New Approaches to Synthetic Tabular Data Generation

Shengzhe Xu

Dissertation submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy
in
Computer Science & Application

Naren Ramakrishnan, Chair
Chang-Tien Lu
Danfeng Yao
Ruoxi Jia
Manish Marwah

May 30, 2025
Alexandria, Virginia

Keywords: Generative Neural Models, Tabular Data, Large Language Models.

New Approaches to Synthetic Tabular Data Generation

Shengzhe Xu

(ABSTRACT)

Synthetic data generation, while already becoming well-known as part of Generative AI (GenAI), has been primarily focused on images, voice, and text, which mostly have homogeneous data formats. This dissertation focuses on the modeling and generation of synthetic tables, which involve a range of characteristics: numerous variables, diverse attribute types, functional dependencies across columns, and temporal dependencies across rows. We aim to explore how to generate higher-quality synthetic tabular data through the following subproblems: (1) auto-regressive DNNs for synthetic table generation (STG), (2) large language models (LLMs) for adaptive STG with higher fidelity, (3) reducing in-context learning burden in STG via LLM priors, (4) embedding isotropy as a trust indicator for STG with LLMs, and (5) STG for next-generation wireless as a telecom application. Through Problems 1 and 2, we aim to improve the quality of generated synthetic tables; in Problem 3, we reduce the computational cost while maintaining quality; Problem 4 proposes a trust indicator for evaluating synthetic data quality by analyzing the isotropy of the model's internal embeddings; and Problem 5 demonstrates an application scenario in wireless telecommunications.

New Approaches to Synthetic Tabular Data Generation

Shengzhe Xu

(GENERAL AUDIENCE ABSTRACT)

Synthetic data generation, while already becoming well-known as part of Generative AI (GenAI), has been primarily focused on images, voice, and text, which mostly have homogeneous data formats. This dissertation focuses on the modeling and generation of synthetic tables, which involve a range of characteristics: numerous variables, diverse attribute types, functional dependencies across columns, and temporal dependencies across rows. We aim to explore how to generate higher-quality synthetic tabular data through the following subproblems: (1) auto-regressive DNNs for synthetic table generation (STG), (2) large language models (LLMs) for adaptive STG with higher fidelity, (3) reducing in-context learning burden in STG via LLM priors, (4) embedding isotropy as a trust indicator for STG with LLMs, and (5) STG for next-generation wireless as a telecom application. Through Problems 1 and 2, we aim to improve the quality of generated synthetic tables; in Problem 3, we reduce the computational cost while maintaining quality; Problem 4 proposes a trust indicator for evaluating synthetic data quality by analyzing the isotropy of the model's internal embeddings; and Problem 5 demonstrates an application scenario in wireless telecommunications.

Dedication

To my maternal Grandma, my Dad, and my Wife. And our cats: Hina, Kimi, and Subaru.

I am who I am, well loved by you.

Acknowledgments

I devoted some of my best years to pursuing this Ph.D. journey, which has been far from smooth sailing. Yet, I was also fortunate; there were always people who brought light into my life as we walked this road together. I would like to express my deepest gratitude to everyone who came into my life and stood by me through these challenging, yet invaluable, fighting days. Anyway, even though the journey took much longer than expected, I can now acknowledge it was still a meaningful experience. From my early days of competitive algorithm programming in primary school through my undergraduate years, to now having conducted academic research at the frontier of computer science, from both its bottom layer in compiler systems to its upper layer in artificial intelligence. Now, it is time to move on.

First and foremost, there is no doubt that without my advisor, Dr. Naren Ramakrishnan, I could not have completed this journey. To me, he is not only a respected scholar but also an educator and a true gentleman. I am deeply thankful for all that he has done for me over the years. He never gave up on his students, including me, even when the research didn't yield results, and even during my darkest times of depression and anxiety in the COVID-19 lockdown. Trying to do better, I hope that one day I'll become someone he can be proud of, and pass on everything I've learned from him to younger students.

I would also like to thank my research mentors: Dr. Manish Marwah, Dr. Nikhil Muralidhar, and Dr. Khoa D. Doan; my lab-mates Dr. Mandar Sharma, Raquib Bin Yousef, Amanda Cho-ting Lee, Dr. Andreea Sistrunk; our research faculty Dr. Patrick Butler, Nathan Self, and Brian Mayer; our collaborators Dr. Walid Saad, Dr. Christo K. Thomas, Dr. Rashed Shelim. We shared many precious and unforgettable memories, and together we wrote many interesting papers. I believe our story is still continuing.

Thank you to all the members of my dissertation committee. Dr. Chang-Tien Lu shared many valuable life experiences and personal wisdom with me. I cherish every time we had an opportunity for a deep conversation. Thank you to Dr. Danfeng Yao and Dr. Ruoxi Jia for encouraging me to stay on my research path and pursue impactful work with ambition, both in theory and in application. Dr. Manish Marwah, he has already given me the best possible guidance in this research area, and I hope I can continue to rely on his wisdom even after graduation. I am thankful for their insightful comments and critiques during my proposal exam and final defense. I also sincerely thank Dr. Cal Ribbens, Dr. Cliff Shaffer, and Dr. Eli Tilevich for their suggestions and support when I completely changed my research direction mid-way through the Ph.D. program. I deeply respect them as true educators.

A heartfelt thank you to Dr. Julie Kaplan and Dr. Joseph Frieben at the Cook Counseling Center. Over the past three years, you have played a crucial role in my life. Your patience and companionship helped me find a new mental life and healing. Thank you to Sharon Kinder-Potter, Laura Cooper, Klaudia Escobar Villatoro, Corinne Julien, and Wanawsha Shalaby for your always timely and dependable administrative support over the years.

Thank you to my old friends: Dr. Dake Chen, Liang Meng, Wang Xi, Chenxi Zhang, Wenting Zhao, Dr. Hang Hu, Dr. Xinwei Fu, Dr. Zhuang Wang, and Dr. Boyu Lyu. Having you makes me feel I am not alone, and that there is always a friend willing to lend a shoulder.

For encouraging me to make the bold decision to completely change my research direction in the summer of 2018, I want to thank my uncle and mentor, Yusong Huang, who first guided my journey into deep learning research, along with my mentors and friends at MSRA — Dr. Jiang Bian, Dr. Jia Zhang, Dr. Zichuan Lin, Peng Zhang, Lintao Lv, Zhiyuan Zhao, Yifan Zhou, Mingqi Huang, Tianyu Zhang, Yuanjing Shi, Lewen Wang, and Yiping Wang. I already miss you all. Thank you for showing me the bright and warm side of academia.

Thank you to my friends at the MMA/BJJ gym for being such a positive and warm environment. I am deeply grateful to Coach Zachary G. Queirolo for helping me regain my strength and well-being when I was at my weakest. Uncle Bill, thank you for always reminding me not to drop my defensive guard. Colin, it has been inspiring to see how quickly you have grown and improved, even though I can no longer keep up with you in an even match. Mike, Torey, and Joe, these folks are incredibly strong, but always kind and patient in teaching others. Richard, Jhon, and Eli, it's always nice to see you on Saturdays. Training with you all pushed me forward and reminded me that there are places where people are treated as human beings, where work isn't everything of life, and tomorrow is not the end of the world.

Finally, to my family: my maternal Grandma, my late paternal Grandpa, my Parents, and my Uncles and Aunts. The child you all once cared for has finally graduated.

Should You Be Trapped in a Windless Land

Fly, fly away,
Like a bird in the sky.
See the world on my behalf,
To the heaven may you fly.

Contents

List of Figures

List of Tables

Chapter 1

Introduction & Organization of the Dissertation

With the rise of large AI models, practitioners in various domains now aspire to leverage data to train their own machine learning models. However, obtaining authentic datasets presents various obstacles, even within the same organization or company. Some researchers aim to protect user privacy by anonymizing original data. Another school of thought aims to synthesize data using generative models, which is the focus of this dissertation.

Synthetic data generation, while a burgeoning topic in machine learning, has been primarily restricted to images and text. Our focus here is on tabular data generation, which have been traditionally approached using joint distribution modeling techniques such as GAN [140] or diffusion [146]-based autoregressive generators that rely on conditional distributions to enhance the generation of dependencies between columns or rows.

In this dissertation, we propose, describe, and evaluate methods to improve the generation of synthetic tabular data at each of the aforementioned stages. Specifically, we address the following problems:

1.1 Problem 1: Auto-Regressive DNNs for Synthetic Table Generation (STG)

This chapter introduce the problem of sythetic traffic generation and presents STAN (Synthetic network Traffic generation with Autoregressive Neural models), a tool to generate realistic synthetic network traffic datasets for subsequent downstream applications. Our novel neural architecture captures both temporal dependencies and dependence between attributes at any given time. It integrates convolutional neural layers with mixture density neural layers and softmax layers, and models both continuous and discrete variables. We evaluate the performance of STAN in terms of quality of data generated, by training it on both a simulated dataset and a real network traffic data set. Finally, to answer the question—can real network traffic data be substituted with synthetic data to train models of comparable accuracy?—we train two anomaly detection models based on self-supervision. The results show only a small decline in accuracy of models trained solely on synthetic data. While current results are encouraging in terms of quality of data generated and absence of any obvious data leakage from training data, in the future we plan to further validate this fact by conducting privacy attacks on the generated data. Other future work includes validating capture of long term dependencies and making model training more efficient.

1.2 Problem 2: LLMs for STG with Higher Fidelity

Synthetic data generation is integral to ML pipelines, e.g., to augment training data, replace sensitive information, and even to power advanced platforms like DeepSeek. While LLMs fine-tuned for synthetic data generation are gaining traction, synthetic table generation, a critical data type in business and science, remains underexplored compared to text and image

synthesis. This chapter shows that LLMs, whether used as-is or after traditional fine-tuning, are inadequate for generating synthetic tables. Their autoregressive nature, combined with random-order permutation during fine-tuning, hampers the modeling of functional dependencies and prevents capturing conditional mixtures of distributions essential for real-world constraints. We demonstrate that making LLMs permutation aware can mitigate these issues.

1.3 Problem 3: Reducing In-Context Learning Burden in STG via LLM Priors

Large language models (LLMs) have demonstrated strong in-context learning (ICL) capabilities, where a few representative examples guide content generation. This idea has been applied to great effect in synthetic tabular data generation, where LLMs can generate data that approximate samples from complex, heterogeneous distributions based on representative examples. However, ensuring high-fidelity synthetic data often requires a very large number of ICL examples. At the same time, as LLMs get larger and larger, their in-built prior knowledge becomes vast and can potentially substitute for specific data examples. In this chapter, we ask the question 'how many examples can a prompt substitute for?' and explore knowledge-guided prompting (KGP) where domain knowledge, either inferred or available, is explicitly injected into the prompt, reducing dependence on ICL examples. Our experiments systematically explore the trade-off between ICL and KG, revealing an empirical scaling law that quantifies how quality of generated synthetic data varies with increasing domain knowledge and decreasing example count. We classify prior knowledge into strong knowledge (e.g., symbolic constraints, statistical priors) versus weaker knowledge (e.g., monotonicity constraints, dependency relationships) and explore relationships between both forms and

in-context examples. Our results demonstrate that knowledge-guided prompting can be a scalable alternative, or addition, to in-context examples, unlocking new approaches to synthetic data generation.

1.4 Problem 4: Embedding Isotropy as a Trust Indicator for STG with LLMs

Recent studies have shown that vector representations of contextual embeddings learned by pre-trained large language models (LLMs) are effective in various downstream tasks in numerical domains. Despite their significant benefits, the tendency of LLMs to hallucinate in such domains can have severe consequences in applications such as energy, nature, finance, healthcare, retail and transportation, among others. To guarantee prediction reliability and accuracy in numerical domains, it is necessary to open the black-box and provide performance guarantees through explanation. However, there is little theoretical understanding of when pre-trained language models help solve numeric downstream tasks. This chapter seeks to bridge this gap by understanding when the next-word prediction capability of LLMs can be adapted to numerical domains through a novel analysis based on the concept of isotropy in the contextual embedding space. Specifically, we consider a log-linear model for LLMs in which numeric data can be predicted from its context through a network with softmax in the output layer of LLMs (i.e., language model head in self-attention). We demonstrate that, in order to achieve state-of-the-art performance in numerical domains, the hidden representations of the LLM embeddings must possess a structure that accounts for the shift-invariance of the softmax function. By formulating a gradient structure of self-attention in pre-trained models, we show how the isotropic property of LLM embeddings in contextual embedding space preserves the underlying structure of representations, thereby resolving

the shift-invariance problem and providing a performance guarantee. Experiments show that different characteristics of numeric data and model architecture could have different impacts on isotropy.

1.5 Problem 5: STG for Next-Gen Wireless: A Telecom Application

Large language models (LLMs) and foundation models have been recently touted as a game-changer for 6G systems. However, recent efforts on LLMs for wireless networks are limited to a direct application of existing language models that were designed for natural language processing (NLP) applications. To address this challenge and create wireless-centric foundation models, this chapter presents a comprehensive vision on how to design *universal foundation models* that are tailored towards the unique needs of next-generation wireless systems, thereby paving the way towards the deployment of *artificial intelligence (AI)-native networks*. Diverging from NLP-based foundation models, the proposed framework promotes the design of *large multi-modal models (LMMs)* fostered by three key capabilities: 1) processing of *multi-modal sensing* data, 2) *grounding* of physical symbol representations in real-world wireless systems using causal reasoning and retrieval-augmented generation (RAG), and 3) enabling *instructibility* from the wireless environment feedback to facilitate dynamic network adaptation thanks to logical and mathematical reasoning facilitated by neuro-symbolic AI. In essence, these properties enable the proposed LMM framework to build universal capabilities that cater to various cross-layer networking tasks and *alignment* of intents across different domains. Preliminary results from experimental evaluation demonstrate the efficacy of grounding using RAG in LMMs, and showcase the alignment of LMMs with wireless system designs. Furthermore, the enhanced rationale exhibited in the responses

to mathematical questions by LMMs, compared to vanilla LLMs, demonstrates the logical and mathematical reasoning capabilities inherent in LMMs. Building on those results, we present a sequel of open questions and challenges for LMMs. We then conclude with a set of recommendations that ignite the path towards LMM-empowered AI-native systems.

Chapter 2

Literature Review - the STG Landscape

2.1 Synthetic Tabular Generation (STG)

Synthetic data generation. Generating synthetic data to make up for the lack of real data is a common solution. Compared to modeling image data [86], learning distributions on multi-variate time-series data poses a different set of challenges. Multi-variate data is more diverse in the real world, and such data usually has more complex dependencies (temporal and spatial) as well as heterogeneous attribute types (continuous and discrete).

Synthetic data generation models often treat each column as a random variable to model joint multivariate probability distributions. The modeled distribution is then used for sampling. Traditional modeling algorithms [11, 48, 118] have the restraint of distribution data types and due to computational issues, the dependability of synthetic data generated by these models is extremely limited. GAN-based (Generative Adversarial Network-based) approaches augment performance and flexibility to generate data [69, 91, 140].

However, they are still either restricted to a static dependency without considering the temporal dependence usually prevalent in real world data [108, 140], or only partially build temporal dependence inside GAN blocks [58, 69]. By 'partially' here, we mean that the

temporal dependencies generated by GANs is based on RNN/LSTM decoders, which limits the temporal dependence in a pass or a block that GAN generates at a time. In real *netflow* data, traffic is best modeled as infinite flows which should not be interrupted by blocks of the synthesizer model. Approaches such as [58] (which generates embedding-level data, such as the embedding of URL addresses) and [69] are unable to reconstruct real *netflow* data characteristics such as IP address or port numbers, and other complicated attribute types. We are not aware of any prior work that models both entire temporal and between-attribute dependencies for infinite flows of data.

Furthermore, considering domain specialty, such as *netflow*, domain-specific data should not be classified simply as "tabular data" due to their insight and value complexity. For example, [108] passes the *netflow* data into a pre-trained IP2Vec encoder [107] to obtain a continuous representation of IP address, port number, and protocol for the GAN usage. STAN doesn't need a separate training component, and instead successfully learns IP address, port number, protocol, TCP flags characteristics naturally by simply defining the formats of those special attributes.

2.1.1 Deep Generative Model for STG

Non-autoregressive generative models. Lei et al. [140] proposed CTGAN where rows are independent of each other; a conditional GAN architecture ensures that the dependency between columns is learned. Tabsyn [146] showcased remarkable advancements in joint-distribution learning via a VAE plus diffusion approach, surpassing previous models of similar lineage, in terms of distributional correlation measures and machine learning efficiency. DoppelGanger [69] uses a combination of an RNN and a GAN to incorporate temporal dependencies across rows but this method has been tested in traditional, low-volume settings

such as Wikipedia daily visit counts. For high-volume applications, STAN [141] utilizes a combination of a CNN and Gaussian mixture neural networks to generate synthetic network traffic data. GraphDF [24] conducts multi-dimensional time series forecasting. GOGGLE [72] employs a generative modeling method for tabular data by learning relational structures.

Autoregressive generative models. PixelRNN [86, 127] have been successfully applied to signal data, image data, and natural language data. They attempt to iteratively generate data elements: previously generated elements are used as an input condition for generating the subsequent data. Compared to GAN models, autoregressive models emphasize two factors during the distribution estimating: 1) the importance of the time-sequential factor; 2) an explicit and tractable density. In this dissertation, we apply the autoregressive idea to learn and generate time-series multi-variable data.

Mixture density networks. Unlike modeling discrete attributes, some continuous numeric attributes are relatively sparse and span a large value range. Mixture Density Networks [14] is a neural network architecture to learn a Gaussian mixture model (GMM) that can predict continuous attribute distributions. This architecture provides the possibility to integrate GMM into a complex neural network architecture.

2.1.2 LLMs for STG

Use of Language Models (LLMs) for tabular data generation. Most modern LLMs are based on the transformer architecture [128] with parameters ranging from few millions to billions [54], and researchers have developed creative ways to harness LLMs in traditional machine learning and data contexts. LIFT [33] initially transforms a table row into a sentence, such as 'An Iris plant with sepal length 5.1cm, sepal width 3.5cm', and employs an LLM as a learning model for table classification, regression, and generation tasks.

GReaT [15], introduced earlier, utilizes a GPT-2 model that has been fine-tuned using a specific corpus for synthetic data generation. A general benefit of utilizing LLMs is the elimination of customized preprocessing pipelines. OmniPred [117], provides a framework for training language models as universal end-to-end regressors over (x, y) data from arbitrary formats. Similarly, Treutlein et al. [125] exhibit the ability of *inductive out-of-context* reasoning (OOCR) in a regression fine-tuning task of a language model. (A key difference between these works and this chapter is that in the regression setting, the prompt conditions the output to only predict the target label whereas we are attempting data conforming to the entire joint distribution at once.) Recent works [15, 116, 142, 148, 151] have shown the ability to use fine tuning to inject controlled distribution into LLMs, but these approaches are inflexible and do not leverage prior LLM's pre-trained knowledge. Curated LLM [113] is an approach that prompts LLMs with specific domain requirements (in English) but this approach is primarily intended for low-data regimes. In recent times, instruction-tuned models have shown great strides in 'following instructions' (and some forms of reasoning) but they are still limited at generating a diversity of datasets as considered here (some recent efforts [6, 31, 55, 104], e.g., BARE [154], aim to combine base models with post-training to address this issue).

Feature Ordering. Although not well-studied in the context of tabular data generation, the notion of feature ordering has been investigated in the context of graph-to-text translation [114] wherein to learn effective graph encodings, vertices are linearized via combinations of different graph traversal mechanisms, e.g., topological & breath-first strategies [34], and top-down & bottom-up approaches [105]. As a second example, 'permutation-invariant tabular data synthesis [155] examines the influence of the arrangement of table columns on convolutional neural network (CNN) training and organizes them according to the correlation among columns. Nevertheless, it is important to acknowledge that relying on mere

correlation to establish column orders can be limiting. There are also several approaches (e.g., [30, 61]) that synthesize, discover, or aggregate features from relational databases, leveraging order information when possible, for use in machine learning pipelines.

The Prompt vs the Example. The idea of modeling tradeoffs between prompts and in-context learning (ICL) examples has been studied before [65], but primarily in the context of NLP tasks and for a single prompt, not a range of knowledge levels in prompting as studied here. Our work is the first to systematically explore the tradeoff between knowledge and ICL examples for synthetic tabular data generation.

2.2 Data Science Insights for STG GenAI

Mining and Modeling Functional Dependencies. While tabular data representation learning for tables have been extensively studied [36, 37, 42, 77, 79, 91, 92, 115, 133, 134, 146], one key theme has been the discovery of functional dependencies (FDs) which has been studied by the data mining community from both theoretical and application points of view [23, 76, 83, 94, 149, 152]. Yunjia et al. [149] relax the notion of strict functional dependencies to include noisy functional relationships by utilizing probabilistic graphical models. Chen et al. [23], in their FakeTables approach use the discovery of functional dependencies in a GAN formulation; they first use a generator to create a set of columns (set A) and an autoencoder to cast another set (set B), which are then used by a discriminator to calculate the gradient loss. Muralidhar et al. [83] proposed to use Granger causality to incorporate functional *invariants* across multiple time series.

Data Privacy for Cybersecurity. In recent decades, machine learning has been applied to multiple tasks in cybersecurity, such as automatically detecting malicious activity and stopping attacks [18, 22]. Such machine learning approaches usually require a large amount

of training data with specific features. However, a training model using real user data leads to privacy exposure and ethics problems. Previous work on anonymizing real data [103] has failed to provide satisfactory privacy protection or degrades data quality too much for machine learning model training.

This dissertation takes a different approach that, by learning and generating realistic synthetic data, ensures that real data can be substituted for when training machine learning models. Prior work on generating synthetic network traffic data includes Cao et al. [20] who use a network simulator to generate traffic data while we do not generate data through workload or network simulation; both Riedi et al. and Paxson [93, 106] use advanced time-series methods, however, these are for univariate data, and different attributes are assumed to be independent. We model multivariate data, that is, all attributes at each time step, jointly. Mah [75] models distributions of HTTP packet variables similar to baseline 1 (B1) in our thesis. Again, unlike our method, it does not jointly model the network data attributes.

Applications of LLM and STG in wireless telecommunication. LLMs for wireless networks have been studied in [12, 73, 120]. However, the LLMs of [73, 120] are confined to processing a single mode of textual data, which restricts their role to network chatbots. Accordingly, such LLMs cannot capture the multi-modal data arising from the multiple functions (e.g., sensing, communication, etc.) of future wireless networks. Although [12] focuses on utilizing *multi-modal* LLMs, their approach relies on LLMs like GPT-x, LLaMA, or Falcon tailored for natural language processing (NLP) tasks. To become effective, such multi-modal LLMs must be fine-tuned as wireless tasks change, and, thus, they cannot act as a *universal* solution to different interrelated, cross-layer tasks in AI-native networks.

In addition, the works in [12, 73, 120] overlook how artificial intelligence (AI)-native networks can fuse, at scale, the environmental *sensing* data that drive their multi-modal LLMs. Moreover, state-of-art solutions like [12] neglect the fact that even textual wireless data can

be structured in specific formats such as tables and network performance evaluations can be presented in the form of graphs or images. This limitation in handling structure and modality, restricts the range of functionalities (e.g., wireless chatbots, text to image conversion) for which LLMs can be effectively adopted.

Moreover, current LLMs [12, 73, 120] lack the essential *grounding* abilities that connect their abstract, language-based knowledge to real-world experiences. In fact, LLMs majorly gain their knowledge upon being trained on extensive corpuses of text data. Hence, these LLMs cannot capture the complex physics governing the wireless environment, such as the propagation of wireless signals, thereby leading to potentially inconsistent decisions and predictions. The absence of grounding impedes AI-native networks from carrying out logical, causal, and mathematical reasoning operations, necessary for achieving goals such as resilience, intent management, non-linear signal processing, and others. Therefore, it is necessary to ensure that the representations acquired by LLMs accurately interpret information from the real world and adhere to the network goals.

Another persistent challenge of LLMs is their tendency to hallucinate by generating "human-like" outputs that do not connect to reality, essentially fabricating false information[102]. If such hallucinations occur, AI-native networks driven by LLMs may generate inaccurate information. For example, LLMs may propose power allocations that violate the regulated thresholds of transmit powers of base stations, leading to alignment problems. *Alignment* here pertains to fulfilling network objectives, adhering to physical constraints like radiated power or environmental sustainability goals, and ensuring compliance with governmental regulations. Finally, existing LLMs [12, 73, 120] lack precise instructibility from the environment and, hence, they cannot perform dynamic problem-solving. *Instructibility* is the capability of LLMs to dynamically adjust their behavior based on explicit feedback from user equipment, system engineers, or operators. This adaptability should also ensure that

LLM decisions are explainable.

Chapter 3

Auto-Regressive DNNs for Synthetic Table Generation (STG)

Following the introduction in Chapter §1.1, this chapter will first propose an approach to table generation using auto-regressive deep neural networks. This will be the first part of the dissertation, focusing on Numerical Inference-based synthetic table generation.

Traditionally, to address data privacy issues, three main approaches are usually explored [4, 5]: 1) non-cryptographic anonymization methods; 2) cryptographic anonymization methods; and, 3) perturbation methods, such as differential privacy. However, 1) leaks private information in most cases; 2) is currently impractical (e.g., using homomorphic encryption) for large data sets; and, 3) degrades data quality making it less suitable for machine learning tasks.

In this chapter, we take an orthogonal approach. We generate synthetic data that is realistic enough to replace real data in machine learning tasks. Specifically, we consider multivariate time-series data and, unlike prior work, capture both temporal dependencies and dependencies between attributes. Figure 3.1 illustrates our approach, called STAN. Given real historical data, we first train a CNN-based autoregressive generative neural network that learns the underlying joint data distribution. The trained STAN model can then generate any amount of synthetic data without revealing private information. This realistic synthetic data can replace real data in the training of machine learning models applied to cybersecurity

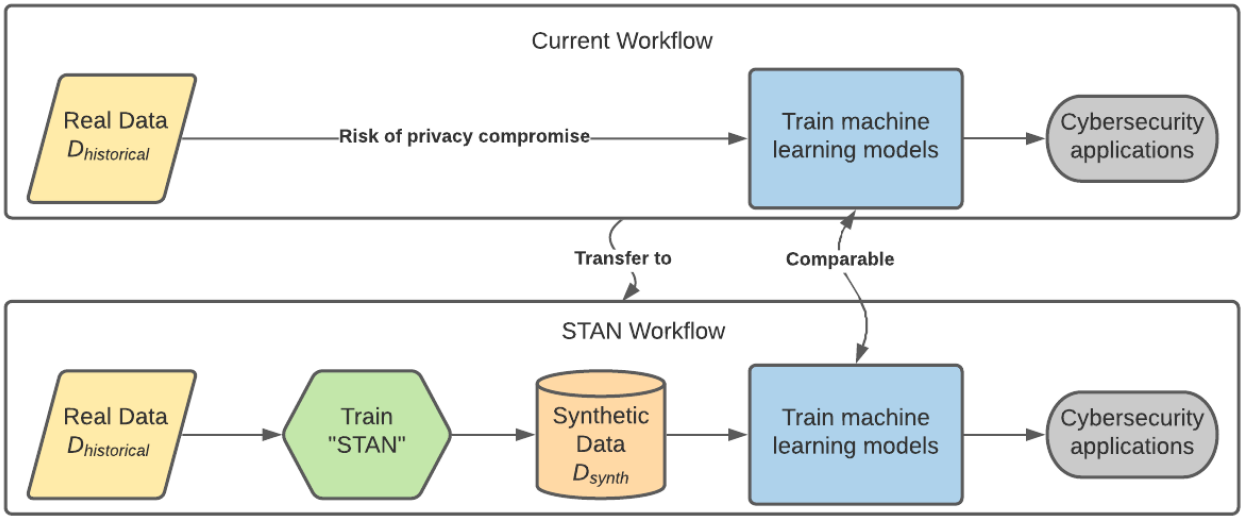


Figure 3.1: STAN The top figure shows a simplified workflow where real data ($D_{historic}$) is used to train a machine learning model for cybersecurity applications; however, use of real data may result in a privacy compromise. The bottom figure shows the proposed workflow, where machine learning models are trained using realistic synthetic data (D_{synth}) generated by STAN.

applications, with performance[1] comparable to models trained on real data.

To evaluate the performance of STAN, we use both simulated data and a real publicly-available network traffic data set. We compare our method with four selected baselines using several metrics to evaluate the quality of the generated data. We also evaluate the suitability of the generated data as compared to real data in training machine learning models. Self-supervision is commonly used for anomaly detection [26, 129]. We consider two such tasks – a classification task and a regression task for detecting cybersecurity anomalies – that are trained on both real and synthetic data.

We show comparable model performance after entirely substituting the real training data with our synthetic data: the F-1 score of the classification task only drops by 2% (99% to 97%), while the mean square error only increases by about 13% for the regression task.

[1]Here performance refers to a model evaluation metric such as precision, recall, F1-score, mean squared error, etc.

In this chapter, we make the following key contributions:

- STAN is a new approach to learn joint distributions of multivariate time-series data—data typically used in cybersecurity applications—and then to generate synthetic data from the learned distribution. Unlike prior work, STAN learns both temporal and attribute dependencies. It integrates convolutional neural layers (CNN) with mixture density neural layers (MDN) and softmax layers to model both continuous and discrete variables. Our code is publicly available.[2]
- We evaluated STAN on both simulated data and a real publicly available network traffic data set, and compared with four baselines.
- We built models for two cybersecurity machine learning tasks using only STAN generated data to train and which demonstrate model performance comparable to using real data.

3.1 Problem Definition

We assume the data to be generated is a multivariate time-series. Specifically, data set $\mathbf{x}$ contains n rows and m columns. Each row $\mathbf{x}_{(i,:)}$ is an observation at time point i and each column $\mathbf{x}_{(:,j)}$ is a random variable j, where $i \in [1..n]$ and $j \in [1..m]$. Unlike typical tabular data, e.g., found in relational database tables, and unstructured data, e.g., images, multivariate time-series data poses two main challenges: 1) the rows are generated by an underlying temporal process and are thus not independent, unlike traditional tabular data; 2) the columns or attributes are not necessarily homogeneous, and comprise multiple data types such as numerical, categorical or continuous, unlike say images.

[2] https://github.com/ShengzheXu/stan.git

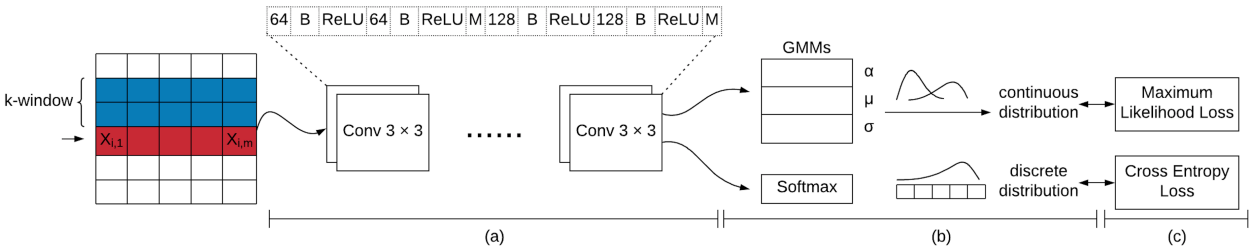


Figure 3.2: STAN components: (a) window CNN, which crops the context based on a sliding window and extracts features from context; The CNN architecture includes 14 layers where numeric values notes are the number of 3*3 convolutional filters; B notes batch normalization layers; ReLU notes activation layers; and M notes max pooling layers. (b) mixture density neural layers and softmax layers learn to predict the distributions of various types of attributes; (c) the loss functions for different kinds of layers.

The data $\mathbf{x}$ follows an unknown, high-dimensional joint distribution $\mathbb{P}(\mathbf{x})$, which is infeasible to estimate directly. The goal is to estimate $\mathbb{P}(\mathbf{x})$ by a generative model $\mathbb{S}$ which retains the dependency structure across rows and columns. Values in a column typically depend on other columns, and temporal dependence of a row can extend to many prior rows. Once model $\mathbb{S}$ is trained, it can be used to generate an arbitrary amount of data, $\mathbf{D}_{synth}$.

Another key challenge is evaluating the quality of the generated data, $\mathbf{D}_{synth}$. Assuming a data set, $\mathbf{D}_{historical}$, is used to train $\mathbb{S}$, and an unseen test data set, $\mathbf{D}_{test}$, is used to evaluate the performance of $\mathbb{S}$, we use two criteria to compare $\mathbf{D}_{synth}$ with $\mathbf{D}_{test}$:

1. Similarity between a metric M evaluated on the two data sets, that is, is $M(\mathbf{D}_{test}) \approx M(\mathbf{D}_{synth})$?

2. Similarity between performance P on training the same machine learning task T, in which the real data, $\mathbf{D}_{test}$, is replaced by the synthetic data, $\mathbf{D}_{synth}$, that is, is $P[T(\mathbf{D}_{test})] \approx P[T(\mathbf{D}_{synth})]$?

3.2 Proposed Method

We model the joint data distribution, $\mathbb{P}(\mathbf{x})$, using an autoregressive neural network.

The model architecture, shown in Figure 3.2, combines CNN layers with a density mixture network [14]. The CNN captures temporal and spatial (between attributes) dependencies, while the density mixture neural layer uses the learned representation to model the joint distribution. More architecture details will be discussed in Section 3.2.2. During the training phase, for each row, STAN takes a data window prior to it as input. Given this context, the neural network learns the conditional distribution for each attribute. Both continuous and discrete attributes can be modeled. While a density mixture neural layer is used for continuous attributes, a softmax layer is used for discrete attributes.

In the synthesis phase, STAN sequentially generates each attribute in each row. Every generated attribute in a row, having been sampled from a conditional distribution over the prior context, serves as the next attribute's context.

3.2.1 Joint distribution factorization

$\mathbb{P}(\mathbf{x})$ denotes the joint probability of data $\mathbf{x}$ composed of n rows and m attributes. We can expand the data as a one-dimensional sequence $\mathbf{x}_1, ..., \mathbf{x}_n$, where each vector $\mathbf{x}_i$ represents one row including the m attributes $x_{i,1}, ..., x_{i,m}$. To estimate the joint distribution $\mathbb{P}(\mathbf{x})$ we express it as the product of conditional distributions over the rows. We start from the joint distribution factorization with no assumptions:

$$\mathbb{P}(\mathbf{x}) = \prod_{i=1}^{n} \mathbb{P}(\mathbf{x}_i | \mathbf{x}_1, ..., \mathbf{x}_{i-1}) \tag{3.1}$$

Unlike unstructured data such as images, multivariate time-series data usually corresponds to underlying continuous processes in the real world and do not have exact starting and ending points. It is impractical to make a row probability $\mathbb{P}(\mathbf{x}_i)$ depend on all prior rows as in Equation 3.1. Thus, a k-sized sliding window is utilized to restrict the context to only the k most recent rows. In other words, a row conditioned on the past k rows is independent of all remaining prior rows, that is, for $i > k$, we assume independence between $\mathbf{x}_i$ and $\mathbf{x}_{<i-k}$. We can thus rewrite the joint distribution $\mathbb{P}(\mathbf{x})$ as the product of the conditional distributions over the prior k rows:

$$\mathbb{P}(\mathbf{x}) = \prod_{i=1}^{k} \mathbb{P}(\mathbf{x}_i|\mathbf{x}_1, ..., \mathbf{x}_{i-1}) \prod_{i=k+1}^{n} \mathbb{P}(\mathbf{x}_i|\mathbf{x}_{i-k}, ..., \mathbf{x}_{i-1}) \tag{3.2}$$

Note that a suitable value of k needs to be picked based on empirical evidence or domain knowledge. While all the probabilities in the second term on the RHS of Equation 3.2 are conditioned on k variables, the same is not true for the probabilities in the first term. To make these consistent, we add zero padding and then symbolically define that $\mathbf{x}_i$ where $i \leq 0$ represents a padding row, as Equation 3.3 shows.

$$\begin{aligned} \mathbb{P}(\mathbf{x}) &= \prod_{i=1}^{k} \mathbb{P}(\mathbf{x}_i|\mathbf{x}_{i-k}, ..., \mathbf{x}_1, ..., \mathbf{x}_{i-1}) \prod_{i=k+1}^{n} \mathbb{P}(\mathbf{x}_i|\mathbf{x}_{i-k}, ..., \mathbf{x}_{i-1}) \\ &= \prod_{i=1}^{n} \mathbb{P}(\mathbf{x}_i|\mathbf{x}_{i-k}, ..., \mathbf{x}_{i-1}) \end{aligned} \tag{3.3}$$

The joint distribution of a row can be factorized in two ways: 1) Equation 3.4 assumes conditional independence of attributes in a row, given all attributes in the previous k rows; 2) Equation 3.5 makes no conditional independence assumptions of attributes in the same row.

$$\mathbb{P}(\mathbf{x}) = \prod_{i=1}^{n} \prod_{j=1}^{m} \mathbb{P}(\mathbf{x}_{i,j} | \mathbf{x}_{i-k}, ..., \mathbf{x}_{i-1}) \tag{3.4}$$

$$\mathbb{P}(\mathbf{x}) = \prod_{i=1}^{n} \prod_{j=1}^{m} \mathbb{P}(\mathbf{x}_{i,j} | \mathbf{x}_{i-k}, ..., \mathbf{x}_{i-1}; x_{i,1}, ..., x_{i,j-1}) \tag{3.5}$$

While (3.4) provides a good approximation, we found (3.5) performs slightly better.

During generation, initialization is a key concern for our generative model due to temporal dependence. In order to generate data without supplying any real data or specific seed, we begin the above autoregressive chain process with the marginal distribution $\mathbb{P}(\mathbf{x}_1)$. In practice, as described later in section (3.2.2), the marginal distribution is approximated by all zero conditional: $\mathbb{P}(\mathbf{x}_1|0)$. Then beginning from the second step, Equation 3.4, 3.5 holds.

3.2.2 Neural network architecture

As shown in Figure 3.2, the input window goes through the *convolutional layers* followed by *mixture density neural layers* or *softmax layers* sequentially to learn the joint distribution. We define two *loss* functions for the two distribution modeling layers separately. Algorithms 1 and 2 provide details on model training and data synthesis. Note that the training phase allows for parallelization while the synthesis phase is sequential.

Window convolutional layers (wCNN). The CNN layers, which we call window CNN since they operate on a sliding window of data, perform a two-dimensional convolution. For one row $\mathbf{x}_i$ the layers capture a rectangular context above the row as shown in Figure 3.2 STAN uses multiple convolutional layers that preserve the spatial and temporal resolution in a sliding time window box. Each number in Figure 3.2 represents the number of $3 * 3$ filters in that layer. Batchnorm, ReLU and max pooling layers are also used, marked as BN,

Algorithm 1 Model training process for each attribute j

Input $D_{Historical}$, window size k, attribute type T_j.
Output STAN model $\mathbb{S}_{stan}$;

Construct window data
  X_i^{window} = concatenate X_{i-k},...,X_i;
  $y_i^{window} = X_i$;
for epoch in 1 ... EPOCH **do**
  $X_i^{window} *= Mask$
  if T_j is continuous **then**
    $\mathbb{P}_{gmm\_pred} = mdn(wCNN(X_{window}))$;

    $loss = nll(\mathbb{P}_{gmm\_pred}, y_{window})$;

  else
    $\mathbb{P}_{softmax\_pred} = softmax(wCNN(X_{window}))$;

    $loss = cross\_entropy(\mathbb{P}_{softmax\_pred}, y_{window})$;

  Using *Adam* optimizer to update $\mathbb{S}_{stan}$ with $loss$;

$ReLU$, and M, respectively.

Convolution mask. Based on which factorization is selected, we have mask A for Equation 3.4 and mask B for Equation 3.5.

Mixture density neural layer (mdn). learns a conditional *Gaussian mixture distribution*. It consists of three parallel fully connected layers, modeling $\alpha_i, \sigma_i, \mu_i$ separately, where the parameter α_i represents for the component weights of an *Gaussian mixture model*, and the μ_i and σ_i^2 are the mean and variance parameters of the Gaussian distribution components. The α_i parameters output go to a softmax, so that the weights of all the Gaussian mixture components sum to one.

Loss functions. We define loss functions for *mixture density neural layer* and *softmax layer* separately. Note that the two losses have different scales, and while multitask learning has its advantages, we match each *mixture density neural component* or *softmax component* with

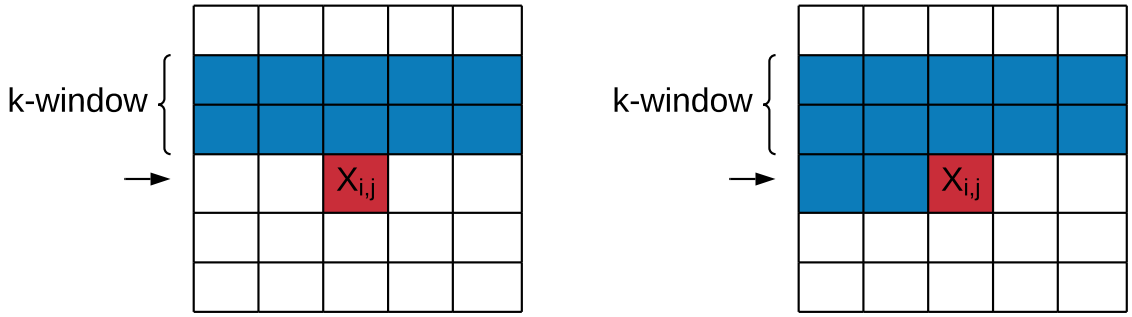
Algorithm 2 Data synthesis process

Input Trained STAN model $\mathbb{S}_{stan}$.
Output D_{synth};

Init context X^{window} = marginal sampling()
while condition(target row number or time stamp) **do**
    $X_i^{window} *= Mask$
    $P_{pred} = \mathbb{S}_{stan}(X_i^{window})$;

    y_{sample} = sample from distribution $\mathbb{P}_{pred}$;

    $X_{i+1}^{window} = X_i^{window}[1:,:] + y_{sample}$



(a) Mask A for conditional independence assumption between attributes in same row

(b) Mask B for no conditional independence assumption in the same row

Figure 3.3: Masks for context window convolution

an individual *wCNN component*.

A Negative Log-Likelihood Loss (NLL) is used for the mixture density layers, which predict a group of mixture density parameters that can compose a Gaussian mixture model as Equation 3.6: $\alpha_i, \sigma_i, \mu_i$. We use maximum likelihood loss to estimate a true distribution: the label of the input, which is the new row that to be generated, is supposed to have the highest probability in the estimated distribution. Cross entropy loss is used for the softmax layer.

$$NLL(x|\mu, \sigma^2) = -\log \sum \alpha_i * \mathcal{N}(x|\mu_i, \sigma_i^2) \tag{3.6}$$

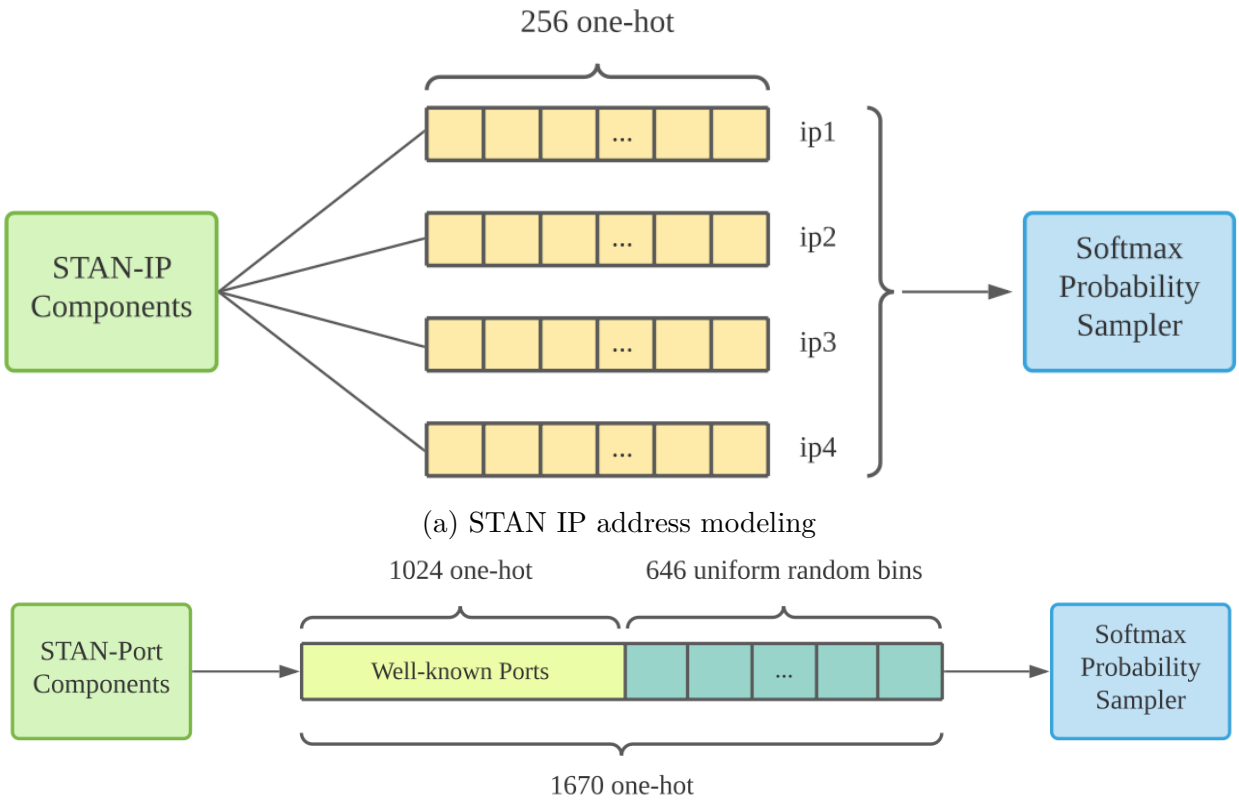
3.2.3 IP address and port number Modeling

IP addresses and port numbers are key to network traffic data. However, naively modeling them as continuous or discrete variables gives poor results.

