G. E. P., Jenkins, G. M., and Reinsel, G. C. (2015). *Time Series Analysis: Forecasting and Control*. Wiley.

[23] Breiman, L. (2001). Random forests. *Machine Learning*, 45(1):5–32.

[24] Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., and Amodei, D. (2020). Language Models are Few-Shot Learners. *arXiv:2005.14165*.

[25] Carlini, N. et al. (2023). Extracting training data from diffusion models. *USENIX Security Symposium*.

[26] Chen, T. . (2016). Xgboost: A scalable tree boosting system. *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794.

[27] Chen, T. et al. (2022). Large-scale deep learning systems. *Proceedings of Machine Learning Research*.

[28] Chen, Z. et al. (2019). Numeracy-600k: Learning numerical reasoning over text. *arXiv preprint arXiv:1908.08947*.

[29] Cheng, H.-T., Koc, L., Harmsen, J., et al. (2016). Wide & deep learning for recommender systems. In *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems*.

[30] Child, R. et al. (2019). Generating long sequences with sparse transformers. *arXiv preprint arXiv:1904.10509*.

[31] Christ, M., Braun, N., Neuffer, J., and Kempa-Liehr, A. W. (2018). Time series feature extraction on basis of scalable hypothesis tests (tsfresh - a python package). *Neurocomputing*, 307:72–77.

[32] Dal Pozzolo, A. and Bontempi, G. (2015). Credit scoring and the need for explainable models. *IEEE Intelligent Systems*, 30(3):69–75.

[33] Dal Pozzolo, A., Bontempi, G., and Snoeck, M. (2017). Adversarial drift detection for credit card transactions. *IEEE Computational Intelligence Magazine*, 12(4):15–29.

[34] Dal Pozzolo, A. et al. (2015). Calibrating probability with undersampling for unbalanced classification. *IEEE Symposium Series on Computational Intelligence*.

[35] Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2019). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *NAACL,*.

[36] Dhariwal, P. and Nichol, A. (2021). Diffusion models beat gans on image synthesis. *Advances in Neural Information Processing Systems*.

[37] Doshi-Velez, F. and Kim, B. (2017). Towards a rigorous science of interpretable machine learning. *arXiv preprint arXiv:1702.08608*.

[38] Dwork, C. and Roth, A. (2014). *The Algorithmic Foundations of Differential Privacy*. Now Publishers.

[39] Elman, J. L. (1990). Finding structure in time. *Cognitive Science*, 14(2):179–211.

[40] Erhan, D., Courville, A., Bengio, Y., and Vincent, P. (2010). Why does unsupervised pre-training help deep learning? *Journal of Machine Learning Research*, 11:625–660.

[41] Esteban, C., Hyland, S. L., and Rätsch, G. (2017). Real-valued (medical) time series generation with recurrent conditional gans. *arXiv preprint arXiv:1706.02633*.

[42] et al., A. H. (2020). On the degeneration of neural text generation. *ICLR*.

[43] European Parliament and Council of the European Union (2016). General data protection regulation. *Official Journal of the European Union*.

[44] Fan, A., Lewis, M., and Dauphin, Y. (2018). Hierarchical neural story generation. *ACL*.

[45] Fawcett, T. (1997). Analysis and visualization of categorical time series. *Proceedings of KDD*.

[46] Freund, Y. and Schapire, R. (1997). A decision-theoretic generalization of on-line learning. In *European Conference on Computational Learning Theory*, pages 23–37.

[47] Friedman, J. (2001). Greedy function approximation: A gradient boosting machine. *Annals of Statistics*.

[48] Friedman, J., Hastie, T., and Tibshirani, R. (2001). *The Elements of Statistical Learning*. Springer.

[49] Fursov, I., Morozov, M., Kaploukhaya, N., Kovtun, E., Rivera-Castro, R., Gusev, G., Babaev, D., Kireev, I., Zaytsev, A., and Burnaev, E. (2021). Adversarial attacks on deep models for financial transaction records. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining (KDD)*.

