On LLM-Enhanced Mixed-Type Data Imputation with High-Order Message Passing

Jianwei Wang
Schools of Computer Science and Engineering, University of New South Wales
jianwei.wang1@unsw.edu.au

Kai Wang*
Antai College of Economics & Management, Shanghai Jiao Tong University
w.kai@sjtu.edu.cn

Ying Zhang*
Common Prosperity Visualization and Policy Simulation Lab, Zhejiang Gongshang University
ying.zhang@zjgsu.edu.cn

Wenjie Zhang
Schools of Computer Science and Engineering, University of New South Wales
wenjie.zhang@unsw.edu.au

Xiwei Xu
Data61, CSIRO
xiwei.xu@data61.csiro.au

Xuemin Lin
Antai College of Economics & Management, Shanghai Jiao Tong University
xuemin.lin@sjtu.edu.cn

ABSTRACT

Missing data imputation, which aims to impute the missing values in the raw datasets, is crucial for modern data-driven models like large language models (LLMs). Despite its importance, existing solutions either 1) only support numerical and categorical data or 2) show an unsatisfactory performance due to their design prioritizing text data and overlooking intrinsic characteristics of tabular data.

In this paper, we propose UnIMP, a **Un**ified **IMP**utation framework that leverages LLM and high-order message passing to enhance the imputation of mixed-type data, including numerical, categorical, and text data. Specifically, we first introduce a cell-oriented hypergraph to model the table. We then propose BiHMP, an efficient Bidirectional High-order Message-Passing network to aggregate global-local and high-order information while capturing the inter-column heterogeneity and intra-column homogeneity. To align the capacity of the LLM with the information aggregated by BiHMP, we introduce Xfusion, which, together with BiHMP, acts as adapters for the LLM. We follow a pre-training and fine-tuning pipeline to train UnIMP, integrating two optimizations: chunking technique, which divides tables into smaller chunks to enhance efficiency; and progressive masking technique, which gradually adapts the model to learn more complex data patterns. Both theoretical proofs and empirical experiments on 10 real-world datasets highlight the superiority of UnIMP over existing techniques.

PVLDB Reference Format:
Jianwei Wang, Kai Wang, Ying Zhang, Wenjie Zhang, Xiwei Xu, and Xuemin Lin. On LLM-Enhanced Mixed-Type Data Imputation with High-Order Message Passing. PVLDB, 18(10): 3421 - 3434, 2025.
doi:10.14778/3748191.3748205

PVLDB Artifact Availability:
The source code, data, and/or other artifacts have been made available at https://github.com/guaiyoui/UnIMP.

*Kai Wang and Ying Zhang are the joint corresponding authors.

Proceedings of the VLDB Endowment, Vol. 18, No. 10 ISSN 2150-8097.
doi:10.14778/3748191.3748205

Table 1: Comparison of representative imputation methods. "—" denotes not applicable. Abbr: GAN (generative adversarial network), GNN (graph neural network), OT (optimal transport), AE (autoencoder), LLMs (large language models).

Methods	Handle Data Types			Generalization	Backbone
	Num.	Cate.	Text		
Mean/Mode [25]	★★	★★	–	★★★	Statistics
KNNI [73]	★★★	★★★	–	★★★	Similarity
MICE [60]	★★★	★★★	–	★	Regression
GAIN [68]/VGAIN [40]	★★★	★★★	–	★	GAN
GRAPE [69]/IGRM [75]	★★★	★★★	–	★	GNNs
ReMasker [13]	★★★	★★★	–	★	AE
NOMI [57]	★★★	★★★	–	★	Similarity
DFMs [45]	★	★★	★★★	★★★	LLMs
Table-GPT [31]/Jellyfish [71]	★	★★	★★★	★★★	LLMs
UnIMP (ours)	★★★	★★★	★★★	★★★	GNNs+LLMs

1 INTRODUCTION

Data quality has gained increasing attention due to its pivotal role in developing and training data-driven models, particularly those built on artificial intelligence (AI) technology. For instance, the performance of large language models (LLMs) can be significantly enhanced by high-quality training datasets, without requiring substantial changes to the model architecture [15]. The missing data problem, as one of the most critical issues of data quality, is ubiquitous in real-world raw datasets due to various factors such as poor data collection practices and device malfunctions [40]. To handle these issues, missing data imputation (a.k.a. missing value imputation) [57, 69] has been introduced, which aims to fill in missing values using information from observed data samples and features.