STAN specifically learns IP address and port number characteristics.

As shown in Figure 3.4 (a), STAN treats an IP address as four 256-categorical discrete attributes. With the help of the STAN Mask definition, even though one IP address attribute is split across four intermediate attributes, it is considered together as one variable for attribute dependence.

Since port numbers can vary between 0 and 64K, that is too many values to model as a discrete variable. On the other hand, treating it as a continuous variable would result in inaccuracies especially for well-known ports (those less than 1024), where being off even by one can mean something completely different. Therefore, we take a hybrid approach. STAN treats port numbers up to 1024 individually as discrete values; beyond that it models ports in bins of size 100, as shown in Figure 3.4 (b). In all, port numbers are represented as a 1670-categorical discrete attribute. After being generated by STAN, if a port number is less than 1024 it corresponds to that particular port number, else a port is sampled from a uniform distribution of port numbers in the corresponding bin during post processing.



(a) STAN IP address modeling

(b) STAN port number modeling

Figure 3.4: With the benefit of flexible STAN continuous and discrete generator architecture, special domain attributes (such as IP address, port, protocol, and TCP flags), can be learned by purely modifying the configure parameters.

3.2.4 Baselines

We selected four different methods to serve as baselines for our method. This range for basic Gaussian Mixture Model, Bayesian Network to two recent deep learning approaches that use GANs for synthetic data generation, which for brevity we refer to as GMM, BN, WPGAN, and CTGAN, respectively. We compare STAN with these baselines and analyze the distribution factorization.

Gaussian Mixture Model (GMM). This assumes all attributes at a particular time step are independent of each other, and further that each row is independent. Thus it can be

factorized as following:

$$\mathbb{P}(\mathbf{x}) = \prod_{i=1}^{n} \mathbb{P}(\mathbf{x}_i) \qquad (3.7a)$$

$$\mathbb{P}(\mathbf{x}_i) = \prod_{j=1}^{m} \mathbb{P}(\mathbf{x}_{i,j}) \qquad (3.7b)$$

Bayesian Network (BN). As a traditional statistical approach, limited temporal or attributes dependence can be learnt based on the domain knowledge from experts. For example, if $\mathbf{x}_{i,j_1}$ is dependent on $\mathbf{x}_{i-1,j_1}$ and $\mathbf{x}_{i,j_2}$, we can write it as a product of the conditional distributions (see Equation 3.8). The value $\mathbb{P}(\mathbf{x}_{i,j_1}|\mathbf{x}_{i-1,j_1}, \mathbf{x}_{i,j_2})$ is the probability of the j_1 attributes of the i-th observation row, given the $(i-1)$-th j_1 attribute and the i-th j_2 attribute. Considering the edge situation as well as utilizing the Bayes rule, we rewrite the distribution $\mathbb{P}(\mathbf{x}_{i,j_1}|\mathbf{x}_{i-1,j_1}, \mathbf{x}_{i,j_2})$ as:

$$\begin{aligned}\mathbb{P}(\mathbf{x}) &= \prod_{i=1}^{n} [\mathbb{P}(x_{i,j_1}|x_{i,j_2}, x_{i-1,j_1}) \prod_{j=1, j\neq j_1}^{m} \mathbb{P}(x_{i,j})] \\ &= \mathbb{P}(x_1) \cdot \prod_{i=2}^{n} [\mathbb{P}(x_{i,j_1})\mathbb{P}(x_{i-1,j_1}|x_{i,j_1})\mathbb{P}(x_{i,j_2}|x_{i,j_2})] \\ &\quad \cdot \prod_{j=1, j\neq j_1}^{m} \mathbb{P}(x_{i,j})\end{aligned} \qquad (3.8)$$

WPGAN [108] utilizes GAN to specifically generate network traffic flow data, while **CTGAN** [140] utilizes GAN to generate general tabular data that contains both discrete and continuous attributes. Both B3 and B4 assume attribute dependence at a certain time step but ignore temporal-wise dependence. Thus the joint distribution can be factorized as Equation 3.7a only, while the factorization inside each row is untractable due to the GAN

mechanism.

3.2.5 Evaluation Metrics

The evaluation of generative models is challenging and subjective. We use multiple metrics to compare them: likelihood, distribution comparison, domain knowledge rules test, and machine learning tasks performance comparison.

Likelihood. The likelihood function measures the goodness of a statistical model fitting a data sample. However, the intrinsic difference between explicit density methods (GMM, BN, and STAN) and implicit density methods (WPGAN and CTGAN) makes it more challenging to compare them. Goodfellow et al. [46] states that there is no fair approach to directly compare the likelihoods of GAN models. Thus in this chapter, we only compare the likelihoods between explicit density models, that is, GMM, BN and, STAN.

Distribution and JS divergence Although the goal of our work is to model joint distribution of a window of data, we also compare the marginal distributions of the individual attributes. As a quantitative metric, we calculate Jensen-Shannon divergence between the distributions of the generated data $\mathbf{D}_{synth}$ and the real data $\mathbf{D}_{test}$ for each attribute.

Domain knowledge test. We use domain knowledge checks to evaluate the synthetic data quality. Since the application data set pertains to network traffic flow, we use several properties that such data needs to satisfy in order to be realistic [108].

In addition to marginal distributions, we also explore network traffic specific distributions such as that of the number of unique destinations (in terms of IP addresses), and of number of bytes per IP address, and compare then with those distributions in real data. Similarly, we compare the top most frequently occurring port numbers.

Machine learning application task. The final goal of generating synthetic data is to build machine learning models without using any real data. To evaluate whether the generated data is able to replace real data in a model training process, we select two tasks used for cybersecurity anomaly detection. One is a classification task while the other is regression; both use self-supervision.

While these tasks may not seem to directly relate to cybersecurity, they are good at detecting anomalies, which may be caused, e.g., by a cybersecurity breach. The basic assumption is that different attributes in network traffic data have underlying dependencies, e,g., between the protocol field and other fields, during normal operation and these relationships may change in the presence of security anomalies. The models are built to capture this normal relationship, and then try to detect any changes in it during deployment.

The first task is predicting the transport protocol field in network traffic flow data, while the second task is to predict the number of bytes in the next network flow. In practice once trained these models are used for marking anomalies when the actual value significantly differs from the real one based on a hyperparameter threshold value. We train a RandomForest model for the classification task, and a neural network model for the regression task.

As the evaluation metric, we use the F1 score, and the mean square error (MSE) for the two tasks, respectively. Since the classification task is an multi-class task, we apply the macro-F1 score which takes the average of all the category F1 scores. This ensures equal treatment of all classes even when the class distribution is skewed, as is likely for transport protocol where TCP dominates. For both tasks we compare the cross-validation performance of the models trained on real and synthetically generated data.

3.3 Experimental Results

To demonstrate its effectiveness, we train and evaluate STAN on a real network traffic data set. However, to experiment with some architectural variations, we first use a simple simulated data set.

3.3.1 Understanding STAN using Simulated Data

We built a simulated dataset with a simple random process whose dependence can be clearly controlled. We simulated a two-variable data distribution with the following formula and sampled 10,000 points data set (X, Y) from it:

$$x_t = 0.9x_{t-1} + 0.1\mathcal{N}$$

and

$$y_t = 0.9x_t + 0.1\mathcal{N},$$

where $\mathcal{N}$ is standard normal distribution. We incorporate both temporal and attribute dependence with two attributes, X and Y.

To train we apply a naive version of STAN, that passes the input to mixture density neural layers directly, and GMM on the simulated data set. Then we generated data using both STAN-a and STAN-b.

To measure how well dependence is captured we use the Pearson correlation coefficient R both for temporal dependence $R(X_t, X_{t-1})$ and attribute dependence $R(X_t, Y_t)$. In addition, we use two machine learning tasks to show that the synthetic data is able to serve as model training source to replace the original data.

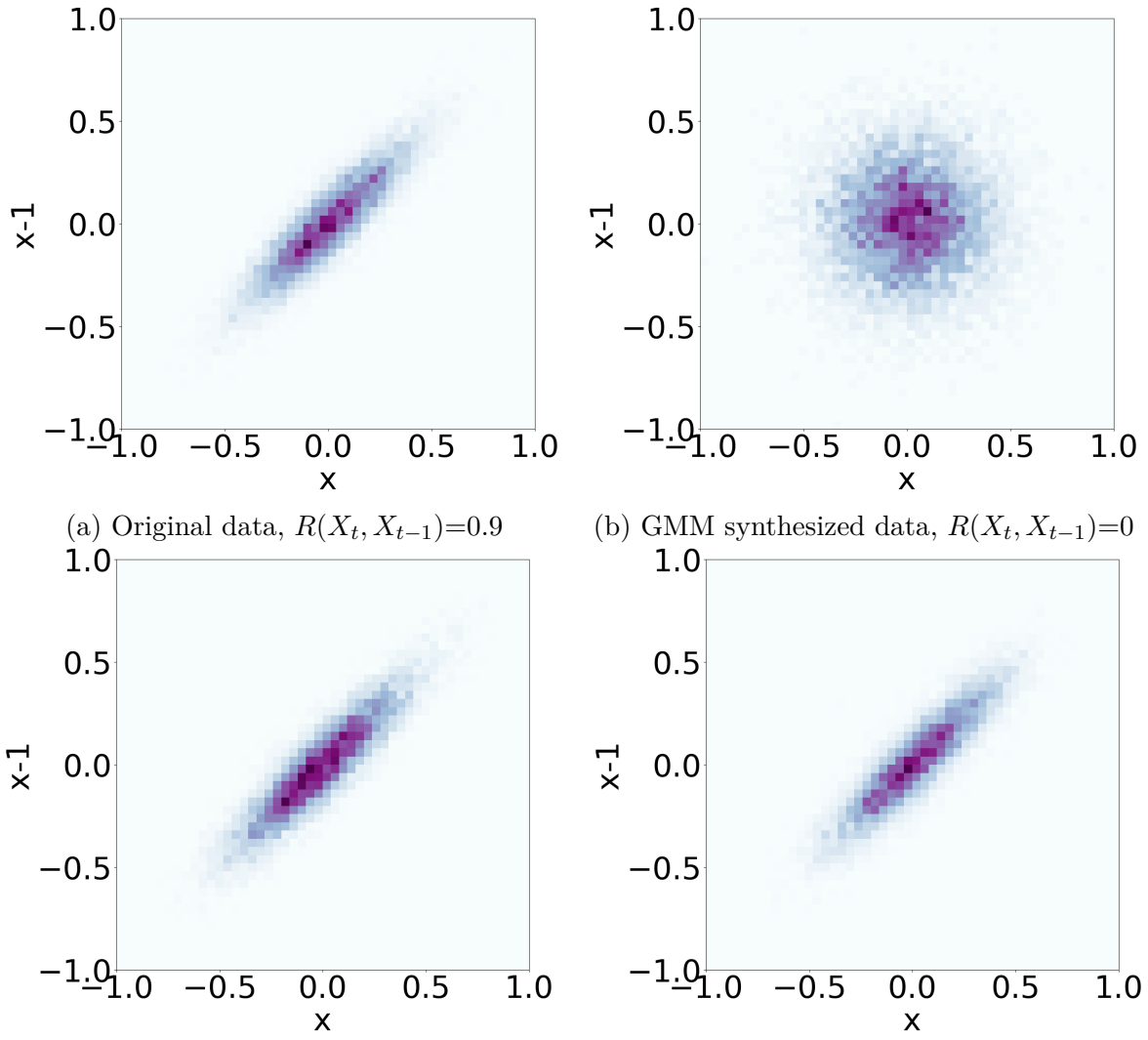
Quantitative evaluation. We evaluated the correlation coefficient R between both temporal dependence $R(X_t, X_{t-1})$ and attribute dependence $R(X_t, Y_t)$. Figure 3.5 shows the scatter plots of X_t and X_{t-1} from four data sources (simulated data, GMM synthetic data, STAN with mask A synthetic data, and STAN with mask B synthetic data), and Figure 3.6 shows that for X_t and Y_t.

Figures 3.5a and 3.6a show the dependence in the original (simulated) data, across time and attribute, respectively, that the synthesizer needs to learn. The scatter plots (x_{t-1}, x_t) and (x_t, y_t) show strong linear relationship. Figures 3.5b and 3.6b show that independently learned marginal distribution is unable to generate data with temporal and attribute-wise dependence. Figure 3.5c and 3.5d show STAN-a and STAN-b perform similarly and are able to generate data with the same temporal dependence as in the simulated (original) data. However, Figure 3.6c and 3.6d show that STAN-b, when explicitly conditioned on the same-row attribute context in its convolution perceptive field, generates better attribute-wise dependence than STAN-a.

Since mask A and mask B represent *conditional independence* and *explicit dependence* respectively, we summarize through this observations:

- Both conditional independence and explicit dependence provide reasonable approximation for temporal dependence. The $R(X_t, X_{t-1})$ of simulated data, synthetic data generated using STAN mask A, and synthetic data generated using STAN mask B are all same (0.9).

- Conditional independence provides a reasonable same-row attribute approximation, while explicit dependence performs better. The $R(X_t, Y_t)$ of simulated data, and synthetic data generated using STAN mask B is 0.9; while that of synthetic data generated using STAN mask A is 0.7.



(a) Original data, $R(X_t, X_{t-1})$=0.9

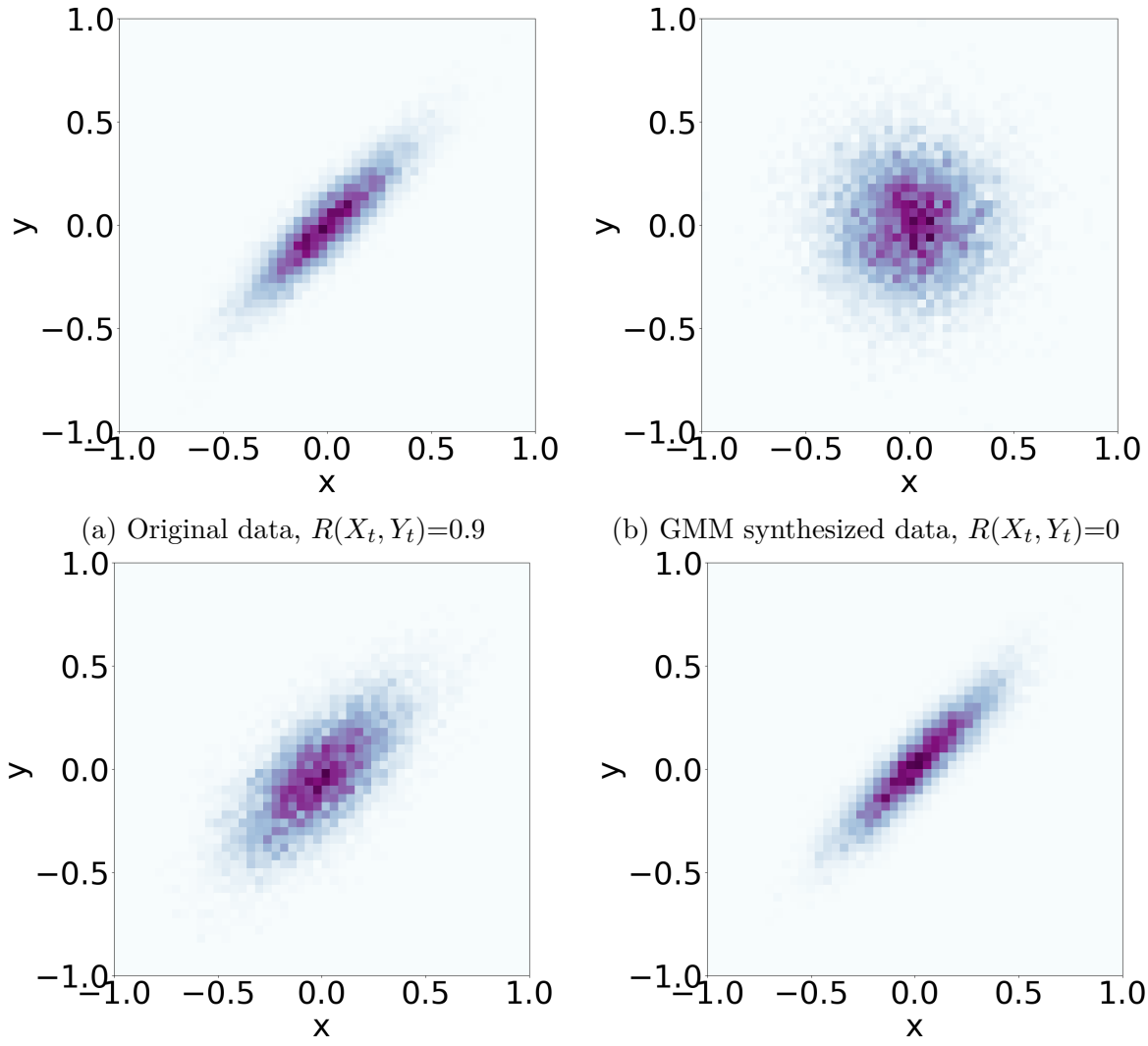
(b) GMM synthesized data, $R(X_t, X_{t-1})$=0

(c) STAN mask A synthesized data, $R(X_t, X_{t-1})$=0.9

(d) STAN mask B synthesized data, $R(X_t, X_{t-1})$=0.9

Figure 3.5: Temporal dependence between (X_t, X_{t-1}). Scatter plots of (X_t, X_{t-1}) values: (a) shows the original (simulated) data, while (b)–(d) show synthetic data. The x- and y-axes represent the values of X_{t-1} and X_t, respectively. The correlation coefficient R (shown in each subplot) quantifies how well the temporal dependency between X_{t-1} and X_t is captured by the synthetic data compared to the real data.

Machine learning tasks. We also used the simulated data and the corresponding synthetic data for training two machine learning tasks (using the scikit-learn Python library). Then we evaluate the performance (Mean Square Error) of the trained machine learning models



(a) Original data, $R(X_t, Y_t)$=0.9

(b) GMM synthesized data, $R(X_t, Y_t)$=0

(c) STAN mask A synthesized data, $R(X_t, Y_t)$=0.7

(d) STAN mask B synthesized data, $R(X_t, Y_t)$=0.9

Figure 3.6: Attribute dependence between (X_t, Y_t). Scatter plots of (X_t, Y_t) values: (a) shows the original (simulated) data, while (b)–(d) show synthetic data. The x- and y-axes represent the values of X_t and Y_t. The correlation coefficient R (shown in each subplot) quantifies how well the dependency between X_t and Y_t is captured by the synthetic data compared to the real data.

on the simulated test data. Simulated training data and simulated test data follow the same distribution.

- **T1**: predict y_t given x_t (row attribute dependence).

- **T2**: predict x_{t+1} given x_t (temporal dependence).

Table 3.1 shows that a machine learning model trained only on synthetic data generated by STAN produces similar test loss as that trained on the original simulated test data.

Training Data	MSE(**T1**)	MSE(**T2**)
Simulated data	0.010	0.01
GMM	0.050	0.05
STAN mask A	0.013	0.01
STAN mask B	0.010	0.01

Table 3.1: Mean Square Error of the two tasks

3.3.2 Real network traffic data

Data set. Network traffic data is typically a multivariate time-series. A common format is called *netflow*, where each row represents a unidirectional network traffic connection or flow. We selected a *netflow* data set for our experiments since it is a good representative format for network traffic data in general. Here we use a large publicly available *netflow* data set. Typically each row consists of the following attributes: timestamp at the end of a flow (te), duration of flow (td), packets exchanged in the flow (pkt), and the corresponding number of bytes (byt), source IP address[3] (sa), destination IP address (da), source port (sp), destination port (dp), flags (flg), and transport protocol (pr). Each row x_i can be expressed as a tuple of $(te_i, byt_i, sa_i, da_i, pr_i,$ etc). Table 3.2 shows typical attributes, their types and example values.

We apply STAN on a publicly available benchmark *netflow* data set, UGR'16 [74], which contains large scale traffic data captured by a Tier-3 ISP cloud service provider. First, we

[3]We only consider IPv4 addresses here.

Attribute	Type	Example
timestamp	continuous	2016-04-11 00:02:15
duration	continuous	0.344
transport protocol	categorical	TCP
source IP address	categorical	85.201.196.53
source port	categorical	19925
dest. IP address	categorical	42.219.145.151
dest. port	categorical	80
bytes	numeric	11238
packets	numeric	11
TCP flags	categorical	.A..SF

Table 3.2: Overview of typical attributes in flow-based data.

selected a week of data (April week3) data to focus on. We selected this week this week since it looked interesting in terms of volume of traffic and the number of events marked. Second, we randomly selected flows related to 90 users (essentially IP addresses) based on the number of traffic flows per user distribution. Third, we extracted one day's (Monday) data to be the $\mathbf{D}_{historical}$ and another day (Tuesday) of the same user group and the same week to be the $\mathbf{D}_{test}$. Following this strategy, we selected 1,531,126 samples for the $\mathbf{D}_{historical}$ and 1,952,702 samples for the $\mathbf{D}_{test}$. Ten percent of $\mathbf{D}_{historical}$ are selected out to serve as training validation data $\mathbf{D}_{validation}$.

***netflow* data processing.** To ensure the trained model is a practical and robust tool to synthesize network traffic flow data, we normalize the raw *netflow* data for ease of processing by the neural network. Also, the neural network predicted values are transformed back into the original scale.

The inputs to the neural model are pre-processed to facilitate training. The numerical attributes are min-max scaled; for the categorical attributes, we apply one-hot encoding. Specifically, for the protocol attribute we use a three-way softmax (for TCP, UDP and other). Note that for simplicity we consider only three protocol categories since TCP and

UDP are the most prevalent; it would be easy to extend it to more categories if needed. For source and destination port number attributes, we handle well-known and other ports differently as described in Section 3.2.3. Instead of modeling timestamps of individual flows, we model the time deltas between them.

Training hyperparameters. Our models are trained on four Tesla P100 GPUs using the Pytorch toolbox. From the different parameter update rules tried, the Adam [63] algorithm gives best convergence performance and is used for all experiments. The learning rate schedules were manually set to the highest values that allowed fast convergence: 0.001 for mixture density neural layers and 0.01 for softmax layers. The batch sizes are also manually set for the experiments. For UGR16, we use as large a batch size as that showed quick converge; this corresponds to 512 time windows input per batch. We use pre-processing to prepare data batches that can be trained in parallel and accelerate the training and generation process. For mixture density neural layers, we select 10 as the Gaussian components to be learned based on the cross-validation. For the initial convolution network layer parameters, we sample from a Uniform distribution whose boundary is the standard deviation of the kernel size, i.e. $[-\frac{1}{3*3}, \frac{1}{3*3}]$.

Likelihood. For each data point (each row), we can directly calculate the row likelihood by factorization equations. In our case, explicit density generative models (GMM, BN and STAN) clearly define the distribution for each attributes and for those we can evaluate the modeled distribution directly via individual attribute distributions. For simplicity, we discretize continuous variables to validate their negative log-likelihood value for all the baselines and attributes, based on the variable value range and the data set size. In Table 3.3 we report the negative log likelihood (NLL) of a few attributes as modelled by STAN and baselines GMM and BN. Note that we are unable to generate IP addresses and port numbers using GMM and BN, so the NLL for those attributes is not compared in the table. STAN produces

better results for both continuous and discrete attributes.

Model	bytes	packet	time duration	transport protocol
GMM	4.85	3.78	1.81	0.341
BN	3.90	2.62	0.97	0.344
STAN	2.34	1.73	0.59	0.002

Table 3.3: Attribute negative log likelihood of models evaluated on $\mathbf{D}_{validation}$ (lower is better).

Data Synthesis. Once a STAN model is trained on $\mathbf{D}_{historical}$, it is able to sequentially generate any length of *netflow* data $\mathbf{D}_{synth}$. As described in Section 3.2, STAN starts the generation process by sampling from the trained marginal distribution without any input requirement, and then autoregressively generates row by row conditioned on the prior rows context. To fairly compare to $\mathbf{D}_{test}$, we used STAN to generate $\mathbf{D}_{synth}$ that includes 1,208,182 samples for the same set of $\mathbf{D}_{historical}$ users. Note that since we are generating data in one day's range (based on the generated delta time attribute *dt* and the accumulated timestamp), the total number of samples is not directly determined by any hyperparameter. In the rest of this section, we evaluate the comparability between the $\mathbf{D}_{synth}$ and $\mathbf{D}_{test}$.

Distribution and JS divergence. Figure 3.7 shows the individual JS divergence of the marginal distribution of both the continuous and discrete attributes. STAN captures the marginal distribution well for most attributes. Even though GMM precisely models the marginal distribution of the training data set, it does not perform as well as STAN on the test data set. We believe this is because the marginal distribution over days is non-stationary.

Observation 1: *STAN models the marginal distribution better than baseline GMM.*

Domain knowledge test. As described earlier we perform basic sanity checks specified as rules on that need to be satisfied by generated flow-based network data. There are two basic types of rules – those that apply to an individual attribute and those that check relationships

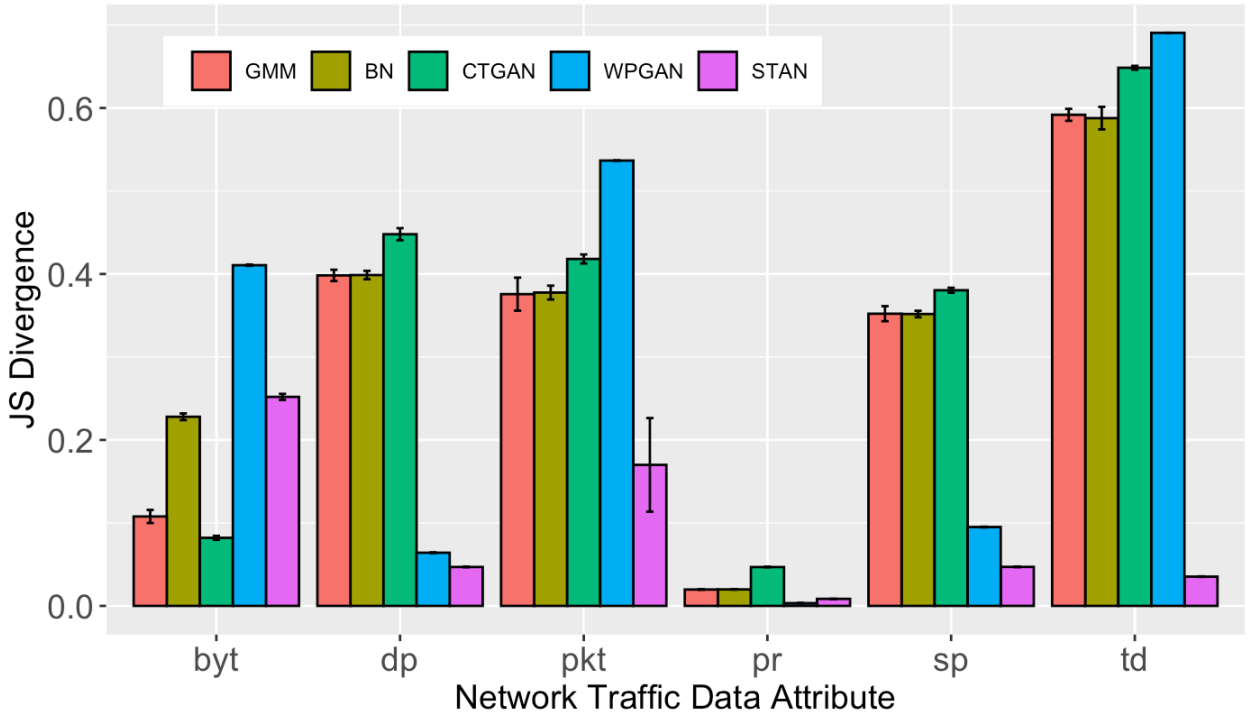


Figure 3.7: JS divergence between attribute marginal distribution between $\mathbf{D}_{test}$ and $\mathbf{D}_{synth}$ from STAN as well as that from baselines.

between multiple attributes. Note that individual tests on port numbers and protocols are not needed since they are modeled as categorical variables. We highlight five tests here which are summarized in Table 3.4. STAN performs well in all three.

- Test 1: Validity of IP address. Source IP address should not be multicast (from 224.0.0.0 to 239.255.255.255) or broadcast (255.xxx.xxx.xxx); Destination IP address should not be of the form 0.xxx.xxx.xxx. (Note that source address can be all zeros, e.g. for DHCP requests.)

- Test 2: Number of bytes/packets minimum size: The minimum value for *pkt* attribute is 1 and the minimum value for *byt* attribute depends on the transport protocol. For a TCP flow packet, the minimum size is 40 bytes (20 bytes for IP header + 20 bytes for TCP header); similarly for a UDP flow packet, the minimum size is 28 bytes (20

bytes for IP header + 8 bytes for UDP header).

- Test 3: Relationship between number of bytes (byt) and number of packets (pkt). Based on the protocol the following relationship exists between these attributes. For a TCP flow,

$$40 * pkt \leq byt \leq 65535 * pkt$$

Similarly, for a UDP flow,

$$28 * pkt \leq byt \leq 65535 * pkt$$

The maximum packet size for both TCP (assuming maximum MTU) and UDP is 64K bytes.

- Test 4: If the transport protocol is not TCP, then the flow should not have any TCP flags.

- Test 5: If the flow describes normal user behavior and the source port or destination port is 80 (HTTP) or 443 (HTTPS), the transport protocol must be TCP[4].

Domain specific checks are performed to verify semantic consistency in the generated data. Depending on the data set, additional such checks can be added. If a generated data set performs poorly on these tests, modeling of different attributes in the traffic data set such as IP addresses and port numbers can be modified to capture the required semantic constructs, including possibly using distinct bins for modeling specific port numbers above 1024 as is now done for port numbers less than 1024.

Marginal distributions. We explore marginal distributions of *netflow* attributes in the

[4]There is an effort to move HTTP/S to UDP, referred to as QUIC[1]. We use this test here since there was no QUIC traffic in our netflow dataset. However, this test may not be relevant for a different dataset.

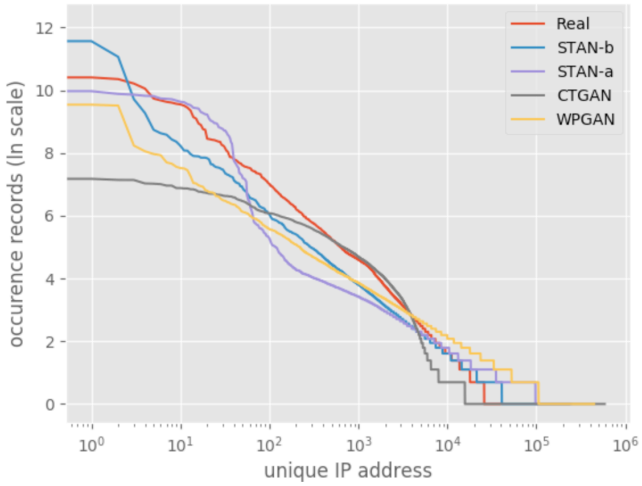
	Test 1	Test 2	Test 3	Test 4	Test 5
Real Data	99	100	100	100	100
GMM	98	66	48	–	77
BN	98	67	49	–	78
WPGAN [108]	99	75	72	97	99
CTGAN [140]	99	93	74	–	99
STAN	**99**	**93**	**93**	**100**	**99**

Table 3.4: Passing percentage of domain knowledge tests. Dash (-) means the methods are not able to generate 'flags' attribute.

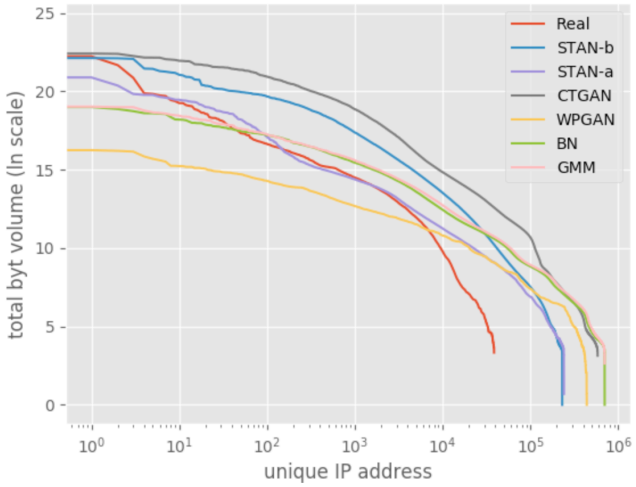
generated data. Figure 3.8a shows the distribution of number of unique IP addresses each user communicates with. Typically, such distributions follow a power law distribution [80]. STAN generated synthetic data comes closest to the real network data, including the unique IP address maximum occurrence value, the total number of different unique IP addresses, and the distribution curve. Meanwhile, Figure 3.8b shows the distribution of the total number of bytes exchanged related to each user (IP address). Again, STAN synthetic data follows the real data distribution closer than the baseline methods. It is worth noting that STAN learned these distributions without any explicit design choices on our part, e.g., in the loss function. It correctly inferred by marginal distribution by virtue of learning the joint distribution.

We also compare port number distribution between real and synthetic data. Based on the real data, we select the top 5 TCP services and top 3 UDP services, which appear most frequently and the occurrence ratio are greater than 1% in the entire TCP or UDP traffic. Figure 3.9a shows the occurrence probability of that service in the entire TCP traffic records. We have 80/443 port for HTTP/HTTPS service (Hypertext Transfer Protocol / Hypertext Transfer Protocol Secure), 25 port for SMTP (Simple Mail Transfer Protocol), 53 port for DNS (Domain Name System), 110 port for POP3 (Post Office Protocol, version 3), and 22 port for SSH (Secure Shell). Similarly in Figure 3.9b, we have 53 port for DNS service , 161 port for SNMP (Simple Network Management Protocol), and 123 port for NTP (Network Time

Protocol). We find STAN performs well at generating a port number distribution similar to real data. Further, this implies that STAN does a good job of capturing application level traffic, which can be mapped to different ports. While WPGAN with IP2Vec component is the second best, the rest of baselines (GMM, BN, CTGAN) perform poorly.



(a) Unique IP address occurrence distribution. The y-axis (ln-scale) is the number of distinct IP addresses contacted.



(b) Unique IP volume (*bytes*) distribution. The y-axis (ln-scale) is the total traffic in bytes related to a user (unique IP address).

Figure 3.8: IP address characteristics. The x-axis (log-scale) represents unique user (IP address) that occurs in the *netflow* traffic.

Figures 3.10a and 3.10b show the marginal distribution of protocol and flags attributes respectively. As expected, TCP is the dominant protocol followed by UDP. For simplicity, we group the other protocols as 'other'. For both protocol and flags, STAN generated data shows a similar distribution to real data.

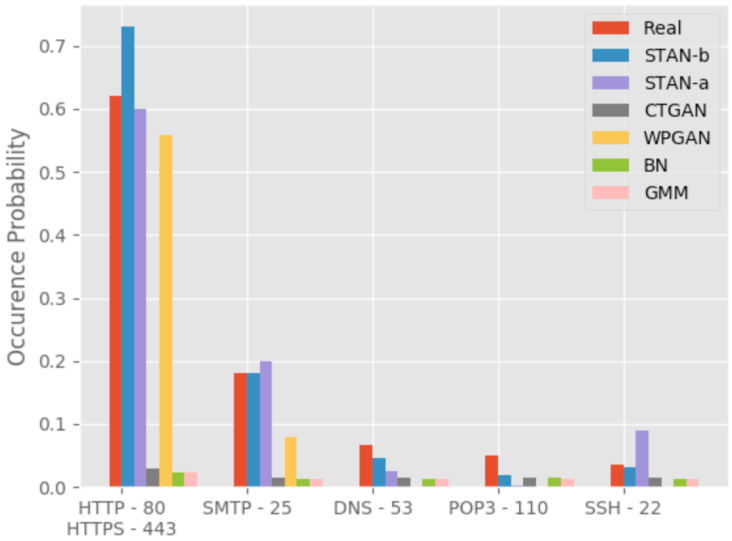
Observation 2: *Compared to baselines, STAN can learn the IP and port characteristics better without using domain knowledge or other design tuning.*

Cybersecurity application tasks. Finally, we test our synthetic data on two cybersecurity machine learning applications, to detect anomalies using self-supervision. One of the tasks is a classification problem, and the other is a regression problem. The goal is to figure out whether it is possible to fully substitute real data with synthetic data for training machine learning models.

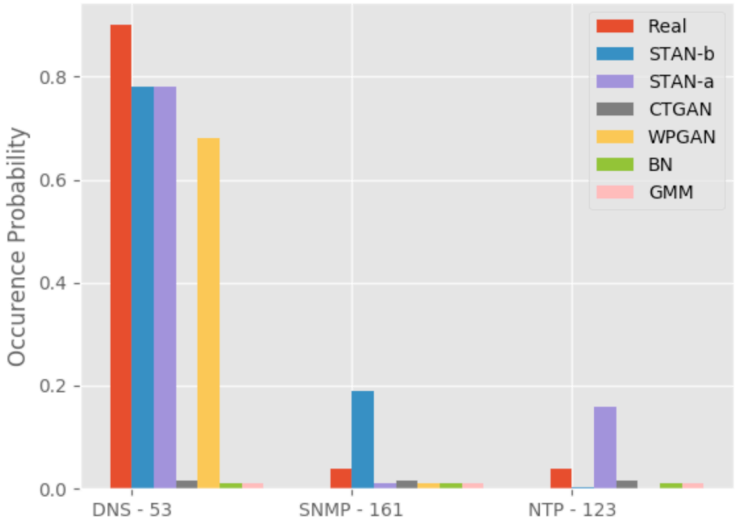
A series of models are trained on real test data. We start our training from using a complete $\mathbf{D}_{test}$ (real data) and successively decrease the amount of real data until no data from $\mathbf{D}_{test}$ is used. Another series of models are trained similarly using the real test data; however, instead of simply removing certain amount of data from $\mathbf{D}_{test}$, we substitute the indicated amount of data with our synthetic data $\mathbf{D}_{synth}$, so that the total amount of data is kept unchanged.

In the following two tasks, we use $\mathbf{D}_{test}$, which is unseen and never used in the synthesizer training process. For the synthetic data $\mathbf{D}_{synth}$, every synthesizer model generates five sets of synthetic data sample, so we can compute error bars. Five-fold cross validation is used to get a robust estimate of the measurements.

Task1: protocol forecasting. Fig. 3.11a shows the F-1 scores achieved by Random Forest models. There are six sets of models. 'Real-Data': these are random forest models trained by reducing the real data; 'STAN': these are random forest models trained by reducing the



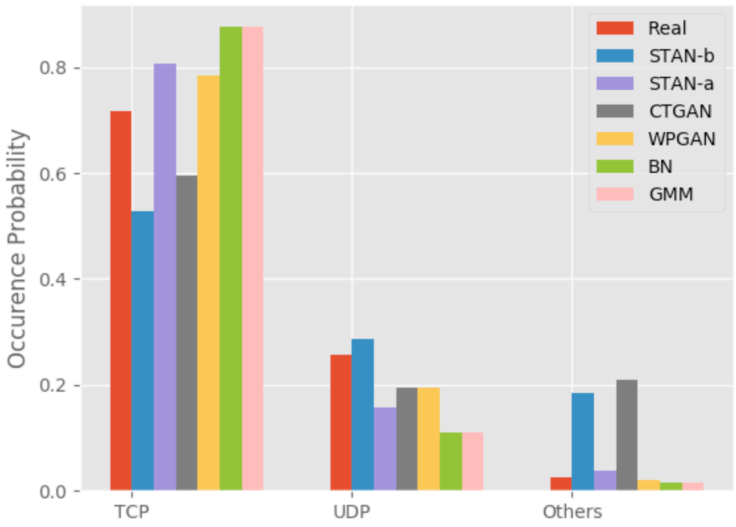
(a) Top 5 most frequently occurring TCP ports and their related service



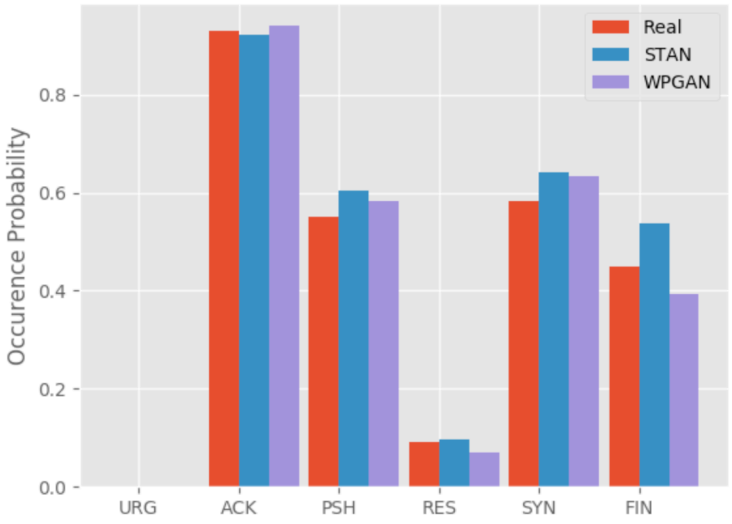
(b) Top 3 most frequently occurring UDP ports and their related service

Figure 3.9: Port number characteristics.

real data, but substituting the reduced data by synthetic data generated by STAN; 'GMM', 'BN', 'WPGAN', and 'CTGAN': these are similar to the 'STAN' models but obtained by substituting the reduced data by the four baselines respectively. The x-axis represents how much real data is used from 100% down to 0%.