[50] Garuti, F., Sangineto, E., Luetto, S., Forni, L., and Cucchiara, R. (2024). Diffusion transformers for tabular data time series generation. *arXiv*.

[51] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative adversarial nets. In *NeurIPS*.

[52] Gorishniy, Y., Rubachev, I., Khrulkov, V., and Babenko, A. (2021). Revisiting deep learning models for tabular data. In *Advances in Neural Information Processing Systems (NeurIPS)*.

[53] Grinsztajn, L. et al. (2022). Why do tree-based models still outperform deep learning on tabular data? *Advances in Neural Information Processing Systems*.

[54] Guo, H., Tang, R., Ye, Y., and Li, Z. (2017). Deepfm: A factorization-machine based neural network for ctr prediction. *IJCAI*.

[55] Han, H., Xu, J., Zhou, M., Shao, Y., Han, S., and Zhang, D. (2022). LUNA: language understanding with number augmentations on transformers via number plugins and pre-training. *arXiv:2212.02691*.

[56] He, K. et al. (2020). Momentum contrast for unsupervised visual representation learning. *Proceedings of CVPR*.

[57] Herzig, J. et al. (2020). Tapas: Weakly supervised table parsing via pre-training. *Proceedings of ACL*.

[58] Ho, J., Jain, A., and Abbeel, P. (2020). Denoising diffusion probabilistic models. In *Advances in Neural Information Processing Systems (NeurIPS)*.

[59] Ho, J. and Salimans, T. (2022). Classifier-free diffusion guidance. *arXiv preprint arXiv:2207.12598*.

[60] Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*.

[61] Holtzman, A., Buys, J., Du, L., Forbes, M., and Choi, Y. (2020). The curious case of neural text degeneration. *arXiv preprint arXiv:1904.09751*.

[62] Huang, X. et al. (2020). Tabtransformer: Tabular data modeling using contextual embeddings. *arXiv preprint arXiv:2012.06678*.

[63] Jordon, J., Yoon, J., and van der Schaar, M. (2019). Pate-gan: Generating synthetic data with differential privacy guarantees. In *International Conference on Learning Representations*. Often cited as 2018 preprint; final version at ICLR 2019.

[64] Karras, T. et al. (2022). Elucidating the design space of diffusion-based generative models. *Advances in Neural Information Processing Systems*.

[65] Ke, G. et al. (2017). Lightgbm: A highly efficient gradient boosting decision tree. *Advances in Neural Information Processing Systems*.

[66] Kingma, D. P., Salimans, T., Poole, B., and Ho, J. (2021). Variational diffusion models. In *Advances in Neural Information Processing Systems (NeurIPS)*.

[67] Kingma, D. P. and Welling, M. (2014). Auto-encoding variational bayes. *International Conference on Learning Representations (ICLR)*.

[68] Kontschieder, P., Fiterau, M., Criminisi, A., and Rota Bulò, S. (2015). Deep neural decision forests. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*.

[69] Kotelnikov, A., Baranchuk, D., Rubachev, I., and Babenko, A. (2023). TabDDPM: modelling tabular data with diffusion models. In *ICML*.

[70] LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *nature*, 521(7553):436–444.

[71] Lessmann, S. et al. (2015). Benchmarking state-of-the-art classification algorithms for credit scoring. *European Journal of Operational Research*.

[72] Li, S., Jin, X., Xuan, Y., Zhou, X., Chen, W., Wang, Y.-X., and Yan, X. (2019). Enhancing the locality and breaking the memory bottleneck of transformer on time series forecasting. *Advances in Neural Information Processing Systems*, 32. Often cited as 2020 due to arXiv/early versions.

[73] Liang, X., Zou, T., Guo, B., Li, S., Zhang, H., Zhang, S., Huang, H., and Chen, S. (2015). Assessing Beijing's PM2.5 pollution: severity, weather impact, APEC and winter heating. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 471(2182).

[74] Liu, T., Qian, Z., Berrevoets, J., and van der Schaar, M. (2023). GOGGLE: Generative modelling for tabular data by learning relational structure. In *ICLR*.