When applying imputation techniques to real-world datasets, it is desirable that these techniques can achieve high accuracy and support a variety of data types. High imputation accuracy is crucial, as existing studies have demonstrated that a higher accuracy would generally lead to a higher downstream task performance [40, 57, 68, 74]. Supporting multiple data types is also important, as real-world datasets in fields like e-commerce [65] and healthcare [26] often contain a mix of numerical, categorical and text data [49, 66]. Moreover, with advances in the Internet of Things (IoT [36]) and big data, the proportion of mixed-type data continues to grow [1, 59].

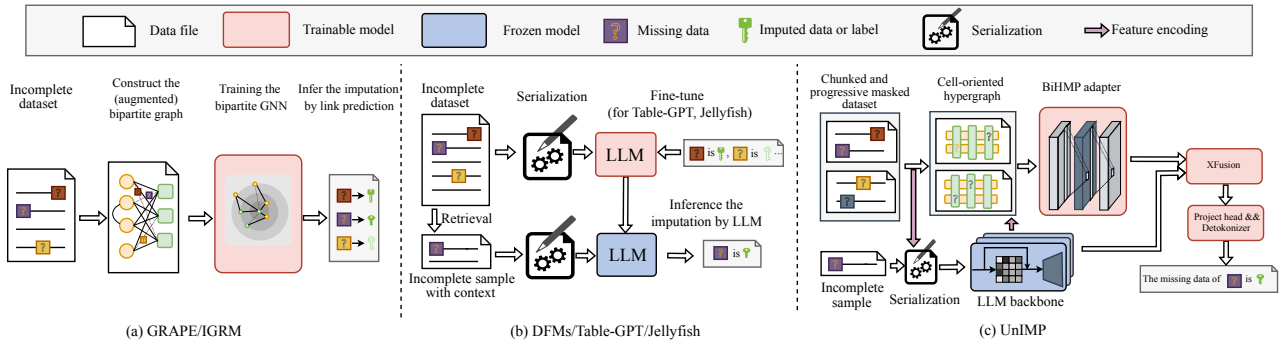


Figure 1: Comparisons of frameworks: (a) GRAPE and IGRM construct an (augmented) bipartite graph from the incomplete dataset and train a bipartite GNN, modeling imputation as link prediction. (b) DFMs, Table-GPT and Jellyfish select incomplete samples with context and infer imputations using LLMs. Table-GPT and Jellyfish fine-tune LLMs for tabular tasks, while DFMs use them directly. (c) Our model builds a cell-oriented hypergraph and encodes features with an LLM. The incomplete sample is serialized, and information is aggregated by both the LLM and BiHMP before predicting the imputation.

Existing works. Given the importance and practical value of missing data imputation, a set of techniques has been developed [13, 57, 69]. Some representative and recent methods are summarized in Table 1, and more details can be found in the surveys [35, 40]. The early-stage imputation methods were primarily rule-based, including statistics-based approaches such as Mean and Mode [25], and similarity-based methods such as K Nearest Neighbors Imputation (KNNI) [73]. These methods have good generalization capacity, as the rules can be applied to various datasets. However, their imputation accuracy is often limited since these fixed rules often struggle to capture the underlying patterns in diverse datasets effectively.

To better model relationships among samples and features within datasets for imputation, various learning-based methods are introduced, such as regression algorithm in MICE [60], generative adversarial net (GAN) [63] in GAIN [68] and VGAIN [40], optimal transport (OT) [62] in TDM [74], auto-encode (AE) in ReMasker [13], and graph neural networks (GNNs) in GRAPE [69] and IGRM [75]. The GNN-based method, which stands out for its feature aggregation capacity, is illustrated in Figure 1(a). While these methods achieve outstanding performance in handling numerical and categorical data, they struggle to process text data effectively. Furthermore, these approaches typically train a separate model for each dataset or even for an individual feature, hindering their generalizability

Recently, LLM-based methods are emerging in the field, leveraging knowledge encoded in LLM to predict missing data. LLM-based imputation can be broadly categorized into two primary categories: in-context-learning-based methods (e.g., [5, 45, 46, 70]) and fine-tuning-based methods (e.g., [31, 71]). The in-context-learning-based methods, with Direct Foundation Models (DFMs) [45] as a prominent representative, directly infer missing data by pre-trained LLM with carefully designed prompts. The fine-tuning-based methods, with Table-GPT and Jellyfish as prominent representatives, fine-tune LLM to adapt the model for tabular tasks including imputation. The LLM-based methods are illustrated in Figure 1(b).