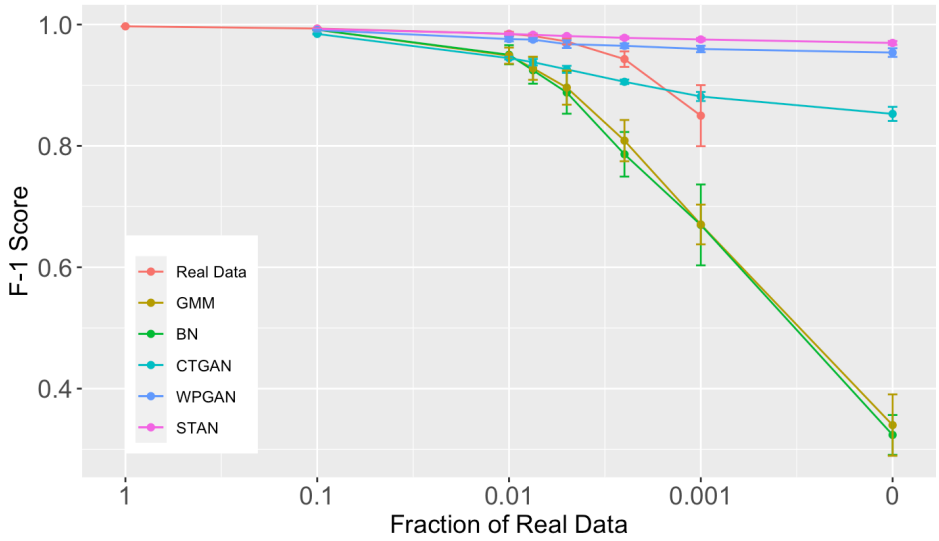
(a) Marginal protocol distribution



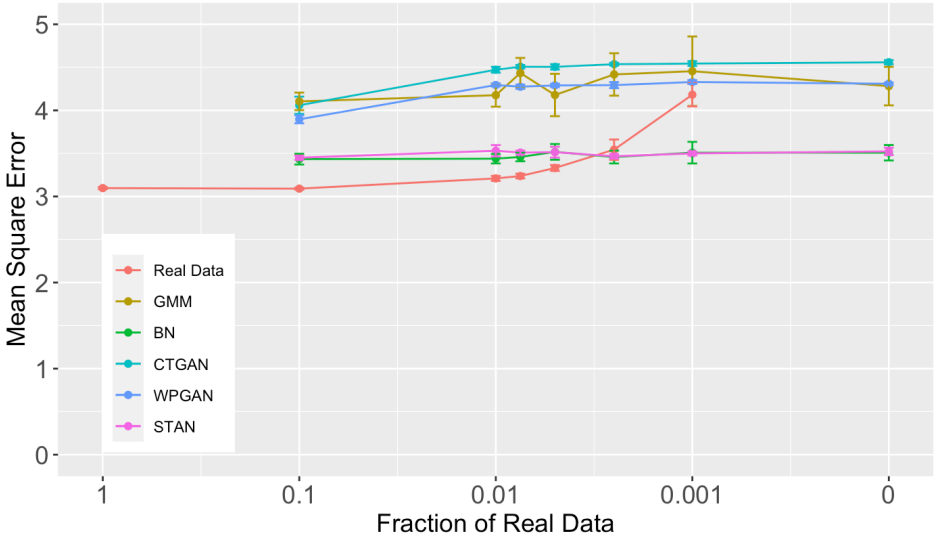
(b) TCP flag distribution, includings six flags: URG, ACK, PSH, RES, SYN, and FIN

Figure 3.10: Transport protocol and TCP flags characteristics.

If we only use real data, the F1 score drops from 0.99 down to 0.97 as the amount of data decreases. Clearly, with no real data, we are unable to train a model. When we substitute real data with that generated by the baselines, the performance drops even quicker, because they do a poor job of capturing the temporal and attribute dependence. Even in the absence of any real data, data generated by STAN results in an F1 score of 0.97, where the drop in



(a) F1-score of same-row Protocol prediction Task



(b) Mean Square Error of *bytes* Value Forecasting Task

Figure 3.11: Real application task performance.

performance is only 2%. That is, the model built with only synthetic data retains 98% of the performance of the all real data trained model.

Task2: *bytes* **value forecasting.** follows a similar setup of experiments as Task1. Figure 3.11b shows the mean square error achieved by a neural network regression model. The plot shows that STAN and Bayesian network (BN) outperform the other three baseline

models. Building a Bayesian network with domain knowledge typically performs better than GANs [140].

In our experiments, BN is optimized specifically for the *bytes* sequential value. However, STAN has two advantages over the Bayesian network. First, users do not need the domain knowledge required for Bayesian network implementation. Secondly, there is no inherent bias attributable to an expert unlike traditional Bayesian networks. Similar to the first task, the penalty for using only STAN generated data (with no real data) is low, an increase of 13% in the mean square error.

Observation 3: *Compared to BN, STAN performs better on task1 and as well on task2 without requiring any domain knowledge.*

Observation 4: *Even with 0% real data, STAN models task1 and task2 with only a small drop in accuracy.*

3.4 Discussion

In this chapter, we discussed the use of a novel CNN combined with a mixture density network to auto-regressively generate synthetic tabular data with both row and column dependencies. However, due to inherent limitations of deep neural networks—such as the need for complex feature engineering—in the next chapter 4, we explore fine-tuning a large language model to generate synthetic tabular data and overcome these limitations. The approach in chapter 4 aims to enhance the fidelity of the generated tables, particularly in terms of row consistency and validity.

Chapter 4

LLMs for STG with Higher Fidelity

While Chapter 3 demonstrated promising performance using deep neural networks for synthetic tabular data generation, certain limitations—such as the need for complex feature engineering—remain. To address these, this chapter explores the use of large language models (LLMs), leveraging their strong priors and adaptability to reduce manual effort and further improve table fidelity, particularly in terms of column dependencies and semantic consistency.

Large language models (LLMs) have found applicability in a rich diversity of domains, well beyond their original roots [3, 17, 81, 87, 99, 109, 124]. As so-called foundation models [68] they have been shown to be re-targetable to a variety of downstream tasks. Our focus here is to view LLMs as raw synthetic tabular data generators rather than as supporting an analysis or discovery task. Arguably, LLMs are adept at synthetic generation of text, images, videos, code, documentation, and many other modalities. The use of LLMs to generate tabular data is quite understudied.

Such synthetic tabular data generation is integral to ML pipelines, e.g., too augment training data, replace sensitive information, and even to power advanced platforms like DeepSeek [51, 70]. However, the unique characteristics of tabular data manifest as challenges in an LLM-based generative context. The most popular incarnations of language models are auto-regressive models, e.g., LLama [124], GPT-x [87], DeepSeek [51, 70], wherein each word or token is generated conditional on past tokens in a sequential manner using attention

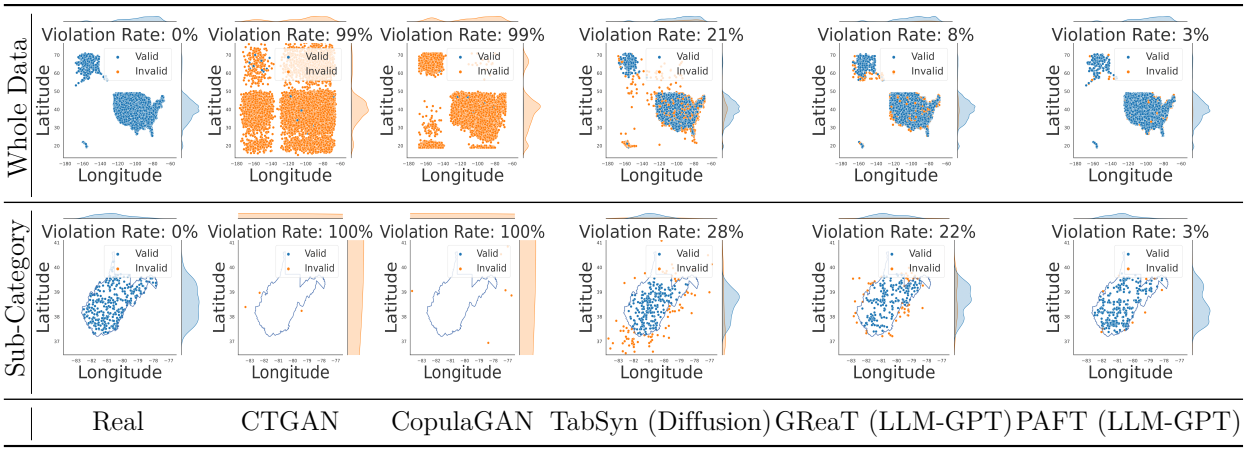
models. In a synthetic data context, each 'sentence' typically represents a row of tabular data, and each 'word' corresponds to an attribute in that row. The previous state-of-the-art models (GReaT [15]), has advocated the use of random feature orders but as we show in Table 4.1, when fine-tuning is done with random feature orders, key relationships are often not captured or, worse, violated. In particular, with tabular data, there are numerous functional dependencies at play and as a result, generating tokens in random orders is bound to cause violations. There is thus an **'impedance mismatch' between autogressive LLMs and synthetic data generation.**

The main contributions of this chapter are:

- We highlight an important deficiency with using LLMs for synthetic tabular data generation and explore the performance of many state-of-the-art generation models in the context of composite and multi-category tabular schema.
- We inject knowledge of pre-existing functional relationships among columns into the autoregressive generation process, so that the generated synthetic data respects more real constraints. In particular, we present a taxonomy of functional dependencies (FDs) whose discovery and organization into a column dependence graph supports their incorporation into the LLM fine-tuning process via a permutation function, leading to our approach dubbed Permutation-aided Fine-tuning.(PAFT).
- We evaluate the performance of PAFTon a range of datasets featuring a diverse mix of attribute types, functional dependencies, and complex relationships. Our results demonstrate that PAFTis the state-of-the-art in reproducing underlying relationships in generated synthetic data.
- Finally, we demonstrate through rigorous experiments that relying just on standard univariate distribution, bivariate correlation, and even the evaluation of downstream

machine learning models (which primarily focuses on predicting a single column in a dataset) is grossly insufficient for assessing the quality of synthetic data and propose systematic remedies like measuring violation rates of known domain rules.

Table 4.1: Comparison of multiple synthetic data generation approaches. The second row showcases the state of West Virginia (WV), which is a subset of the whole data. In the figures, the solid line represents the official border of West Virginia and the Blue and Orange colors indicate the legal and illegal samples in the synthetic data respectively.



4.1 Challenges to Synthetic Table Generation in the Current LLM Paradigm

Admittedly, the sure-fire way to check if a dataset has been faithfully modeled is to see if the joint distribution of its features is captured accurately. Most current tests of synthetic data generation quality focus on fidelity to single-column distributions, or to multi-column distributions. For multiple variables, measures such as machine learning efficiency (MLE) [15, 33, 140, 146] are frequently used to construct classifiers or regressors.

A key lesson from probabilistic graphical model research [64] is that factorizing joint distributions into products of conditionals (e.g., Bayesian networks) dramatically helps reduce the

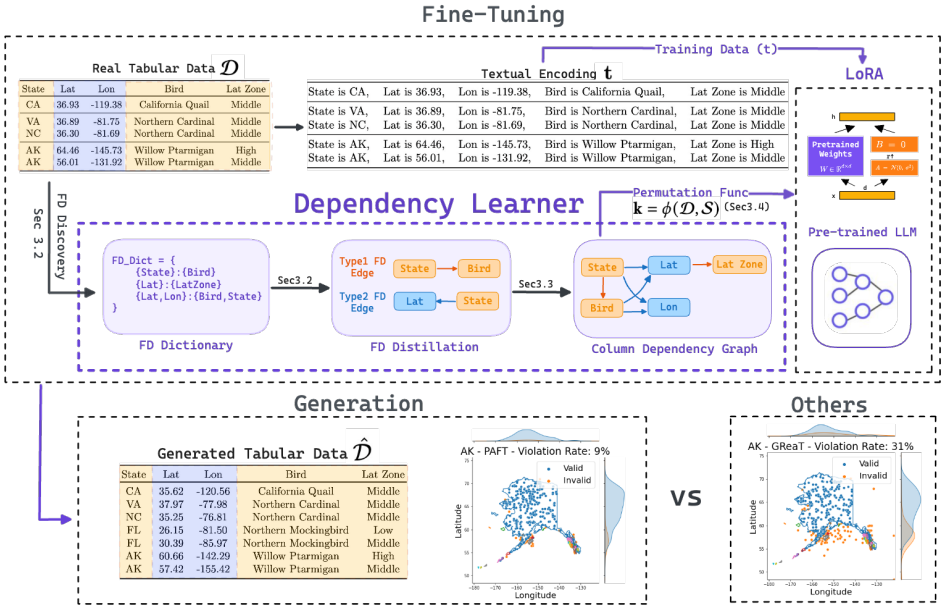


Figure 4.1: Overview of the proposed Permutation-Aided Fine-tuning (PAFT) approach.

number of parameters necessary to capture the underlying data characteristics. A similar lesson from database research [44] is that modeling functional dependencies (FDs) in data helps reduce redundancy in modeling and storage. These lessons, i.e., that **order matters**, continue to apply in the LLM era but are not internalized in our prompt ordering, fine-tuning, or evaluation methodologies. Other researchers have noted the importance of ordering in LLMs [27, 95] but this lesson has not been leveraged to improve synthetic table generation by LLMs. In the absence of leveraging good feature orders, existing approaches either focus on 'one order', 'no order', or 'all orders'. All of these approaches severely limit the quality of generated synthetic data.

As an example, Table 4.1 presents a case study on using models to generate synthetic data of locations in various states of the USA. The data is of the form (state, latitude, longitude) where the attributes adhere to the FD: $\{\text{latitude}, \text{longitude}\} \rightarrow \text{state}$. Existing methods can generate satisfactory univariate distributions (as shown in the border of the top row plots) but fail to capture the joint distribution (center of the top row plots) and conditional

distributions across subcategories (bottom row plots).

In summary, feature ordering can be both a nuisance and a gift. It is a nuisance because it demands additional constraints to be modeled. It can be a gift because it suggests ways to sequentially generate data even by autoregressive LLMs. Our proposed approach (PAFT) aims to achieve an optimal permutation order for fine-tuning LLMs.

Figure 4.1 shows an overview of the proposed permutation aided fine-tuning approach (PAFT). A typical workflow is 1) Textual Encoding (Section 4.2.1) 2) Functional Dependency (FD) discovery (Section 4.2.2) 3) FD Distillation (Section 4.2.2) and 4) Feature Order Permutation Optimization (Section 4.2.3). Our fine-tuning and sampling strategy is explained in Section 4.2.4.

4.2 PAFT: Permutation-Aided Fine-Tuning

Problem Setup. Let $\mathcal{D}$ represent a table with n rows (i.e., records) and m columns (i.e., attributes a.k.a schema). Let each record be represented by vector $\mathbf{x}_i$ and further let x_{ij} represent the element value of the j^{th} attribute of record $\mathbf{x}_i$. Hence each row $\mathbf{x}_i \in \mathcal{D}$ represents an individual record and each column $\mathbf{x}_{(:,j)} \sim \mathcal{X}_j$ can be considered sampled from a random variable $\mathcal{X}_j$ that governs the distribution of attribute j. Finally, let $i \in [1..n]$ and $j \in [1..m]$. Realistically, tabular data $\mathcal{D}$ is frequently a mixture of categorical and continuous attributes, hence each $\mathcal{X}_j$ can be a categorical or continuous random variable. If $\mathcal{A} = \{\mathcal{X}_1, \mathcal{X}_2, \ldots, \mathcal{X}_m\}$ represents the collection of random variables, then the table generation process aims to sample from a joint distribution $\mathbb{P}(\mathcal{A}) = \mathbb{P}(\mathcal{X}_1, \mathcal{X}_2, \ldots, \mathcal{X}_m)$. This joint distribution is usually a complex, high-dimensional distribution and, most importantly, unknown. The goal of learning an effective tabular data generator $p_\theta(\cdot)$ is to enable $p_\theta(\cdot)$ to learn a *faithful* approximation $\mathbb{P}(\mathcal{A}|\mathcal{D})$ of the data generation process distribution $\mathbb{P}(\mathcal{A})$

using the data sample $\mathcal{D}$ such that $\mathbb{P}(\mathcal{A}|\mathcal{D}) \approx \mathbb{P}(\mathcal{A})$. Once such an effective model $p_\theta(\mathcal{D})$ is trained, it can be employed to generate large volumes of seemingly *realistic* synthetic data $\hat{\mathcal{D}} \sim \mathbb{P}(\mathcal{A}|\mathcal{D})$.

4.2.1 Tabular Data Generation with LLMs

While training $p_\theta(\cdot)$, it is usually assumed that all records $\mathbf{x}_i \in \mathcal{D}$ are independent. Generating new data samples $\hat{\mathbf{x}}_i \in \hat{\mathcal{D}}$ can be done in various ways (e.g., see [139, 140, 150]) which aim to directly estimate the joint distribution $\mathbb{P}(\mathcal{A})$ or, as is done here in PAFT, where $\mathbb{P}(\mathcal{A})$ is estimated by an autoregressive LLM based generative process, as a product of multiple conditional densities governed by the input ordering.

Autoregressive LLM models are pre-trained to maximize the likelihood of *target* token $x_{ij} \in \mathcal{D}$, conditioned upon the autoregressive context $\mathbf{x}_{(i,1:j-1)} \in \mathcal{D}$ where $\mathcal{D}$ is the training corpus comprising a large amount of textual data (in the pre-training context). Eq. 4.1 defines the general training criterion of LLM training using the self-supervised next-token prediction task with 'w' denoting the context length.

$$\mathcal{L}(\theta; \mathcal{D}) = -\sum_{\mathbf{x}_i \in \mathcal{D}} \sum_{j=1}^{w} \log \mathbb{P}(x_i | \mathbf{x}_{(i,1:j-1)}). \tag{4.1}$$

The generation of a single instance (i.e., database record) $\mathbf{x}_i \in \mathcal{D}$ is given by Eq. 4.2:

$$\mathbb{P}(\mathbf{x}_i) = \mathbb{P}(x_{i,1}, ..., x_{i,m}) \simeq \prod_{j=1}^{m} \mathbb{P}(x_{i,j} | x_{i,1}, .., x_{i,j-1}). \tag{4.2}$$

Specifically, each database record is generated as a product of conditional distributions.

Input Encoding. To support the processing of our records $\mathbf{x}_i \in \mathcal{D}$ by a pre-trained LLM,

we adopt the following encoding:

$$
\begin{aligned}
t_{i,j} &= [c_j, \text{`}is\text{'}, x_{i,j}, \text{`,'}], i \in \{1, .., n\}, j \in \{1, .., m\}, \\
\mathbf{t}_i &= [t_{i,1}, t_{i,2}, .., t_{i,m}], i \in \{1, .., n\}.
\end{aligned}
\tag{4.3}
$$

In Eq. 4.3, c_j represents the attribute name of the j^{th} database column while $x_{i,j}$ represents the actual value of the j^{th} column for the i^{th} record. Further, we can assume we have a mechanism to obtain a *feature order permutation* $\mathbf{k}$ to govern the order of the attributes in $\mathbf{t}_i$, such that $\mathbf{t}_i(\mathbf{k}) = [t_{i,k_1}, t_{i,k_2}, .., t_{i,k_m}]$ (where $i \in \{1, .., n\}, k_j \in \{1, .., m\}$.), represents the same record but with the attribute order governed by the permutation $\mathbf{k}$, This definition admits the random feature order as a special case in which $\mathbf{k}$ is a random permutation.

Since we consider autoregressive LLM-based generative models, employing the chain rule to sequentially produce each column of a table record $\mathbf{t}_i$, we can view each generation step as *approximating* the joint distribution of the table columns as a product of conditional distributions (i.e., $\mathbb{P}(t_{i,1}, ..., t_{i,m}) \simeq \prod_{j=1}^{m} \mathbb{P}(t_{i,j}|t_{i,1}, .., t_{i,j-1})$). However, as the number of columns increases and the relationships between columns get more conditional, the likelihood of encountering training and sampling bias due to class imbalance also rises [140]. To minimize such adverse effects, we can consider injecting knowledge of the pre-existing functional relationships among columns, to govern the autoregressive generation process. To infer such functional relationships, we leverage a learned dependency graph derived from functional dependency (FD) relations which enables us to effectively determine the appropriate training and sampling sequence. This, in turn, allows us to alleviate potential biases during training by establishing a *generation curriculum* leading to improved estimation accuracy of the joint distribution $\mathbb{P}(\mathbf{t}_i)$ in auto-regressive prediction $\mathbb{P}(t_{i,k_1}, ..., t_{i,k_m}) \simeq \prod_{j=1}^{m} \mathbb{P}(t_{i,k_j}|t_{i,k_1}, .., t_{i,k_{j-1}})$, where the ordering $t_{i,k_1} \ldots, t_{i,k_m}$ is obtained by a feature order permutation function $\mathbf{k} = \phi(\mathcal{D}, \mathcal{S})$.

We detail the requisite background and design of $\phi(\mathcal{D}, \mathcal{S})$ in sections 4.2.2 and 4.2.3.

4.2.2 Discovery and Distillation of Functional Dependencies (FD)

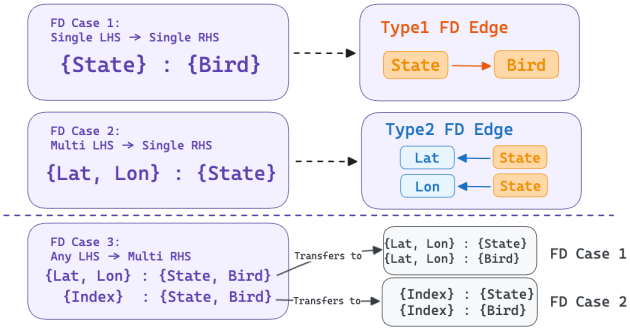
A functional dependency (FD) is a relationship R in schema S that exists when a subset of attributes $A \subset S$ uniquely determines another subset $B \subset S$ of attributes. We succinctly represent an FD as $R : A \rightarrow B$ which specifies that B is functionally dependent on A.

Definition 4.1 (Schema-Level FD)**.** With A, B being two disjoint subsets of the schema (columns) of table $\mathcal{D}$, a schema-level FD F associated with $\mathcal{D}$ has the form: $F : A \rightarrow B$.

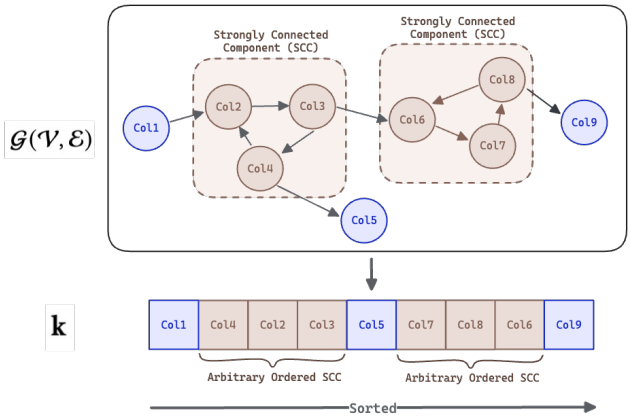
We leverage FD discovery techniques to govern the order of the autoregressive data generation process in PAFT. A large body of research from the database literature on FD discovery [88, 89, 149] can be leveraged in PAFT, including methods that account for noisy FDs [149]. In this work, we focus on leveraging schema-level FDs discovered using a state-of-the-art FD discovery algorithm [88] to govern the autoregressive data generation process.

FD Distillation. The result of traditional FD discovery, yields complex (i.e., multi-attribute) functional dependencies between columns which are ambiguous to resolve in an autoregressive generation setting. Hence we undertake an intermediate *FD distillation* step to simplify multi-attribute functional dependencies into multiple single attribute FDs as detailed below.

We first construct a dependency graph model $G(\mathcal{V}, \mathcal{E})$ where $\mathcal{V}$ represents the set of vertices with each $v \in \mathcal{V}$ representing an attribute in $\mathcal{S}$ and $e_{ij} \in \mathcal{E}$ representing an edge relation from attribute v_i to attribute v_j. Further, we consider two types of edges in $\mathcal{E}$, specifically each e_{ij} may be a *type-1* edge or a *type-2* edge (defined next). The two edge types (i.e., *type-1*, *type-2*) in $\mathcal{E}$ are derived from three classes of FDs, as shown in Fig. 4.2a. Subsequently, we



(a) There are two types of column dependency edges for three types of functional dependencies (FDs), which are distinguished by the left-hand side (LHS) and right-hand side (RHS) in the FD.



(b) DAG for column functional dependency derived by expanding SCC super nodes and retrieving a fully flattened, ordered structure.

Figure 4.2: FD-Distillation and Dependency Graph Sorting for automatically extracting order permutations from tables.

proceed to examine each individual case: 1) Single attribute left-hand side (LHS) and single attribute right-hand side (RHS) 2) Multi-attribute LHS and single-attribute RHS. 3) single or multi-attribute LHS and multi-attribute RHS. We shall use the example table in Fig. 4.1 to define each FD case.

[**Type-1 Edge**]. Let us consider an example of FD Case 1, wherein the column *State* functionally determines column *Bird*. In such a case, we enforce that the value for the attribute *State* be generated prior to the value for the attribute *Bird*. Accordingly, a forward

directed edge from *State* to *Bird* is created in the column dependency graph $\mathcal{G}$. We term such forward directed edges as type-1 edges in $\mathcal{G}$. [**Type-2 Edge**]. The other type of edge in $\mathcal{G}$, arises when we encounter an FD with a multi-attribute LHS and a single attribute RHS (i.e., FD Case 2). As per FD Case 2, the values of multiple columns in the LHS would collectively decide the value of the column on the RHS. As an example in Fig. 4.2a, the tuple of columns *Latitude, Longitude* functionally determines *State*. For such FDs, two backward edges are added in the column dependency graph $\mathcal{G}$, connecting *State* to both *Latitude* and *Longitude*. We term such backward directed edges as type-2 edges in $\mathcal{G}$.

FD Case 3 relationships are ones where the RHS has multiple attributes and the LHS could have single or multiple-attributes. Such relationships do not directly result in an edge in our dependency graph $\mathcal{G}$. Instead, as classically done in FD literature [88], we subject such FD relationships to an intermediate decomposition step. Specifically, the multi-attribute RHS of FD Case 3 relationships is decomposed into multiple single-attribute RHS dependencies each comprising the original LHS. Further, in each of these new decomposed relationships, if the LHS is single-attribute, it is treated as a FD *Case 1* relationship (i.e., a directed edge from LHS to RHS is added to $\mathcal{G}$), else it is handled as an FD *Case 2* relationship, wherein for each attribute in the multi-attribute LHS, a *backward* dependency edge from the single-attribute RHS is added to the dependency graph $\mathcal{G}$. Therefore, for every column in the right-hand side (RHS), we employ either *type-1* or *type-2* edge construction, depending on the value of its left-hand side (LHS). Full procedure detailed in Appendix A.2.1 Algorithm 3.

4.2.3 Putting It All Together

Until this point, our construction of the graph $\mathcal{G}$ has only been limited to considering pairwise relationships between columns in $\mathcal{D}$. Graph properties like functional dependency tran-

sitivity, require us to obtain a total ordering on the nodes $v \in \mathcal{V}$ of $\mathcal{G}(\mathcal{V}, \mathcal{E})$ that is deterministic in nature for effective auto-regressive LLM training. This implies that in order to obtain a feature order permutation ($\mathbf{k}$) from the derived functional dependency relationships (Sec. 4.2.2), a computation must be performed on the entire dependency graph $G(\mathcal{V}, \mathcal{E})$.

We define this task of obtaining a total feature order $\mathbf{k}$ from $\mathcal{G}(\mathcal{V}, \mathcal{E})$ as an optimization step which seeks to produce $\mathbf{k}$ while minimizing the number of violated relationships in $\mathcal{G}(\mathcal{V}, \mathcal{E})$. Appendix A.2.1 Algorithm 4 outlines this procedure.

Consider the trivial case of having an empty FD graph $\mathcal{G}$ i.e., $|\mathcal{E}| = \emptyset$. If the columns of a table are not functionally dependent on each other, then the order of generation is not important and $\mathbf{k}$ can be some arbitrary permutation of the columns in $\mathcal{S}$. For all other cases, our total feature ordering algorithm operates in three phases, as shown in Fig. 4.2b. [**Phase 1: Condensation**]. It is apparent that if a graph is not a directed acyclic graph (DAG), there is no optimal solution to the total feature ordering problem. In other words, there must be FDs that cannot be satisfied in the resulting total order permutation $\mathbf{k}$. In such cases, we compute the strongly connected components (SCC), and condense them into super nodes, thus transforming the original graph into a DAG. [**Phase 2: Ordering**]. An application of a topological sort onto the DAG from Phase 1 will result in a total feature ordering with all SCCs in $\mathcal{G}$ compressed into super nodes. [**Phase 3: Expansion of SCC**]. Once the topological sort is conducted in Phase 2, we finally expand the SCC super nodes (via. arbitrary ordering) such that although the intra-SCC ordering of the nodes within the SCC is arbitrary, their ordering relative to non-SCC nodes is maintained.

4.2.4 Synthetic Data Generation using PAFT

After the optimized feature order permutation is obtained (via Appendix A.2.1 Algorithm 4), we fine-tune the LLM with the textually encoded table record $\mathbf{t}_i$ such that the auto-regressive generation process is governed by the optimal feature order permutation $\mathbf{k} = \phi(\mathcal{D}, S)$. Specifically, we generate the table governed by order $\mathbf{k}$ as defined in Eq. 4.4.

$$\mathbb{P}(\mathbf{t}_i) = \mathbb{P}(t_{i,k_1}, ..., t_{i,k_j}) \simeq \prod_{j=1}^{m} \mathbb{P}(t_{i,k_j} | t_{i,k_1}, .., t_{i,k_{j-1}}). \tag{4.4}$$

We employ the Low-Rank Adaptation (LoRA) fine-tuning strategy [56]. To generate synthetic rows, we first sample the initial token $p(t_{i,k_1})$ from the marginal distribution of variable k_1 in actual training data, and then use Eq. 4.4 to sequentially sample subsequent tokens $p(t_{i,k_j})$, where $j \in 2, ..., m$.

4.3 Experimental Evaluation

We conduct an exhaustive empirical evaluation of PAFTto assess its ability to reproduce realistic data distributions, superiority over other competing approaches, and most importantly substitutability of data generated by PAFTin the context of a larger ML pipeline. More specifically, the questions we seek to answer are:

1. Does PAFT-generated synthetic data accurately capture conditional distributions within categories? (Sec 4.3.1)

2. Does PAFT generate data respect the consistency of intrinsic data characteristics? (Sec 4.3.2)

3. Does the synthetic data generated by PAFT pass the *sniff* test? (Sec 4.3.3)

4. Can data generated by PAFT replace real data in downstream ML model training? (Sec 4.3.4)

5. Do the data sets generated by PAFTadhere to real distributions and possess mode diversity? (Sec 4.3.5)

6. Can PAFT enhance the stability in generating high quality tables, resulting in a faster sampling phase (Sec 4.3.6)?

7. Do newer generations of LLMs obviate the need for PAFT? (Sec 4.3.7)

Datasets. We evaluate the effectiveness of PAFTthrough experiments on six real datasets commonly used in synthetic table generation studies such as GReaT [15] CTGAN [140]. These are Beijing [25], US-locations [41], California Housing [85], Adult Income [13], Seattle [112], and Travel [121]. Separately, we also generate a set of four simulated datasets for class-mixture distributions (as described in Algorithm 5 Appendix A.3.1). Details of dataset statistics are in the Appendix A.3.1.

Baselines. For benchmarking, we organize baselines that utilize current deep learning approaches for synthetic data generation (CTGAN [140], CopulaGAN [140], TabSyn [146]) and the most advanced synthetic table generator with LLM fine-tuning GReaT [15]. To guarantee an equitable comparison, we employ the Distill-GReaT model for both LLM techniques in all tests.

Reproducibility. Each baseline (CTGAN, CopulaGAN, TabSyn, GReaT) adheres to the recommended hyperparameters and utilizes officially released API tools: Synthetic Data Vault [92] and GReaT [15]. For a fair comparison of GReaT and PAFT, the LoRA fine-tuning parameters are set the same as: Lora attention dimension $r = 16$, alpha parame-

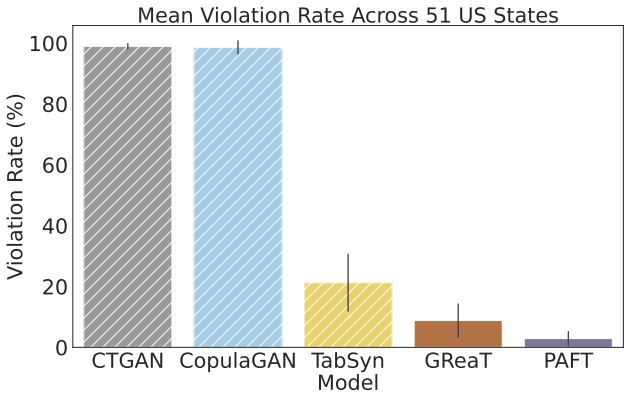


Figure 4.3: For a composite dataset, US-locations, this comparison examines state-specific violation rates across different synthetic data generation approaches. The error bars represent standard deviation. The states on the x-axis are ordered by decreasing violation rates. PAFT significantly reduces state-specific violations in the composite dataset.

ter for Lora scaling $lora\_alpha = 32$, the names of the modules to apply the adapter to $target\_modules = c\_attn$, the dropout probability for Lora layers $lora\_dropout = 0.05$, $bias = none$.

Parameters for MLE and Discriminator Models. We utilize neural network, linear/logistic regression, and random forest models from the Scikit-Learn package for the ML efficiency and discriminator experiments. The exact hyperparameters for each model are detailed in Appendix Table A.2. Every result is evaluated through the 5-fold cross-validation process.

Low-order statistics [146] of column-wise data distribution and pair-column correlation are calculated with the SDV library [1].

Table 4.2: Violation rates across different categories in the same dataset highlight the effectiveness of PAFT in addressing conditional distribution challenges in mixed-category datasets. Notably, even though baseline models may perform well in the MLE task or distribution evaluation, they often fail in practical boundary checks, such as functional dependencies (FDs). **We can see that states with the lowest violation rates are the easiest to model, i.e. rectangular.**

States with the **Highest** Violation Rates (↓) (Sorted by LLM baseline: GReaT)					
Category (State)	MO	AK	KY	FL	WV
					
Real Data	0%	0%	0%	0%	0%
CTGAN	98.0%	99.3%	97.4%	99.9%	100%
CopulaGAN	99.3%	84.6%	99.3%	97.8%	100%
TabSyn	10.5%	17.2%	22.9%	25.1%	28.1%
GReaT	17.1%	18.4%	20.6%	21.9%	22.3%
PAFT	**1.9%**	**5.1%**	**3.4%**	**3.4%**	**3.4%**

States with the **Lowest** Violation Rates (↓) (Sorted by LLM baseline: GReaT)					
Category (State)	CO	KS	NM	SD	UT
					
Real Data	0%	0%	0%	0%	0%
CTGAN	96.5%	97.0%	99.5%	97.5%	99.3%
CopulaGAN	98.3%	98.8%	98.4%	98.3%	98.4%
TabSyn	11.3%	18.4%	6.2%	17.3%	16.6%
GReaT	0.5%	0.9%	1.2%	1.5%	2.1%
PAFT	**0.0%**	**0.3%**	**0.9%**	**0.9%**	**0.0%**

4.3.1 RQ1: Does PAFT-generated synthetic data accurately capture conditional distributions within categories?

Capturing and generating the diversity in a multi-class setting (composite dataset) has been shown to be a challenging practical problem [77]. In addition to Table 4.1, Fig. 4.3 and Table 4.2 further illustrate the widespread nature of this issue within the same composite dataset, where the GAN-generated results exhibit an almost 100% violation rate. The term 'composite dataset' refers to a dataset in which the distributions across different subcategories show significant variances. The LLM model GReaT, which employs random order permutation, shows a significant improvement in maintaining conditional distributions. Nevertheless, its performance remains inconsistent due to unevenness within categories, resulting in violation rates ranging from 0.5% to 22.3%. In contrast, PAFTconsistently controls deviations from the real facts of conditional distributions, maintaining them within a range of 0% to 5%. This reliability holds even in challenging subcategory cases where baseline methods underperform.

4.3.2 RQ2: Does PAFT generate data respecting the consistency of intrinsic data characteristics?

In addition to the dissimilar sub-category challenge, we also investigate whether unsatisfactory conditional distributions exist in general across various datasets, and whether the PAFT method can address these issues. In line with this, we conducted rule checks that were derived from real-world scenarios and subsequently evaluated the generated data from all models. Table 4.3 displays the violation rate in the generated data. From the table, we observe that PAFTadheres to data's characteristics more faithfully (i.e., significantly fewer

[1] https://docs.sdv.dev/sdmetrics/reports/quality-report

Table 4.3: Datasets have intrinsic characteristics like functional dependencies, range restrictions, and other domain knowledge. Results are averaged over five random runs, with variance detailed in Appx. Table A.3.

Intrinsic Fact (Dataset)	**Fact Violation Rate (↓)**				
	CTGAN	Cop.GAN	TabSyn	GReaT	PAFT
Lat-long→State (US-locations)	99.2%	98.5%	21.5%	8.2%	**2.9%**
Lat-long → CA (California)	47.6%	99.9%	8.8%	5.4%	**1.3%**
Med. house price → $[1.4e^5, 5e^5]$ (California)	1.5%	0.01%	**0.0%**	**0.0%**	**0.0%**
education→ education-num (Adult)	83.9%	19.1%	1.4%	1.2%	**0.5%**
Zipcode→Seattle (Seattle)	**0.0%**	99.9%	**0.0%**	**0.0%**	**0.0%**

rule violations) than baseline methods, by learning together with functional dependencies.

4.3.3 RQ3: Does the synthetic data generated by PAFT pass the *sniff* test?

Similarly to the analysis conducted in recent work [15] (GReaT), we employ the random forest (RF) algorithm to train discriminators to distinguish real data (labeled True) and synthetically generated data (labeled False). Subsequently, we test performance on an unseen set (consisting of 50% synthetically generated data and 50% real data). In this experiment, scores represent the percentage of correctly classified entities. In this case, an ideal accuracy score would be close to 50%, which means the discrimniator fails to distinguish between real and synthesized data. The scores are shown in Table 4.4 and indicate that the data generated

by PAFTare most indistinguishable from real data, even by powerful discriminative models.

Table 4.4: Privacy Discriminator Performance. The scores stand for the accuracy for detecting real or fake data, where the ML models are trained using 50% real data and 50% random data. An ideal accuracy score is 50, indicating the model cannot distinguish between real and synthesized data. The best results are marked in **bold**, the second-best results are underlined. Results are averaged over five random runs, with variance detailed in Appx. Table A.7.

Data Sniff Test - ML Discriminator Accuracy (Values closest to 50% are best.)					
Method	CTGAN	Co.GAN	TabSyn	GReaT	**PAFT**
Beijing	99.16%	98.69%	50.97%	51.1%	**50.09%**
US-locations	99.94%	97.74%	51.97%	50.47%	**50.01%**
California	98.35%	86.64%	50.64%	53.74%	**49.89%**
Adult	94.43%	59.82%	51.64%	**51.12%**	48.75%
Seattle	87.61%	85.7%	**50.12%**	68.27%	47.21%
Travel	77.96%	74.14%	**50.66%**	62.49%	48.18%

4.3.4 RQ4: Can data generated by PAFT replace real data in downstream ML model training?

We next assess the effectiveness of the generated (synthetic) data by comparing the performance of discriminative models trained on synthetic data versus real data for their target tasks. Models tested include random forests (RF), linear regression (LR), and multi-layer perceptron (NN). As shown in Table 4.5, PAFT is best or second best in over 80% of (dataset, method) combinations.

Table 4.5: MLE performance. MLE performance: For datasets with regression tasks (marked *), performance is evaluated using MAPE (where lower scores are better). For datasets with classification tasks, accuracy is used (where higher scores are better). The best results are marked in **bold** and the second-best results are underlined. Results are averaged over five random runs, with variance in Appx. Table A.4.

Regression		**MAPE (↓)** Over Different Methods					
Dataset(*)		Orig.	CTGAN	Co.GAN	TabSyn	GReaT	**PAFT**
Beijing	**RF**	0.41%	2.49%	2.15%	0.7%	0.57%	**0.52%**
	LR	1.37%	2.23%	1.55%	1.25%	**0.97%**	1.34%
	NN	0.99%	2.44%	2.83%	1.01%	1.16%	**0.95%**
Calif.	**RF**	0.18%	0.65%	0.39%	0.22%	0.25%	**0.20%**
	LR	0.30%	0.54%	0.5%	0.30%	**0.29%**	0.31%
	NN	0.34%	0.53%	0.47%	0.29%	0.3%	**0.27%**
Seattle	**RF**	0.33%	0.76%	0.38%	0.30%	0.35%	**0.28%**
	LR	0.29%	0.74%	0.32%	**0.23%**	0.33%	0.29%
	NN	0.28%	0.71%	0.38%	0.28%	0.33%	**0.27%**
Classif.		**Accuracy (↑)** Over Different Methods					
Dataset		Orig.	CTGAN	Co.GAN	TabSyn	GReaT	**PAFT**
US-loc.	**RF**	99.95%	7.17%	45.33%	**99.99%**	99.84%	99.91%
	LR	46.1%	5.11%	31.08%	43.69%	45.65%	**49.41%**
	NN	99.85%	7.56%	53.34%	**99.64%**	98.94%	99.44%
Adult	**RF**	84.97%	71.15%	81.33%	83.69%	**83.89%**	83.06%
	LR	78.53%	75.68%	78.18%	**78.38%**	76.1%	77.24%
	NN	76.9%	75.69%	76.6%	78.36%	78.23%	**79.16%**
Travel	**RF**	88.95%	56.35%	67.18%	84.09%	79.78%	**85.19%**
	LR	82.87%	70.17%	79.56%	**83.31%**	78.34%	82.76%
	NN	81.77%	71.05%	79.56%	81.88%	80.77%	**83.20%**

4.3.5 RQ5: Do the data sets generated by PAFTadhere to real distributions and possess mode diversity?

We also evaluate how closely the density and diversity of the true data distribution are matched by PAFTgenerated data using correlation metrics and density-based distance met-

rics. Specifically, we employ the Kolmogorov-Smirnov Test (KST) to evaluate the density estimate of numerical columns, and the Total Variation Distance (TVD) for categorical columns. When calculating the correlation between columns, we employ Pearson correlation for numerical columns and contingency similarity for categorical columns. The results for both density estimate similarity and correlation based analysis are detailed in Table 4.6. As shown, PAFTsynthetic data closely matches the real data in terms of univariate distribution and bivariate correlation, outperforming the baseline.

Table 4.6: Low-order statistics [146] of column-wise data density and pair-wise column correlation[1]. Scores range from 0 to 1. Higher values indicate more accurate estimation. PAFToutperforms the best generative baseline model in most case. The best results are marked in **bold**, the second-best results are underlined. Results are averaged over five random runs, with variance in Appx. Table A.5 and A.6.

Single-Column Shape Score (↑)					
Dataset	CTGAN	Co.GAN	TabSyn	GReaT	**PAFT**
Adult	0.81	0.92	**0.98**	0.88	0.90
Beijing	0.89	0.79	**0.98**	0.93	0.97
California	0.87	0.77	**0.98**	0.89	0.83
US-locations	0.83	0.82	0.96	0.93	**0.97**
Seattle	0.83	0.73	0.93	0.90	**0.93**
Travel	0.84	0.90	0.93	**0.93**	**0.93**
Two-Column Pair Trends score (↑)					
Dataset	CTGAN	Co.GAN	TabSyn	GReaT	**PAFT**
Adult	0.81	0.86	**0.93**	0.80	0.78
Beijing	0.92	0.94	**0.99**	0.95	0.98
California	0.84	0.87	**0.97**	0.87	0.91
US-locations	0.50	0.55	0.93	0.89	**0.94**
Seattle	0.74	0.72	0.80	0.76	**0.81**
Travel	0.77	0.80	**0.87**	0.85	0.82

Visual examples are depicted in Figure 4.4. PAFThas the ability to generate a wide range of diversity, encompassing both continuous and discrete variables, which closely resembles real data.

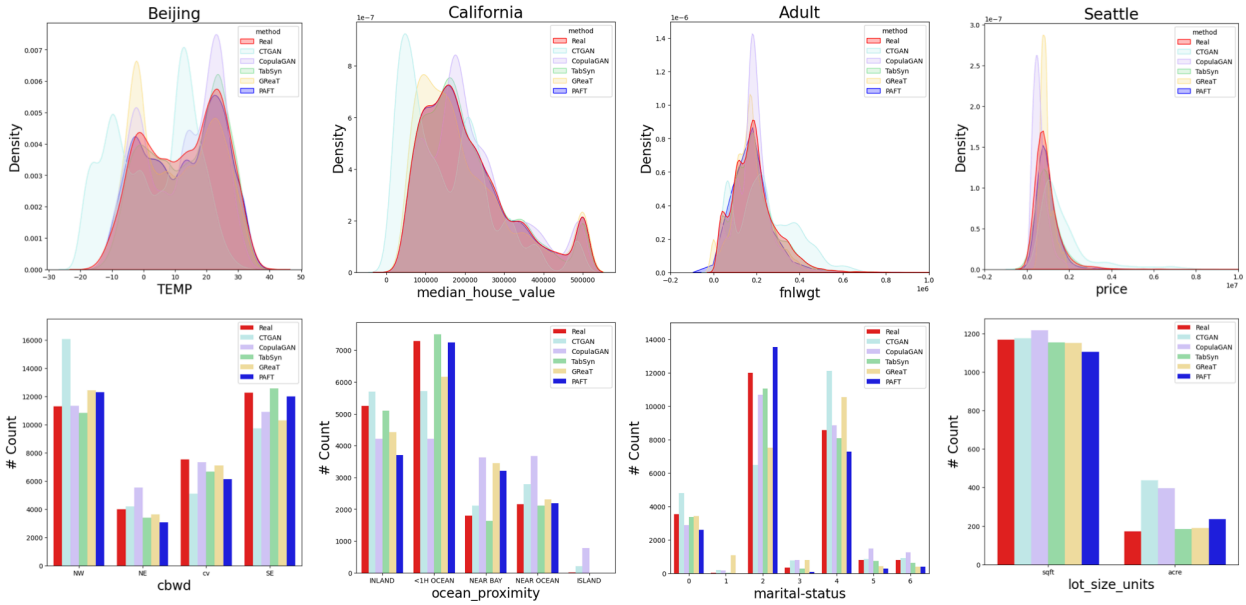


Figure 4.4: Column distributions visualization for each dataset generated by CTGAN, CopulaGAN, GReaT, and PAFT. The top row displays examples of numerical columns, while the bottom row presents examples of categorical columns. Overall, PAFT(Blue) has the closest distribution to real data (Red) compared to other synthesis methods. PAFTalso showcase the ability to generate a wide range of diversity.