[75] Lu, Z., Wang, Z., Huang, D., Wu, C., Liu, X., Ouyang, W., and Bai, L. (2024). FiT: Flexible Vision Transformer for Diffusion Model. *arXiv:2402.12376*.

[76] Luetto, S., Garuti, F., Sangineto, E., Forni, L., and Cucchiara, R. (2023). One Transformer for All Time Series: Representing and Training with Time-Dependent Heterogeneous Tabular Data.

[77] Lütkepohl, H. (2005). New Introduction to Multiple Time Series Analysis. *Springer Books*.

[78] Mehrabi, N., Morstatter, F., Saxena, N., Lerman, K., and Galstyan, A. (2021). A survey on bias and fairness in machine learning. *ACM Computing Surveys*, 54(6).

[79] Mildenhall, B., Srinivasan, P. P., Tancik, M., Barron, J. T., Ramamoorthi, R., and Ng, R. (2020). NeRF: representing scenes as neural radiance fields for view synthesis. In *ECCV*.

[80] Narayanan, A. and Shmatikov, V. (2008). Robust de-anonymization of large sparse datasets. *IEEE Symposium on Security and Privacy*.

[81] Nichol, A. Q. and Dhariwal, P. (2021). Improved denoising diffusion probabilistic models. In *International Conference on Machine Learning (ICML)*.

[82] Nowozin, S., Cseke, B., and Tomioka, R. (2016). f-gan: Training generative neural samplers using variational divergence minimization. In *Advances in Neural Information Processing Systems (NeurIPS)*.

[83] Padhi, I., Schiff, Y., Melnyk, I., Rigotti, M., Mroueh, Y., Dognin, P. L., Ross, J., Nair, R., and Altman, E. (2021). Tabular transformers for modeling multivariate time series. In *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP*.

[84] Patki, N., Wedge, R., and Veeramachaneni, K. (2016). The synthetic data vault. In *IEEE International Conference on Data Science and Advanced Analytics (DSAA)*.

[85] Peebles, W. and Xie, S. (2023). Scalable diffusion models with transformers. In *International Conference on Computer Vision (ICCV)*.

[86] Pineau, J., Vincent-Lamarre, P., Sinha, K., Larivière, V., Beygelzimer, A., d'Alché Buc, F., Fox, E., and Larochelle, H. (2021). Improving reproducibility in machine learning research. *Journal of Machine Learning Research*, 22(164):1–20.

[87] Prokhorenkova, L. O., Gusev, G., Vorobev, A., Dorogush, A. V., and Gulin, A. (2018). Catboost: unbiased boosting with categorical features. In *Advances in Neural Information Processing Systems (NeurIPS)*.

[88] Qin, Y., Song, D., Chen, H., et al. (2017). A dual-stage attention-based recurrent neural network for time series prediction. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence (IJCAI)*.

[89] Quionero-Candela, J. et al. (2009). Dataset shift in machine learning. *MIT Press*.

[90] Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., Krueger, G., and Sutskever, I. (2021). Learning Transferable Visual Models From Natural Language Supervision.

[91] Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., and Liu, P. J. (2020). Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67.

[92] Ramesh, A., Pavlov, M., Goh, G., Gray, S., Voss, C., Radford, A., Chen, M., and Sutskever, I. (2021). Zero-shot text-to-image generation. In *ICML*.

[93] Rasul, K., Seward, C., Schuster, I., and Vollgraf, R. (2021). Autoregressive denoising diffusion models for multivariate probabilistic time series forecasting. *arXiv preprint arXiv:2110.08049*.

[94] Ribeiro, M. T. et al. (2016). "why should i trust you?": Explaining the predictions of any classifier. *Proceedings of KDD*.

[95] Rombach, R., Blattmann, A., Lorenz, D., Esser, P., and Ommer, B. (2022). High-resolution image synthesis with latent diffusion models. In *CVPR*.

[96] Shankaranarayana, S. M. and Runje, D. (2021). Attention augmented convolutional transformer for tabular time-series. In *2021 International Conference on Data Mining, ICDM 2021 - Workshops*.