Motivations. By serializing tabular data into text and generating the next token auto-regressively from prompts, existing LLM-based methods can handle mixed-type data with good generalization capacity. However, they typically exhibit limited performance, especially with numerical data. They directly recast the problem of imputation as text generation while the intrinsic characteristics of tabular data remain insufficiently explored: 1) *Global-local information.* The value of a cell is influenced by both global patterns across the entire table and localized details of the cell. These LLM-based methods focus on individual samples and several examples as prompt, overlooking global information; 2) *High-order dependencies.* The relationships in the table are not necessarily pairwise but may involve three or more entities simultaneously. However, LLM-based methods, built on the attention mechanism [54], mainly focus on pairwise relationships. 3) *Inter-column heterogeneity and intra-column homogeneity.* Features across columns can be highly diverse, and cells within the same column generally exhibit consistent semantics. LLM-based methods, designed for sequential inputs, struggle to align the specific contextual relationships between columns and within columns.

As summarized in Table 1, existing imputation methods generally fall into two categories: 1) LLM-based methods exhibit an unsatisfactory performance, particularly for numerical data, and 2) other learning-based methods and rule-based methods only support numerical and categorical data. To better support real-world applications, an imputation method capable of handling mixed-type data while maintaining high accuracy is in demand. A straightforward approach would be to impute each data type using the outperforming method for that type, such as applying MICE [48] or NOMI [57] for numerical and categorical data, and Jellyfish [71] for text data. However, imputing each data type separately may ignore the intricate dependencies between them, leading to inconsistencies and reduced overall accuracy.

Challenges. *Challenge I: How to aggregate global-local and high-order information for imputation while capturing inter-column and intra-column patterns?* The existence of missing data and large-scale tables complicates the aggregation process. A direct solution is to input the entire table into the LLM. However, computing complexity grows quadratically as the size of input increases. Another promising alternative is to apply the global-to-local framework [58]

to learn the multi-level features. However, it relies on similarity search on samples with missing features, leading to inaccurate results. Methods based on retrieval-augmented generation [30] show promise for aggregating information. However, they primarily focus on local information while overlooking global information.

Challenge II: How to effectively integrate the aggregation module and train the model to support mixed-type imputation? The presence of mixed-type data complicates the integration process. To handle mixed-type data, a direct approach would be to sequentially apply LLM, aggregation module and then another LLM. This pipeline first uses the LLM to embed the input, followed by information aggregation and finally generates the output by another LLM. However, treating LLM and the aggregation module as separate components may limit the alignment of the capabilities encoded in LLM with the aggregated information. Moreover, training this sequential pipeline on tables of varying sizes may fail to adequately utilize computational resources, resulting in inefficiencies.

Our solutions. Guided by the above challenges, we propose UnIMP, a **Un**ified **IMP**utation framework as depicted in Figure 1(c) that employs LLM and the high-order message passing for accurate mixed-type data imputation.

To address *Challenge I*, we design a cell-oriented hypergraph in UnIMP to model tabular datasets and introduce BiHMP, an efficient and effective Bidirectional High-order Message-Passing network, to aggregate information on the constructed hypergraph. By leveraging high-order message passing on the hypergraph, BiHMP learns to aggregate global-local and high-order information while capturing intra- and inter-column relationships. The cell-oriented hypergraph, where each cell in the dataset is modeled as a node and nodes within the same row or column are connected by hyperedges, is used to model the table structure following [8]. This structure enables the flexible integration of missing and mixed-type data. Furthermore, BiHMP is designed to efficient yet effective aggregate information. It comprises iteratively applied linear hyperedge-to-node and node-to-hyperedge layers to refine and propagate features.

To address *Challenge II*, UnIMP is designed with the LLM as the backbone, while BiHMP and Xfusion (a fusion module based on attention mechanism [54]) serve as adapters [23, 72] for the LLM. Specifically, prompts are together propagated by both the LLM and the BiHMP on the embedded hypergraph. The information aggregated by BiHMP, along with the LLM-generated data, is sent to the Xfusion module to enhance the alignment and integration of both sets of information. The fused data is subsequently sent to the projection head of LLM to generate output. Moreover, The adapter strategy trains only the BiHMP, Xfusion and projection head, freezing most LLM parameters to enhance efficiency and better preserve pre-trained knowledge in LLM while incorporating BiHMP-aggregated information.