4.3.6 RQ6: Can PAFT enhance the stability in generating high quality tables, resulting in a faster sampling phase?

When it comes to the comparison between fine-tuning approaches in LLM, the quality of the generated table rows can be reflected in the sampling process. This is particularly evident when generating an i.i.d row that involves auto-regressive generation. If the generated row cannot be decoded back into a real table row, then another sample needs to be redone. Given that the device condition and hyperparameters of the GReaT and PAFT are identical, a

shorter sampling time indicates a higher probability of accepting a generated row, meaning improved generation quality. (See Table 4.7.) Furthermore, the time measurement is based solely on a single-core GPU to ensure a fair comparison of different baselines. Using multiple GPU parallel computing can significantly speed up the fine-tuning process in practice.

Admittedly, PAFTand all LLM fine-tuning based table generators share the identical challenge of time consuming (both training and sampling), comparing to classic DL or GAN based table generators. In exchange for the investment of time, there are several advantages to consider. These include the elimination of data preprocessing which is a significant time cost for a human expert in charge of data preparation, a deep understanding of real-world knowledge, and finally DL-bsaed generators have been shown to fail when the table generation task involves generating columns with textual sentences as their values.

Table 4.7: A run time comparison of all generative models. Models were trained and fine-tuned using comparable hyperparameters, and generated samples were of the same size as the real dataset. For certain datasets that pose difficulties for auto-regressive generation (such as Adult), PAFT can significantly enhance the quality of the generation process, resulting in reduced generation time. For typical datasets with fewer challenges, the time-efficiency performance of GreaT and PAFTis comparable. (The standard variance of time in the five random experiments was smaller than one secondary unit, so it has been omitted).

	CTGAN		**CopulaGAN**		**TabSyn**	
	Training Time	Sampling Time	Training Time	Sampling Time	Training Time	Sampling Time
Adult	50:38 sec	0:47 sec	11:26 min	1:04 sec	46.35 min	7.38 sec
Beijing	55:72 sec	1:07 sec	13:94 min	2:67 sec	54:11 min	9:89 sec
California Housing	2:79 min	5:68 sec	5:72 min	1:02 sec	33:27 min	4:81 sec
US-locations	18:53 sec	0:17 sec	4:72 min	0:39 sec	28:6 min	4:47 sec
Seattle	3:56 sec	0:07 sec	24:93 sec	0:13 sec	18:43 min	0:64 sec
Travel	0:40 sec	0:04 sec	12:49 sec	0:06 sec	16:44 min	0:63 sec

	GReaT		**PAFT**	
	Training Time	Sampling Time	Training Time	Sampling Time
Adult	3:52 hr	37:47 min	3:49 hr	**5:15 min**
Beijing	4:10 hr	5:24 min	4:08 hr	5:21 min
California Housing	2:23 hr	4:16 min	2:22 hr	**2:58 min**
US-locations	55:48 min	58 sec	54:33 min	58 sec
Seattle	7:42 min	10 sec	7:43 min	11 sec
Travel	3:30 min	5 sec	3:33 min	5 sec

4.3.7 RQ7: Do newer generations of LLMs obviate the need for PAFT?

The choice of foundation models for synthetic table generation involves two key considerations: in-context learning and fine-tuning, especially with large-parameter models such as GPT-4 and LLaMA. A recent survey [42] indicates that the in-context learning approach of GPT-4 is well-suited for augmenting tabular data in low-data regimes, as highlighted by CLLM [113]. However, GPT-4 suffers from limitations such as the disappearance of column-wise tail distributions and a low success rate in accurately extracting output cell values [42].

In contrast, the state-of-the-art GReaT [15] and other synthetic data generation approaches [116, 148, 151] typically employ smaller models like GPT-2 or DistilGPT2, which effectively address attribute encoding while reducing feature engineering efforts. These smaller models already outperform traditional GAN and traditional statistics-based approaches. For example, DistilGPT2 can be trained using LoRA on GPUs as modest as the Tesla P100.

Importantly, fine-tuning larger models such as GPT-4 entails significant computational cost. For instance, fine-tuning a GPT-4 model for three epochs on a dataset with 50k rows and a five-column table costs approximately $300 (OpenAI) or requires equivalent high-end GPU resources. Therefore, while the same fine-tuning schema can be applied to models like GPT-4 or LLaMA, it is not a cost-effective solution compared to the solutions considered here.

In summary, newer generation LLMs do not obviate the need for PAFT. Our method directly addresses the impedance mismatch between autoregressive LLMs and synthetic table generation—particularly by preserving functional dependencies through permutation-aware fine-tuning—a challenge that persists even with advanced models.

4.4 Discussion

This chapter has brought LLMs closer to the goal of generating realistic synthetic datasets. By learning FDs and leveraging this information in the fine-tuning process, we are able to align the auto-regressive nature of LLMs with the ordering of columns necessary for generating quality synthetic data. While PAFTis quite broadly applicable by itself, it can be extended in several directions. First, what are other, perhaps more expressive, types of tabular constraints that can be utilized in the fine-tuning process? Second, what is the internal basis for regulating orders inside a transformer architecture and can we more directly harness it? Third, can we theoretically prove the (im)possibility of generating specific synthetic datasets by LLM architectures? Fourth, despite the numerous advantages of LLM in learning and generating tabular data, scalability remains an acknowledged challenge [15, 33, 42], encompassing concerns such as context window and training speed. And finally, privacy-preserving methods have been implemented in table generators based on GANs but remains understudied in LLM fine-tuning. These questions will be the focus of our future work.

Limitations. The row-wise generation cost of our method, particularly when employing fine-tuning, is affected by the dataset sample size and computational resources (GPU). Moreover, the capacity of PAFTto generate columns is affected by context window sizes. These limitations can be overcome by the newer generation of LLMs or by exploring partial row generation, i.e., generating a row in multiple steps using an LLM.

Overall in this chapter, we discussed the use of LLM fine-tuning for generating synthetic tabular data with minimal feature preprocessing. Based on the proposed PAFT pipeline, this approach promises more robust row fidelity. However, in practical data science scenarios, the scale of data can be large, and the computational cost of training large language models cannot be considered trivial. Therefore, in the next chapter 5, we propose leveraging

prior domain knowledge, referred to as KGP (Knowledge-Guided Prompting), to guide in-context learning for tabular data generation. This approach aims to strike a balance between generation quality and the computational cost of using LLMs.

Chapter 5

Reducing In-Context Learning Burden in STG via LLM Priors

While Chapter 4 showed that fine-tuned LLMs can generate high-fidelity synthetic tables with minimal preprocessing, their computational cost remains a major concern at scale. To address this, we now explore an alternative approach using prior domain knowledge to guide in-context learning, aiming to balance generation quality with efficiency.

Synthetic data generation is a key ingredient in many data science pipelines, e.g., to help overcome privacy limitations [2, 60], to support machine learning in domains where there are imbalanced classes [52], to enable data augmentation when real data is scarce [29], and to simulate rare or extreme events that are difficult to capture in real-world datasets [38]. Many powerful ML algorithms, e.g., generative adversarial networks (GANs) [47, 57, 131, 138, 140] rely on synthetic data generation as a key ingredient to their workflow.

Recently, large language models (LLMs), especially the latest variants such as GPT-4o and the LLaMA series, have been examined for their potential as structural data regressors or generators. Most modern LLMs are based on the transformer architecture [128] with parameters ranging from few millions to billions [54], and researchers have developed creative ways to harness LLMs in traditional machine learning and data contexts. For instance, LIFT [33] transforms table rows of raw numerical data into sentences such as 'An Iris plant with sepal length 5.1cm, sepal width 3.5cm...', and employs an LLM to solve traditional

machine learning taasks like classification, regression, and generation. GReaT [15] fine-tunes an LLM for synthetic data generation and show that even small-scale models such as Distill-GPT [97] are capable of synthetic data generation [15].

While the above works fine-tune an LLM to support data generation, newer variants support synthetic data generation out-of-the-box, i.e., with in-context learning (ICL) [113]. Using just a few example rows in the context window, an LLM can be made to generate synthetic data conforming to inferred properties of the supplied rows. However ensuring fidelity to complex, heterogeneous distributions by finding representative examples remains a challenge and requires careful experimentation. Just as sampling points on a curve might require more points where there are shifts in behavior (versus regions where there is more normalcy), we will require more ICL examples for some regions versus others to support improved generalization.

In this chapter, we investigate if/how much an LLM's in-built prior knowledge can help in synthetic data generation and, more specifically, whether it can replicate behavior that previously required an inordinate number of ICL examples. We ask the question: 'how many examples can a prompt substitute for?' and aim to capture this tradeoff by defining knowledge levels and studying their interplay with the number of ICL examples. Our approach is dubbed knowledge-guided prompting (KGP), where domain knowledge, either inferred or available, is explicitly injected into the prompt, reducing dependence on ICL examples. This enables new approaches to synthetic data generation than purely data-driven or purely knowledge-driven approaches.

Our contributions are:

1. We propose an automatic knowledge-guided prompting (KGP) approach to improve the quality of structured data generation while at the same time limiting the number of

ICL examples required. This approach is especially valuable in scenarios where there is data paucity or we wish to reduce the number of tokens while maintaining synthetic data generation quality.

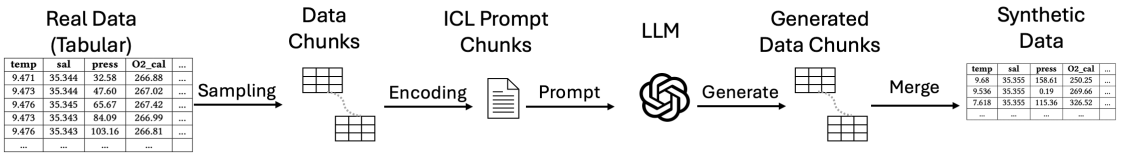
2. The KGP approach proposed here systematizes prior knowledge into strong knowledge (e.g., symbolic constraints, statistical priors) versus weaker knowledge (e.g., monotonicity constraints, dependency relationships), and explores their interplay w.r.t. the number of ICL examples.

3. Through numerous experiments, we demonstrate that our KGP approach yields better generation quality than using purely ICL examples, unlocking new hybrid approaches to synthetic data generation. Most importantly, we demonstrate how the KGP framework provides a framework to think about scaling laws that predict the number of examples needed for given levels of prior knowledge.

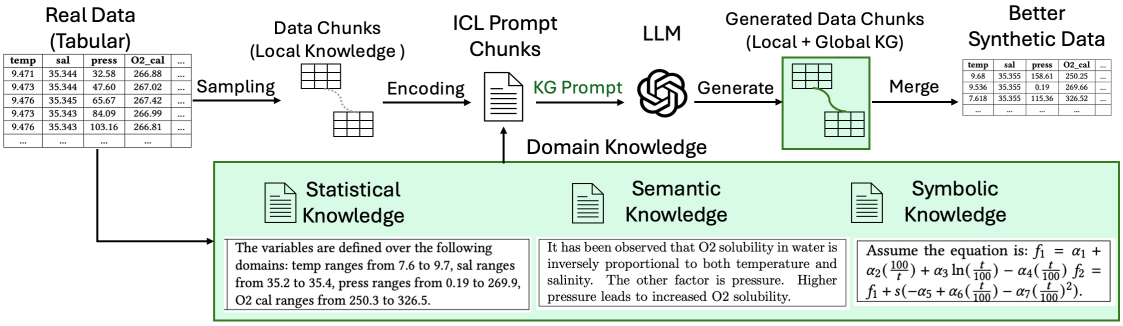
5.1 Knowledge Guided Tabular Data Generation with LLMs

Synthetic tabular data generation typically comprises a generation function $\mathcal{G}(\cdot) : \mathcal{D}_{\text{train}} \rightarrow \mathcal{D}_{\text{out}}$ where $\mathcal{D}_{\text{train}}$ is the set of samples supplied to $\mathcal{G}$ as input and $\mathcal{D}_{\text{out}}$ is the target data distribution. The main objective of the tabular data generation task is to generate synthetic data $\mathcal{D}_{\text{syn}}$ conditioned upon $\mathcal{D}_{\text{in}}$ such that $\mathcal{D}_{\text{syn}} \sim \mathcal{D}_{\text{out}}$ i.e., the generated data captures the joint distribution inherent in $\mathcal{D}_{\text{out}}$.

Recently, LLMs owing to their semantic recognition capabilities as well as pre-trained knowledge, have demonstrated effectiveness in the synthetic tabular data generation task. In the



(a) Traditional synthetic tabular data generation pipeline.



(b) Knowledge-guided prompting (KGP) pipeline.

Figure 5.1: (a) a traditional synthetic tabular data generation pipeline using LLMs encodes sample data as in-context learning examples to drive the generation process. (b) Our knowledge-guided prompting (KGP) approach, incorporates automatically inferred domain knwoledge, providing the LLM-based generator a complementary context in addition to ICL examples. Our experimental findings indicate that such global property conditioning via. KGP leads to a significant improvement in synthetic data generation quality.

context of LLM based tabular data generation, we can think about the LLM as a few-shot generator where the few-shot nature of the problem arises from the in-context learning (ICL) examples $\mathcal{D}_{\text{train}}$ supplied as input to the LLM-based generator $\mathcal{G}$ as part of the input query $q = [< \text{prompt} >; \mathcal{D}_{\text{train}}]$. The query 'q' comprises the prompt along with $\mathcal{D}_{\text{train}}$ ICL examples.

In the LLM tabular data generation task, owing to the limited effective context windows in LLMs, the input data is chunked into 'c' chunks, each a group of k rows $\mathcal{D}_{\text{train}} = \{\mathcal{D}^{(1)}_{\text{train}}, \ldots, \mathcal{D}^{(c)}_{\text{train}}\}$ and each chunk is supplied to the LLM as a set of in-context learning (ICL) examples, in addition to a prompt i.e., $q_i = [< prompt >; \mathcal{D}^{(i)}_{\text{train}}]$. The result of all the queries $q_i | i = 1 \ldots c$ are merged to form the final generated table $\mathcal{D}_{\text{syn}} = \bigcup_{i=1}^{c} \mathcal{D}^{(i)}_{\text{syn}}$. Thus, the LLM, conditioned upon the prompt and ICL examples (i.e., $\mathcal{D}^{(i)}_{\text{train}}$), generates new table

rows similar to $\mathcal{D}^{(i)}_{\text{train}}$.

However, the properties of data in each chunk, $\mathcal{D}^{(i)}_{\text{train}}$ strongly influence the quality of data generated and issues such as *lack of full distributional coverage* and *process noise* may affect the data $\mathcal{D}^{(i)}_{\text{train}}$ thereby carrying over to the generated output chunk $\mathcal{D}^{(i)}_{\text{out}}$, and also create intra-chunk and inter-chunk inconsistensies. To alleviate these adverse effects, we propose knowledge-guided prompting (KGP) as a method to inject prior domain knowledge about (global) properties prevalent in the ground-truth data distribution in addition to the ICL examples $\mathcal{D}^{(i)}_{\text{train}}$. Essentially, this entails augmenting each query $q_i = [<\text{knowledge}-\text{guided}\, \textit{prompt}\ >; \mathcal{D}^{(i)}_{\text{train}}]$ Prior domain knowledge may occur in many forms and we now detail the various types of domain knowledge and how to inject each into the LLM tabular data generation pipeline via KGP. The full LLM-based tabular data generation pipeline with KGP is depicted in Fig. 5.1.

5.1.1 Encoded Knowledge Types

A *knowledge guided prompt (KGP)* accompanying a chunk $\mathcal{D}^{(i)}_{\text{train}}$ of a dataset $\mathcal{D}_{\text{train}}$, holds for all of $\mathcal{D}_{\text{train}}$. Otherwise stated, KGP encodes global knowledge while a data chunk holds local knowledge.

Prior domain knowledge may appear as symbolic relationships, functional dependencies, semantic descriptions of the data as well as statistical knowledge about the data distribution. The various types of domain knowledge we categorize are illustrated in Table 5.1, along with KGP examples of each. More detailed examples in the context of specific datasets investigated in this chapter, are included in Table 5.2. We specifically focus on three major types of knowledge guidance in this work:

1. Symbolic KGP: In this form of KGP, we assume access to the symbolic (theoretical)

relationship governing the (possibly noisy) data generation process.

2. Semantic KGP: In this form of KGP, we assume we can encode (partial) knowledge of the data distribution in terms of common prior to take advantage of the semantic recognition capabilities of the LLM.

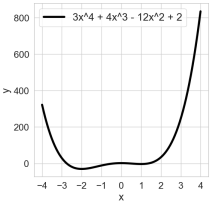
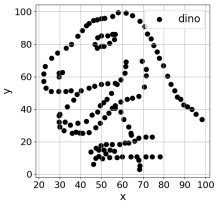
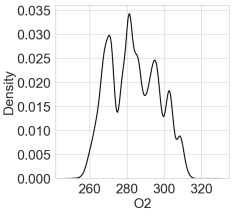
3. Statistical KGP: In this form of KGP, we assume (weak) knowledge about ranges of specific columns in our tabular data.

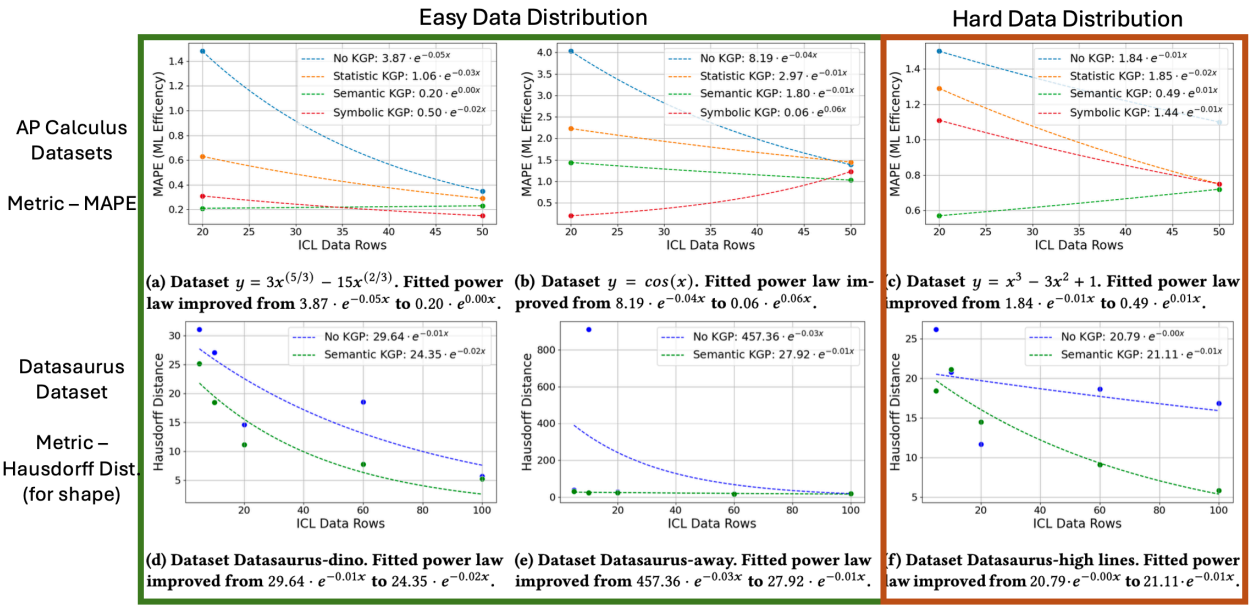
Fig. 5.1b depicts the proposed KGP pipeline with the various types of domain knowledge considered. Throughout our experimentation, we do not treat all three types of domain knowledge equally, we assume 'Statistical KGP' as weak domain knowledge that is the most prevalent, 'Semantic KGP' also as weak domain knowledge with relatively lower prevalence than Stastical knowledge and finally we assume 'Symbolic KGP' as the strongest as well as the least prevalent type of domain knowledge.

Table 5.1: Types of domain knowledge along with examples of how each type can be incorporated into KGP.

Type	**Knowledge**	**Example**
Strong	Symbolic	Equation: $3x^4 + 4x^3 - 12x^2 + 2$.
Strong	Distribution	The data follows a specific form of the Bohachevsky function.
Strong	Functional Dependency	If Protocol is TCP, then packet size is between 40 to 65,535 bytes.
Weak	Semantic Description	X and y coordinates of points when plotted visually depict a dinosaur.
Weak	Statistical Knowledge	The variables are defined over the following domains: temp ranges from 7.6 to 9.7, press ranges from 0.19 to 269.9.

Table 5.2: Example setup of different types of datasets and different levels of knowledge. In practice, the data contains more digits; however, for presentation purposes, we only display up to two to three decimal places.

Example Data	W/o KGP	Statistical (Stat.) KGP	Semantic (Sem.) KGP	Symbolic (Sym.) KGP	Preview
AP Calculus (Math)	x is 2.4278, y is -0.8169. x is 0.2925, y is 1.9153. x is 1.1009, y is -0.1916. [...More]	The variables are defined over the following domains: x ranges from -4.0 to 4.0.	The function f is decreasing if x<=-2, increasing if -2<=x<=0, decreasing if 0<=x<=1, and increasing if x>=1.	Consider the equation: $3x^4 + 4x^3 - 12x^2 + 2$.	
Datasaurus Dozen (Graphical)	x is 55.3846, y is 97.1795. x is 51.5385, y is 96.0256. [...More]	The range of x is from 31.10686656 to 85.4461864, and the range of y is from 4.57766135 to 97.83761472.	x and y coordinates of points when plotted visually depict a dinosaur.	N/A	
O_2 Sensing (Real World)	temp is 9.471, sal is 35.344, press is 32.58, O2 cal is 266.88. temp is 9.473, sal is 35.344, press is 47.60, O2 cal is 267.02. [...More]	The variables are defined over the following domains: temp ranges from 7.6 to 9.7, sal ranges from 35.2 to 35.4, press ranges from 0.19 to 269.9, O2 cal ranges from 250.3 to 326.5.	It has been observed that O2 solubility in water is inversely proportional to both temperature and salinity. The other factor is pressure. Higher pressure leads to increased O2 solubility.	N/A	



(a) Dataset $y = 3x^{(5/3)} - 15x^{(2/3)}$. Fitted power law improved from $3.87 \cdot e^{-0.05x}$ to $0.20 \cdot e^{0.00x}$. **(b) Dataset $y = cos(x)$. Fitted power law improved from $8.19 \cdot e^{-0.04x}$ to $0.06 \cdot e^{0.06x}$.** **(c) Dataset $y = x^3 - 3x^2 + 1$. Fitted power law improved from $1.84 \cdot e^{-0.01x}$ to $0.49 \cdot e^{0.01x}$.**

(d) Dataset Datasaurus-dino. Fitted power law improved from $29.64 \cdot e^{-0.01x}$ to $24.35 \cdot e^{-0.02x}$. **(e) Dataset Datasaurus-away. Fitted power law improved from $457.36 \cdot e^{-0.03x}$ to $27.92 \cdot e^{-0.01x}$.** **(f) Dataset Datasaurus-high lines. Fitted power law improved from $20.79 \cdot e^{-0.00x}$ to $21.11 \cdot e^{-0.01x}$.**

Figure 5.2: Showcasing the MAPE and Hustoff distance between the synthetic data and the real data. X-axis represents different ICL data sizes. The green curve represents the semantic KGP and the blue curve represents the No-KGP setting. Take (a) for example, by incorporating the visual knowledge phrase "x and y coordinates of points when plotted visually depict a dinosaur." into the prompt, the quality of the generated data improves when the dataset is limited. The quantitative metric Hausdorff Distance decreased from 18.54 to 7.72 indicating a significant improvement when using 60 In-Context Samples.

5.2 Experimental Results

In this section, we design experiments on synthetic tabular data generation tasks to investigate the effectiveness of knowledge-guided prompting (KGP) using LLMs. We evaluate our approach using numerous datasets across mathematical, geometric, and real-world applications. Specifically, we wish to investigate the following research questions:

RQ1: What is the trade-off between domain-knowledge and ICL examples? (Section 5.2.2)

RQ2: Can domain-knowledge alleviate effects of poor data coverage or help with (out-of-domain) OOD generalization? (Section 5.2.3)

RQ3: Which type of knowledge injection is the most effective? (Section 5.2.4)

RQ4: How does KGP affect the quality of the synthetic data generated? (Section 5.2.5)

RQ5: (Case Study) Can we characterize the effectiveness of KGP in a real-world cyber-physical scenario? (Section 5.2.6)

5.2.1 Setup & KGP Scope

Datasets. We have adopted real datasets across three application domains. We manually extracted datasets from the AP Calculus textbook (specifically, Section 4 [9]), featuring variations of equations and descriptions of function characteristics, Thirteen datasets from the Datasaurus Dozen exhibiting distinct visual characteristics [78], and an O_2 sensing dataset from real-world applications related to cyber-physical systems [1].

Baselines. We aim to investigate the potential of in-context prompting techniques utilizing large language models, and in this experiment the flagship OpenAI model GPT-4o is utilized as the foundation model. In accordance with the system outlined in the previous section, three levels of knowledge-guidance prompts will be introduced and analyzed in an ablation study: Statistical KGP, Semantic KGP, and Symbolic KGP.

KGP Scope. It is important to clarify that for the purpose of this evaluation, we treat the levels of knowledge as a set of concentric circles. In other words, "Semantic KGP" denotes the **combination of Statistical and Semantic KGP**. Similarly, "Symbolic KGP" includes **all three forms of knowledge**. Notably, Without knowledge implies no guidance from knowledge is utilized, serving as the traditional baseline for synthetic data generation.

Metrics. The traditional metrics for synthetic data from the tabular data generation community are utilized, including machine learning utility (MLU), negative log likelihood (NLL), KL divergence, and distance to the closest record (DCR). Additionally, for datasets contain-

[1] https://www.bco-dmo.org/dataset/3426

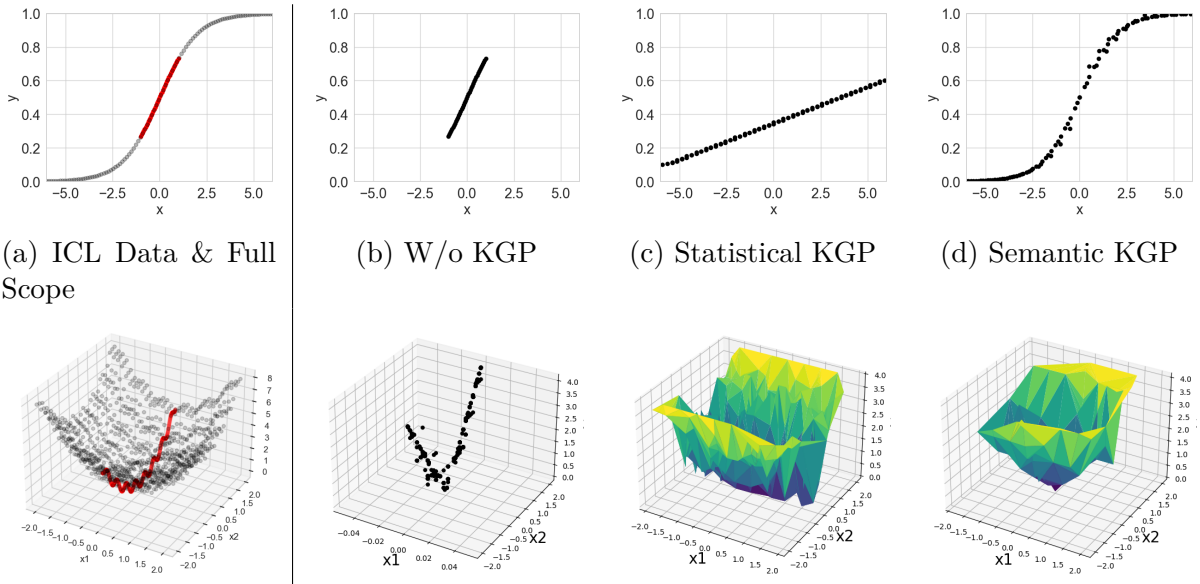
ing ground-truth symbolic equations, we employ mean square error (MSE) as the primary metric for assessing data record validity. For datasets characterized by shape-focused distributions, we employ Hausdorff distance to assess the similarity of the shapes.

5.2.2 RQ1: What is the trade-off between domain-knowledge and ICL examples?

Encoding structural data within a text sentence for LLM utilization incurs a substantial token load. This underscores the rationale for saving example data tokens to maintain performance expectations or to enhance performance in scenarios where example data is insufficient.

Although LLM developers continue to explore the upper limits of context windows for both input and output, their efforts remain insufficient in the domain of synthetic structural data generation. Therefore, it is important to consider strategies for conserving tokens utilized by in-context data samples, as well as identifying ways to assist a language model when a user's example data is limited. Here are two common cases:

Simple Data Distributions. When the joint distribution of the data is easy to model, using KGP will decrease data requirements and thereby reduce token demand, leading to financial and time saving. Figure 5.2a,b,c and e investigate the impact of KGP on data generation when the target data follows a simple joint distribution. Specifically, Figure 5.2a, b have been evaluated on datasets from the AP calculus [9] data corpus, where 4 KGP variants have been evaluated enabling us to investigate the full range of knowledge-guidance granularity (i.e., statistical, semantic and symbolic knowledge guidance). For each KGP variant, one context with 20 ICL examples and another with 50 ICL examples has been evaluated. The plots in Figure 5.2a,b both clearly demonstrate the benefit of KGP over the



(a) ICL Data & Full Scope (b) W/o KGP (c) Statistical KGP (d) Semantic KGP

(e) ICL Data & Full Scope (f) W/o KGP (g) Statistical KGP (h) Semantic KGP

Figure 5.3: Visualization of out-of-distribution (OOD) generation, featuring two mathematical functions: Sigmoid and Bohachevsky. In the ICL Real Data figure (a) & (e), the red data points represent the observed field, whereas the grey data points indicate the complete ground truth field. Figure (b)-(d), (f)-(h) showcase the generated synthetic data under corresponding KGP settings.

'No KGP' variant in low data (i.e., 20 ICL examples) scenarios.

Finding: *Semantic KGP, Symbolic KGP require 40% fewer ICL examples to achieve the same generation quality as a variant without KGP.*

Further, a similar experiment is carried out on the Datasaurus corpus [78] employing the popular Hausdorff distance metric to test data generation quality. In this scenario, we compare the 'No KGP' variant with the 'Semantic' KGP variant. The two variants are each evaluated in two ICL contexts namely, one with 10 and another with a 100 random ICL data points. The goal is once again to evaluate how the KGP affects tabular data generation quality and its utility in low-data scenarios.

Figure 5.2d evaluated in the context of the Dino dataset from the Datasaurus corpus, illus-

trates that 'Semantic KGP' achieves equivalent generation quality as 'No-KGP' with a 40% reduction in the number of ICL examples. Figure 5.2e is a similar comparison performed on the *Away dataset* from the Datasaurus corpus and demonstrates an even higher reduction (ie., 90%) in ICL examples in the Semantic KGP context to achieve thee same generation quality as the No KGP context.

Finding: *Overall, KGP improves synthetic-data generation quality with a 40% - 90% reduction in ICL examples while achieving the same generation quality as a variant without KGP, even in for simple data distributions.*

Complex Data Distributions. In Figure 5.2c and 5.2f, we evaluate datasets from the AP Calculus and Datasaurus corpora respectively except here, we consider datasets where the data exhibits a more complex (i.e., harder to model) joint distribution. Figure 5.2f illustrates an example of modeling a relatively difficult joint distribution (i.e., *High Lines* dataset) which is difficult owing to the data being distributed in disparate statistical modes. Here, we notice that despite being conditioned on 100 in-context samples, even a state-of-the-art LLM like GPT-4o alone (i.e., (with No KGP) does not generate a valid synthetic joint distribution (as evidenced by high Hausdorff distance of the blue line even at ICL 100). However, by simply injecting a semantic KGP statement such as 'x and y are visually looking like high lines', the model can not only significantly improve the quality of generated data but achieves the same generation quality as 'No KGP' with 80% fewer ICL examples.

Figure 5.2 represents a cubic polynomial function, also hard to model in a purely data-driven manner. We notice that incorporating any form of KGP (statistical, semantic or Symbolic) leads to a significant reduction in data generation error and a 50% reduction in ICL examples to achieve the same generation quality as the 'No KGP' variant.

Overall Finding: *KGP results in a significant reduction in ICL examples for synthetic tabular*

data generation, both in contexts where the data follows an easy and a hard joint distribution. Specifically, knowledge guidance leads to a 40%-90% reduction in ICL examples in the easy data context and between 50%-80% reduction in the hard data context.

5.2.3 RQ2: Can domain-knowledge alleviate effects of poor data coverage or help with OOD generalization?

Table 5.3 showcases the capability of generating previously unobserved structural data on the basis of the 'Statistical' and 'Semantic' KGP provided, while Figure 5.3 illustrates the visual representation. When calculating the MSE of the uncovered field of the 'No KGP' variant, noise will be examined. 'Statistical' KGP and 'Semantic' KGP will generate data to cover those region utilizing the injected domain knowledge. The mean squared error (MSE) of the sigmoid function can be significantly reduced by 98%. Furthermore, with respect to the Bohachevsky function, the absolute value of the error decreased from 1.62 to 0.44 due to its higher dimensionality and increased complexity. In the realm of complex functions, delving into unfamiliar areas demands increased caution, as the LLM generator may mistakenly treat unknown fields as similar to known ones, as illustrated in Figure 5.3c and 5.3g.

Table 5.3: MSE for OOD generalization.

Math Function	W/o KGP	Statistical KGP	Semantic KGP	MSE Impr.
Sigmoid (2d)	0.11	0.09	**0.002**	↓ **98%**
Bohachevsky (3d)	1.62	2.23	**0.44**	↓ **73%**

Overall Finding: *With the support of domain knowledge, KGP is capable of generating out-of-distribution (OOD) data and augmenting datasets that suffer from poor coverage or missing values. The data generated through 'Statistical' plus 'Semantic' KGP exhibits an error rate that is 78% to 90% lower compared to the plain 'No KGP' method when exploring*

to unknown data feild.

5.2.4 RQ3: Which type of knowledge injection is the most effective?

Table 5.4 shows that the injection of 'Statistical' KGP and 'Semantic' KGP generally leads to consistent and improved data quality. Utilizing the complete equation, notably 'Symbolic' KGP, does not always produce beneficial outcomes due to the limited grasp of complex mathematics.

Overall Finding: *The integration of statistical KGP and semantic KGP in data generation involving various mathematical function relationships can produce a consistent improvement in quality (i.e., a reduction in MSE), ranging from 35% to 70%, without occasionally causing negative error.*

Table 5.4: Mean Squared Error at the 50-ICL setting with various levels of KGP. Arrows indicate the trend of the effect (MSE) as higher levels of KGP (i.e., more granular knowledge-guidance rules) are injected.

Math Function	W/o KGP	Statistical KGP	Semantic KGP	Symbolic KGP
$y = 3x^{(5/3)} - 15x^{(2/3)}$	0.35	(↓)0.29	(↓)0.23	(↓)**0.15**
$y = x^3 - 3x^2 + 1$	1.10	(↓)0.75	(∼)**0.72**	(↑)0.75
$y = 2x^3 - 15x^2 + 36x$	0.02	(∼)0.02	(∼)0.02	(↓)**0.01**
$y = x + 2sin(x)$	0.40	(↓)0.14	(∼)**0.12**	(↑)0.57

5.2.5 RQ4: How does KGP affect the quality of generated synthetic data?

In addition to the savings on the number of ICL examples, it is equally crucial for KGP based generation pipelines to ensure high-quality synthetic data. To investigate this, we quantitatively evaluate the synthetic data quality with well accepted metrics: low-order statistics (Sec 5.2.5), machine learning utility (MLU) (Sec 5.2.5), and closest distance to record (DCR) (Sec 5.2.5).

How close is synthetic data to the full distribution joint (low-order statistics)?

Table 5.5 provides a quantitative assessment of the performance of the synthetic table. The negative log likelihood (NLL) quantifies the resemblance between synthetic data and real data. The low KL-divergence simultaneously guarantees the mode diversity of the synthetic data.

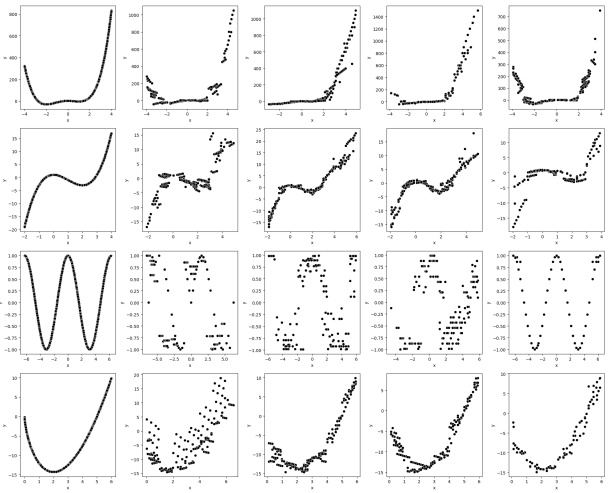
Figure 5.4 compares real and synthetic data, highlighting shape trends. Analyzing columns two (no KGP) to five (Symbolic KGP) shows that better knowledge guidance yields more realistic results within the same sample. Figure (a) represents the ICL20 conditions, while Figure (b) illustrates the ICL50 conditions. Comparison of subfigures (a) and (b) in Figure 5.5 shows that the Semantic KGP form in graphs requires a more detailed context in English. Otherwise, the Semantic KGP can similarly contaminate synthetic data, akin to a reverse de-noising process.

How does distance to the closest record change when knowledge is incorporated?

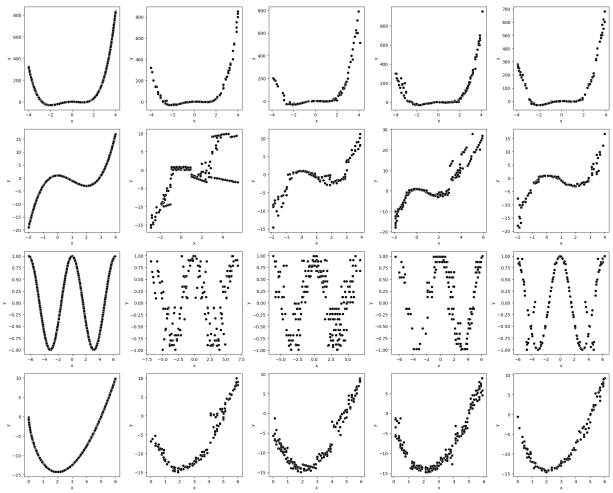
Table 5.6 shows improved row similarity without adding a new leak record.

Table 5.5: Low-order statistics, evaluated by negative log likelihood (NLL) and KL-divergence. Both of the metrics are smaller the better.

Negative Log Likelihood (NLL) (↓)			
Dataset	W/o KGP	Statistical KGP	Semantic KGP
AP Calculus	4.75±0.90	**4.73±0.73**	4.91±1.14
Datasaurus Dozen	8.99±0.18	9.09±0.25	**8.88±0.04**
O_2 Sensing	31.18	11.62	**8.54**
KL-Divergence (↓)			
Dataset	W/o KGP	Statistical KGP	Semantic KGP
AP Calculus	**0.02±0.02**	0.04±0.04	0.07±0.09
Datasaurus Dozen	0.06±0.07	0.10±0.06	**0.01±0.01**
O_2 Sensing	4.43	0.70	**0.20**



(a) ICL-20 for four functions, one per row: $(3x^4+4x^3-12x^2+2)$; (x^3-3x^2+1); $(cos(x))$; $(3x^{(5/3)}-15x^{(2/3)})$.

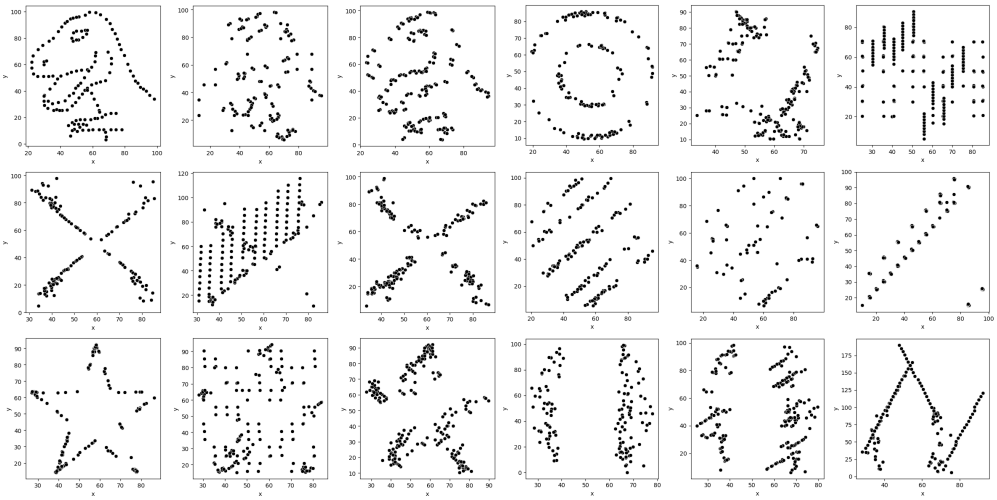


(b) ICL-50 for four functions, one per row: $(3x^4+4x^3-12x^2+2)$; (x^3-3x^2+1); $(cos(x))$; $(3x^{(5/3)}-15x^{(2/3)})$

Figure 5.4: Diversity of modes in synthetic data. Five columns from left to the right are real data, No KGP, statistical KGP, semantic KGP, and symbolic KGP.

Can we use synthetic data in ML pipelines?

Table 5.7 illustrates the performance of machine learning using synthetic data. The analysis reveals that synthetic data can effectively replace original data for training two commonly



(a) Good Semantic Knowledge: 'Dinau-sour', 'x shape', 'star'.

(b) Misleading Semantic Knowledge: 'bullseye', 'slant up', 'wide lines'.

Figure 5.5: Diversity of modes in synthetic data. Three columns from left to the right represent real data, No KGP, Semantic KGP.

used machine learning models, random forest and linear regression, yielding low MAPE errors on actual test data.

Overall Finding: *Using KGP (Statistical and Semantic) resulted in optimal performance across all three standard synthetic table metrics, with an average enhancement of 50% for each metric.*

5.2.6 Case Study: Characterizing Effectiveness of KGP in a Real-World Cyber-Physical Scenario

This section uses a dataset from a noisy cyber-physical system recording temperature, salinity, and pressure to predict water's oxygen solubility. Table 5.8 presents the evaluation of generating synthetic data from noisy raw data using Statistical KGP and Semantic KGP.

Finding: *Statistical KGP is vital for preserving a valid row joint distribution in scenarios*

Table 5.6: Distance to the closest record: A lower distance yields a better record in terms of validity; however, the occurrence of a zero value, which indicates a leak of raw data, is unacceptable.

Distance to the closest record. (↓)			
Dataset	W/o KGP	Statistical KGP	Semantic KGP
AP Calculus	**0**	0	0
Datasaurus Dozen	0.21±0.20	0.41±0.24	**0.12±0.09**
O_2 Sensing	0.57	0.46	**0.38**

characterized by noisy data.

One of the advantages of using modern LLMs is the availability and flexibility of the agent-embedded framework. Considering that even explicitly including the instruction "does not copy the original data" in the prompt, the generated data may still include some, see Table 5.6. Given that the LLM generator can utilize foundational and supplementary domain knowledge (statistical and semantic KGPs) to correct errors, we will initially introduce noise to the original data and subsequently employ the LLM to correct this noise, a process referred to as **noise-and-refix**. When the real original data are not presented to the LLM agent, concerns regarding the copying or leaking of data are eliminated.