[97] Shavitt, I. and Segal, E. (2018). Regularization learning networks: Deep learning for tabular datasets. *Advances in Neural Information Processing Systems*, 31.

[98] Shukla, S. and Marlin, B. (2021). Neural networks for time series forecasting. *Foundations and Trends in Machine Learning*.

[99] Shwartz-Ziv, R. and Tishby, N. (2017). Opening the black box of deep neural networks via information. *arXiv preprint arXiv:1703.00810*.

[100] Sohl-Dickstein, J., Weiss, E., Maheswaranathan, N., and Ganguli, S. (2015). Deep unsupervised learning using nonequilibrium thermodynamics. *Proceedings of ICML*.

[101] Solatorio, A. V. and Dupriez, O. (2023). REaLTabFormer: Generating realistic relational and tabular data using transformers. *arXiv:2302.02041*.

[102] Song, H. et al. (2019). Attentive neural processes. In *International Conference on Learning Representations (ICLR)*.

[103] Song, J., Meng, C., and Ermon, S. (2021a). Denoising diffusion implicit models. *arXiv preprint arXiv:2010.02502*.

[104] Song, Y., Sohl-Dickstein, J., Kingma, D. P., Kumar, A., Ermon, S., and Poole, B. (2021b). Score-based generative modeling through stochastic differential equations. *International Conference on Learning Representations (ICLR)*.

[105] Spithourakis, G. et al. (2021). Numerical reasoning in language models. In *Proceedings of EMNLP*.

[106] Stahlberg, F. (2019). Neural machine translation: A review. *Journal of Artificial Intelligence Research*.

[107] Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., and Wojna, Z. (2016). Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*.

[108] Theis, L., van den Oord, A., and Bethge, M. (2016). A note on the evaluation of generative models. *International Conference on Learning Representations*.

[109] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2017). Attention is All you Need. In *Advances in Neural Information Processing Systems (NeurIPS)*.

[110] Wallace, E. et al. (2019). Do nlp models know numbers? probing numeracy in embeddings. *Proceedings of EMNLP*.

[111] Wang, Z. et al. (2021). Generating synthetic tabular data using transformers. In *Proceedings of NeurIPS Workshops*.

[112] Wiegreffe, S. and Pinter, Y. (2019). Attention is not not explanation. *Proceedings of EMNLP*.

[113] Xie, Q., Dai, Z., Hovy, E., Luong, M.-T., and Le, Q. V. (2020). Self-supervised learning for time series analysis: Taxonomy, progress, and prospects. *arXiv preprint arXiv:2005.00000*. Survey-style reference commonly used for SSL principles.

[114] Xu, L. et al. (2019). Modeling tabular data using conditional gan. *Advances in Neural Information Processing Systems*.

[115] Xu, Z. (2020). Loan default prediction with Berka dataset. https://towardsdatascience.com/loan-default-prediction-an-end-to-end-ml-project-with-real-bank-data-part-1-1405f7aecb9e.

[116] Yang, Y. et al. (2022). Self-supervised learning for tabular data. *arXiv preprint arXiv:2207.08919*.

[117] Yang, Y., Morillo, D., and Hospedales, T. (2018). Deep neural decision trees. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*.

[118] Yin, P., Neubig, G., and Yih, W.-t. (2020). Tabert: Pretraining for joint understanding of textual and tabular data. In *Proceedings of ACL*, pages 841–851.

[119] Yoon, J., Jarrett, D., and van der Schaar, M. (2019). Time-series generative adversarial networks. In *Advances in Neural Information Processing Systems (NeurIPS)*.

[120] Zhang, H., Zhang, J., Shen, Z., Srinivasan, B., Qin, X., Faloutsos, C., Rangwala, H., and Karypis, G. (2024). Mixed-type tabular data synthesis with score-based diffusion in latent space. In *ICLR*.

[121] Zhou, J. et al. (2023). A survey on foundation models. *arXiv preprint arXiv:2302.06675*.