To more effectively and efficiently train UnIMP, we follow the pre-train and fine-tune strategy [2, 34, 55] and integrate two optimizations, i.e., the chunking and the progressive masking motivated by [13, 67, 69]. The chunking technique splits all tables into smaller, uniform chunks to handle large tables and facilitate efficient batch processing, allowing for more effective use of computational resources, such as GPU memory and parallel computation capabilities. The progressive masking technique, on the other hand, gradually increases the complexity of the imputation task by progressively masking more data during training. It enables the model to learn complex patterns progressively, improving its understanding of data relationships and generalization capacity.

Table 2: Symbols and Descriptions

Notation	Description
$X, \check{X}, \overline{X}$	raw dataset, imputed dataset and ground-truth
n, d	the row number and column number
x_i, x_{ij}	i-th sample in X and its j-th column data
$M \in \{0,1\}^{n\times d}$	the mask matrix indicating incompleteness
m_i, m_{ij}	i-th sample in M and its j-th column data
$HG(\mathcal{V}, \mathcal{E})$	hypergraph with nodes $\mathcal{V}$ and hyperedges $\mathcal{E}$
θ	model parameters.
z_{v_i}, z_{e_i}	the hidden embedding of node v_i & hyperedge e_i
σ	non-linear function (ReLU)
f, g	functions or layers in the neural networks

Theoretical and empirical studies. We theoretically and empirically demonstrate the superiority of UnIMP. Theoretically, we prove that a model capable of capturing global-local information, high-order dependencies, and inter- and intra-column patterns leads to improved imputation accuracy. Furthermore, we show that UnIMP effectively incorporates these critical properties. Empirical evaluations on 10 real-world datasets with varying table sizes and data types also validate the excellent performance of UnIMP in terms of both accuracy and efficiency. For numerical and categorical data, UnIMP reduces imputation RMSE by an average of 12.11% to 45.22% compared to previous state-of-the-art methods and other baselines. In the case of imputing text data, UnIMP improves imputation accuracy by 30.53, 28.04 and 23.47 percentage points (pp) over DFMs [45], Table-GPT [31] and Jellyfish [71], respectively, as measured by ROUGE-1_{F1}. Furthermore, UnIMP can handle large datasets and exhibits robust generalization ability.

Contributions. The main contributions are as follows.

- We propose a unified framework UnIMP to accurately impute mixed-type data, including numerical, categorial and text data.
- We propose a cell-oriented hypergraph to model tabular data and design BiHMP, an efficient and effective bidirectional high-order message-passing network, to aggregate information while capturing the key intrinsic characteristics of tabular data.
- We introduce Xfusion, along with BiHMP, as adapters for LLM to process diverse data and enable accurate mix-type imputation. Additionally, we integrate chunking and progressive masking techniques to train UnIMP, enhancing performance.
- We theoretically prove the critical role of global-local information, high-order relationships, and inter- and intra-column patterns to the accuracy of imputation.
- Extensive empirical experiments on 10 real-world datasets validate the outstanding performance of UnIMP.

2 PRELIMINARIES

2.1 Problem Statement

Following previous works [40, 57], the task of missing data imputation is defined over tabular datasets $\mathcal{X}$ [31, 40, 57, 74], where each $X \in \mathcal{X}$ consists of n data samples (rows) and d features (columns). We denote the j-th feature of the i-th sample as x_{ij}, which can be

of numerical, categorical or text type. A mask matrix $M \in R^{n \times d}$ is used to indicate missing components in X. m_{ij} equals 1 if x_{ij} is observed and 0 if x_{ij} is missing, i.e., $X = X_{obs} \cup X_{miss}$ where $X_{obs} = \{x_{ij} | x_{ij} \in X, m_{ij} = 1\}$ and $X_{miss} = \{x_{ij} | x_{ij} \in X, m_{ij} = 0\}$. The imputed data and the ground-truth data are denoted as $\tilde{X}$ and $\overline{X} = X_{obs} \cup \overline{X}_{miss}$, respectively. Note that the indexing in this paper starts from zero, which is a common convention in many programming languages.