Finding: *A modicum of in-context example data is essential to generate the hidden distribution, while knowledge guidance is more effective in providing dependency, correcting errors, or establishing boundaries.*

5.3 Discussion

This chapter proposes the use of Knowledge-Guided Prompt (KGP) to enhance the generation of structural tabular data by a Large Language Model (LLM). Although examples of

Table 5.7: MLU- Random Forest and Linear Regression.

Machine Learning Utility (MLU) - Random Forest (↓)			
Dataset	W/o KGP	Statistical KGP	Semantic KGP
AP Calculus	0.47±0.45	0.30±0.32	**0.27±0.31**
Datasaurus Dozen	0.90±0.30	0.88±0.25	**0.60±0.19**
O_2 Sensing	0.031	0.029	**0.0225**
Machine Learning Utility (MLU) - Linear Regression (↓)			
Dataset	W/o KGP	Statistical KGP	Semantic KGP
AP Calculus	1.61±2.02	1.61±2.02	2.13±2.90
Datasaurus Dozen	0.90±0.13	0.84±0.08	**0.79±0.08**
O_2 Sensing	0.039	0.053	**0.023**

Table 5.8: Distance to the closest record.

Distance to the closest record under Noisy Case. (↓)			
Dataset	W/o KGP	Statistical KGP	Semantic KGP
O_2 Sensing W/o Noise	0.57	0.46	**0.38**
O_2 Sensing W/ Noise	1.06	**0.61**	0.70

in-context learning data are limited in a chunk size and offer localized knowledge only, the comprehensive domain knowledge of the entire dataset can be incorporated as an English prompt using statistical KGP and semantic KGP. Experiments demonstrate that the KGP strategy can reduce ICL data (that is, tokens) by 40%, improve data with unknown regions (out-of-distribution generation) or improve the quality of synthetic data while utilizing the same level of ICL data.

Future work will be aimed at developing a multimodal learning framework encompassing KGP with visual and semantic facets. A user's inconsistent semantic KGP, when compared to the example data, may result in a decrease in generation quality, constituting a form of

model poisoning. Leveraging associations across modalities via shared parameters will lead to more resilient approaches for knowledge-guided applications.

Overall, we have addressed DNN-based STG in Chapter 3, LLM fine-tuning for STG in Chapter 4, and LLM in-context learning with data priors in Chapter 5. In the next chapter 6, we propose using embedding isotropy as a trust indicator for STG with LLMs: leveraging observations of LLM embeddings to assess the quality of the synthetic data generated.

Chapter 6

Embedding Isotropy as a Trust Indicator for STG with LLMs

Chapters 3–5 explored methods to improve synthetic tabular data generation using DNNs, LLM fine-tuning, and in-context learning with data priors. In this chapter, we shift focus to a new question: when will an STG model generate higher-quality data? To address this, we propose using embedding isotropy as an indicator of synthetic data quality in LLM-based generation.

Large language models have demonstrated impressive success in adapting to various downstream tasks in numerical domains, such as finance [45, 144], energy [43], climate science [59], healthcare [130], wireless communications [143], synthetic tabular generation [16, 33, 143], among others. Inspired by the success of pre-trained LLMs, several methods have been developed recently in [8, 35, 49, 59, 84, 101, 136] by adapting LLM to numerical domains. For many of these numeric downstream tasks, training a linear classifier on top of the hidden-layer representations generated by the pre-trained LLMs have already shown near state-of-the-art performance [8, 59]. However, these models in [8, 35, 49, 59, 84, 101, 136] are treated as 'black-box' where numeric forecasts are controlled by complex nonlinear interactions between many parameters. This makes it difficult to understand how models arrive at their predictions and makes it challenging for users to trust the model outputs.

LLMs' tendency to hallucinate can have serious consequences in critical numeric applications.

For example, prediction errors in fraud detection in finance can lead to huge financial losses and errors in protection onset of sepsis or cardiac arrest in healthcare can result in patient deaths. Thus, to guarantee prediction reliability and accuracy in numerical domains, it is necessary to open the black-box and provide performance guarantees through explaination. Although recent empirical studies [59, 71, 84] demonstrate the benefits of vector representations of embedding learned by LLMs in various numeric downstream tasks, there is little theoretical understanding of their empirical success. Thus, a fundamental question arises: *"when (or how) can the next-word prediction capability of LLMs be effectively adapted to numerical domains?"*

The main contribution of this chapter is to provide an approach to answer this question by exploiting the isotropic property of LLM hidden representations in the contextual embedding space. To achieve state-of-the-art performance in numerical domains, we show that the hidden representations of LLMs must exhibit *a structured form* in contextual embedding space that accounts for the shift-invariance of the softmax function (i.e., the softmax output remains unchanged when all logits are shifted by a constant). Without such structure, the model can shift the logits while keeping the training loss unchanged, thereby leaving the logits ineffective for numeric downstream tasks. By formulating a gradient structure of self-attention in pre-trained models, we show how the isotropic property of LLM embeddings in contextual embedding space preserves the underlying structure of representations, thereby resolving the shift-invariance problem of the softmax function.

Our key contributions include: (i) We consider a log-linear model for LLMs and demonstrate theoretically why hidden representations must exhibit structure to address the shift-invariance problem of the softmax function. (ii) We take a deeper look into the hidden representations of pre-trained models and show how isotropy preserves the structural integrity of representations. In particular, we derive an upper bound for the Jacobian matrix

which collects all first-order partial derivatives of self-attention with respect to the input pattern and show that m largest eigenvectors of the LLM hidden representations minimize the gradient norm of self-attention. Then by projecting the representations into lower dimensions using these m largest eigenvectors, we find the isotropy within the clusters in the contextual embedding space. (iii) Finally, we provide a comprehensive evaluation across 12 real and 10 synthetic time series datasets over 6 different LLMs.

6.1 Problem Setup in Temporal Tabular Generation

Time Series Tokens and Similarity Measure. Similar to next-word prediction by LLMs, the next-value prediction in the numerical domain can be modeled by *time series forecasting* techniques [8, 59] which are widely

adopted in the machine learning literature. Formally, given a time series $\mathbf{x}_{1:T+L} = [x_1, \ldots, x_T, \ldots, x_{T+L}]$, where the first T time instances give the historical context, the next L time instances constitute the forecast region, and $x_t \in \mathbb{R}$ is the observation of each time instance, we are interested in predicting the joint distribution of next L time instances, $p(\mathbf{x}_{T+1:T+L}|\mathbf{x}_{1:T})$. Since, the pre-trained models operate on tokens from a finite vocabulary, using them for time series data requires mapping the observations to a finite set of tokens. Based on different numeric applications and LLM architectures, various tokenization techniques, e.g., quantization and scaling [8, 101], patching [59, 84, 136], and adaptation of language model tokenizer in numeric domians [35, 49], can be applied to tokenize the time series and create a time series vocabulary $\mathcal{V}$ of N time series tokens, i.e., $|\mathcal{V}| = N$, as shown in Figure 6.1. Then, the realization of the next L time instances can be obtained by autoregressively sampling from the predicted distribution $p(k_{T+l+1} \mid \boldsymbol{k}_{1:T+l})$, for $l \in \{1, \ldots, L\}$, where $\boldsymbol{k}_{1:T+l}$ is the tokenized time series and k_i denotes a time series token from the vocabulary of size $|\mathcal{V}|$.

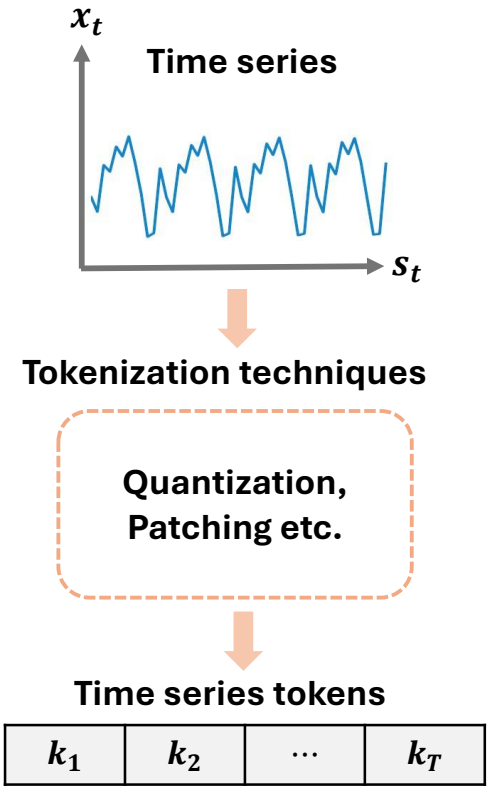


Figure 6.1: Time series tokenization.

Let $\tilde{\mathbf{\Psi}}(k_i) = \{\boldsymbol{\psi}_1(k_i), \boldsymbol{\psi}_2(k_i), \ldots\}$ be the set of all LLM contextual embedding instances of the time series token k_i. Here, different contexts in the time series sequences yield different LLM embeddings of k_i. By constructing $\sum_k |\tilde{\mathbf{\Psi}}(k)| = |\mathcal{V}|$, we define the inter-token cosine similarity as:

$$\zeta_{\cos} \triangleq \mathbb{E}_{i \neq j} \left[\cos \left(\boldsymbol{\psi}(k_i), \boldsymbol{\psi}(k_j) \right) \right], \tag{6.1}$$

where $\boldsymbol{\psi}(k_i)$ and $\boldsymbol{\psi}(k_j)$ are random samples from $\tilde{\mathbf{\Psi}}(k_i)$ and $\tilde{\mathbf{\Psi}}(k_j)$, respectively. The expectation is taken over all pairs of different tokens. The inter-token cosine similarity metric describes the similarity between different tokens based on the contexts. For ease of reading,

we express $T+l$ as T_l and $T+l+1$ as T_{l+1} for the remainder of the chapter.

Model. We consider a general pre-trained model for numerical data and open the black box of the pre-trained model. Specifically, we assume that the observation probability of $k_{T_{l+1}}$ given $\boldsymbol{k}_{1:T_l}$ satisfies the log-linear model [10]

$$p^*(k_{T_{l+1}} = i \mid \boldsymbol{k}_{1:T_l}) \propto \exp\left(\langle \boldsymbol{\psi}^*(\boldsymbol{k}_{1:T_l}), \boldsymbol{\psi}^*(k_i)\rangle\right), \tag{6.2}$$

where $\psi^*(k_i) \in \mathbb{R}^D$ is a vector that only depends on the time series token $k_{i\in\mathcal{V}}$, and $\psi^*(\mathbf{k}_{1:T_l})$ is a function that encodes the tokenized time series sequence $\mathbf{k}_{1:T_l}$ into a vector in $\mathbb{R}^D$. The log-linear modeling aligns with the commonly used LLMs networks whose last layer is typically a softmax layer. Moreover, we do not consider any prior distribution for input, making our model more general than previous latent models [10, 135].

To define the numeric downstream task, let $z_i^*(k,l) := \langle \psi^*(\mathbf{k}_{1:T_l}), \psi^*(k_i)\rangle$ be the i-th logit of the ground-truth model, and assume that the numeric downstream tasks are defined by a function of the logits, i.e., $f^*(\mathbf{z}^*)$. Also let $Z^*(k,l) = \sum_{i=1}^{N} \exp(z_i^*(k,l)) = \sum_{i=1}^{|\mathcal{V}|} \exp(\langle \psi^*(\mathbf{k}_{1:T_l}), \psi^*(k_i)\rangle)$ be the partition function [10], i.e., normalization factor. In LLMs, the partition function is often used to normalize the output probabilities of the model, ensuring that they sum to 1. Then, the normalized ground-truth model $\forall i \in \mathcal{V}$ is given by

$$p(k_{T_{l+1}} = i \mid \mathbf{k}_{1:T_l}) = \frac{\exp(\langle \psi^*(\mathbf{k}_{1:T_l}), \psi^*(k_i)\rangle)}{\sum_{i=1}^{|\mathcal{V}|} \exp(\langle \psi^*(\mathbf{k}_{1:T_l}), \psi^*(k_i)\rangle)} = \frac{\exp(z_i^*(k,l))}{Z^*(k,l)}$$

Since we do not know the ground-truth model in reality, we do not have access to the ground-

truth model $\psi^*(k_i)$ and $\psi^*(\mathbf{k}_{1:T_l})$. Instead, we only have access to the student model $\psi(k_i)$ and $\psi(\mathbf{k}_{1:T_l})$ that aims to achieve low pre-training loss. We can define the student logits as $\mathbf{z}(k,l) := \{\langle \psi(\mathbf{k}_{1:T_l}), \psi(k_i) \rangle\}_{i=1}^{|\mathcal{V}|}$. Intuitively, $\mathbf{z}$ are the contextualized representations learned by the student-model during pre-training. Then, the solution of the downstream task is to learn a function $f(k,l)$. Then, the output of the student model $\forall i \in \mathcal{V}$ can be defines as

$$p\left(k_{T_{l+1}} = i \mid \mathbf{k}_{1:T_l}\right) = \frac{\exp(\langle \psi(\mathbf{k}_{1:T_l}), \psi(k_i) \rangle)}{Z(k,l)}. \tag{6.3}$$

Loss Function. As typical in language models, we use the categorical distribution over the elements in the time series vocabulary $\mathcal{V}$ as the output distribution $p(k_{T_{l+1}}|\mathbf{k}_{1:T_l})$, for $l \in \{1, \ldots, L\}$, where $\mathbf{k}_{1:T_l}$ is the tokenized time series. The student model is trained to minimize the cross entropy between the distribution of the tokenized ground truth label and the predicted distribution. The loss function for a single sequence of tokenized time series is given by [8, 137]

$$\begin{aligned} \mathcal{L} &= -\sum_{l=1}^{L+1} \sum_{i=1}^{|\mathcal{V}|} p^*(k_{T_{l+1}} = i \mid \mathbf{k}_{1:T_l}) \log p(k_{T_{l+1}} = j \mid \mathbf{k}_{1:T_l}) \\ &= \sum_{l=1}^{L+1} \sum_{i=1}^{|\mathcal{V}|} \mathcal{D}_{\mathrm{KL}}(p^*(k_{T_{l+1}} = i|\mathbf{k}_{1:T_l}) \| \, p(k_{T_{l+1}} = j|\mathbf{k}_{1:T_l})) + H(p^*(k_{T_{l+1}} = j|\mathbf{k}_{1:T_l})), \end{aligned} \tag{6.4}$$

where $p(k_{T_{l+1}} = i \mid \mathbf{k}_{1:T_l})$ is the categorical distribution predicted by the student model parametrized by $v_{1:T_l}$, $p^*(k_{T_{l+1}} = i|\mathbf{k}_{1:T_l})$ is the distribution of ground-truth model, $\mathcal{D}_{\mathrm{KL}}$ is the KL divergence, and $H(p^*(k_{T_{l+1}} = i| \, \mathbf{k}_{1:T_l}))$ is the entropy of distribution $p^*(k_{T_{l+1}} = i|\mathbf{k}_{1:T_l})$ which is a constant. We assume that student model achieves a small loss so that the KL-divergence term in (6.4) is also small.

Downstream Numerical Task. We consider a simple downstream task whose prediction on categorical distribution is linear in $\psi^*(\mathbf{k}_{1:T_l})$, that is, $f^*(k,l) = \langle \psi^*(\mathbf{k}_{1:T_l}), u^* \rangle = \sum_{i=1}^{|\mathcal{V}|} a_i^* z_i^*(k,l)$, where $u^* = \sum_{i=1}^{|\mathcal{V}|} a_i^* \psi^*(k_i) \in \mathbb{R}^D$ and a_j is the coefficient. This model is still not sufficient to provide a performance guarantee to generalize to downstream task in unseen scenarios. However, the log probability difference is proportional to the difference in the value of the perfect model (i.e., ground-truth) $f^*(k,l)$. This allows the student model to alter the signs of $f^*(k,l)$ without resulting in a large KL divergence [137]. Then, it is more reasonable to model the numeric downstream task as

$$f^*(k,l) = \sum_{i=1}^{|\mathcal{V}|} a_i^* \sigma(z_i^*(k,l) - b_i^*) = \sum_{i=1}^{|\mathcal{V}|} a_i^* \sigma(\langle \psi^*(\mathbf{k}_{1:T_l}), \psi^*(k_i) \rangle - b_i^*),$$

where σ is the ReLU function and b_j^* denotes the threshold for the logits. The numeric downstream task only considers the logits that are above the threshold, and thus ignores all the entries with very small probabilities.

6.2 The Role of Isotropy in Adapting LLMs to Numerical Data

As previously discussed in Section 6.1, we consider LLM networks whose last layer is usually a softmax layer and the numeric downstream task is determined by the function of the logits. The underlying relation between the logits and softmax function determines the performance of the numeric downstream tasks. However, the softmax function is shift-invariant, that is, the output of the softmax function remains unchanged when all logits are shifted by

a constant. Since we do not have any control over the logit shift of the student model on unseen data, good performance during training does not necessarily provide any performance guarantee for the numeric downstream task on unseen scenarios.

Theorem 6.1. *Let the logits of the ground-truth model be bounded. Then for any $f^*(k,l)$, there exists a set of functions $\{\hat{z}_i(k,l)\}_{i=1}^{|\mathcal{V}|}$ such that for all k and T_{l+1}, the predictive distribution of the student model $\hat{p}\left(k_{T_{l+1}} \mid \mathbf{k}_{1:T_l}\right)$ matches that of ground-truth model $p^*(k_{T_{l+1}} \mid \mathbf{k}_{1:T_l})$ and $\hat{f}(k,l) = 0$. In other words, there exists a student model with the same pre-training loss as the ground-truth model, but its logits are ineffective for the numeric downstream tasks.*

Theorem 6.1 demonstrates that without any structure in the hidden representations of LLM embeddings, the student model is able to shift the logits for any sample while keeping the pre-training loss unchanged and leaving logits ineffective for the numeric downstream tasks. Consequently, a theoretical guarantee for the numeric downstream task performance needs structure in the LLM representations learned by the pre-trained model. One way to prevent the shift-invariance problem from influencing the performance of the numeric downstream tasks is to keep the partition function stable. Let $= (\psi_1(k), \ldots, \psi_{|\mathcal{V}|}(k))^\top \in \mathbb{R}^{|\mathcal{V}| \times D}$ be the hidden representations of input time series sequence. Then the stability of the partition function can be assessed through the isotropy in the contextual embedding space [10, 82] as follows

$$I\left(\{\boldsymbol{\psi}(k)\}\right) = \frac{\min_{\boldsymbol{\psi}(k) \in \mathcal{C}} Z(k,l)}{\max_{\boldsymbol{\psi}(k) \in \mathcal{C}} Z(k,l)}, \tag{6.5}$$

where $\mathcal{C} = \boldsymbol{\Psi}^\top \boldsymbol{\Psi}$ is the input correlation matrix of input pattern and $l = 1, \ldots, L$. From (6.5), we can see that when the partition function is constant (i.e., stable) for different samples, $I(\{\psi(k)\})$ becomes close to 1 which indicates that the contextual embedding space $\{\psi(k)\}$ is more isotropic[10, 82]. Note that in (6.3), the probability of a value in any time instance is

the exponential of the corresponding logit $z_i(k,l)$ divided by the partition function $Z(k,l)$. If the partition function remains stable for different samples, the logits can be solely determined by the probabilities, thereby resolving the shift-invariance problem of the softmax function.

6.3 Study of isotropy in LLM hidden representations

Analysis settings. For illustrative purposes, in this section, we present two examples from a specific numerical domain, such as transport, and for a specific language model (i.e., GPT-2) that illustrate the conditions under which isotropy is preserved in LLM representations.

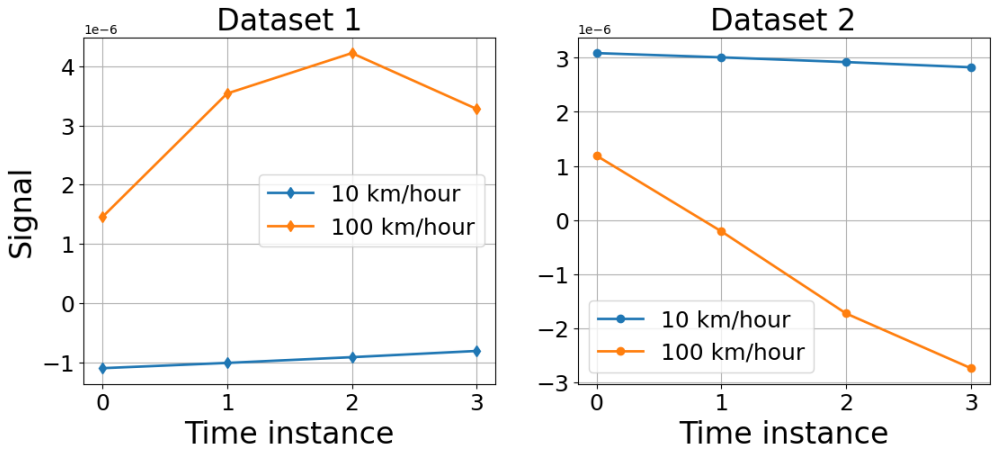


Figure 6.2: Visualization of transport Dataset 1 (Left) and Dataset 2 (Right) labels for two extreme cases of velocities, i.e., 10 km/hour and 100 km/hour.

In Section 6.4, we provide comprehensive evaluations across 22 different datasets from various numerical domains and other 5 different language models with different architectures and tokenization techniques. For this specific analysis, we select two datasets from transport as they are dynamic, noisy, time-varying, and thus hold all primary characteristics of the numerical data across various domains. Specifically, we use two ways of sharing signals which

are typically used for signal transmission in transportation settings, as shown in Figure 6.2. We call these two datasets as "Dataset 1" and "Dataset 2". The downstream task here is to predict the transport signalling property over wireless channel using LLM, where Dataset 1 causes good downstream performance (i.e., near optimal signal prediction), while the Dataset 2 causes bad downstream task performance (i.e., high error in signal prediction). We use NMSE as a performance metric for the numeric downstream task because it is widely used for signal prediction. We deploy the first six layers of GPT2 [97] and use the datasets and simulation setups from [71], which are the standard settings for time series forecasting. We predict $L = 4$ future signaling properties based on the historical $T = 16$ signal properties through time series forecasting using GPT2. The training and validation dataset contains $8,000$ and $1,000$ samples, respectively, with user velocities uniformly distributed between 10 km/hour and 100 km/hour. The test dataset contains ten velocities ranging from 10 km/hour to 100 km/hour, with $1,024$ samples for each velocity.

Table 6.1: The effective dimension $d(0.8)$

Layer	1	2	3	4	5	6
Dataset 1	4	4	4	4	4	4
Dataset 2	1	1	1	1	1	1

Effective Dimensions. In each layer of each model, we start with a data matrix $\mathbf{A} \in \mathbb{R}^{|\mathcal{V}| \times D}$, where $|\mathcal{V}|$ represents the number of tokens in the input time series sequence, and D corresponds to the embedding dimension. We apply PCA to reduce the dimensionality from D to m i.e., $\tilde{\mathbf{A}} \in \mathbb{R}^{|\mathcal{V}| \times m}$. Then

the fraction of variance captured by the reduced representation is given by: $r_m = \frac{\sum_{i=0}^{m-1} \sigma_i}{\sum_{i=0}^{D-1} \sigma_i}$ where σ_i denotes the i-th largest eigenvalue of the covariance matrix of $\mathbf{A}$. We define the ϵ-effective dimension as $d(\epsilon) \triangleq \arg\min_m r_m \geq \epsilon$. For instance, if $d(0.8) = 3$, it means that three principal dimensions retain 80% of the variance. A higher d suggests a more isotropic space [19], where information is spread across multiple dimensions rather than

being concentrated in a narrow subspace. Table 6.1 presents the values of $d(0.8)$ for different layers and models. Surprisingly, GPT-2 has so few effective dimensions, with $d(0.8) = 4$ for Dataset 1 and $d(0.8) = 4$ for Dataset 1, for layers 1 through 6, as compared to its original embedding dimensions $D = 768$. The reduced dimensionality suggests that GPT-2's embeddings lie in a subspace defined by a very narrow cone [40], and hence, there would be hardly any model capacity. Surprisingly, these language models are as successful as they are in numerical domains, given that most of their embedding vectors are as similar to one another as they are. These observations motivate us to look deeper into the contextual embedding space.

6.3.1 Clusters in the Contextual Embedding Space

Let $G(\ ) = (g_1(\ ), \ldots, g_{|\mathcal{V}|}(\ ))^\top : \mathbb{R}^{|\mathcal{V}| \times D} \mapsto \mathbb{R}^{|\mathcal{V}| \times D}$ be the function for self-attention, i.e., $g_i(\ ) = \text{softmax}(\ \ ^\top)$, where $\ = \mathbf{W}_Q \mathbf{W}_K^\top \in \mathbb{R}^{D \times D}$, and $\mathbf{W}_Q \in \mathbb{R}^{D \times m}$, $\mathbf{W}_K \in \mathbb{R}^{D \times m}$ are the parameter matrices for the query and key matrices of self-attention. The lemma below contributes to understanding why the isotropic property of pre-trained LLMs help to generalize to numerical domains. The proof of this lemma closely aligns with the approach in [62]. The proof of this lemma follows the analysis in [62] is provided in Appendix B.2 for completeness.

Lemma 6.2. *Consider the Jacobian matrix* $\mathbf{J} = \left[\frac{\partial g_i(\ )}{\partial \psi_j}\right]_{i,j=1}^{|\mathcal{V}|}$, *which represents the gradient of the self-attention mapping* $G(\ )$ *with respect to the input time series token embeddings. Then the spectral norm of* $\mathbf{J}$ *satisfies* $\|\mathbf{J}\|_2 \leq |\ |_2 \sum_{i=1}^{|\mathcal{V}|} \left(p_{i,i} + \frac{1}{2}\right) \left|\psi_i - \sum_{j=1}^{|\mathcal{V}|} p_{i,j} \psi_j\right|^2 + \Delta$, *where the residual term* Δ *is given by* $\Delta = |\ |_2 \sum_{i \neq j}^{|\mathcal{V}|} p_{i,j} \left|\psi_j - \sum_{q=1}^{|\mathcal{V}|} p_{i,q} \psi_q\right|^2 + \frac{|\ |_2}{2} \sum_{j=1}^{|\mathcal{V}|} |\psi_j|^2$, *and the attention weights* $p_{i,j}$ *are defined as* $p_{i,j} = \frac{\exp(\psi_i^\top \ \psi_j)}{\sum_{k=1}^{|\mathcal{V}|} \exp(\psi_i^\top \ \psi_k)}$.

From Lemma 6.2, we can see that to minimize the norm of the gradient $\|\mathbf{J}\|_2$, we essentially need to make $\sum_{i=1}^{|\mathcal{V}|} \left|\psi_i - \sum_{j=1}^{|\mathcal{V}|} p_{i,j} \psi_j\right|^2$ small. When $\ $ is small and all the input time series

token embeddings are centered at the origin, $\sum_{i=1}^{|\mathcal{V}|} \psi_i = 0$, we have $\sum_{i=1}^{|\mathcal{V}|} \left|\psi_i - {}^{\top} p_{i,:}\right|^2 \approx \sum_{i=1}^{|\mathcal{V}|} \left|\psi_i - {}^{\top}\ \psi_i\right|^2$ (see Appendix B.2).

The theorem below shows that minimizes the objective $\sum_{i=1}^{|\mathcal{V}|} \left|\psi_i - {}^{\top}\ \psi_i\right|^2$ and contains the m largest eigenvectors of correlation matrix ${}^{\top}$ of time series token embeddings, where m is the rank of . The proof of Theorem 6.3 is provided in Appendix B.3.

Theorem 6.3. *Let the eigenvalues of the correlation matrix* ${}^{\top}$ *be ordered as* $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_D$*, and let* $\gamma_i \in \mathbb{R}^D$ *for* $i = 1, \ldots, D$ *denote their associated eigenvectors. Then, the matrix* * *that minimizes the quantity* $\sum_{i=1}^{|\mathcal{V}|} \left|\psi_i - {}^{\top}\ \psi_i\right|^2$ *has the optimal form* $= \sum_{i=1}^{m} \frac{1}{\lambda_i} \gamma_i \gamma_i^{\top}$.

Theorem 6.3 shows that the self-attention learns to perform a function for the numeric downstream tasks through training, which are closely related to the m largest eigenvectors of LLM hidden representations. In other words, the self-attention mechanism effectively projects input time series tokens onto a low-dimensional contextual embedding space defined by the top eigenvectors of correlation matrix ${}^{\top}$.

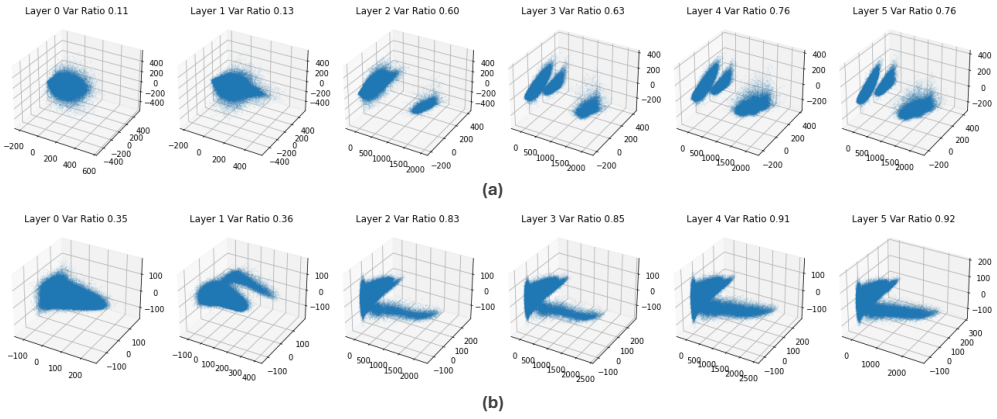


Figure 6.3: Variations in GPT-2's hidden representation for different datasets from the same domain: (a) PCA plot of contextual embedding space for transport Dataset 1. (b) PCA plot of contextual embedding space for transport Dataset 2.

Motivated by the findings from Lemma 6.2 and Theorem 6.3, we project the models' hidden representations of into a lower-dimensional space by utilizing the $m = 3$ largest eigenvectors

through PCA, as shown in Figure 6.3. The three axes of the figure represent the first three principal components of the covariance matrix of LLM representations of each layer. For instance, in layer 6, the first three principal components account for 76% and 92% of the total variance for Dataset 1 and Dataset 2, respectively. From Figure 6.3 a and Figure 6.3 b, we can see that there are disconnected or slightly overlapped islands that are far away from each other through layer 3 to layer 6. Note that the first principal dimension value spans from 0 to 2000, significantly wider than the other 2 dimensions, and dominates the total variance. A similar analogy can also be observed for transport Dataset 2 in Figure 6.3 b. In (6.1), the space isotropy is measured on pairs of arbitrary time series token representations, which could reside in two disconnected clusters. However, given that the variance is dominated by distances between clusters, such estimation would be biased by the inter-cluster distances. Hence, it is more meaningful to consider a per-cluster (i.e., local) investigation rather than a global estimate.

We start by performing clustering on the LLM representations in the contextual embedding space. There are various methods for performing clustering, such as k-means, DBSCAN [39]. We select K-means clustering method because it is reasonably fast in high embedding dimensions. We use the classical silhouette score analysis [110] to determine the number of clusters $|C|$ in the contextual embedding space (see Appendix B.4 for details). Since each LLM contextual embedding instance $\psi(k_i)$ belongs to a particular cluster through clustering, the cosine similarity should be measured after shifting the mean to the origin [82]. Accordingly, we subtract the mean for each cluster (i.e., centroid) and calculate the adjusted $\zeta_{\cos}$ in Section 6.1. Assuming we have a total of $|C|$ clusters, let $\psi_c(k_i) = \{\psi_c^1(k_i), \psi_c^2(k_i), \dots\}$ be the set of token k's contextual embeddings in cluster $c \in C$, and $\psi_c(k_i)$ be one random sample in $\psi_c(k_i)$. We define the adjusted inter-token cosine similarity as

$$\zeta'_{\cos} \triangleq \mathbb{E}_c \left[\mathbb{E}_{i \neq j} \left[\cos \left(\bar{\psi}_c(k_i), \bar{\psi}_c(k_j) \right) \right] \right] \tag{6.6}$$

where $\bar{\psi}_c(k_i) = \psi_c(k_i) - \mathbb{E}_{\psi_c}[\psi_c(k_i)]$. Here $\mathbb{E}_c$ is the average over different clusters, and $\bar{\psi}_c(k_i)$ is the original contextual embedding shifted by the mean, with the mean taken over the samples in cluster c [62]. The inter-token cosine similarity takes values between -1 and 1. A value close to 0 indicates strong isotropy and ensures the existence of structure in the LLM representations.

Isotropy in Dataset 1. In this section, we provide an example of Dataset 1 where a high prediction accuracy (i.e., low NMSE) is achieved by GPT2 based model. For instance, in Figure 6.4a, we compare the NMSE performance of our GPT2 based LLM with non-language models for different user velocities. From Figure 6.4 a, we can observe that the NMSE performance of all models gradually increased along with the increase in user velocity. This is because, with the increase in velocity, the transport signal characteristics rapidly change within a very short coherence time, resulting in increased prediction difficulty for the prediction model. The GPT2 based model consistently outperforms other baselines and demonstrates its high prediction accuracy.

For illustrative purposes, we pick three user velocities: 10 km/hour, 50 km/hour, and 100 km/hour, for isotropy assessment of Dataset 1. The GPT2 based model achieves good NMSE performance for all of these three velocities, as shown in Figure 6.4a. We apply inter-type cosine similarity $\zeta'_{\cos}$ in (6.6) to measure the isotropy in GPT2 embedding space. From Figure 6.4b, we can see that the GPT2 has consistent near-zero cosine similarity values for all layers, including layer 6. This indicates that nearly perfect isotropy exists in the GPT2 embedding space for the Dataset 1, which preserves the structure in the GPT2's hidden

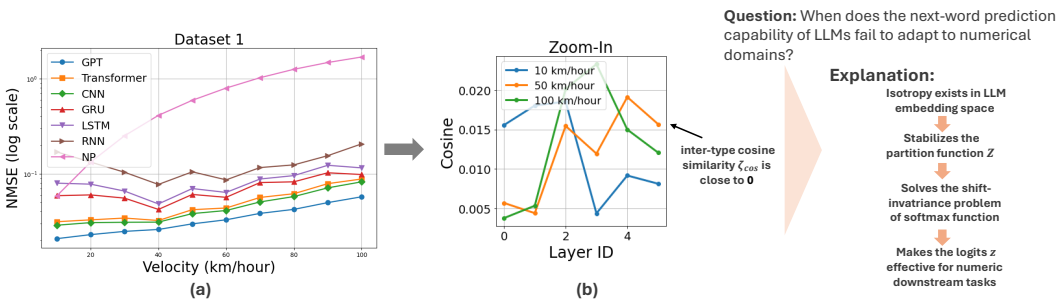


Figure 6.4: (a) LLM outperforms all other baselines for all of the ten different velocities for Dataset 1. (b) Inter-type cosine similarities for Dataset 1 with different velocities. ζ'$_{\cos}$ are close to zero for all the layers, including layer 6, indicating that nearly perfect isotropy exists in the LLM embedding space for the Dataset 1, which preserves the structure in the LLM's hidden representations and causes the high prediction accuracy.

representations and causes good downstream task performance.

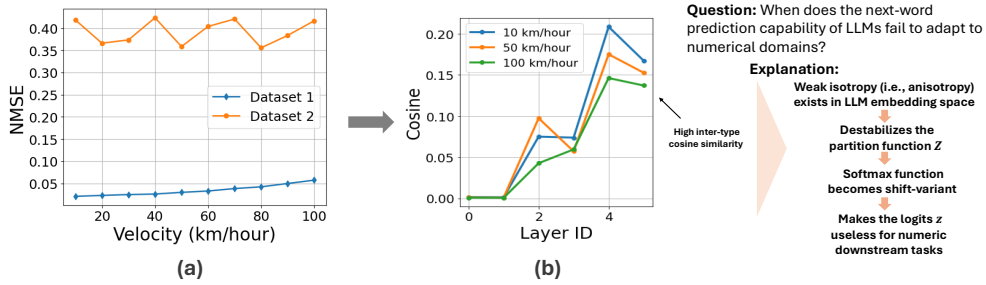


Figure 6.5: (a) The NMSE performance of the LLM based model for Dataset 2 deteriorates significantly compared to Dataset 1. (b) Inter-type cosine similarities for Dataset 2 for different velocities. Higher $\zeta_{\cos}$ values indicate a weak isotropy (i.e., anisotropy) exists in the LLM embedding space which causes a lack of structure in the hidden representations, yielding poor prediction accuracy.

Isotropy in Dataset 2. In this section, we provide an example of Dataset 2 where a poor prediction accuracy (i.e., high NMSE) is achieved by GPT2 based model. As shown in Figure 6.5 a, the NMSE performance fluctuates randomly for different velocities, while the NMSE performance for Dataset 1 is gradually increasing with increase in the velocities. The NMSE performance for Dataset 2 deteriorates significantly compared to the Dataset 1. As before, with the three user velocities, the NMSE performance for Dataset 2 for all of these velocities is worse as compared to Dataset 1, as shown in Figure 6.5 a. From Figure 6.5 b,

we can observe a weak isotropy (i.e., anisotropy) in the LLM embedding space for Dataset 2, causing a lack of structure in the GPT2 hidden representations, and thus leading to bad downstream performance.

6.4 Experiments

Baselines. We consider popular pre-trained LLMs as the baselines for numeric downstream tasks, including PatchTST [84], Lag-Llama [101], Moirai-1.0-R [136], Chronos-T5 [8] and Choronos-Bolt (https://huggingface.co/autogluon/chronos-bolt-base). The models use different architectures, time series tokenization techniques and hyperparameters for numeric downstream tasks as summarized in Table B.1 in Appendix B.5.

Datasets. We conduct a comprehensive evaluation using 12 different real time series datasets from various numerical domains, including energy, nature, finance, healthcare, retail and transportation (data sources can be found in Table B.3 of Appendix B.5). We also illustrate our findings using KernelSynth [8], a method that generates 10 more synthetic datasets via Gaussian processes in Section 6.4.

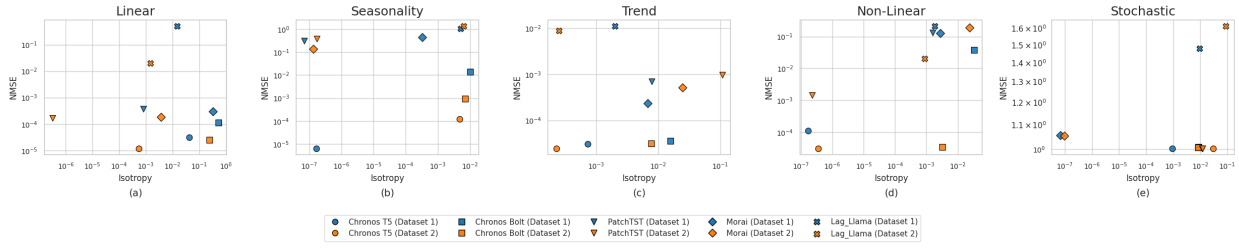


Figure 6.6: NMSE vs isotropy analysis for 10 different synthetic datasets of 5 different domains.

6.4.1 Qualitative Analysis

In this section, we analyze the time series forecasting by the baseline LLMs qualitatively. We focus on synthetically generated time series for a controlled analysis of different types of time series patterns which belong to 5 different domains, such as linear, seasonality, trend, non-linear and Gaussian perturbation. We are particularly interested in the isotropic measurement in the LLM's last layer as it is related to the logits and probabilistic inference. So all isotropic measure provided in this section is based on the last layer of the baselines.

We begin by analyzing time series forecasting performance (i.e., NMSE) for different baselines and its relation with isotropy in Figure 6.6. For instance, in Figure 6.6 b, the inter-type cosine similarity in hidden embedding spaces of Chronos-T5 for seasonality (Dataset 1) is -0.00007 and for seasonality (Dataset 2) is 0.0047. This indicates that stronger isotropy exists (i.e., inter-type cosine similarity value is close to 0) for Chronos-T5 for seasonality (Dataset 1) which preserves the structure in the Chronos-T5's hidden representations and causes good downstream task performance. On the other hand, a weaker isotropy exists (i.e., inter-type cosine similarity value is far from 0) for Chronos-T5 in seasonality (Dataset 2), causing a lack of structure in Lag-Llma's hidden representations, and thus leading to bad downstream task performance. Moreover, the NMSE for seasonality (Dataset 1) is lower for Chronos-T5, PatchTST and Lag-Llma, while higher for Chronos-Bolt and Morai, while for non-linear (Dataset 2), the NMSE is higher for all the baselines except Morai. A similar analysis can also be observed for all other synthetic datasets and baselines in Figure 6.6. This shows that the same dataset from any particular domain may cause different forecasting performances for different baselines, as it generates different contextual embedding spaces for language models with different architectures and tokenization techniques (See Appendix B.7).

Next, we examine the influence of isotropy on forecasting performance in two important

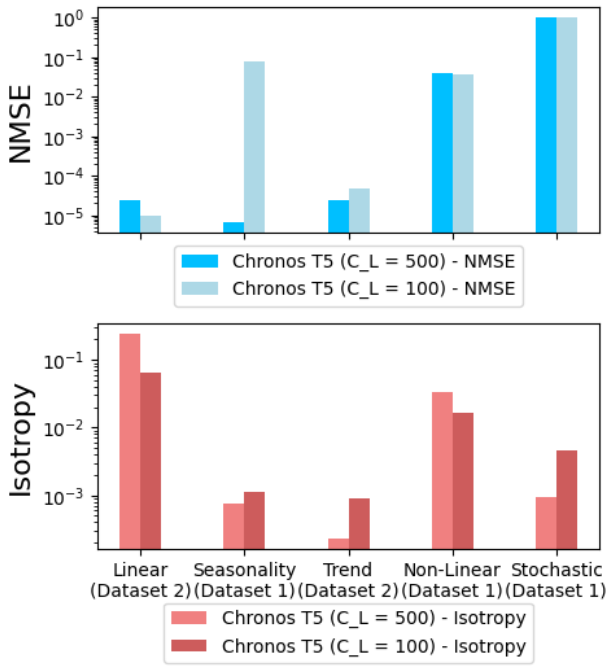


Figure 6.7: NMSE vs isotropy comparison across different input context lengths for synthetic datasets.

scenarios: a) different input context lengths, and b) different levels of noises in the input data. The first scenario is important as it provides an analysis on selecting proper input context length rather than selecting the length through random trials and errors. The second scenario is important as it gives us ideas on how the level of noise in noisy data impacts performance, since the data in the real world is mostly noisy.

Isotropy in different input context lengths. We first analyze the impacts of isotropy under varying input context lengths. We select Choronos-T5 as an example model and non-linear (Dataset 1) as an example dataset. In Figure B.3 in Appendix B.7, we show how the hidden representations of Chronos-T5 vary for two different input context lengths, such as $L = 500$ and $L = 500$, for non-linear (Dataset 1), which gives an indication of having different isotropic measures for different context lengths.

In Figure 6.7, we compare the NMSE vs isotropy across two different input context lengths,

$L = 500$ and $L = 100$, for different synthetic datasets. As can be seen from the figure, the isotropy values vary across different input context lengths and datasets. For instance, in seasonality (Dataset 1), we have (NMSE= 0.0000066, cosine similarity= -0.00076) and (NMSE= 0.0793, cosine similarity= 0.0011) for $L = 500$ and $L = 100$, respectively. The decrease in isotropy significantly increases the NMSE for the input context length $L = 100$. In contrast, in Linear (Dataset 2), the isotropy increases for the input context length $L = 100$, which causes the decreases in NMSE for chornos-T5. In practice, the input context length is often selected randomly or through trial and error, which may cause higher forecasting errors for different datasets. Isotropy analysis enables us to understand how varying input context lengths influence the hidden representations of the language model. This insight helps guide improvements in forecasting performance by examining the isotropic properties of the contextual embedding space.