The hypergraph is a generalization of the graph where each hyperedge may contain an arbitrary number of nodes. We denote a hypergraph by $HG(\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is a set of nodes, and $\mathcal{E}$ is the set of hyperedges. $\mathcal{E} \subseteq P^*(\mathcal{V}) \setminus \{\varnothing\}$ where $P^*(\mathcal{V})$ is the power set on the nodes [8]. We use x_v to denote the cell w.r.t. the node v. With the above notations, we define the task of mixed-type missing data imputation as follows.

Definition 2.1:(Mixed-type Missing Data Imputation [57, 69]). *Mixed-type missing data imputation* aims to impute the unobserved elements in the raw data, i.e., X_{miss}, and make the imputed matrix $\tilde{X}$ as close to the real complete dataset $\overline{X}$ as possible. The raw data matrix X may contain numerical, categorical and text data.

2.2 Large Language Models

LLMs, such as GPT [6] and Llama [15], which contain billions of parameters, show remarkable emergent behaviors and impressive zero-shot capabilities across diverse tasks. The execution process of LLMs typically includes four key steps: 1) a tokenizer that segments input text into discrete tokens, 2) an LLM backbone that is used for feature propagation, 3) a projection head that predicts the next token, and 4) a decoding process that translates token IDs back into human-readable text. These LLMs are often auto-regressive, trained on large text corpora with the objective of maximizing the log-likelihood of the next word given the previous words.

$$\theta_{LLM} = \arg\max_{\theta} \sum_{i} \log P(t_i | t_{i-k}, \cdots, t_{i-1}; \theta) \tag{1}$$

where t_i stands the token and k is the context window size. There are two typical ways to adapt the model to new tasks and boost performance: supervised-fine-tuning (SFT) and in-context-learning. For SFT, given a training query $T = \{t_0, t_1, \cdots, t_s\}$ and training target y, the objective is to maximize the following log-likelihood: $\sum_{(T,y)} \log P(y | t_0, t_1, \cdots, t_s)$. The in-context-learning-based methods handle new tasks during inference by constructing a prompt that includes examples (i.e., context). Given a set of example queries $\{T^0, T^1, \cdots, T^w\}$, targets $\{y^0, y^1, \cdots, y^w\}$, and a new query text T^{w+1}, a prompt $(T^0, y^0, \cdots, T^w, y^w, T^{w+1})$ is constructed and sent to the LLM to infer the results.

3 THE OVERALL FRAMEWORK

In this section, we first introduce three key intrinsic characteristics of tabular data and prove its critical role for imputation. Then, we introduce the details of UnIMP.

3.1 Intrinsic Characteristics of Tabular Data

Global-local information. The global-local information suggests that the value of a cell is influenced by both neighboring cells (local context) and cells distributed across the entire table (global context) [57, 58]. For example, as illustrated in Figure 2, the name of the president is determined not only by the associated nation but also by the sequential relationship of terms. Hence, we can infer that the missing value is the president succeeding President Trump.
High-order relationship. The high-order relationship suggests that the value of the cell is influenced simultaneously by multiple columns as a whole. For example, as shown in Figure 2, the name of a president can be determined simultaneously by the nation and the term as a whole. However, neither the nation nor the term alone is sufficient to fully determine the value.
Intra-column heterogeneity and intra-column homogeneity. Features across columns can be highly diverse, and cells within the same column typically exhibit consistent semantics. For example, as shown in Figure 2, the name is text data and the nation is categorical data. Furthermore, the name remains consistent across rows.

Next, we formally define the imputation error.

Definition 3.1:(Imputation Error). Given a model $\theta_{X_{obs}}$ trained on the observed data X_{obs}, the imputation error $\Psi(\theta_{X_{obs}}, X_{miss})$ captures the expected performance of $\theta_{X_{obs}}$ on the test unobserved data X_{miss}, i.e., $\Psi(\theta_{X_{obs}}, X_{miss}) = \mathbb{E}[\|\theta_{X_{obs}}(X_{miss}) - \overline{X}_{miss}\|]$. When the context is clear, we denote $\theta_{X_{obs}}$ as θ to improve readability.

We then have the following three theorems highlighting the importance of global-local information, high-order information and column patterns, respectively. The proofs are in Section 4.

Theorem 3.1: *Consider two imputation models, θ^{g+l} and θ^{l} (resp. θ^{g}), where θ^{g+l} simultaneously captures both global and local information, and θ^{l} (resp. θ^{g}) captures only the local (resp. global) information. Assuming that interactions of global and local information are independent, then:*

$$\Psi(\theta^{g+l}, X_{miss}) \leq \Psi(\theta^{l}, X_{miss}) \text{ or } \Psi(\theta^{g}, X_{miss})$$

indicating that a model capable of capturing both global and local information achieves a lower imputation error.

Our work systematically combines both types of information and theoretically demonstrates the value of this integration.

Theorem 3.2: *Consider two imputation models, $\theta^{[0:r]}$ and $\theta^{[0:s]}$, where $\theta^{[0:r]}$ captures interactions up to order r in the latent space, and $\theta^{[0:s]}$ captures interactions up to order s, with $r > s$. We have*

$$\Psi(\theta^{[0:r]}, X_{miss}) \leq \Psi(\theta^{[0:s]}, X_{miss}),$$

indicating that the model capable of capturing higher-order interactions exhibits a lower imputation error.

Theorem 3.3: *Consider two imputation models, θ^{cp} and θ, where θ^{cp} captures the column patterns including intra-column heterogeneity and intra-column homogeneity, while θ does not. Then, we have:*

$$\Psi(\theta^{cp}, X_{miss}) \leq \Psi(\theta, X_{miss}),$$

indicating that model θ^{cp} capturing the column patterns achieves a lower imputation error.

These claims motivate us to design a graph-based method with multi-hop message passing for effective information aggregation. The BiHMP in UnIMP integrates global-local information to enhance imputation, while hypergraph modeling and BiHMP capture

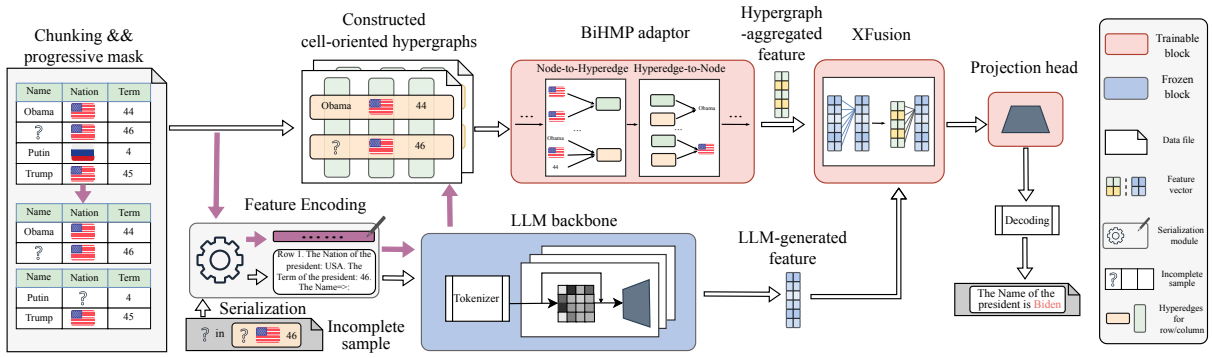


Figure 2: Framework overview of UnIMP. Given an incomplete dataset, UnIMP chunks the data and applies progressive masking. A hypergraph is constructed for each chunk, with node and hyperedge features extracted by an LLM from serialized data. The incomplete sample is serialized, processed by the LLM, and aggregates information on the hypergraph via the BiHMP adaptor, which iteratively applies Node-to-Hyperedge and Hyperedge-to-Node layers. Features from the LLM and hypergraph are fused through the Xfusion module, followed by a projection head for imputation.

high-order dependencies. Additionally, the cell-oriented, row-level, and column-level designs in UnIMP enable the model to capture inter-column heterogeneity and intra-column homogeneity.

3.2 Overview of UnIMP

Guided by the above characteristics of tabular data, we propose UnIMP for accurate mixed-type data imputation. The overall architecture is illustrated in Figure 2. Given the incomplete raw dataset, UnIMP first utilizes the chunking technique and progressive masking to process data. Then, cell-oriented hypergraphs are constructed. UnIMP utilizes the tokenizer and LLM backbone for feature initialization and propagation, embedding both raw data and query prompts. These obtained features are processed through the BiHMP module, which iteratively runs the node-to-hyperedge layer and the hyperedge-to-node layer to aggregate both local and global information. The aggregated feature is then fused with the embedded prompts using an XFusion module. Finally, a projection head maps the predicted data back to token space, and a decoding process is employed to translate the tokens into human-readable results.