Isotropy in varying noise levels in datasets. Next, we focus on the second scenario to see the impact of noisy datasets on LLM's performance. Again, we use the Chornos-T5 as an example language model. In Figure 6.8, we compare the NMSE vs isotropy across two different noise levels,

one without noise, and the other with Gaussian noise with $\sigma = 0.05$ standard deviation. From Figure 6.8, we can see consistently lower isotropy (i.e., inter-type cosine similarity far from 0) for all noisy synthetic datasets as compared to the datasets without noise. For instance, in trend (Dataset 2), we have (NMSE= 0.000024, cosine similarity= -0.00022) and (NMSE= 0.0012, cosine similarity= 0.0040) for $\sigma = 0$ and $\sigma = 0.05$, respectively. The decrease in isotropy significantly increases the NMSE for the noisy dataset. In practice, the environments of many real numerical domains, such as nature and energy, are noisy and dynamic. In such an environment, it is not always possible to measure the noise in real time and take the necessary steps to clean the input time series for better performance. However,

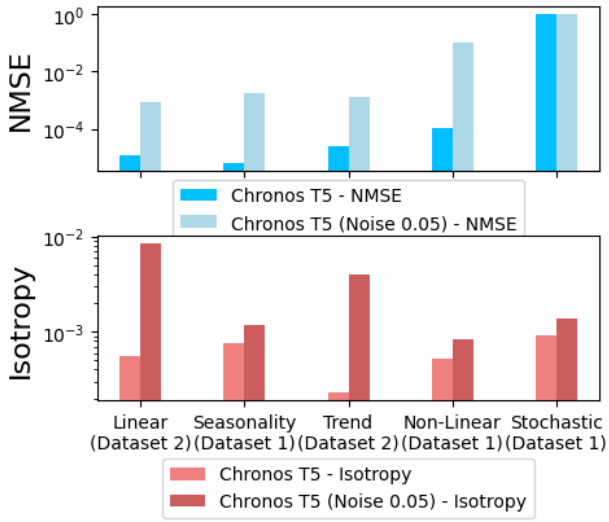


Figure 6.8: NMSE vs isotropy comparison across different noise levels in synthetic datasets.

it is always possible to measure the isotropy from LLM hidden representations, which can be used as a measure of noise in the input datasets, and thus, help to improve the forecasting performance.

6.4.2 Quantitative Analysis

In Figure B.1 in Appendix B.6, we analyze the time series forecasting performance for 12 real datasets with different baselines and its relation with isotropy. For instance, in Figure B.1 e, the inter-type cosine similarity in hidden embedding spaces of Morai for retail (Dataset 1) is 0.002 and for retail (Dataset 2) is 0.1931. This indicates that stronger isotropy exists for Morai for retail (Dataset 1) which preserves the structure in the Morai's hidden representations and causes good downstream task performance. On the other hand, a weaker isotropy exists for Morai in retail (Dataset 2), causing a lack of structure in Morai's hidden representations, and thus leading to bad downstream task performance. Moreover, the NMSE for retail (Dataset 2) is lower for all baselines except the Morai as compared to retail (Dataset

1), while the NMSE for energy (Dataset 1) is lower for all baselines except Lag-Llma as compared to energy (Dataset 2). A similar analysis can also be observed for all other real datasets and baselines in Figure B.1. This again shows that a dataset may have different impacts on forecasting performances for different baselines, and isotropy can be used as a measure of how the dataset generates different contextual embedding spaces for different language models based on their architectures and tokenization techniques.

Finally, in Figure B.2 in Appendix B.6, we compare the NMSE vs isotropy for varying input context lengths to observe its impact on the real datasets. We select Lag-Llma as our example model. We compare the results for two different input context length: 1) the recommended input context length $L = 144$ and the reduced input context length $L = 96$. As can be seen from the figure, the violation in recommended input context length by reducing it from $L = 144$ to $L = 96$ not only decreases the NMSE performances, but also increases for some datasets. For instance, the inter-type cosine similarity values become close to 0, i.e., from 0.1091 to 0.0012 and from 0.2014 to 0.0396, respectively, for nature (Dataset 2) and finance (Dataset 1), which in turn improves the NMSE performances.

6.5 Discussion

The isotropy in embeddings as studied here can serve as a foundation for future research on a deeper understanding of LLMs and their applications in various domains. Beyond isotropy, there could be other methods to approximate the partition function with a constant and make the logits useful for the numeric downstream tasks. Moreover, our isotropy study only ensured the existence of structure in the LLM hidden representations and provides a performance guarantee when the structure is preserved by isotropy. Improving the numeric downstream task performance when structure is not preserved in the LLM representations

is a topic of future work.

Overall, as of Chapters 3–6, we have proposed three approaches for synthetic tabular data generation (STG) and one indicator to anticipate the trustworthiness of the performance of the STG model. In Chapter 7, we present an application-oriented perspective, demonstrating how STG can be applied to next-generation wireless telecommunications, specifically in combination with large multimodal models and retrieval-augmented generation (RAG).

Chapter 7

STG for Next-Gen Wireless: A Real Telecom Application

As the final problem in this dissertation, Chapter 7 shifts from method development to real-world application. Building on the insights from Chapters 3–6, we provide a real future vision case study about how STG can be applied in next-generation wireless telecommunications, highlighting its integration with large multimodal models (LMM) and retrieval-augmented generation (RAG).

7.1 STG in Telecom for Privacy and Adaptivity

One of the notable motivations for adapting AI models, especially the proposed large multimodal foundation models, to the beyond-next-G wireless telecommunication framework is that both the frontline application scenarios and the backend processes,such as protocol and proxy design, are constantly evolving. Especially after 5G, with the emergence of 6G and beyond, the pace of technological development is outstripping human decision-making in protocol design. This is where the proposed vision of large multimodal models (LMMs) in the wireless economy finds broad and grounded applications.

However, such an AI-driven approach is, by nature, also data-driven. In the context of large multimodal models, various sources of wireless data can be utilized in the pipeline, including

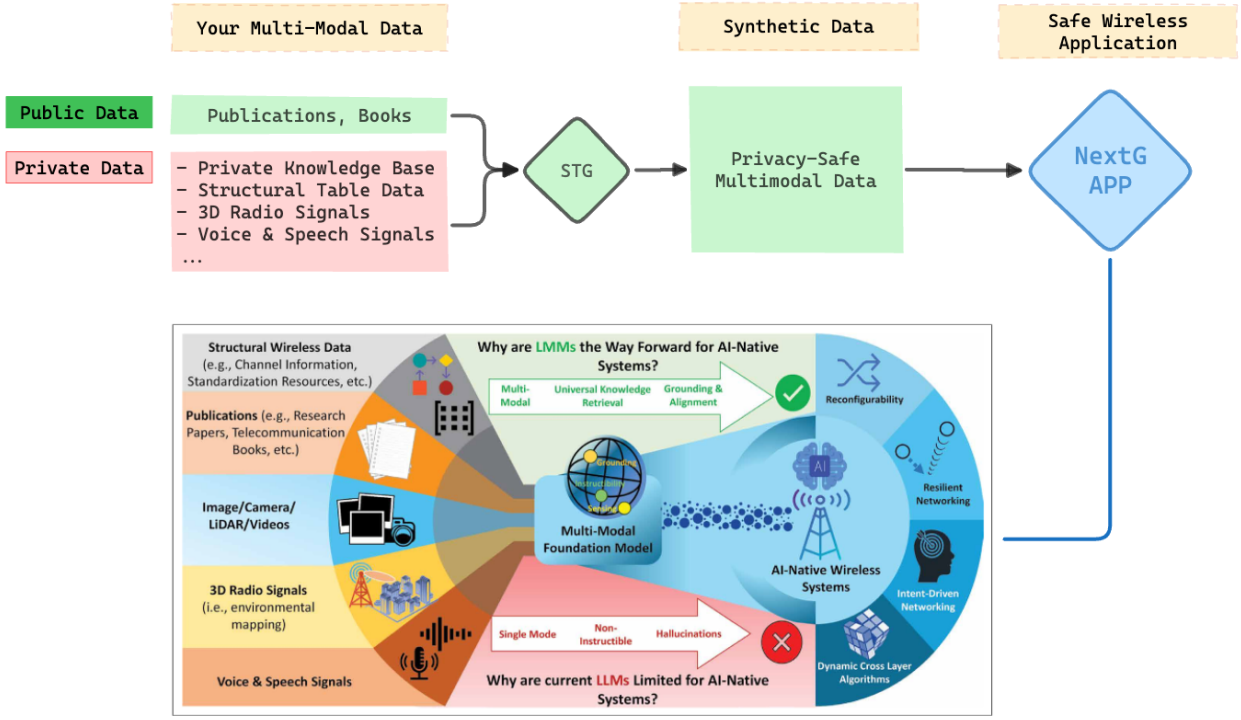


Figure 7.1: Illustrative figure of the proposed framework for STG and LMM empowered AI-native wireless systems with broad and grounded applications.

both public literature and private application-level data such as wireless traffic, 3D radio signals, and voice or speech signals, as shown in Figure 7.1.

Although public literature — similar to what powers advanced models like ChatGPT, DeepSeek, and LLaMA — provides a rich foundation of background knowledge, in wireless telecommunication, privately-owned data, both textual and numerical, plays an even more critical role in making domain-specific AI modeling effective. These AI models may be deployed as dialogue-based AI chatbots for question answering on private knowledge bases, or used to train specialized agents such as reinforcement learning-based planners, or models for suggesting optimal reconfigurations in next-generation network deployments.

Remember, the initial motivation of this dissertation is to enable the safe sharing of private data between different organizations, or even between different internal groups within the

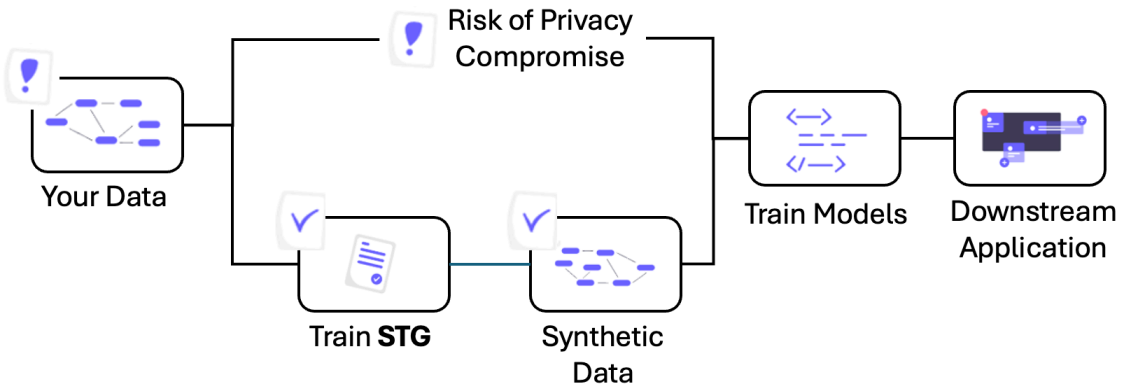


Figure 7.2: Illustrative figure of the proposed Synthetic Tabular Generation (STG) framework, enabling data sharing without compromising privacy. STG simulates realistic tabular data while preserving structural patterns and statistical properties, making it suitable for training AI models across organizational boundaries.

same organization. In conclusion, STG, as in Figure 7.2, allows the safe use of realistic synthetic large-scale data in heterogeneous formats, which is an essential condition for making the proposed LMM pipeline feasible for telecommunication tasks.

7.2 Scope of Synthetic Data and LMM in Next-G framework

Future AI-native wireless systems (e.g., 6G and beyond) must leverage machine learning (ML) and AI algorithms to design, optimize, and operate various facets of the network, including resource allocation, transceiver design, and others [111]. Consequently, cross-layer network functionalities implemented by AI models could enable advanced network capabilities, including: 1) *resilience*, enabling 6G networks to withstand disruptions and maintain connectivity even in challenging scenarios; 2) *intent management*, allowing networks to autonomously translate high-level business intents into closed-loop network configurations; 3) *big-data analytics*, enabling diagnostics using historical wireless data, addressing software or

hardware failures, improving communication and computing resource usage, and predicting future user and network behavior; and 4) *non-linear signal processing*, allowing networks to process multi-modal signal characteristics.

To achieve the above goals, a promising avenue is to explore generative AI's universal knowledge retrieval and generation capabilities, particularly foundation models such as large language models (LLMs). Trained on diverse datasets, LLMs can discern intricate patterns and offer insights for optimizing end-user experience in future wireless applications.

The main contribution of this chapter is the introduction of *grounded and instructible large multi-modal models (LMMs)* that are *universal* and have *alignment* capabilities . Here, *universal foundation model* for wireless systems are AI models tailored to handle a wide array of tasks and applications within the wireless domain, irrespective of the network architecture and standards. Our key contributions include:

- We propose the usage of Synthetic Tabular Generation (STG) in the pipeline of wireless-centric foundation models.

- We propose a novel framework for universal, wireless-centric foundation models, that goes beyond [12, 73, 120] by integrating the following capabilities into LMMs: 1) *Multi-modal data fusion*: fusing multi-modal sensing information to a shared semantic space thus enabling efficient training of universal foundation models, 2) *Grounding:* involving the creation of a wireless-specific language through retrieval augmented generation (RAG) [21] and leveraging causal reasoning, and 3) *Instructibility:* facilitating transparent interactions between the wireless environment and LMMs through online reinforcement learning (RL) and neuro-symbolic AI to perform logical and mathematical reasoning. This approach ensures *alignment* by developing trustworthy LMMs that can explain the reasons behind wireless data, propose cross-layer network actions

aligned with network goals, and accommodate physical constraints.

- Initial experimental results using RAG demonstrate that infusing more wireless context improves the accuracy of LMM responses, thereby reducing hallucinations compared to responses generated without wireless context. Furthermore, for mathematical questions, an LMM delivers accurate responses with proper rationale, showcasing its ability to reason effectively when grounded in the right context.

- We present a use case of intent-based management employing LMMs, comprising problem formulation, intent assurance, and a validation phase. The improved logical and mathematical reasoning capabilities (shown in experiments) enable LMMs to function as dynamic problem solvers. We demonstrate that logical and mathematical reasoning capabilities enable continuous monitoring of network performance–a critical aspect of building resilient networks. In contrast to deep RL-based methods, which may be constrained to specific domains, LMMs can accelerate network service recovery during failures by proposing a sequence of remediation actions.

- We highlight challenges in constructing universal foundation models, covering aspects such as network planning, acquiring diverse datasets, and adapting to evolving standards.

7.3 LMM-Empowered AI-Native Wireless Systems: Proposed Framework

To build universal foundation models the proposed multi-modal LMM framework is built upon the principles of multi-modal data fusion, grounding, and instructibility. The components of the proposed universal foundation models framework are discussed next.

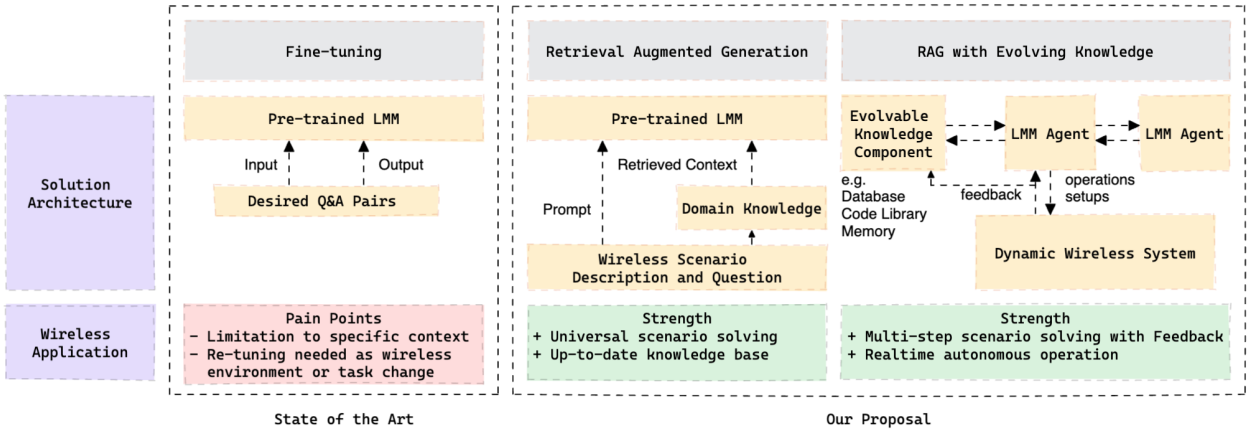


Figure 7.3: Applying LMM solutions for wireless applications: 1) Fine-tuning, 2) RAG, and 3) RAG with evolving knowledge.

Fusion of Multi-Modal Sensing Information: A Tradeoff Between Minimality and Redundancy

For capturing the real-world wireless environment, there is a need for precise sensing and mapping of its diverse surroundings. Prior works like [12] (and references therein) discussed exploiting visual generative AI models such as meta-transformers to map multi-modal wireless sensing information to a semantic latent space. By capturing the characteristics of the wireless environment, such models can enhance contextual and situational awareness for sensing applications in AI-native networks. However, providing the entire mapped sensing information (received from diverse sources) as input to train the LMM is resource-intensive and requires substantial time for retraining. This hinders the timely execution of dynamic updates essential for maintaining seamless connectivity. To address this limitation of meta-transformers inherent in [12], we propose to convey a compressed sequence of pertinent information to the LMM using dimensionality reduction. To achieve this, we start with the identification of physical symbols present across the multi-modal data. *Physical symbols* refer to abstract entities present in the data that have relevant semantics with respect to the wireless network. For example, this may involve associating symbols with various extrinsic

and intrinsic elements. *Extrinsic elements* encompass dynamic objects, such as scattering elements in the environment and users. Meanwhile, *intrinsic elements* involve static network features like network addresses and signal processing methods underlying wireless transmission or reception, among others. There could be redundancy among the information conveyed by the symbols across multiple modalities of data. To mitigate redundancy, the shared semantic latent space's construction should adhere to the information bottleneck principle [7]. In this framework, it is assumed that a dominant mode of information, called *prime modality*, exists within a dataset, serving as the primary source of information. Other modalities complement or enhance the information provided by this prime modality. Here, the compact representation for any modality should convey as little information as possible about the raw data, and, simultaneously, prime modality representation should convey maximum information about other modalities. This ensures that the resulting semantic latent space is of *minimal dimension* while *avoiding redundant information*. Further, for LMM training, we advocate using this filtered representations in the shared semantic space as inputs. Additionally, filtering also determines when to perform dynamic updates of the neural network (NN) parameters of the LMM, taking into account the nature of the captured data, which can be either static (e.g., 3GPP standards) or dynamic (e.g., wireless channel information).

While the fusion and filtering of multi-modal information and the training of LMMs is important, on its own, simply identifying symbols is not enough if LMMs are to be universal. While a vanilla LLM can effectively predict the events following an observed sequence of sensing information, it lacks precise understanding of what causes the event and its implications from a wireless system perspective. For example, translating images of trees in a wireless environment into a set of angles of arrival or departure that describe the RF signal propagation environment requires associating meaning from a wireless perspective with each

extracted physical symbol. This aspect, called *grounding*, is detailed next.

Causal Reasoning for Grounding in LMMs: Reducing Hallucinations and Bolstering Trustworthiness

Traditional grounding approaches typically entail creating a knowledge base, which represents an instance of symbolic AI. The knowledge base captures the possible logical relations among physical symbols, such as scattering objects, users, network topology, and transmission or reception parameters. Nevertheless, the use of knowledge base methods faces scalability challenges as the number of relations and physical symbols expands. To overcome this limitations in conventional grounding methods, we propose that LMMs infer the relations among various physical symbols identified using *causal reasoning* [123], as discussed next.

While specific experiments demonstrate that language models might exhibit causality, it is predominantly attributed to the causal knowledge ingrained in the training data, rather than indicative of LLMs possessing inherent causal understanding. In [147], a gradient-based, transformer-type algorithm for zero-shot optimal covariate balancing for causal treatment effect is introduced. We propose to advance [147] by incorporating theoretical methods to construct causal foundation models, focusing on wireless concepts as the relevant physical symbols. Here, one may ask: *how to identify the causal relations among physical symbols and how to ensure that the learned relations are aligned with the wireless concepts in standards and textbooks?* One common approach is to perform fine-tuning [12] that takes a pre-trained language model trained on large amounts of general text and then continue to train it on a small-scale task-specific text. Fine-tuning is appropriate if the user specifically knows the ground-truth causal relations. For wireless scenarios, fine-tuning can be beneficial for constructing a wireless specific chatbot capable of extracting valuable information from its

knowledge base. However, fine-tuning LLMs may impose limitations on the wireless applications supported. This limitation arises from the narrow set of NN parameters that are tuned during the fine-tuning process (limited degrees of freedom). This, in turn, requires re-tuning as the wireless environment or task change. To address these limitations, we suggest the use of RAG coupled with causal discovery, as discussed next and in Fig. 7.3.

How to perform causal discovery through RAG?

Through querying from a wireless-specific database that includes wireless textbooks, research papers, 3GPP standard, or any device instruction handbook, RAG [67] enables the LMM to understand the domain-knowledge context. Once this information is retrieved, the generation component of RAG can help formulate new content that infers or expresses causal relations among the identified physical symbols. For example, when the LMM is tasked with deducing causal relationships between scattering objects in the environment and channel parameters like angle of arrival (AoA) or angle of departure (AoD), RAG can map these wireless observations to the underlying physical concepts from the database.

Through an evolvable external knowledge component and multi-agent cooperation, RAG can allow the implementation of emerging applications that require multi-tasking, like in connected homes or industrial robots. Moreover, performing retrieval from an evolving knowledge base can enable universal knowledge retrieval for semantic communications [122], and intent management. With its continuous learning capability, RAG, with evolving knowledge, enables continually updating the wireless algorithms across all open systems interconnection (OSI) layers while ensuring compatibility with the advancements in the semiconductor industry and software solutions. This proves advantageous, particularly for intent management and resilience (see Section 7.4 and 7.4). Apart from the continual learning capability, RAG with evolving knowledge enables the knowledge retrieved to be dynamically adjusted to cater

to the specific application demands. For example, in a multi-user communication system, the retrieved literature on signal processing algorithms must differ from what might be required in a single-user scenario.

What does causal discovery through RAG entail for LMMs?

Grounding via causal discovery entails endowing LMMs with the capability to comprehend the causal relationships among physical symbols and subsequently engage in causal inference through interventions and counterfactuals [123]. Through interventions and counterfactuals, the LMM can indulge in chain-of-thought kind of reasoning, where it analyzes a sequence of causal state-action pairs $(\boldsymbol{s}_t, \boldsymbol{a}_t)$ and their effects, $\boldsymbol{s}_0 \xrightarrow{\boldsymbol{a}_0} \boldsymbol{s}_1 \xrightarrow{\boldsymbol{a}_1} \boldsymbol{s}_2 \cdots \xrightarrow{\boldsymbol{a}_{N-1}} \boldsymbol{s}_N$. This ability facilitates long-term planning for wireless resource allocation, signaling schemes for transmission and reception (and may include beamforming, modulation, coding, and control signaling, among others), and quality-of-service (QoS) management, thereby contributing to the establishment of robust and resilient wireless systems.

Instructiblity from Environmental Feedback

For instructibility, LMMs should be able to dynamically adjust resource allocation, signaling policies, and many other cross-layer network functionalities in real time, catering to diverse tasks, environments, and optimization objectives. Additionally, they should be able to continuously monitor wireless observations to identify and address any unforeseen issues that might impede seamless network connectivity. A standard approach for dynamically adjusting wireless resource management and signaling schemes based on user feedback using deep RL. However, deep RL techniques are task-specific and require retraining when the wireless environment and optimization objectives change. While multi-task RL solutions

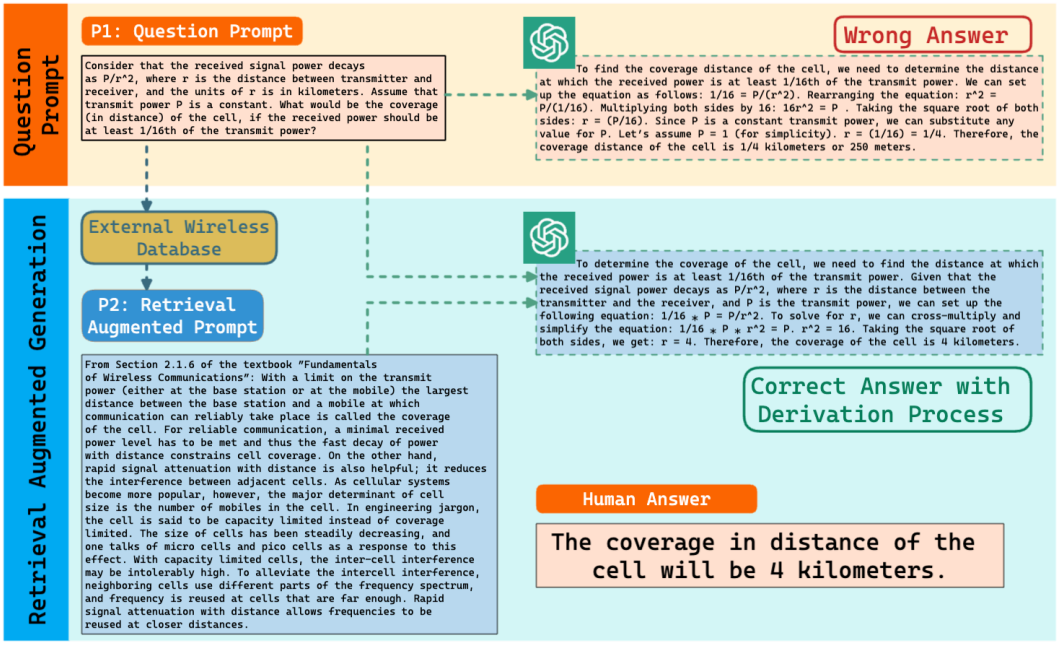


Figure 7.4: A sample mathematical Q/A pair from the dataset.

exist, they are mostly limited to specific domains or OSI layers. Moreover, they lack the ability to continuously evolve their state and action space to cope with changes in standards or advancements in wireless technology. Conventional multi-task RL solutions also cannot perform abductive reasoning, a crucial aspect for making inferences about missing data or determining the best explanations for observed data. These features are essential for achieving dynamic adaptability and reconfigurability, necessary for resilience and intent management. They are also crucial for supporting abductive reasoning capabilities required for semantic communications and other related tasks. We next discuss the key components needed to provide instructibility to LMMs. We begin by detailing the framework incorporating communication context, prompting, and an online LMM with wireless environment feedback, contributing to establishing an instructible system. Subsequently, we explore how to instill LMMs with logical and mathematical reasoning, that are essential for constructing self-evolving and dynamically adaptable systems, thus achieving instructibility.

Communication context

Causal representations that are used to represent the physical symbols, similar to tokens in NLP-based LLMs, form the *communication context* for an LMM. This context encapsulates critical aspects of the wireless communication scenario. The components of the LMM context include:

- **Network setup** including details about 1) *communicating devices*, 2) *communication link* (downlink or uplink), and 3) *physical topology* that describes the antenna configuration, as well as any miscellaneous network architecture.

- **Communication constraints** including constraints on the total power and shared communication/computation resources aross frequency, time, and other dimensions.

- **Wireless standards/text snippets** read using RAG (Section 7.3), that include excerpts from relevant wireless communication standards or documents, providing a contextual basis for the communication scenario.

- **End-to-end optimization objectives** that may include quality-of-service (QoS) measures such as average throughput, delay, reliability/quality-of-experience.

- **Historical wireless data** that may involve diverse measurements such as uplink pilots, user feedback on channel quality indication, various sensing measurements, and received uplink signal measurements, among other relevant parameters.

Next, we explain how to construct an online LMM by instructing it with environmental feedback.

Online LMM with wireless environment feedback using neuro-symbolic AI

One common apporoach to instill instructiblity is to use an iterative prompting mechanism in which an LMM is guided through multiple rounds of interaction with human prompts. In each iteration, the model refines and improves subsequent responses using the feedback (e.g., the QoS results based on the wireless policy of the LMM) from the previous round. However, iterative prompting requires human intervention. We propose building an online LMM framework to address this limitation and enable the development of autonomous wireless systems. In this setup, LMM functions as the wireless policy and is operationally embedded within an interactive setting using online RL. This entails utilizing gathered wireless observations and feedback from the environment to iteratively enhance its functionality, aligning with goals expressed in wireless language. The formulation of the LMM-powered cross-layer network functionalities can be represented as a partially observable Markov decision process. Here, the states are defined by the communication context and prompts, actions are represented by the wireless policy suggested by the LMM, and rewards are determined using performance metrics obtained from the wireless environment. If available, the network goal or intent can be articulated in natural language by the network operator. To ensure continuous operation without disrupting connectivity, online LMMs should possess the ability to explain wireless observations and infer any missing data, necessitating logical reasoning capabilities. Furthermore, given that many wireless concepts can be expressed mathematically, LMMs must inevitably be capable of performing mathematical reasoning. This includes tasks such as channel predictions, beamforming vector computations, channel quality measurements, and many other cross-layer network computations.

Here, multi-task RL can be an alternative approach to perform diverse wireless tasks. However, since they lack logical and mathematical reasoning capabilities, we advocate incorporating them through the use of *neuro-symbolic AI* [122]. In our setting, symbolic AI

serves to evaluate diverse logical and mathematical formulas, while the neural component is responsible for learning the logical and mathematical equations from wireless observations and context information. When prompted with communication context and grounded wireless observations using causal discovery (Section 7.3), symbolic AI connects facts and data through rules and algorithms, resembling the cognitive operations of the human brain in storing high-level concepts and engaging in nuanced inference. To prevent hallucination, LMMs must have the ability to explain wireless observations and infer any missing data. Additionally, they should understand the connections between various physical symbols through symbolic AI. Here, a viable approach is to develop a formal logical language that can encapsulate the exhaustive ontology of wireless concepts and articulate rules governing the functioning of wireless systems (and is the symbolic part here). This strategy is reminiscent of the Cyc concept [66], which serves a similar purpose for web-based data. Beyond their lack of logical reasoning abilities, existing LLMs face challenges in accurately capturing mathematical formulas and executing mathematical derivations. To overcome this limitation, a promising approach involves leveraging a neuro-symbolic problem solver [96], having three main components. First, a *problem reader* encodes math word problems, presented as textual prompts, into vector representations. Second, a *programmer* generates symbolic grounded equations, which are executed to produce answers. Lastly, a *symbolic executor* obtains final results. In this setup, the programmer learns the weights (neural part) that establish connections between various mathematical symbols. The resulting neuro-symbolic problem solver enables the construction of dynamic problem solvers, a critical component for intent management and resilience, as discussed in Section 7.4 and 7.4.

7.4 Experimental Validation: A Case Study with Private Knowledge Base Q/A.

Here, we first demonstrate illustrative experiments conducted on a dataset specific to wireless scenarios using RAG. We highlight the enhanced performance of LMM compared to vanilla LLMs which does not have any wireless context, characterized by succinct explanations (resulting in *reduced hallucinations*), precise answers (demonstrating *grounding in wireless concepts*), and well-founded rationales (illustrating *mathematical reasoning* capabilities). Additionally, we discuss how the results indicate the potential application of the proposed LMM in addressing specific challenges in future wireless networks.

Application 1: Private Knowledge Base Q/A

To evaluate the efficacy of RAG in wireless contexts, we conducted certain Q/A experiments, where a sample question from the dataset is shown in Fig. 7.4. In the RAG process, relevant paragraphs are extracted from [126] to serve as wireless context information. This information encompasses a combination of textual content and mathematical symbols and equations. This structured wireless information represents a simple multi-modality case. Table 7.1 shows an evaluation using 16 human participants on responses from different prompting methods of 4 conceptual wireless questions and 7 mathematical wireless questions. With the help of retrieved knowledge context, common LMM evaluation metrics (see Table 7.1 for definitions), including precision, recall, F1 score, and ROUGE-L measure, indicate a performance improvement over vanilla LLMs ranging between 15% to 30%. We also interpret the human evaluation result in the following way: 1) For conceptual questions, the standard *Question Prompt* can retrieve reasonable rationale, while RAG can refine the assertion, leading to an 8% improvement in the assertion metric. Furthermore, the over-

explaining metric, which gauges the alignment of responses with human expectations, shows a remarkable almost three-fold improvement for RAG compared to vanilla LLMs. Improved assertions imply that LMMs can mitigate hallucinations in their responses, thereby aligning more effectively with the goals of the network. 2) In the context of mathematical questions, *Retrieval Augmented Prompt* consistently provides the correct answer and can offer more detailed mathematical derivatives. Specifically, the rationale exhibits a 22% improvement with RAG, while the derivative steps are 81% more detailed than vanilla LLMs. This indicates that LMMs exhibit enhanced logical and mathematical reasoning abilities.

As discussed earlier, the demonstrated logical and mathematical reasoning capabilities, evidenced by improved rationale measures, enable LMMs to map wireless observations and context information fed as input to them into a mathematical problem formulation. This empowers LMMs to function as dynamic problem solvers (defined in Section 7.3). To exemplify the application of these capabilities in future wireless systems, we next discuss a few use cases, including intent management and resilience.

Application 2: LMMs for Intent Management

A recent work that exploits LLMs for intent management appeared in [120]. However, this prior work is limited to using an LLM as a chatbot to convert human specified intent in natural language to infrastructure level intents as network service descriptors. Furthermore, the authors in [120] utilize human feedback to enhance the configurations generated by the LLM, hindering their ability to ensure intent assurance autonomously. In contrast to [120], we propose to employ LMMs for various phases in intent management spanned across the OSI layers:

- *Problem formulation phase:* The network must autonomously translates the opera-

tor specified intents into an optimization problem, considering multiple objectives and physical constraints. An LMM can facilitate dynamic problem formulation without human intervention since they possess better logical and mathematical reasoning abilities. Moreover, LMM responses are grounded in wireless physics, as discussed in Section 7.3, and clarified in Fig. 7.4. It can thus formulate precisely wireless optimization problems, for resource allocation, signaling schemes, or a combination of cross-layer objectives.

- *Intent assurance phase:* Leveraging a continuous stream of wireless measurements, LMMs can function as intent *assurance agents*. Using neuro-symbolic AI capabilities, these agents can assess logical formulas representing the desired intent (typically defined by QoS targets). If the intent is not fulfilled, LMMs can identify and articulate the specific issues that need resolution, guiding efforts towards achieving intent assurance within a specified timeframe.

- *Validator agent:* For solving the LMM-designed problem formulation provided we can use multiobjective RL with causal reasoning games building on [123]. Validating solutions against regulatory norms and physical constraints for long-term intent fulfillment is crucial. This is the *alignment goal* described previously in Section I. To autonomously manage intent, a validator role can be fulfilled by LMM, possessing a solid understanding of wireless concepts.

Next, we discuss the impact of minimizing hallucinations through improved assertions and providing precise answers (as reflected in overexplaining metric). This capability is essential for swiftly recovering from network service disruptions and thereby ensuring resilience.

Evaluation Measure	Question Prompt	Retrieval Augmented Prompt
Precision (↑)	0.06	**0.08**
Recall (↑)	0.59	**0.65**
F1 Score (↑)	0.11	**0.14**
ROUGE-L (F-measure) (↑)	0.17	**0.20**
Over Explaining (↓)	0.34	**0.12**
Conceptual Question Rationale (↑)	**0.89**	0.84
Conceptual Question Assertion (↑)	0.88	**0.95**
Mathematical Question Rationale (↑)	0.77	**0.94**
Mathematical Question Assertion (↑)	0.76	**0.97**
Mathematical Question Derivative Steps (↑)	0.48	**0.87**

Table 7.1: Prompting GPT-3.5 Turbo with retrieval-augmented context shows a general advancement over purely prompting with questions in 4 quantitative measurements (upper section) and 6 human-evaluated measurements (lower section). For each metric (row), the symbol (↑) indicates that higher scores are better, and better results are highlighted in **bold** for the two prompting methods. **Precision**: the number of shared words to the total number of words in the generated answers; **Recall**: the number of shared words to the total number of words in the human answers; **F1 score**: $\frac{2 \cdot Precision \cdot Recall}{Precision + Recall}$; **ROUGE-L (F-measure)**: based on the longest common subsequence (LCS) between the generated answer and human answer, which indicates that a longer shared sequence should indicate more similarity between the two sequences. **Human Evaluated Score**: Participants are asked to rate each Q/A sample without knowing the source of it. They will give score 0 for no and score 1 for yes for each of the Rationale, Assertion, and Over Explaining items. Mathematical questions require an additional derivative step to be scored.

Application 3: LMMs for Resilient Networking

Resilience is the ability of wireless networks to: a) detect or predict in advance any failures or performance disruptions arising due to network functionality issues across any OSI layer, changing wireless environment, user dynamics, or external malicious influences; and b) recover back to their normal functionality within a stipulated time frame, thereby ensuring seamless connectivity for all connected devices. In [123], we proposed a robust framework for building resilient wireless networks causal Bayesian optimization. However, the application of our solution in [123] is limited, because it mainly focuses around quickly recovering from QoS deviations in the network. However, network service disruptions can stem from changes in the wireless environment or malfunctions in hardware or software functionalities across diverse edge devices. To address this challenge, we suggest leveraging LMMs equipped with causal knowledge, not only pertaining to the wireless environment but also grounded in wireless standards and cross-layer network functionalities. Such a universal foundation model can handle service disruptions across multiple domains and tasks by operating in a closed loop fashion as discussed next.

- *Continuous monitoring of service disruptions:* To detect network service disruptions, the LMM should continuously monitor and predict potential issues across OSI layers. For example, consider a situation where the software code representing functionality at any OSI layer on an edge device becomes corrupted due to processor malfunctions. Alternatively, critical information intended for storage on an edge server might face corruption due to jamming attempts. Herein, since LMMs are grounded in wireless concepts, they can adeptly analyze error messages and descriptions of software malfunctions or data corruptions, offering suggestions aligned with network standards to rectify the issues, without any human intervention. This approach enhances the model's capability to provide context-aware solutions across diverse tasks or domains

or environments. Furthermore, LMMs can be consistently prompted to check for potential wireless environment issues that might lead to performance deviations in the near future. Such network issues can be formulated as either logical formulas or mathematical equations. For example, a potential logical formula could be $p : X \rightarrow (Y < \tau)$, signifying that with probability p, the wireless observation X results in a performance Y below the expected target τ. However, vanilla LLMs face challenges in handling such logical and mathematical problems, as discussed in Section 7.3. In this context, logical and mathematical reasoning capabilities using neuro-symbolic problem solvers play a crucial role in assessing performance quality. In contrast to [123], which necessitates the construction of specific causal models for monitoring particular tasks or QoS targets, the universal nature of LMMs allows a single model to be used for monitoring performance deviations and software or hardware malfunctions across any OSI layer. Given that a failure is detected, we next look at how the LMM can help the network functionalities quickly (within a stipulated time) recover back to the expected performance.

- *Network service recovery via LMM:*

  As detailed in Section 7.3, RAG enables the model to comprehend the causal implications of network actions by grounding wireless observations to the extracted knowledge. This enables LMMs to execute the minimal interventions required to restore the network to a normal functioning state. Further, as discussed in Section 7.3, instructibility allows LMMs to generate a sequence of network actions in response to the feedback from the wireless environment. These actions can involve repairing malfunctioning code, adjusting resource allocation, or refining signaling schemes to restore the network to normal functioning. In contrast to [123], which might require separate causal AI models to monitor diverse functionalities across OSI layers, the universal-

ity of LMMs can possible enable faster switching between tasks requiring repair or refinement, utilizing a single AI model.

7.5 STG Data Repository to Support Universal Wireless Foundation Model Training

Training a foundation model for wireless communication enables domain-specific optimization, efficiency gains through reduced NN weights, and enhanced performance. However, achieving this long-term goal requires collaboration among stakeholders in wireless communication and computer science. The associated challenges include the need for seamless interdisciplinary cooperation, addressing diverse communication standards, and incorporating evolving technologies to ensure the model's adaptability and effectiveness.

- *Diverse and representative datasets:* To ensure seamless connectivity across different wireless environments and diverse applications, an LMM should be trained under diverse signal conditions, interference patterns and fading scenarios.

- *Adaptability to evolving standards and technologies:* The incorporation of evolving 3GPP standards into the foundation models is crucial. This integration ensures that the decisions made by the LMM comply with both the network and the unified 3GPP standards. This alignment contributes significantly to enhancing the trustworthiness of the language model by ensuring its compatibility and compliance with the latest industry standards. In addition to leveraging LMMs for understanding and adhering to existing standards, they can also play a crucial role in the creation of standards, especially in scenarios where technologies are not standardized. LMMs, with their capacity for natural language processing and generation grounded in wireless concepts,

can contribute to the formulation and documentation of wireless standards, fostering innovation and clarity in technology development.

7.6 Discussion

This chapter developed a new framework for Synthetic Table Generation (STG) with designing AI-native wireless systems (6G and beyond) for multiple tasks using foundation models built on multi-modality, grounding, and instructibility principles. We conclude with following key recommendations:

- **Building a repository of synthetic (STG) wireless datasets:** While the results based on RAG offer unique insights into LMMs' capabilities, it is essential to acknowledge the challenges associated with generating a comprehensive dataset. To address this, we recommend the creation of an open-source ontology for wireless concepts and algorithms, sourced from a curated selection of textbooks and wireless literature, encompassing 3GPP standards. This approach ensures the quality, reliability, and trustworthiness of the dataset, making it applicable for research and development across the entire wireless community.

- **Speeding up next-G standardization to system design:** LMMs can assist in the swift prototyping of diverse system design scenarios. Leveraging the capabilities of RAG, LMMs can retrieve pertinent text-based descriptions and specifications by considering the provided input, whether it be a network intent or system design goals. This enables LMMs to actively contribute to rapidly exploring design alternatives and their associated implications.

Chapter 8

Conclusion and Future Work

In this dissertation, we have proposed and developed novel approaches to synthetic tabular data generation (STG), addressing key challenges related to data fidelity, model efficiency, and trustworthiness. Our work spans deep learning, large language models (LLMs), and domain-informed prompting, culminating in both methodological contributions and real-world applicability.

8.1 Conclusion

In Chapter 3, we introduced an auto-regressive DNN-based approach leveraging convolutional networks and mixture density networks to capture both row and column dependencies in tabular data and generate the synthetic tabular data. In Chapter 4, we proposed an LLM fine-tuning framework (PAFT) to improve generation fidelity while minimizing the need for manual feature engineering, addressing limitations in DNN-based models. In Chapter 5, we tackled the computational cost of LLM fine-tuning by introducing Knowledge-Guided Prompting (KGP), which leverages prior domain knowledge to guide in-context learning for efficient and scalable STG. In Chapter 6, we shifted focus to model evaluation by proposing embedding isotropy as a trust indicator for LLM-based STG, offering a lightweight, model-internal signal for anticipating generation quality. In Chapter 7, we demonstrated an application of STG in next-generation wireless telecommunications, showcasing its integra-

tion with large multimodal models and retrieval-augmented generation (RAG), highlighting the practical relevance and extensibility of our methods.

In each of the aforementioned problems, we have demonstrated through rigorous qualitative and quantitative results the effect of incorporating structural priors, adaptive modeling techniques, and trust indicators into the STG pipeline. Together, these contributions pave the way for more reliable, scalable, and high-fidelity synthetic data generation in complex real-world domains.

8.2 Future Works

Looking ahead, we identify four key directions for advancing Synthetic Tabular Generation (STG) in future research. First, we aim to enhance the functionality of STG-Toolbox by advancing STG Benchmarking and Isotropy-Aware Data Augmentation. Beyond this, we propose three specialized research directions: 1) Localized Editing STG (LE-STG) for flexible, region-specific table modifications; 2) Multimodal STG (MM-STG) for integrating tables with text and images; and 3) Agent Dialogue-Driven STG (AD-STG) for embedding structured tables into AI-agent systems to support dialogue, planning, and simulation.

8.2.1 STG Benchmarking and Isotropy-Aware Data Augmentation in the STG-Toolbox

To maximize the practical value of Synthetic Tabular Generation, it is pioneering to develop two key components:

- **STG Benchmarking**: Build benchmark datasets to evaluate different STG meth-

ods across a range of metrics, enabling standardized and comprehensive comparisons. For instance, a running example dataset could be used to demonstrate synthetic data quality improvements across entirely different approaches, including GANs, deep neural networks (DNNs), diffusion models, large language model (LLM) fine-tuning, and LLM prompting. Moreover, for different types of tabular data, domain-specific evaluation metrics should be carefully designed to capture relevant aspects of data quality.

- **Isotropy-Aware Data Augmentation**: Given a dataset with poor isotropy, learn to generate a complementary synthetic sub-dataset that, when combined with the original data, can improve model performance.

8.2.2 Localized Editing Synthetic Tabular Generation (LE-STG)

Although most previous work in synthetic tabular generation has focused on generating entire tables, single table or multitable, our approach introduces a new paradigm: *Localized Editing Synthetic Tabular Generation (LE-STG)*. This extends the generation task to support precise, region-specific edits within tabular data while maintaining statistical validity and semantic coherence. Unlike AI-assisted spreadsheet systems that automate cell-level operations, this method emphasizes the preservation of the joint value distributions and structural relationships inherent in the data. Potential use cases and applications for this technique include the following:

- **Data augmentation**: Local edits to a subset of the table can be propagated to surrounding entries to maintain a realistic and valid distribution.
- **Data security and validation**: Enables consistency checks between partial regions of a table to determine whether they likely originate from the same data source.

- **Data poisoning and adversarial generation**: Facilitates the injection of targeted, adversarial content that may appear legitimate to human reviewers but is crafted to compromise downstream AI models. For example, inserting realistic but fabricated financial records that undermine fraud detection systems.

8.2.3 Multimodal Synthetic Tabular Generation (MM-STG)

Large multi-modal models have significantly extended the applications of generative AI due to their rich representation capabilities, including contrastive language-image generation [98], document editing [119], document information retrieval [28, 50], diagram generation [145], and wireless 6G telecommunications [143]. From a technical perspective, models such as ViT-backed Masked AutoEncoder (MAE) [53], T5 [100], and BERT [32] have become well-known for their efficient parameterization, adaptable architectures, and strong learning capabilities.

We are developing a unified model, UMAP, that is designed to integrate text, image, and tabular data in various formats. It supports both discriminative learning and the generation of structured data. UMAP leverages the functional correlations between textual mathematical descriptions, graphical function plots, and tabular data samples to model all three modalities using a single, unified representation.

The core of UMAP is an innovative Vision-Text-Csv Transformer that combines pretraining with a wide range of domain-specific downstream tasks within a sequence generation framework. To the best of our knowledge, this represents the first approach in tabular generative AI to support both multidirectional modality synchronization on STG and STG-image editing simultaneously.

Potential use cases and target audiences for this technique include:

- Editing existing tabular data based on conditioning from a distribution plot (partial distribution).
- Injecting specific behaviors by editing partial segments of a table.
- Generating image patches corresponding to modified portions of tabular data.
- Converting high-level natural language descriptions into targeted table edits.

8.2.4 Agent Dialogue-Driven Synthetic Tabular Generation (AD-STG)

AI agents have emerged as a powerful paradigm with the rapid advancement of Large Language Models (LLMs) and Large Multimodal Models. One particularly notable example is *Generative Agents: Interactive Simulacra of Human Behavior* [90], which introduced a sandbox village populated with autonomous agents. Other agent-based frameworks have demonstrated diverse applications, such as open-ended gaming [132] and embodied interaction [153]. Although these systems have explored dialogue generation, planning, and interaction, the use of *Synthetic Tabular Generation (STG)* within AI-agent contexts has remained largely unexplored. To our knowledge, we're the first to propose incorporating STG into AI-agent dialogue systems.

For example, rather than defining an agent's persona solely with a descriptive paragraph, we propose using a multi-dimensional tabular format to represent their attributes, behaviors, and preferences. Similarly, structured elements within the environment, such as the inventory list of a village storekeeper, can be represented and generated through STG. This enables greater consistency, flexibility, and fidelity in representing the agent state and the interaction context.

As always, the core principle of STG remains the preservation of joint value distributions and structural relationships within the generated data. We argue that, in future AI-agent dialogue systems, STG will play an essential role in grounding agent memory, reasoning, and decision making with structured, realistic, and editable representations.

8.3 Ethics Statement

Synthetic data generation, whether in the form of tabular data, images, or text, presents both opportunities and challenges in the field of artificial intelligence. Among these, Synthetic Tabular Data Generation (STG) is especially powerful, with the potential to improve lives in meaningful ways. This is because a person's personal information, such as financial, medical, educational, family, behavioral, and personal interests, can all be effectively represented in a few simple structured tables.

It is essential that individuals always have the right to decide whether and how their personal data are used by AI models. The primary motivation behind this dissertation is the use of STG as a tool to protect privacy. From the beginning, starting with Chapter 3, this work has emphasized privacy as a fundamental evaluation metric. No matter how realistic the synthetic data may appear, they must always pass strict privacy-preserving tests.

Looking to the future, STG has the potential to become a mature and widely accepted technology, much like synthetic image and text generation today. When that time comes, it is critical that society also advances, establishing clear policies and legal protections to ensure that the privacy and rights of every individual are fully respected.

Bibliography

[1] Quic. https://en.wikipedia.org/wiki/QUIC. Accessed: 2020-11-20.

[2] Nazmiye Ceren Abay, Yan Zhou, Murat Kantarcioglu, Bhavani Thuraisingham, and Latanya Sweeney. Privacy preserving synthetic data release using deep learning. In *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2018, Dublin, Ireland, September 10–14, 2018, Proceedings, Part I 18*, pages 510–526. Springer, 2019.

[3] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

[4] Charu C. Aggarwal and Philip S. Yu. *A General Survey of Privacy-Preserving Data Mining Models and Algorithms*, pages 11–52. Springer US, Boston, MA, 2008.

[5] Mohammad Al-Rubaie and J Morris Chang. Privacy-preserving machine learning: Threats and solutions. *IEEE Security & Privacy*, 17(2):49–58, 2019.

[6] Chenxin An, Jun Zhang, Ming Zhong, Lei Li, Shansan Gong, Yao Luo, Jingjing Xu, and Lingpeng Kong. Why does the effective context length of llms fall short?, 2024. URL https://arxiv.org/abs/2410.18745.

[7] Anonymous. Neuro-inspired information-theoretic hierarchical perception for multimodal learning. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=Z9AZsU1Tju.

[8] Abdul Fatir Ansari, Lorenzo Stella, Caner Turkmen, Xiyuan Zhang, Pedro Mercado, Huibin Shen, Oleksandr Shchur, Syama Syndar Rangapuram, Sebastian Pineda Arango, Shubham Kapoor, Jasper Zschiegner, and Maddix et al. Chronos: Learning the language of time series. *arXiv preprint arXiv:2403.07815*, 2024.

[9] H. Anton, I.C. Bivens, and S. Davis. *Calculus Early Transcendentals, 10th Edition*. Wiley, 2011. ISBN 9781118210130.

[10] Sanjeev Arora, Yuanzhi Li, Yingyu Liang, Tengyu Ma, and Andrej Risteski. A latent variable model approach to PMI-based word embeddings. volume 4, pages 385–399, Cambridge, MA, 2016. MIT Press. doi: 10.1162/tacl_a_00106.

[11] Laura Aviñó, Matteo Ruffini, and Ricard Gavaldà. Generating synthetic but plausible healthcare record datasets. *arXiv preprint arXiv:1807.01514*, 2018.

[12] L. Bariah, Q. Zhao, H. Zou, Y. Tian, F. Bader, and M. Debbah. Large Language Models for Telecom: The Next Big Thing? . *arXiv preprint arXiv:2306.10249*, 2023.

[13] Ronny Kohavi Barry Becker. Adult, Apr 1996. URL https://archive.ics.uci.edu/dataset/2/adult.

[14] Christopher M Bishop. Mixture density networks. 1994.

[15] Vadim Borisov, Kathrin Seßler, Tobias Leemann, Martin Pawelczyk, and Gjergji Kasneci. Language models are realistic tabular data generators. *arXiv preprint arXiv:2210.06280*, 2022.

[16] Vadim Borisov, Kathrin Sessler, Tobias Leemann, Martin Pawelczyk, and Gjergji Kasneci. Language models are realistic tabular data generators. In *The Eleventh International Conference on Learning Representations*, 2023.

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

[18] Anna L Buczak and Erhan Guven. A survey of data mining and machine learning methods for cyber security intrusion detection. *IEEE Communications surveys & tutorials*, 18(2):1153–1176, 2015.

[19] Xingyu Cai, Jiaji Huang, Yuchen Bian, and Kenneth Church. Isotropy in the contextual embedding space: Clusters and manifolds. In *International Conference on Learning Representations*, 2021.

[20] Jin Cao, William S Cleveland, Yuan Gao, Kevin Jeffay, F Donelson Smith, and Michele Weigle. Stochastic models for generating synthetic http source traffic. In *IEEE INFOCOM 2004*, volume 3, pages 1546–1557. IEEE, 2004.

[21] T. Carta, C. Romac, T. Wolf, S. Lamprier, O. Sigaud, and P. Y. Oudeyer. Grounding large language models in interactive environments with online reinforcement learning. *arXiv preprint arXiv:2302.02662*, 2023.

[22] Carlos A Catania and Carlos García Garino. Automatic network intrusion detection: Current techniques and open issues. *Computers & Electrical Engineering*, 38(5):1062–1072, 2012.

[23] Haipeng Chen, Sushil Jajodia, Jing Liu, Noseong Park, Vadim Sokolov, and VS Subrahmanian. Faketables: Using gans to generate functional dependency preserving tables with bounded real data. In *IJCAI*, pages 2074–2080, 2019.

[24] Hongjie Chen, Ryan A Rossi, Kanak Mahadik, Sungchul Kim, and Hoda Eldardiry.

Graph deep factors for probabilistic time-series forecasting. *ACM Transactions on Knowledge Discovery from Data*, 17(2):1–30, 2023.

[25] Song Chen. Beijing pm2.5 data, Jan 2017. URL https://archive.ics.uci.edu/dataset/381/beijing+pm2+5+data.

[26] X Chen, B Li, M Shamsabardeh, R Proietti, Z Zhu, and SJB Yoo. On real-time and self-taught anomaly detection in optical networks using hybrid unsupervised/supervised learning. In *2018 European Conference on Optical Communication (ECOC)*, pages 1–3. IEEE, 2018.

[27] Xinyun Chen, Ryan A Chi, Xuezhi Wang, and Denny Zhou. Premise order matters in reasoning with large language models. *arXiv preprint arXiv:2402.08939*, 2024.

[28] Jaemin Cho, Debanjan Mahata, Ozan Irsoy, Yujie He, and Mohit Bansal. M3docrag: Multi-modal retrieval is what you need for multi-page multi-document understanding. *arXiv preprint arXiv:2411.04952*, 2024.

[29] Edward Choi, Siddharth Biswal, Bradley Malin, Jon Duke, Walter F Stewart, and Jimeng Sun. Generating multi-label discrete patient records using generative adversarial networks. *arXiv preprint arXiv:1703.06490*, 2017.

[30] Milan Cvitkovic. Supervised learning on relational databases with graph neural networks. *arXiv preprint arXiv:2002.02046*, 2020.

[31] Damai Dai, Yutao Sun, Li Dong, Yaru Hao, Shuming Ma, Zhifang Sui, and Furu Wei. Why can gpt learn in-context? language models implicitly perform gradient descent as meta-optimizers. *arXiv preprint arXiv:2212.10559*, 2022.

[32] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings*

of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers), pages 4171–4186, 2019.

[33] Tuan Dinh, Yuchen Zeng, Ruisu Zhang, Ziqian Lin, Michael Gira, Shashank Rajput, Jy-yong Sohn, Dimitris Papailiopoulos, and Kangwook Lee. Lift: Language-interfaced fine-tuning for non-language machine learning tasks. *Advances in Neural Information Processing Systems*, 35:11763–11784, 2022.

[34] Bayu Distiawan, Jianzhong Qi, Rui Zhang, and Wei Wang. Gtr-lstm: A triple encoder for sentence generation from rdf data. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1627–1637, 2018.

[35] Samuel Dooley, Gurnoor Singh Khurana, Chirag Mohapatra, Siddartha Naidu, and Colin White. Forecastpfn: Synthetically-trained zero-shot forecasting, 2023. URL https://arxiv.org/abs/2311.01933.

[36] Lun Du, Fei Gao, Xu Chen, Ran Jia, Junshan Wang, Jiang Zhang, Shi Han, and Dongmei Zhang. Tabularnet: A neural network architecture for understanding semantic structures of tabular data. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pages 322–331, 2021.

[37] Yuntao Du and Ninghui Li. Towards principled assessment of tabular data synthesis algorithms. *arXiv preprint arXiv:2402.06806*, 2024.

[38] Cristóbal Esteban, Stephanie L Hyland, and Gunnar Rätsch. Real-valued (medical) time series generation with recurrent conditional gans. *arXiv preprint arXiv:1706.02633*, 2017.

[39] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, KDD'96, page 226–231. AAAI Press, 1996.

[40] Kawin Ethayarajh. How contextual are contextualized word representations? comparing the geometry of bert, elmo, and gpt-2 embeddings. In *Conference on Empirical Methods in Natural Language Processing*, 2019.

[41] Dominique Evans-Bye. States shapefile. https://hub.arcgis.com/datasets/CMHS::states-shapefile/explore?location=29.721532%2C71.941464%2C3.76, 2015.

[42] Xi Fang, Weijie Xu, Fiona Anting Tan, Jiani Zhang, Ziqing Hu, Yanjun (Jane) Qi, Scott Nickleach, Diego Socolinsky, "SHS" Srinivasan Sengamedu, and Christos Faloutsos. Large language models (llms) on tabular data: Prediction, generation, and understanding — a survey. *Transactions on Machine Learning Research*, 2024.

[43] Kuofeng Gao, Yang Bai, Jindong Gu, Shu-Tao Xia, Philip Torr, Zhifeng Li, and Wei Liu. Inducing high energy-latency of large vision-language models with verbose images. In *ICLR*, 2024.

[44] Hector Garcia-Molina, Jeffrey D. Ullman, and Jennifer Widom. *Database Systems: The Complete Book*. Prentice Hall Press, USA, 2 edition, 2008. ISBN 9780131873254.

[45] Azul Garza and Max Mergenthaler-Canseco. Timegpt-1. *arXiv preprint arXiv:2310.03589*, 2023.

[46] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In

Advances in neural information processing systems, pages 2672–2680, Cambridge, MA, USA, 2014. MIT Press.

[47] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *Communications of the ACM*, 63(11):139–144, 2020.

[48] Cormode Graham. Differentially private spatial decompositions. In *Data engineering (ICDE), 2012 IEEE 28th international conference*, 2012.

[49] Nate Gruver, Marc Finzi, Shikai Qiu, and Andrew Gordon Wilson. Large language models are zero-shot time series forecasters, 2024. URL https://arxiv.org/abs/2310.07820.

[50] Jiuxiang Gu, Jason Kuen, Vlad I. Morariu, Handong Zhao, Nikolaos Barmpalios, Rajiv Jain, Ani Nenkova, and Tong Sun. Unified pretraining framework for document understanding. In *Proceedings of the 35th International Conference on Neural Information Processing Systems*, NIPS '21, Red Hook, NY, USA, 2021. Curran Associates Inc. ISBN 9781713845393.

[51] Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

[52] Haibo He and Edwardo A Garcia. Learning from imbalanced data. *IEEE Transactions on knowledge and data engineering*, 21(9):1263–1284, 2009.

[53] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 16000–16009, 2022.

[54] Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katie Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, Jack W. Rae, Oriol Vinyals, and Laurent Sifre. Training compute-optimal large language models. *CoRR*, abs/2203.15556, 2022. doi: 10.48550/ARXIV.2203.15556. URL https://doi.org/10.48550/arXiv.2203.15556.

[55] Cheng-Ping Hsieh, Simeng Sun, Samuel Kriman, Shantanu Acharya, Dima Rekesh, Fei Jia, and Boris Ginsburg. RULER: What's the real context size of your long-context language models? In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=kIoBbc76Sy.

[56] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. *CoRR*, abs/2106.09685, 2021. URL https://arxiv.org/abs/2106.09685.

[57] Shuodi Hui, Huandong Wang, Tong Li, Xinghao Yang, Xing Wang, Junlan Feng, Lin Zhu, Chao Deng, Pan Hui, Depeng Jin, et al. Large-scale urban cellular traffic generation via knowledge-enhanced gans with multi-periodic patterns. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 4195–4206, 2023.

[58] Steve TK Jan, Qingying Hao, Tianrui Hu, Jiameng Pu, Sonal Oswal, Gang Wang, and Bimal Viswanath. Throwing darts in the dark? detecting bots with limited data using neural data augmentation.

[59] Ming Jin, Shiyu Wang, Lintao Ma, Zhixuan Chu, James Y. Zhang, Xiaoming Shi, Pin-

Yu Chen, Yuxuan Liang, Yuan-Fang Li, Shirui Pan, and Qingsong Wen. Time-LLM: Time series forecasting by reprogramming large language models. 2024.

[60] James Jordon, Jinsung Yoon, and Mihaela van der Schaar. Pate-gan: Generating synthetic data with differential privacy guarantees. 2018.

[61] James Max Kanter and Kalyan Veeramachaneni. Deep feature synthesis: Towards automating data science endeavors. In *2015 IEEE international conference on data science and advanced analytics (DSAA)*, pages 1–10. IEEE, 2015.

[62] Hyunjik Kim, George Papamakarios, and Andriy Mnih. The lipschitz constant of self-attention. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 5562–5571. PMLR, 18–24 Jul 2021.

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

[64] Daphne Koller and Nir Friedman. *Probabilistic graphical models: principles and techniques*. MIT press, 2009.

[65] Teven Le Scao and Alexander M Rush. How many data points is a prompt worth? In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2627–2636, 2021.

[66] D. Lenat and G. Marcus. Getting from Generative AI to Trustworthy AI: What LLMs might Learn from Cyc. *arXiv preprint arXiv:2308.04445*, 2023.

[67] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W. Yih, T. Rocktäschel, et al. Retrieval-augmented generation for

knowledge-intensive NLP tasks. In *Proceedings of Advances in Neural Information Processing Systems*, volume 33, pages 9459–9474, Dec. 2020.

[68] Percy Liang, Rishi Bommasani, Tony Lee, Dimitris Tsipras, Dilara Soylu, Michihiro Yasunaga, Yian Zhang, Deepak Narayanan, Yuhuai Wu, Ananya Kumar, et al. Holistic evaluation of language models. *arXiv preprint arXiv:2211.09110*, 2022.

[69] Zinan Lin, Alankar Jain, Chen Wang, Giulia Fanti, and Vyas Sekar. Using gans for sharing networked time series data: Challenges, initial promise, and open questions. In *Proceedings of the ACM Internet Measurement Conference*, pages 464–483, 2020.

[70] Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, et al. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*, 2024.

[71] Boxun Liu, Xuanyu Liu, Shijian Gao, Xiang Cheng, and Liuqing Yang. Llm4cp: Adapting large language models for channel prediction. *Journal of Communications and Information Networks*, 9(2):113–125, 2024. doi: 10.23919/JCIN.2024.10582829.

[72] Tennison Liu, Zhaozhi Qian, Jeroen Berrevoets, and Mihaela van der Schaar. Goggle: Generative modelling for tabular data by learning relational structure. In *The Eleventh International Conference on Learning Representations*, 2022.

[73] A. Maatouk, F. Ayed, N. Piovesan, A. D. Domenico, M. Debbah, and Z-Q. Luo. TeleQnA: A Benchmark Dataset to Assess Large Language Models Telecommunications Knowledge. *arXiv preprint arXiv:2310.15051*, 2023.

[74] Gabriel Maciá-Fernández, José Camacho, Roberto Magán-Carrión, Pedro García-Teodoro, and Roberto Therón. Ugr '16: A new dataset for the evaluation of cyclostationarity-based network idss. *Computers & Security*, 73:411–424, 2018.

[75] Bruce A Mah. An empirical model of http network traffic. In *Proceedings of INFOCOM'97*, volume 2, pages 592–600. IEEE, 1997.

[76] Panagiotis Mandros, David Kaltenpoth, Mario Boley, and Jilles Vreeken. Discovering functional dependencies from mixed-type data. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1404–1414, 2020.

[77] Andrei Margeloiu, Xiangjian Jiang, Nikola Simidjievski, and Mateja Jamnik. Tabebm: A tabular data augmentation method with distinct class-specific energy-based models. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

[78] Justin Matejka and George Fitzmaurice. Same stats, different graphs: generating datasets with varied appearance and identical statistics through simulated annealing. In *Proceedings of the 2017 CHI conference on human factors in computing systems*, pages 1290–1294, 2017.

[79] Ryan McKenna, Daniel Sheldon, and Gerome Miklau. Graphical-model based estimation and inference for differential privacy. In *International Conference on Machine Learning*, pages 4435–4444. PMLR, 2019.

[80] Leigh Metcalf and William Casey. Chapter 3 - probability models. In Leigh Metcalf and William Casey, editors, *Cybersecurity and Applied Mathematics*, pages 23 – 42. Syngress, Boston, 2016. ISBN 978-0-12-804452-0. doi: https://doi.org/10.1016/B978-0-12-804452-0.00003-8. URL http://www.sciencedirect.com/science/article/pii/B9780128044520000038.

[81] R. E. Miller and P. D. Blair. *Input-output analysis: foundations and extensions*. Cambridge university press, 2009.

[82] Jiaqi Mu and Pramod Viswanath. All-but-the-top: Simple and effective postprocessing for word representations. In *International Conference on Learning Representations*, 2018.

[83] Nikhil Muralidhar, Chen Wang, Nathan Self, Marjan Momtazpour, Kiyoshi Nakayama, Ratnesh Sharma, and Naren Ramakrishnan. illiad: Intelligent invariant and anomaly detection in cyber-physical systems. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 9(3):1–20, 2018.

[84] Yuqi Nie, Nam H Nguyen, Phanwadee Sinthong, and Jayant Kalagnanam. A time series is worth 64 words: Long-term forecasting with transformers. In *The Eleventh International Conference on Learning Representations*, 2023.

[85] Cam Nugent. California housing prices, Nov 2017. URL https://www.kaggle.com/datasets/camnugent/california-housing-prices.

[86] Aaron van den Oord, Nal Kalchbrenner, and Koray Kavukcuoglu. Pixel recurrent neural networks. *arXiv preprint arXiv:1601.06759*, 2016.

[87] OpenAI. Chat-gpt: Optimizing language models for dialogue, Nov 2022. URL https://openai.com/blog/chatgpt/.

[88] Thorsten Papenbrock and Felix Naumann. A hybrid approach to functional dependency discovery. In *Proceedings of the 2016 International Conference on Management of Data*, pages 821–833, 2016.

[89] Thorsten Papenbrock, Jens Ehrlich, Jannik Marten, Tommy Neubert, Jan-Peer Rudolph, Martin Schönberg, Jakob Zwiener, and Felix Naumann. Functional dependency discovery: An experimental evaluation of seven algorithms. *Proceedings of the VLDB Endowment*, 8(10):1082–1093, 2015.

[90] Joon Sung Park, Joseph O'Brien, Carrie Jun Cai, Meredith Ringel Morris, Percy Liang, and Michael S Bernstein. Generative agents: Interactive simulacra of human behavior. In *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology*, pages 1–22, 2023.

[91] Noseong Park, Mahmoud Mohammadi, Kshitij Gorde, Sushil Jajodia, Hongkyu Park, and Youngmin Kim. Data synthesis based on generative adversarial networks. *Proceedings of the VLDB Endowment*, 11(10):1071–1083, 2018.

[92] Neha Patki, Roy Wedge, and Kalyan Veeramachaneni. The synthetic data vault. In *IEEE International Conference on Data Science and Advanced Analytics (DSAA)*, pages 399–410, Oct 2016. doi: 10.1109/DSAA.2016.49.

[93] Vern Paxson. Fast, approximate synthesis of fractional gaussian noise for generating self-similar network traffic. *ACM SIGCOMM Computer Communication Review*, 27 (5):5–18, 1997.

[94] Frédéric Pennerath, Panagiotis Mandros, and Jilles Vreeken. Discovering approximate functional dependencies using smoothed mutual information. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1254–1264, 2020.

[95] Ben Prystawski, Michael Li, and Noah Goodman. Why think step by step? reasoning emerges from the locality of experience. *Advances in Neural Information Processing Systems*, 36, 2024.

[96] J. Qin, X. Liang, Y. Hong, J. Tang, and L. Lin. Neural-symbolic solver for math word problems with auxiliary tasks. *arXiv preprint arXiv:2107.01431*, 2021.

[97] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

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

[99] C. Raffel, N. Shazeer, A. Roberts, K. Lee, Sharan S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21:1–67, 2020.

[100] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21(140): 1–67, 2020.

[101] Kashif Rasul, Arjun Ashok, Andrew Robert Williams, Hena Ghonia, Rishika Bhagwatkar, Arian Khorasani, Mohammad Javad Darvishi Bayazi, George Adamopoulos, Roland Riachi, Nadhir Hassen, Marin Biloš, Sahil Garg, Anderson Schneider, Nicolas Chapados, Alexandre Drouin, Valentina Zantedeschi, Yuriy Nevmyvaka, and Irina Rish. Lag-llama: Towards foundation models for probabilistic time series forecasting, 2024. URL https://arxiv.org/abs/2310.08278.

[102] V. Rawte, A. Sheth, and A. Das. A Survey of Hallucination in Large Foundation Models. *arXiv preprint arXiv:2309.05922*, 2023.

[103] Shukor Razak, Nur Hafizah, and Arafat Al-Dhaqm. Data anonymization using pseudonym system to preserve data privacy. *IEEE Access*, 2020.

[104] Ruifeng Ren and Yong Liu. Towards understanding how transformers learn in-context through a representation learning lens. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

[105] Leonardo FR Ribeiro, Claire Gardent, and Iryna Gurevych. Enhancing amr-to-text generation with dual graph representations. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3183–3194, 2019.

[106] Rudolf H Riedi, Matthew S Crouse, Vinay J Ribeiro, and Richard G Baraniuk. A multifractal wavelet model with application to network traffic. *IEEE transactions on Information Theory*, 45(3):992–1018, 1999.

[107] Markus Ring, Alexander Dallmann, Dieter Landes, and Andreas Hotho. Ip2vec: Learning similarities between ip addresses. In *2017 IEEE International Conference on Data Mining Workshops (ICDMW)*, pages 657–666. IEEE, 2017.

[108] Markus Ring, Daniel Schlör, Dieter Landes, and Andreas Hotho. Flow-based network traffic generation using generative adversarial networks. *Computers & Security*, 82: 156–172, 2019.

[109] Bernardino Romera-Paredes, Mohammadamin Barekatain, Alexander Novikov, Matej Balog, M Pawan Kumar, Emilien Dupont, Francisco JR Ruiz, Jordan S Ellenberg, Pengming Wang, Omar Fawzi, et al. Mathematical discoveries from program search with large language models. *Nature*, pages 1–3, 2023.

[110] Peter J. Rousseeuw. Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics*, 20:53–65, 1987. ISSN 0377-0427.

[111] W. Saad, O. Hashash, C. K. Thomas, C. Chaccour, M. Debbah, N. Mandayam, and Z. Han. Artificial general intelligence (AGI)-native wireless systems: A journey beyond 6G. *arXiv preprint arXiv:2405.02336*, 2024.

[112] samuel Cortinhas. House price prediction - seattle, Dec 2022. URL https://www.kaggle.com/datasets/samuelcortinhas/house-price-prediction-seattle.

[113] Nabeel Seedat, Nicolas Huynh, Boris van Breugel, and Mihaela van der Schaar. Curated LLM: Synergy of LLMs and data curation for tabular augmentation in low-data regimes. In *Forty-first International Conference on Machine Learning*, 2024.

[114] Mandar Sharma, Ajay Kumar Gogineni, and Naren Ramakrishnan. Neural methods for data-to-text generation. *ACM Transactions on Intelligent Systems and Technology*, 2024.

[115] Ananya Singha, José Cambronero, Sumit Gulwani, Vu Le, and Chris Parnin. Tabular representation, noisy operators, and impacts on table structure understanding tasks in llms. *arXiv preprint arXiv:2310.10358*, 2023.

[116] Aivin V Solatorio and Olivier Dupriez. Realtabformer: Generating realistic relational and tabular data using transformers. *arXiv preprint arXiv:2302.02041*, 2023.

[117] Xingyou Song, Oscar Li, Chansoo Lee, Bangding Yang, Daiyi Peng, Sagi Perel, and Yutian Chen. Omnipred: Language models as universal regressors. *Trans. Mach. Learn. Res.*, 2024, 2024. URL https://openreview.net/forum?id=t9c3pfrR1X.

[118] Yi Sun, Alfredo Cuesta-Infante, and Kalyan Veeramachaneni. Learning vine copula models for synthetic data generation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 5049–5057, 2019.

[119] Zineng Tang, Ziyi Yang, Guoxin Wang, Yuwei Fang, Yang Liu, Chenguang Zhu, Michael Zeng, Cha Zhang, and Mohit Bansal. Unifying vision, text, and layout for universal document processing. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 19254–19264, 2023.

[120] S. Tarkoma, R. Morabito, and J. Sauvola. AI-native Interconnect Framework for Integration of Large Language Model Technologies in 6G Systems. *arXiv preprint arXiv:2311.05842*, 2023.

[121] Tejashvi. Tour & travels customer churn prediction, Oct 2021. URL https://www.kaggle.com/datasets/tejashvi14/tour-travels-customer-churn-prediction.

[122] C. K. Thomas and W. Saad. Neuro-Symbolic Causal Reasoning Meets Signaling Game for Emergent Semantic Communications. *IEEE Transactions on Wireless Communications*, 23(5), May. 2024.

[123] C. K. Thomas, C. Chaccour, W. Saad, M. Debbah, and C. S. Hong. Causal Reasoning: Charting a Revolutionary Course for Next-Generation AI-Native Wireless Networks. *IEEE Vehicular Technology Magazine*, 19(1), Mar. 2024.

[124] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

[125] Johannes Treutlein, Dami Choi, Jan Betley, Samuel Marks, Cem Anil, Roger Grosse, and Owain Evans. Connecting the dots: Llms can infer and verbalize latent structure from disparate training data. *arXiv preprint arXiv:2406.14546*, 2024.

[126] D. Tse and P. Viswanath. Fundamentals of Wireless Communication. In *Cambridge University Press*, 2005.

[127] Aaron Van den Oord, Nal Kalchbrenner, Lasse Espeholt, Oriol Vinyals, Alex Graves, et al. Conditional image generation with pixelcnn decoders. In *Advances in neural information processing systems*, pages 4790–4798, 2016.

[128] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *CoRR*, abs/1706.03762, 2017. URL http://arxiv.org/abs/1706.03762.

[129] Apoorv Vyas, Nataraj Jammalamadaka, Xia Zhu, Dipankar Das, Bharat Kaul, and Theodore L Willke. Out-of-distribution detection using an ensemble of self supervised leave-out classifiers. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 550–564, 2018.

[130] Dandan Wang and Shiqing Zhang. Large language models in medical and healthcare fields: applications, advances, and challenges. *Artificial Intelligence Review*, 57(299): 1–27, 2024.

[131] Eason Wang, Henggang Cui, Sai Yalamanchi, Mohana Moorthy, and Nemanja Djuric. Improving movement predictions of traffic actors in bird's-eye view models using gans and differentiable trajectory rasterization. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2340–2348, 2020.

[132] Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models. *arXiv preprint arXiv:2305.16291*, 2023.

[133] Yun Wang, Zhida Sun, Haidong Zhang, Weiwei Cui, Ke Xu, Xiaojuan Ma, and Dongmei Zhang. Datashot: Automatic generation of fact sheets from tabular data. *IEEE Transactions on Visualization and Computer Graphics*, 26(1):895–905, 2020. doi: 10.1109/TVCG.2019.2934398.

[134] Zhiruo Wang, Haoyu Dong, Ran Jia, Jia Li, Zhiyi Fu, Shi Han, and Dongmei Zhang. Tuta: Tree-based transformers for generally structured table pre-training. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pages 1780–1790, 2021.

[135] Colin Wei, Sang Michael Xie, and Tengyu Ma. Why do pretrained language models help in downstream tasks? an analysis of head and prompt tuning. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 16158–16170. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper_files/paper/2021/file/86b3e165b8154656a71ffe8a327ded7d-Paper.pdf.

[136] Gerald Woo, Chenghao Liu, Akshat Kumar, Caiming Xiong, Silvio Savarese, and Doyen Sahoo. Unified training of universal time series forecasting transformers, 2024. URL https://arxiv.org/abs/2402.02592.

[137] Chenwei Wu, Holden Lee, and Rong Ge. Connecting pre-trained language model and downstream task via properties of representation. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

[138] Ming-Kun Xie and Sheng-Jun Huang. Learning class-conditional gans with active sampling. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 998–1006, 2019.

[139] Lei Xu and Kalyan Veeramachaneni. Synthesizing tabular data using generative adversarial networks. *arXiv preprint arXiv:1811.11264*, 2018.

[140] Lei Xu, Maria Skoularidou, Alfredo Cuesta-Infante, and Kalyan Veeramachaneni. Modeling tabular data using conditional gan. In *Advances in Neural Information Processing Systems*, pages 7333–7343, 2019.

[141] Shengzhe Xu, Manish Marwah, Martin Arlitt, and Naren Ramakrishnan. Stan: Synthetic network traffic generation with generative neural models. In *Deployable Machine Learning for Security Defense: Second International Workshop, MLHat 2021, Virtual Event, August 15, 2021, Proceedings 2*, pages 3–29. Springer, 2021.

[142] Shengzhe Xu, Cho-Ting Lee, Mandar Sharma, Raquib Bin Yousuf, Nikhil Muralidhar, and Naren Ramakrishnan. Are llms naturally good at synthetic tabular data generation? *arXiv preprint arXiv:2406.14541*, 2024.

[143] Shengzhe Xu, Christo Kurisummoottil Thomas, Omar Hashash, Nikhil Muralidhar, Walid Saad, and Naren Ramakrishnan. Large multi-modal models (lmms) as universal foundation models for ai-native wireless systems. *arXiv preprint arXiv:2402.01748*, 2024.

[144] Xinli Yu, Zheng Chen, Yuan Ling, Shujing Dong, Zongyi Liu, and Yanbin Lu. Temporal data meets llm–explainable financial time series forecasting. *arXiv preprint arXiv:2306.11025*, 2023.

[145] Abhay Zala, Han Lin, Jaemin Cho, and Mohit Bansal. Diagrammergpt: generating open-domain, open-platform diagrams via llm planning. *arXiv preprint arXiv:2310.12128*, 2023.

[146] Hengrui Zhang, Jiani Zhang, Balasubramaniam Srinivasan, Zhengyuan Shen, Xiao

Qin, Christos Faloutsos, Huzefa Rangwala, and George Karypis. Mixed-type tabular data synthesis with score-based diffusion in latent space. *arXiv preprint arXiv:2310.09656*, 2023.

[147] J. Zhang, J. Jennings, C. Zhang, and C. Ma. Towards Causal Foundation Model: on Duality between Causal Inference and Attention. *arXiv preprint arXiv:2310.00809*, 2023.

[148] Tianping Zhang, Shaowen Wang, Shuicheng Yan, Li Jian, and Qian Liu. Generative table pre-training empowers models for tabular prediction. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 14836–14854, 2023.

[149] Yunjia Zhang, Zhihan Guo, and Theodoros Rekatsinas. A statistical perspective on discovering functional dependencies in noisy data. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*, pages 861–876, 2020.

[150] Zilong Zhao, Aditya Kunar, Robert Birke, and Lydia Y Chen. Ctab-gan: Effective table data synthesizing. In *Asian Conference on Machine Learning*, pages 97–112. PMLR, 2021.

[151] Zilong Zhao, Robert Birke, and Lydia Chen. Tabula: Harnessing language models for tabular data synthesis. *arXiv preprint arXiv:2310.12746*, 2023.

[152] Lei Zheng, Ning Li, Xianyu Chen, Quan Gan, and Weinan Zhang. Dense representation learning and retrieval for tabular data prediction. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 3559–3569, 2023.

[153] Mingkai Zheng, Xiu Su, Shan You, Fei Wang, Chen Qian, Chang Xu, and Samuel Albanie. Can gpt-4 perform neural architecture search? *arXiv preprint arXiv:2304.10970*, 2023.

[154] Alan Zhu, Parth Asawa, Jared Quincy Davis, Lingjiao Chen, Boris Hanin, Ion Stoica, Joseph E. Gonzalez, and Matei Zaharia. Bare: Combining base and instruction-tuned language models for better synthetic data generation. *arXiv preprint arXiv:2502.01697*, 2025.

[155] Yujin Zhu, Zilong Zhao, Robert Birke, and Lydia Y Chen. Permutation-invariant tabular data synthesis. In *2022 IEEE International Conference on Big Data (Big Data)*, pages 5855–5864. IEEE, 2022.

Appendices

Appendix A

Appendix for Problem 2

A.1 Extended Related Work

Owing to the ubiquity of tabular data, the synthetic generation of this type of data in traditional machine learning research. Various approaches have been developed for tabular data generation.

Tabular Data Generation with Neural Networks (i.i.d. rows). Lei et al. [140] proposed CTGAN where rows are independent of each other; a conditional GAN architecture ensures that the dependency between columns is learned. Tabsyn [146] also generates independent rows but with a diffusion approach.

Tabular Data Generation with Neural Networks (non i.i.d. rows). DoppelGanger [69] uses a combination of an RNN and a GAN to incorporate temporal dependencies across rows but this method has been tested in traditional, low-volume settings such as Wikipedia daily visit counts. For high-volume applications, STAN [141] utilizes a combination of a CNN and Gaussian mixture neural networks to generate synthetic network traffic data. GraphDF [24] conducts multi-dimensional time series forecasting. GOGGLE [72] employs a generative modeling method for tabular data by learning relational structures.

Use of Language Models (LLMs) for tabular data generation. Most modern LLMs are based on the transformer architecture [128] with parameters ranging from few millions

to billions [54], and researchers have developed creative ways to harness LLMs in traditional machine learning and data contexts. LIFT [33] initially transforms a table row into a sentence, such as 'An Iris plant with sepal length 5.1cm, sepal width 3.5cm', and employs an LLM as a learning model for table classification, regression, and generation tasks. GReaT [15], introduced earlier, utilizes a GPT-2 model that has been fine-tuned using a specific corpus for synthetic data generation. A general benefit of utilizing LLMs is the elimination of customized preprocessing pipelines.

Feature Ordering. Although not well-studied in the context of tabular data generation, the notion of feature ordering has been investigated in the context of graph-to-text translation [114] wherein to learn effective graph encodings, vertices are linearized via combinations of different graph traversal mechanisms, e.g., topological & breath-first strategies [34], and top-down & bottom-up approaches [105]. As a second example, 'permutation-invariant tabular data synthesis [155] examines the influence of the arrangement of table columns on convolutional neural network (CNN) training and organizes them according to the correlation among columns. Nevertheless, it is important to acknowledge that relying on mere correlation to establish column orders can be limiting. There are also several approaches (e.g., [30, 61]) that synthesize, discover, or aggregate features from relational databases, leveraging order information when possible, for use in machine learning pipelines. It worth to note that even in the LLM community, the task of context sorting for LLM prompting is not trivial and has gained significant attention lately [27].

Mining and Modeling Functional Dependencies. Yunjia et al. [149] relax the notion of strict functional dependencies to include noisy functional relationships by utilizing probabilistic graphical models. Chen et al. [23], in their FakeTables approach use the discovery of functional dependencies in a GAN formulation; they first use a generator to create a set of columns (set A) and an autoencoder to cast another set (set B), which are then used

by a discriminator to calculate the gradient loss. Muralidhar et al. [83] proposed to use Granger causality to incorporate functional *invariants* across multiple time series. The area of FD discovery and of data mining with tabular data have both been extensively studied [76, 94, 152].

A.2 Extended Methodology

A.2.1 Extended Algorithm

Algorithm 3 details FD distillation procedures corresponding to the text in Section 4.2.2 and Algorithm 4 details Feature Order Permutation Optimization procedures corresponding to the text in Section 4.2.3.

Algorithm 3 FD Distillation with Schema-Level FDs

Require: List of Schema-level FDs, $\mathcal{S}$
Ensure: Column Dependency Graph, $\mathcal{G}(\mathcal{V}, \mathcal{E})$

$\mathcal{G} \leftarrow \emptyset$
for $fd \in \mathcal{S}$ **do**
    $LHS, RHS \leftarrow fd$
    if $LHS.length = 1$ **and** $RHS.length = 1$ **then**
        for $u \in LHS$ **do**
            for $v \in RHS$ **do**
                $\mathcal{G}$.add_edge(u, v) ▷ Case 1
    else if $LHS.length > 1$ **and** $RHS.length = 1$ **then**
        for $u \in LHS$ **do**
            $\mathcal{G}$.add_edge(v, u) ▷ Case 2
    else ▷ $RHS.length > 1$, Case 3
        for $v \in RHS$ **do**
            if $LHS.length = 1$ **then**
                $\mathcal{G}$.add_edge(u, v) ▷ Go to Case 1
            else
                for $u \in LHS$ **do**
                    $\mathcal{G}$.add_edge(v, u) ▷ Go to Case 2

Algorithm 4 Feature Order Permutation Optimization

Require: Column Dependency Graphs $\mathcal{G}(\mathcal{V}, \mathcal{E})$
Ensure: Optimal Feature Order Permutation $\mathbf{k}$
1: **if** $|\mathcal{E}| = \emptyset$ **then**
2: Return $\mathbf{k} \leftarrow arbitrary\_permutation(\mathcal{V})$
3: **if** $\mathcal{G}$ is not a Directed Acyclic Graph (DAG) **then** ▷ Phase 1
4: $\mathcal{G} \leftarrow strongly\_connected\_components(\mathcal{G})$
5: $\mathbf{k} \leftarrow topological\_ordering(\mathcal{G})$ ▷ Phase 2
6: $\mathbf{k} \leftarrow arbitrary\_SCC\_expansion(\mathcal{G})$ ▷ Phase 3
7: Return $\mathbf{k}$

A.3 Data and Experiment Description

A.3.1 Experimental Setup

Dataset. We evaluated the efficiency of PAFTthrough experiments on six real datasets commonly used in synthetic table generation studies (GReaT, CTGAN, etc.), as well as a set of four simulated datasets: Beijing [25], US-locations [41], California Housing [85], Adult Income [13], Seattle [112], and Travel [121], Simulated (Algorithm 5). These real-world datasets come from diverse domains and vary in size. The range of the number of functional dependencies spans from 0, indicating *complete independence* between columns, to around 400, indicating a high level of *interdependence*. These data also demonstrate the diverse combinations of categories and numerical columns. The simulated data is customized to adhere to the given functional dependence schema, thus explicitly emphasizing the degree to which a model can precisely represent the functional relationship. The simulated data has four distinct versions, denoted by the variable k, which represents the unique values in the d column. As the value of k increases, the data becomes more complex, which has been demonstrated to make training the generative model more challenging, as evidenced by all the experimental results presented in this chapter. In particular, the functional dependency graph of the simulated data is $[a \to b,\ a \to c, b \to c, b \to d, a \to d, b \to d]$. Table A.1 provides

the FD characteristics of each dataset, while Table A.1 provides an more detailed overview.

Training and Testing. To prevent any data leakage, we partitioned the data sets into 80% training sets and 20% test sets. All models are trained or fine-tuned on the same training data samples. All models undergo cross-validation using 5 generated data sets to validate their results. One advantage of using LLM for creating tabular data is that there is no need for any complex data preparation. This means that the feature names and values are used just as they are supplied in the original data sets.

Dataset	#Rows	#Cat	#Num	**#FD**
Beijing	43,824	1	12	**157**
US-locations	20,400	3	2	**7**
California	20,640	8	0	**362**
Adult	32,561	9	6	**78**
Seattle	2,016	2	6	**10**
Travel	954	4	3	**0**
Simulated,k=[1,5,10,15]	10,000	4	0	**6**

Table A.1: Dataset Descriptors (number of rows, categorical columns, numerical columns, and FDs).

Algorithm 5 Building simulated data: Given a dependency graph G, setting values for a table with n rows and m columns with different statistic complexity based on the initial unique value k for the complexity in the root column of the functional dependency chain.

procedure SETVALUES(n, m, G, k)
    $table[n][m] \leftarrow \{\}$
    $top\_order, node\_layer \leftarrow Algorithm\ 4(G)$
    $unique\_value \leftarrow \{\}$
    for $vertex \in topo\_order$ **do**
        $top\_value[vertex] = 2^{max\_layer - node\_layer[vertex]}$
    for $j \leftarrow 1$ to m **do**
        for $i \leftarrow 1$ to n **do**
            $table[i][j] \leftarrow x_{i\%(top\_value[j]*k)}$

Baselines. In benchmarking suite, we have baselines that consist of current deep learning approaches for synthetic data generation (CTGAN [140], CopulaGAN [140]), and the most

advanced LLM fine-tuning synthetic table generator GReaT [15]. To guarantee an equitable comparison, we employ the Distill-GReaT model for both techniques in all tests, and adjust the hyperparameters as advised by the official GitHub website of GReaT. It ought to mention that GReaT utilizes textual encodings with random feature order permutations. This implies that each sample will undergo a different random order during the training and sampling process. This strategy appears similar to the edge case in Algorithm 4, but in fact, they are distinct. When the FD set is empty, PAFT will suggest a random permutation. Nevertheless, this permutation serves as a comprehensive guide for all samples after an in-depth assessment of FD.

A.3.2 Reproducibility detail

Baselines. Each baseline (CTGAN, CopulaGAN, TabSyn, GReaT) sticks to the recommended hyperparameters and utilizes officially released API tools: Synthetic Data Vault [92] and GReaT [15]. As for the fair comparison of GReaT and PAFT, the LoRA fine-tuning parameters are set as: Lora attention dimension $r = 16$, alpha parameter for Lora scaling $lora\_alpha = 32$, The names of the modules to apply the adapter to $target\_modules = c\_attn$, The dropout probability for Lora layers $lora\_dropout = 0.05$, $bias = none$.

Computational Resources. To ensure fairness in the comparison between the baselines, all baseline models and experiments were executed on a single Tesla P100-PCIE-16GB GPU.

Parameters for MLE and Discriminator Models. We utilize neural network, linear/logistic regression, and random forest models from the Scikit-Learn package for the ML efficiency and discriminator experiments. The exact hyperparameters for each model are detailed in Table A.2. Every result is evaluated through the process of 5-fold cross-validation.

Low-order statistics [146] of column-wise data density is calculated with SDV library.

Table A.2: The parameters we used for MLE and Discriminator models remain the same across all datasets.

	RF		**LR**	**NN**		
	n_est	max_depth	max_iter	max_iter	hidden_layer_sizes	learning_rate
Classification	100	*None*	100	300	(150, 100, 50)	0.001
Regression	100	*None*	100	300	(150, 100, 50)	0.001

A.4 Additional Research Questions and Case Studies

A.4.1 RQ2: Does PAFT generate data respecting the consistency of intrinsic data characteristics?

Table A.3 details the standard deviations from five experimental runs.

Table A.3: Additional Std. details in intrinsic characteristics evaluation.

Dataset	**Intrinsic Fact**	**Fact Violation Rate (↓)**				
		CTGAN	CopulaGAN	TabSyn	GReaT	PAFT
US-locations	State-code → Bird	94.66±0.32%	95.66±0.17%	0.10±0.04%	0.30±0.00%	**0.00±0.00%**
US-locations	Lat-long → State	99.22±0.08%	98.51±0.07%	21.50±0.32%	8.16±0.14%	**2.93±0.15%**
California	Lat-long → CA	47.56±6.37%	99.93±0.00%	8.83±0.13%	5.42±0.16%	**1.26±0.06%**
California	Median house price → $[1.4e^5, 5e^5]$	1.46±1.52%	0.01±0.01%	**0.00±0.00%**	**0.00±0.00%**	**0.00±0.00%**
Adult	education → education-num	83.94± 1.93%	19.09± 0.58%	1.43±0.04%	1.24±0.09%	**0.46± 0.03%**
Seattle	Zipcode → Seattle	**0.00±0.00%**	99.88±0.00%	**0.00±0.00%**	**0.00±0.00%**	**0.00±0.00%**

A.4.2 RQ3: Can data generated by PAFT replace real data in downstream ML model training?

Table A.4 details the standard deviations from five experimental runs.

Table A.4: MLE Performance (%): Comparison of original data to synthetic data.For datasets denoted as (*), we use a regression model for prediction, and calculate MAPE as performance (where lower scores are ideal); For other datasets, classification models are used for prediction and we calculate the accuracy as performance. The best results are marked in **bold** and the second-best results are underlined. RF: random forests; LR: linear regression: NN: a traditional multi-layer perceptron.

Regression Task		**MAPE (↓)** Over Different Methods					
Dataset		**Orig.**	**CTGAN**	**CopulaGAN**	**TabSyn**	**GReaT**	**PAFT**
Beijing (*)	**RF**	0.41%	2.49±0.57%	2.15±0.29%	0.7±0.01%	0.57±0.00%	**0.52±0.00%**
	LR	1.37%	2.23±0.54%	1.55±0.21%	1.25±0.0%	**0.97±0.01%**	1.34±0.01%
	NN	0.99%	2.44±0.74%	2.83±1.18%	1.01±0.14%	1.16±0.56%	**0.95±0.13%**
California (*)	**RF**	0.18%	0.65±0.09%	0.39±0.01%	0.22±0.0%	0.25±0.00%	**0.20±0.00%**
	LR	0.30%	0.54±0.1%	0.5±0.01%	0.30±0.0%	**0.29±0.00%**	0.31±0.00%
	NN	0.34%	0.53±0.11%	0.47±0.02%	0.29±0.02%	0.3±0.01%	**0.27±0.00%**
Seattle (*)	**RF**	0.33%	0.76±0.34%	0.38±0.07%	0.30±0.06%	0.35±0.01%	**0.28±0.03%**
	LR	0.29%	0.74±0.35%	0.32±0.03%	**0.23±0.04%**	0.33±0.00%	0.29±0.03%
	NN	0.28%	0.71±0.33%	0.38±0.08%	0.28±0.01%	0.33±0.00%	**0.27±0.01%**
Classification Task		**Accuracy (↑)** Over Different Methods					
Dataset		**Orig.**	**CTGAN**	**CopulaGAN**	**TabSyn**	**GReaT**	**PAFT**
US-locations	**RF**	99.95%	7.17±1.58%	45.33±2.82%	**99.99±0.01%**	99.84±0.07%	99.91±0.03%
	LR	46.1%	5.11±3.14%	31.08±1.92%	43.69±1.9%	45.65±0.86%	**49.41±1.57%**
	NN	99.85%	7.56±4.61%	53.34±1.59%	**99.64±0.17%**	98.94±1.16%	99.44±0.28%
Adult	**RF**	84.97%	71.15±5.59%	81.33±1.53%	83.69±0.28%	**83.89±0.42%**	83.06±0.35%
	LR	78.53%	75.68±0.14%	78.18± 1.53%	**78.38±0.11%**	76.1±0.29%	77.24±0.09%
	NN	76.9%	75.69±0.10%	76.6±1.26%	78.36±1.23%	78.23±1.31%	**79.16±0.15%**
Travel	**RF**	88.95%	56.35±2.64%	67.18±3.0%	84.09±1.18%	79.78±1.29%	**85.19±1.99%**
	LR	82.87%	70.17±17.46%	79.56±0.00%	**83.31±0.22%**	78.34±2.02%	82.76±1.01%
	NN	81.77%	71.05±17.85%	79.56±0.00%	81.88±1.33%	80.77±1.33%	**83.2±0.90%**

A.4.3 RQ4: Does PAFTadhere to real distribution and possess mode diversity?

Tables A.5 demonstrate that the PAFTsynthetic data closely matches the real data in terms of univariate distribution and bivariate correlation, outperforming the baseline. PAFThas the ability to generate a wide range of diversity, encompassing both continuous and discrete variables, which closely resembles real data.

Dataset	CTGAN	CopulaGAN	TabSyn	GReaT	PAFT
Adult	0.81±0.01	0.92±0.01	**0.98±0.00**	0.88±0.00	0.90±0.00
Beijing	0.89±0.01	0.79±0.01	**0.98±0.00**	0.93±0.00	0.97±0.00
California	0.87±0.03	0.77±0.01	**0.98±0.00**	0.89±0.00	0.83±0.00
US-locations	0.83±0.02	0.82±0.00	0.96±0.00	0.93±0.00	**0.97±0.00**
Seattle	0.83±0.01	0.73±0.02	0.93±0.00	0.90±0.00	**0.93±0.00**
Travel	0.84±0.01	0.90±0.02	0.93±0.01	**0.93±0.01**	**0.93±0.01**

Table A.5: Error rate (%) of column-wise density estimation[1]. Bold Face represents the best score on each dataset. Higher values indicate more accurate estimation (superior results). PAFToutperforms the best generative baseline model in most case. The best results are marked in **bold**, the second-best results are underlined.

A.4.4 RQ5: Does the synthetic data generated by PAFT pass the *privacy* test?

Similar to the analysis conducted in recent work [15] (GReaT), we employ the random forest (RF) algorithm to train discriminators to distinguish real data (labelled as True) and synthetically generated data (labeled as False). Subsequently, we test performance on an unseen set (consisting of 50% synthetically generated data and 50% real data). In this experiment, scores represent the percentage of correctly classified entities. In this case, an ideal accuracy score would be close to 50%, which means the discrimniator fails to distinguish between real and synthesized data. Scores are shown in Table A.7 and indicate that the data generated by PAFTis most indistinguishable from real data, even by powerful discriminative models.

Dataset	CTGAN	CopulaGAN	TabSyn	GReaT	PAFT
Adult	0.81±0.02	0.86±0.01	**0.93±0.00**	0.80±0.01	0.78±0.00
Beijing	0.92±0.01	0.94±0.01	**0.99±0.00**	0.95±0.00	0.98±0.01
California	0.84±0.00	0.87±0.01	**0.97±0.00**	0.87±0.01	0.91±0.02
US-locations	0.50±0.02	0.55±0.00	0.93±0.00	0.89±0.00	**0.94±0.00**
Seattle	0.74±0.02	0.72±0.01	0.80±0.01	0.76±0.03	**0.81±0.01**
Travel	0.77±0.02	0.80±0.02	**0.87±0.01**	0.85±0.01	0.82±0.05

Table A.6: Error rate (%) of pair-wise column correlation score[1]. Bold Face represents the best score on each dataset. PAFToutperforms the best baseline model in most case. The best results are marked in **bold**, the second-best results are underlined.

A.4.5 Case Study: Influence of Statistical and Semantic Factors

Statistical Factor The occurrence of functional dependency in a table is influenced by various factors, including the number of rows and columns, the distinct values in the columns, the relationship between columns, and the presence of duplicate rows, etc. Table 4.1 and A.1 shows different dataset may have different level of statistic difficulties.

Semantic Factors The acquisition of semantic factors is typically not achievable from direct observation of the data's appearance. Typically, this implies that the data's worth will be influenced by real-world expertise in a specific field. For instance, map coordinates are influenced by the geopolitical borders of actual countries and states. Similarly, even if only a subset of data points from a mathematical function are observed, there is a need to comprehend the complete representation of that particular mathematical function.

Fig. A.2 shows the difficulty of capturing the functional dependency can also leaded by the semantic conext of a sub-class in a mixture dataset, such as the state shape and geo location distribution. For this case, previous Table 4.2 have already shown the improvement of utilizing PAFT.

Table A.7: Discriminator Performance (%): Comparison of synthesized data from CTGAN, CopulaGAN, TabSyn, GReaT, and PAFT. The scores stand for the accuracy for detecting real or fake data, where the ML models are trained using 50% real data and 50% random data. An ideal accuracy score is 50, indicating the model cannot distinguish between real and synthesized data. The best results are marked in **bold**, the second-best results are underlined.

Data Privacy - Sniff Test: ML Discriminator Accuracy (Ideal → 50%)					
Method	**CTGAN**	**CopulaGAN**	**TabSyn**	**GReaT**	**PAFT**
Beijing	99.16± 0.08%	98.69± 0.39%	50.97±0.06%	51.1± 0.08%	**50.09± 0.05%**
US-locations	99.94± 0.03%	97.74± 0.22%	51.97±0.18%	50.47± 0.07%	**50.01± 0.01%**
California	98.35± 0.2%	86.64± 0.67%	50.64±0.15%	53.74± 0.27%	**49.89± 0.03%**
Adult	94.43± 0.53%	59.82± 0.9%	51.64±0.14%	**51.12± 0.26%**	48.75± 0.03%
Seattle	87.61± 1.06%	85.7± 2.0%	**50.12±0.9%**	68.27± 1.34%	47.21± 0.48%
Travel	77.96± 1.4%	74.14± 1.64%	**50.66±1.97%**	62.49± 1.2%	48.18± 0.81%

Figure A.1: **Semantic Complexity**: using the random order permutation to modeling the mixture of states data is more challenging when the rectangularity index (left) and compactness index (right) increase.

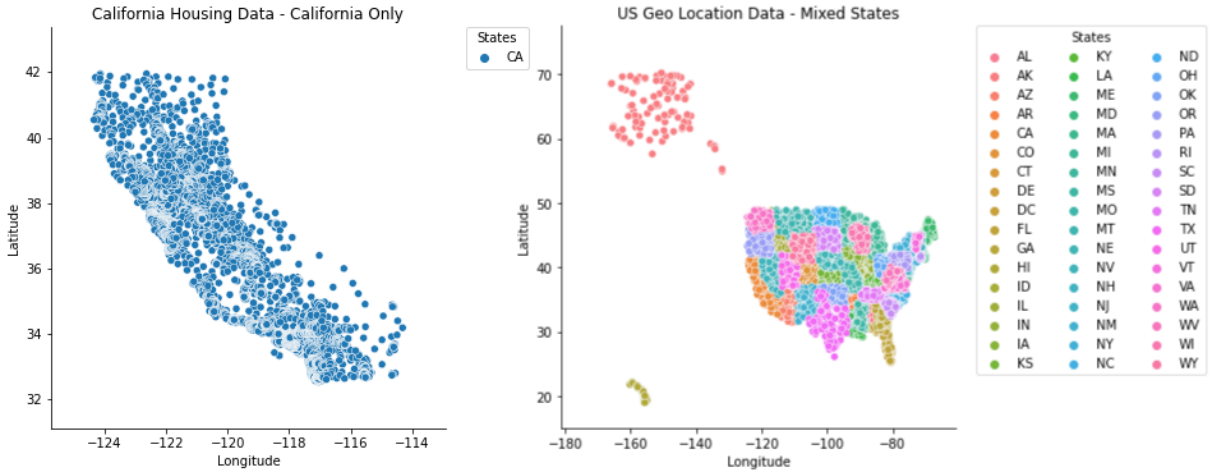


Figure A.2: **Statistical Complexity** can be figured out by analyzing the distribution of the data. For instance, California Housing data is simpler to model concerning the Functional Dependency as it only includes the longitude and latitude for a single state.

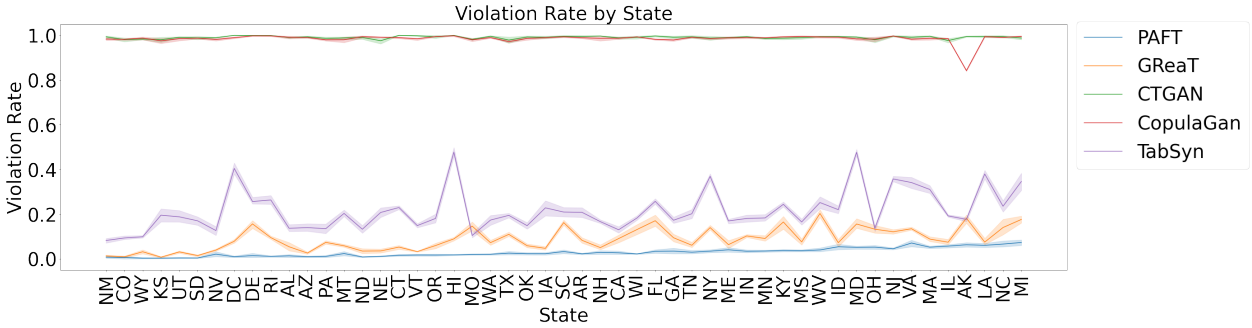


Figure A.3: For a composite dataset, comparison of state-specific violation rates for different synthetic data generation approaches. Here, the states (x-axis) are sorted based on increasing violations. PAFT significantly mitigates state-specific violations in a composite dataset.

Appendix B

Appendix for Problem 4

B.1 Proof of Theorem 6.1

Theorem B.1. *Let the logits of the ground-truth model be bounded. Then for any $f^*(k,l)$, there exists a set of functions $\{\hat{z}_i(k,l)\}_{i=1}^{|\mathcal{V}|}$ such that for all k and T_{l+1}, the predictive distribution of the student model $\hat{p}\,(k_{T_{l+1}} \mid \mathbf{k}_{1:T_l})$ matches that of ground-truth model $p^*(k_{T_{l+1}} \mid \mathbf{k}_{1:T_l})$ and $\hat{f}(k,l) = 0$. In other words, there exists a student model with the same pre-training loss as the ground-truth model, but its logits are ineffective for the numeric downstream tasks.*

Proof. We select $\tau \in \mathbb{R}$ such that $\forall k, T_{l+1}$, $\tau < \min_{j\in\mathcal{V}} b_j^* - \max_{j\in\mathcal{V}} z_i^*(k,l)$, and $\forall k, T_{l+1}, \forall j \in \mathcal{V}$. By setting $\hat{z}_j(k,l) := z_i^*(k,l) + \tau$, we get $\forall j \in \mathcal{V}$,

$$\hat{z}_j(k,l) - b_j^* < z_i^*(k,l) + \min_{j\in\mathcal{V}} b_j^* - \max_{j\in\mathcal{V}} z_i^*(k,T_{l+1}) - b_j^* \leq 0,$$

this implies that $\sigma(\hat{z}_j(k,l) - b_j^*) = 0$. Hence, $\forall k, T_{l+1}$ and we have $\hat{f}(k,l) = 0$. □

B.2 Proof of Lemma 6.2

Lemma B.2. *Consider the Jacobian matrix $\mathbf{J} = \left[\frac{\partial g_i(\,)}{\partial \psi_j}\right]_{i,j=1}^{|\mathcal{V}|}$, which represents the gradient of the self-attention mapping $G(\,)$ with respect to the input time series token embeddings. Then*

the spectral norm of $\mathbf{J}$ *satisfies* $\|\mathbf{J}\|_2 \leq |\,|_2 \sum_{i=1}^{|\mathcal{V}|} \left(p_{i,i} + \frac{1}{2}\right) \left|\psi_i - \sum_{j=1}^{|\mathcal{V}|} p_{i,j}\psi_j\right|^2 + \Delta$, *where the residual term* Δ *is given by* $\Delta = |\,|_2 \sum_{i\neq j}^{|\mathcal{V}|} p_{i,j} \left|\psi_j - \sum_{q=1}^{|\mathcal{V}|} p_{i,q}\psi_q\right|^2 + \frac{|\,|_2}{2} \sum_{j=1}^{|\mathcal{V}|} |\psi_j|^2$, *and the attention weights* $p_{i,j}$ *are defined as* $p_{i,j} = \frac{\exp(\psi_i^\top \psi_j)}{\sum_{k=1}^{|\mathcal{V}|} \exp(\psi_i^\top \psi_k)}$.

Proof. According to the analysis, the gradient of $g_i(\mathbf{\Psi})$ with respect to the variable $\boldsymbol{\psi}_j$ is expressed as $J_{i,j} = \frac{\partial g_i(\,)}{\partial \psi_j} = p_{i,j} I + {}^\top Q^i \left(\ \delta_{i,j} + E_{j,i}\ {}^\top\right)$ where the matrix Q^i is defined by $Q^i = \mathrm{diag}(p_{i,:}) - p_{i,:} p_{i,:}^\top$. Here, $p_{i,:} \in \mathbb{R}_+^{|\mathcal{V}|}$ corresponds to the i-th row of the probability matrix $\boldsymbol{P}$, $E_{j,i} \in \mathbb{R}^{|\mathcal{V}|\times|\mathcal{V}|}$ denotes a matrix with a single entry at the (j,i)-th position and zeros elsewhere, and $\delta_{i,j} \in \{0,1\}$ is the Kronecker delta. We thus have

$$
\begin{aligned}
\|\mathbf{J}\|_2 &\leq \sum_{i,j=1}^{|\mathcal{V}|} |J_{i,j}|_2 \\
&\leq \sum_{i,j=1}^{|\mathcal{V}|} p_{i,j} + \sum_{i=1}^{|\mathcal{V}|} |\ {}^\top Q^i\ |_2|\ |_2 + \sum_{i,j=1}^{|\mathcal{V}|} |\ {}^\top Q^i E_{j,i}\ |_2|\ |_2 \\
&\leq |\mathcal{V}| + |\,|_2 \sum_{i=1}^{|\mathcal{V}|} \left(\sum_{j=1}^{|\mathcal{V}|} p_{i,j} |\psi_j|^2 - \left| \sum_{j=1}^{|\mathcal{V}|} p_{i,j}\psi_j \right|^2 \right) + |\,|_2 \sum_{i,j=1}^{|\mathcal{V}|} |\ {}^\top Q^i e_j \psi_i^\top| \\
&\leq |\mathcal{V}| + |\,|_2 \sum_{i=1}^{|\mathcal{V}|} \sum_{j=1}^{|\mathcal{V}|} p_{i,j} |\psi_j - \sum_{q=1}^{|\mathcal{V}|} p_{i,q}\psi_q|^2 + |\,|_2 \sum_{i,j=1}^{|\mathcal{V}|} p_{i,j} |\psi_i^\top (\psi_j - {}^\top p_{i,:})| \\
&\leq |\,|_2 \sum_{i=1}^{|\mathcal{V}|} \left(p_{i,i} + \frac{1}{2}\right) |\psi_i - {}^\top p_{i,:}|^2 + |\mathcal{V}| + |\,|_2 \sum_{i\neq j}^{|\mathcal{V}|} p_{i,j} |\psi_j - {}^\top p_{i,:}|^2 + \frac{|\,|_2}{2} \sum_{j=1}^{|\mathcal{V}|} |\psi_i|^2 \\
&= |\,|_2 \sum_{i=1}^{|\mathcal{V}|} \left(p_{i,i} + \frac{1}{2}\right) |\psi_i - {}^\top p_{i,:}|^2 + |\mathcal{V}| + |\,|_2 \sum_{i\neq j}^{|\mathcal{V}|} p_{i,j} \left|\psi_j - \sum_{q=1}^{|\mathcal{V}|} p_{i,q}\psi_q\right|^2 + \frac{|\,|_2}{2} \sum_{j=1}^{|\mathcal{V}|} |\psi_i|^2 \\
&= |\,|_2 \sum_{i=1}^{|\mathcal{V}|} \left(p_{i,i} + \frac{1}{2}\right) |x_i - X^\top p_{i,:}|^2 + |\mathcal{V}| + \Delta,
\end{aligned}
$$

where $\Delta = |\,|_2 \sum_{i\neq j}^{|\mathcal{V}|} p_{i,j} \left|\psi_j - \sum_{q=1}^{|\mathcal{V}|} p_{i,q}\psi_q\right|^2 + \frac{|\,|_2}{2} \sum_{j=1}^{|\mathcal{V}|} |\psi_i|^2$. □

The theorem below shows that minimizing the objective $\sum_{i=1}^{|\mathcal{V}|} |\psi_i - {}^\top\ \psi_i|^2$ contains the

largest m eigenvectors of the correlation matrix $^\top$ of input time series token embeddings where m is the rank of .

Lemma 1 implies that one of the key components in the Jacobian's upper bound takes the form $|\psi_i - \sum_{j=1}^{|\mathcal{V}|} p_{i,j}\psi_j|^2$. Consequently, during optimization, it is natural to aim for a reduction in the gradient magnitude, which motivates minimizing the expression $\sum_{i=1}^{|\mathcal{V}|} |\psi_i - \sum_{j=1}^{|\mathcal{V}|} p_{i,j}\psi_j|^2$. This leads to understand the choice of $\mathbf{W}^Q$ and $\mathbf{W}^K$ that minimize $\sum_{i=1}^{|\mathcal{V}|} |\psi_i - \sum_{j=1}^{|\mathcal{V}|} p_{i,j}\psi_j|^2$, which is equivalent to solving the optimization problem $\min_{|\ |_F \leq \rho} \sum_{i=1}^{|\mathcal{V}|} |\psi_i - \sum_{j=1}^{|\mathcal{V}|} p_{i,j}\psi_j|^2$, where the scalar constraint ρ regulates the size of .

To proceed, we consider the objective in the scenario where ρ is small. In this case, we can approximate the attention weights by $p_{i,j} \approx \frac{1}{|\mathcal{V}|} + \frac{1}{|\mathcal{V}|}\psi_i^\top \psi_j$. Now, we define the average of embedding as $\bar{\psi} = {}^\top 1/|\mathcal{V}|$. It then follows that $\sum_{i=1}^{|\mathcal{V}|} |\psi_i - {}^\top p_{i,:}|^2 = \sum_{i=1}^{|\mathcal{V}|} |\psi_i - \bar{\psi} - {}^\top \psi_i|^2$. Assuming all input time series patterns are zero-centered, i.e., $\bar{\psi} = 0$, we have $\sum_{i=1}^{|\mathcal{V}|} |\psi_i - {}^\top \psi_i|^2 = \mathrm{tr}\left((I - {}^\top)^2 {}^\top\right)$. Theorem 6.3 establishes that the optimal that minimizes $\sum_{i=1}^{|\mathcal{V}|} |\psi_i - {}^\top \psi_i|^2$ is spanned by the top m eigenvectors of $^\top$, where m equals the rank of .

B.3 Proof of Theorem 6.3

Theorem B.3. *Let the eigenvalues of the correlation matrix* $^\top$ *be ordered as* $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_D$*, and let* $\gamma_i \in \mathbb{R}^D$ *for* $i = 1, \ldots, D$ *denote their associated eigenvectors. Then, the matrix* * *that minimizes the quantity* $\sum_{i=1}^{|\mathcal{V}|} \left|\psi_i - {}^\top \psi_i\right|^2$ *has the optimal form* $= \sum_{i=1}^{m} \frac{1}{\lambda_i}\gamma_i\gamma_i^\top$.

Proof. Given that $\mathbf{W}_Q \in \mathbb{R}^{D\times m}$ and $\mathbf{W}_K \in \mathbb{R}^{D\times m}$, it follows that the matrix has rank m. Hence, we know $\min \sum_{i=1}^{|\mathcal{V}|} \|\psi_i - {}^\top \psi_i\|^2 \geq \sum_{q=m+1}^{|\mathcal{V}|} \lambda_q$. Now, if we set to $= \sum_{i=1}^{m} \frac{1}{\lambda_i}\gamma_i\gamma_i^\top$, then we obtain $\sum_{i=1}^{|\mathcal{V}|} \|\psi_i - {}^\top \psi_i\|^2 = \mathrm{tr}\left(\left(I - \sum_{i=1}^{m} \gamma_i\gamma_i^\top\right)^2 {}^\top\right) = \sum_{q=m+1}^{D} \lambda_q$.

Therefore, the optimal solution for minimizing $\sum_{i=1}^{|\mathcal{V}|} \|\psi_i - {}^\top \psi_i\|^2$ is essentially character-

ized as a linear combination of the top m eigenvectors of $^\top$. Since a small gradient will prefer a small quantity of $\sum_{i=1}^{|\mathcal{V}|} \|\psi_i - {}^\top \psi_i\|^2$, the self-attention mechanism implicitly drives the weight matrices $\mathbf{W}_Q$ and $\mathbf{W}_K$ to align with the dominant eigen-directions of $^\top$. □

B.4 Clustering in the Contextual Embedding Space

Clustering. We begin with the isotropy assesmment by performing clustering on the LLM representations in the contextual embedding space. There are various methods for performing cultering, such as k-means, DBSCAN [39]. We select K-means clustering method because it is reasonably fast in high embedding dimensions (e.g., $d \geq 768$ for GPT2, ELMo, BERT etc.). We use the celebrated silhouette score analysis [110] to determine the number of clusters $|C|$ in the contextual embedding space. After performing K-means clustering, each observation p (i.e., one of the $\mathbf{J}$ vector representations in $\mathcal{V}$) is assigned to one of C clusters. For an observation p assigned to the cluster $c \in C$, we compute the silhouette score as follows

$$a(p) = \frac{1}{|\mathcal{C}| - 1} \sum_{q \in \mathcal{C}, p \neq q} \text{dist}(p, q); \quad b(p) = \min_{\tilde{c} \neq c} \sum_{q \in \tilde{c}} \text{dist}(p, q); \quad s(p) = \frac{b(p) - a(p)}{\max\{b(p), a(p)\}},$$

where $a(p)$ is the mean distance between an observation p and the rest in the same cluster class p, while $b(p)$ measures the smallest mean distance from p-th observation to all observations in the other cluster class. After computing the silhouette scores $s(p)$ of all observations, a global score is computed by averaging the individual silhouette values, and the partition (with a specific number of clusters) of the largest average score is pronounced superior to other partitions with a different number of clusters. We select the best $|C|$ that belongs to the partition that scores highest among the other partitions.

B.5 Dataset Description

Baselines Architecture, Tokenization Techniques and Hyperparameters.

Table B.1: LLM models architectures, time series tokenization techniques and hyperparameter choices. L stands for context length, d_h for hidden layer dimension, n_L for number of layers, n_H for number of heads, and η for learning rate.

Model	Architecture	Tokenization Technique	Hyperparameters
Chronos-T5	Encoder-Decoder with autoregressive forecasting	Scaling & Quantization	Default
Chronos-Bolt	Encoder-Decoder with multi-step forecasting	Scaling & Quantization	Default
PatchTST	Vanilla Encoder	Patching	Patch length: 16, Stride: 8, $d_h = 32, n_L = 2, n_H = 4$
Moirai	Encoder	Patching	$L = 1024$, Patch length: selected by dataset-specific validation
Lag-Llama	Decoder	Lag Feature	$L = 32$

Real Datsets.

Table B.2: Real and Synthetic Datasets

Data Subset	Domain	Dataset 1	Dataset 2
Real Datasets	Energy	Australian Electricity – Queensland State	Australian Electricity – South Australia
	Weather	Solar Radiation	Rainfall
	Finance	Exchange Rate	NN5 Weekly Cash Withdrawals
	Healthcare	Hospital Patient Counts	COVID-19 Deaths
	Transportation	Transportation Signaling 1	Transportation Signaling 2
	Retail	Car Sales	Dominick
Synthetic Datasets	Linear	DotProduct kernel (C=0)	DotProduct kernel (C=1)
	Seasonality	Seasonality kernel (period = 0.5W)	Seasonality kernel (period = 0.25H)
	Trend	RationalQuadratic kernel ($\alpha = 1$)	RationalQuadratic kernel ($\alpha = 10$)
	Non-Linear	RBF kernel (length scale = 0.1)	RBF kernel (length scale = 1)
	Stochastic	WhiteKernel (noise level = 0.1)	WhiteKernel (noise level = 1)

Synthetic Datasets. We use KernelSynth [8], a method to generate synthetic dataset using Gaussian processes (GPs). KernelSynth allows generation of large, diverse datasets tailored to specific patterns or statistical properties, which is particularly useful when real-world data is scarce or incomplete. In this synthetic data generation process, the GPs are

Table B.3: The complete list of datasets used for our quantitative and qualitative analysis. The table is divided into three sections, representing how the datasets were used for baseline models.

Dataset	Domain	Freq.	Num. Series	Series Length			Prediction
				min	avg	max	Length (H)
Australian Electricity	Energy	30min	5	230736	231052	232272	48
Car Parts	Retail	1M	2674	51	51	51	12
Covid Deaths	Healthcare	1D	266	212	212	212	30
Dominick	Retail	1D	100014	201	296	399	8
Exchange Rate	Finance	1B	8	7588	7588	7588	30
FRED-MD	Economics	1M	107	728	728	728	12
Hospital	Healthcare	1M	767	84	84	84	12
NN5 (Weekly)	Finance	1W	111	113	113	113	8
Weather	Nature	1D	3010	1332	14296	65981	30
Transportaion Signal	Transport	1D	3010	1332	14296	65981	30
Synthetic (10 kernels)	Numerical	-	1000000	1024	1024	1024	64

defined by a mean function, $\mu(t)$, and a positive definite kernel, $\kappa(x_i, x_j)$, which specifies a covariance function for variability across input pairs (x_i, x_j). A kernel bank $\mathcal{K}$ (which consists of linear, RBF, and periodic kernels) is used to define diverse time series patterns. The final kernel $\tilde{\kappa}(x_i, x_j)$ is constructed by sampling and combining kernels from $\mathcal{K}$ using binary operations like $+$ and $\times$. Synthetic time series are generated by sampling from the GP prior, $GP(\mu(t) = 0, \tilde{\kappa}(x_i, x_j))$. The following algorithm presents the pseudocode for KernelSynth which essentially follows the approach in [8].

B.6 Quantitative analysis results

In Figure B.1, we analyze the time series forecasting performance for 12 real datasets with different baselines and its relation with isotropy.

In Figure B.2, we compare the NMSE vs isotropy for varying input context lengths to observe its impact on the real datasets.

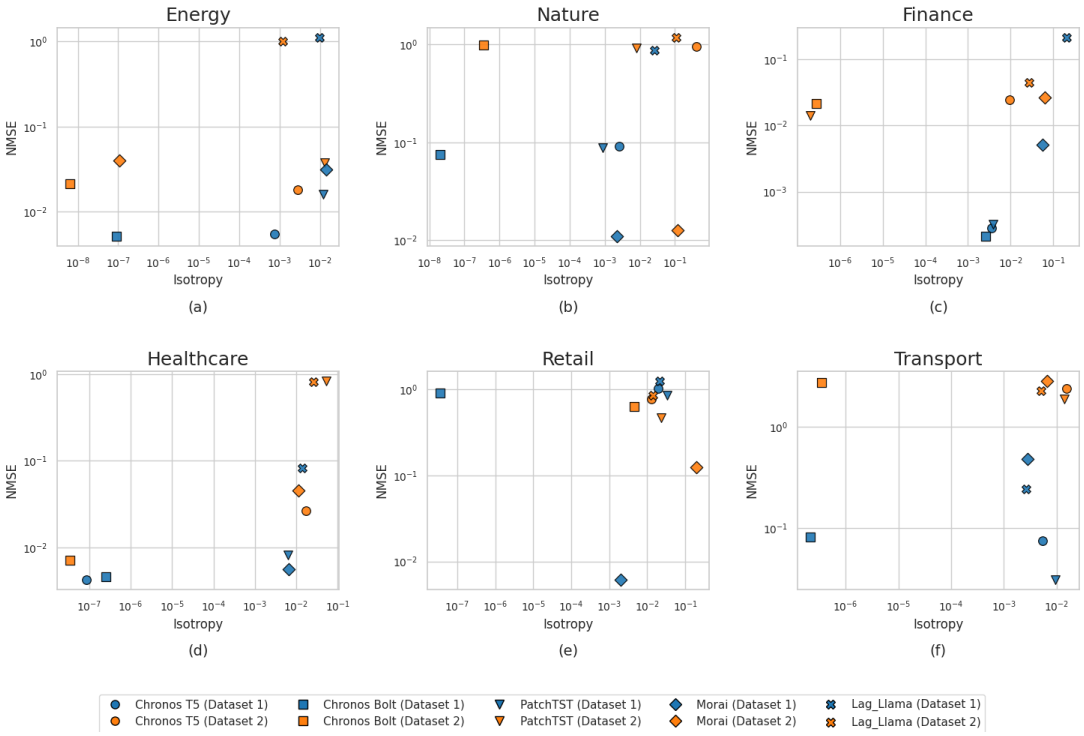


Figure B.1: NMSE vs isotropy analysis for 12 different real datasets of 6 different domains.

B.7 Full Visualization of PCA plots for different models

The Variations in Chornos-T5's hidden representations for different input context lengths is depicted in Figure B.3

B.7.1 Synthetic datasets

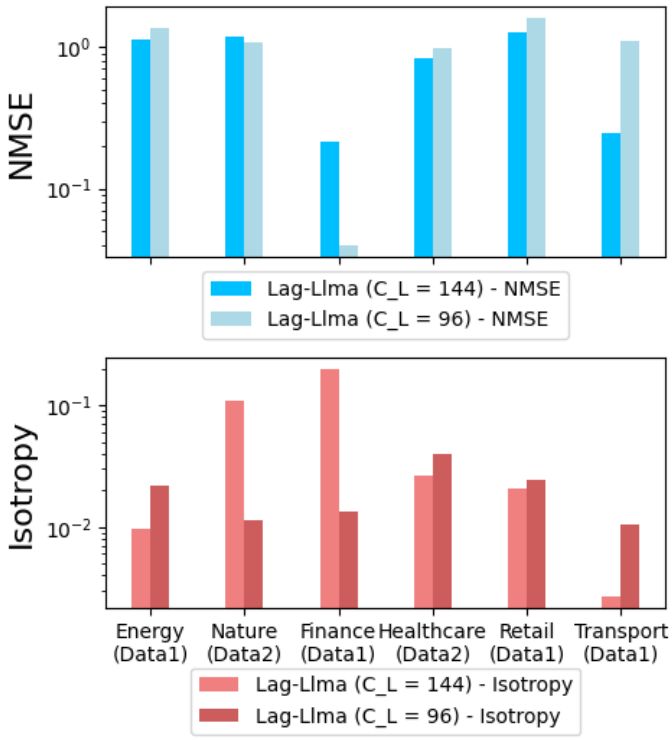


Figure B.2: NMSE vs isotropy comparison across different input context lengths for real datasets.

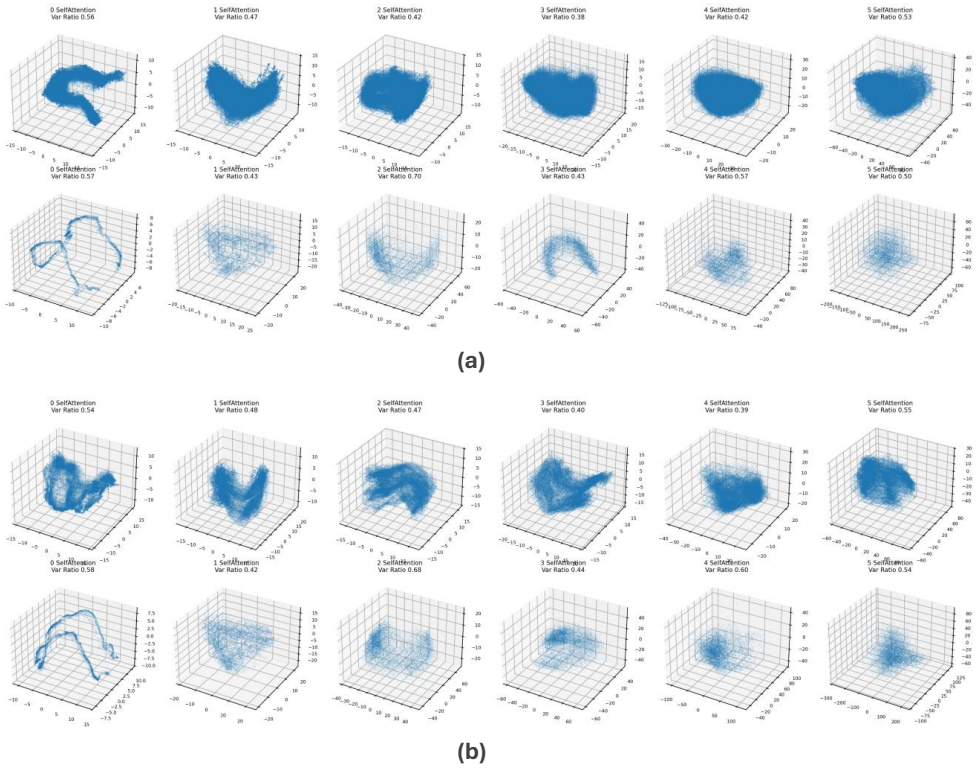
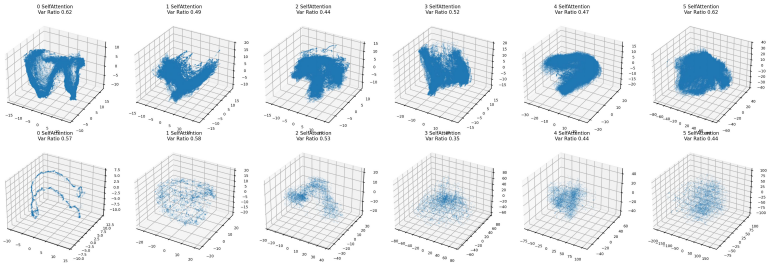


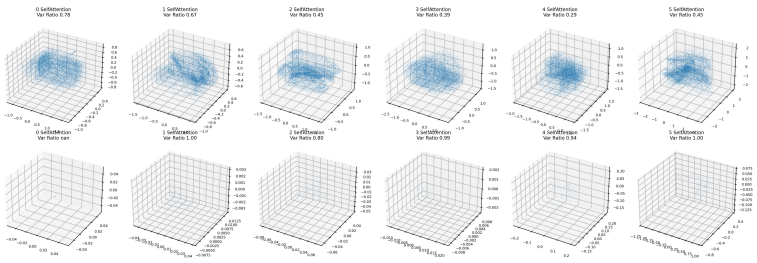
Figure B.3: Variations in Chronos-T5's hidden representations for different input context lengths for the same synthetic dataset "non-linear (Dataset 1)" : (a) Contextual embedding space for input context length $L = 500$. (b) Contextual embedding space for input context length $L = 100$.

Non-Linear (Dataset 1):

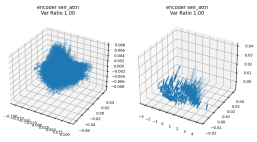
Chronos-T5



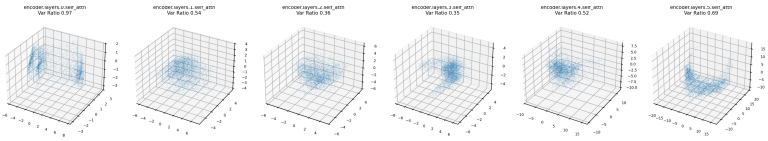
Chronos-Bolt



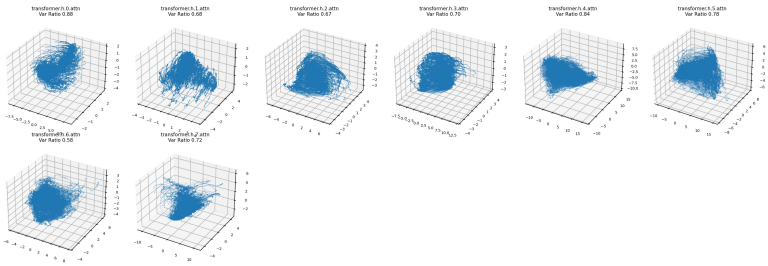
PatchTST



Morai

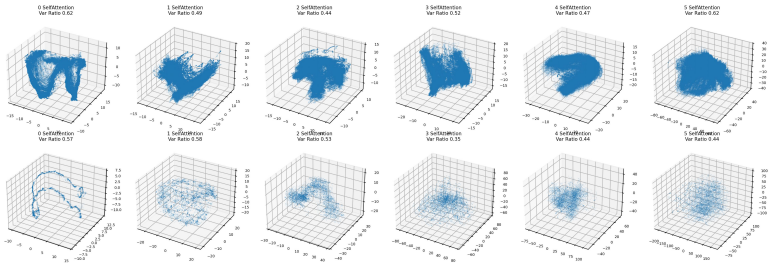


Lag-Llma

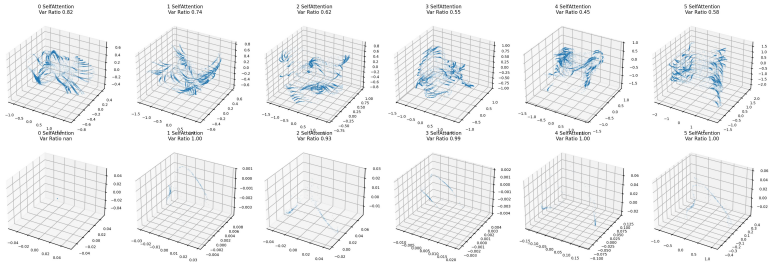


Non-Linear (Dataset 1):

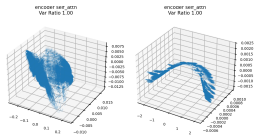
Chronos-T5



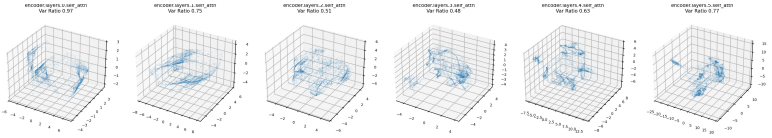
Chronos-Bolt



PatchTST



Morai



Lag-Llma