3.3 Cell-Oriented Hypergraph Modeling

Motivation. To capture the structural properties of tabular data, existing methods often employ graph-based approaches, such as bipartite graphs [69, 75] and similarity graphs [7]. While effective, these methods fall short in comprehensively representing the complex interactions and higher-order relationships inherent in tabular data. To better capture the intrinsic characteristics of tabular data, we adopt a hypergraph-based approach for modeling. While some hypergraph-based methods, such as the cell-oriented hypergraph in [8] and the value-oriented method in [12], have been introduced for general tabular representation learning, they have not yet been employed for imputation. The value-oriented approach models each distinct value as a node, with nodes in the same row sharing a hyperedge. However, this method struggles to deal with mixed-type data and effectively handles missingness. Moreover, missing data imputation focuses on the granularity of individual cells, as missingness occurs at the cell level. Therefore, we adopt the cell-oriented hypergraph following [8]. It is important to note that the work in [8] focuses on tabular representation learning, our work is specifically designed for missing data imputation. Additionally, their method treats hypergraph neural network and language models as separate components, whereas we introduce BiHMP and Xfusion, which serve as the adapters for LLM, to better aggregate information and combine both advantages.

Given a tabular dataset X with n samples, each containing d features, we construct a hypergraph $HG(\mathcal{V}, \mathcal{E})$ as follows:

- For each cell $x_{ij} \in X$, we create a corresponding node $v_{idx} \in \mathcal{V}$, where $idx = i * d + j$.
- For nodes in the same column (i.e., nodes corresponding to $\{x_{0j}, x_{1j}, \cdots\}$), we construct a hyperedge $e_j \in \mathcal{E}$.
- Similarly, nodes in the same row (i.e., nodes corresponding to $\{x_{i0}, x_{i1}, \cdots\}$) form a hyperedge $e_{i+d} \in \mathcal{E}$.

Each node has a degree of 2, connecting to both a row-level and a column-level hyperedge. The edge cardinality of a row-level hyperedge is the number of columns (d), while that of a column-level hyperedge is the number of samples (n). This hypergraph-based modeling effectively captures tabular characteristics. The hyperedge captures the high-order relationship among cells. Moreover, each cell is represented as a distinct node, capturing intra-column heterogeneity, while nodes within the same column are linked by a hyperedge, capturing inter-column homogeneity.

3.4 Feature Encoding

As the neural network models need vectorized inputs, we introduce a vectorization technique to project data from the text space into the vector space. We use the LLM backbone to encode the text data.
Serialization. In order to obtain the embedding, we need to feed the proper prompt to the LLM backbone. Recent research has explored various approaches to serializing tabular data, like the sentences-transformation and the markdown-transformation [17]. Here, we follow the widely-used serialization method of attribute-value pair [17], which is shown as follows:

$$\text{Serial}(x_{ij}) = \text{Row i, col\_name}[d \neq i]\text{: } x_{dj}\text{, col\_name}[i]\text{=>}\{x_{ij}\}\text{ EOS}$$

Here, EOS (i.e., end of the sentence) is a special token to denote the end of the prompt. We use the following serialization prompt for hyperedge data.

$$\text{Serial}(i\text{-th row}) = \text{This is row: i EOS;}$$

$$\text{Serial}(i\text{-th col}) = \text{This is col: col\_name[i] EOS;}$$

Tokenization. Then, the text is sent into the tokenizer to split the prompt into tokens. There are many commonly used tokenizers like BPE [4] and SentencePiece [29]. In our experiments, we use the SentencePiece tokenizer associated with Llama2.

$$\{t_0, t_1, \cdots, t_s\} = \text{tokenizer}(\text{prompt-text})$$

Propagation of LLM backbone. The tokens are processed through the LLM backbone for feature initialization and propagation. Initially, the LLM backbone performs a lookup in the embedding matrix, a predefined matrix with dimensions $|\text{Vocab}| \times d$, where $|\text{Vocab}|$ denotes the vocabulary size and d represents the hidden dimension. Each token ID corresponds to a specific row in this embedding matrix, serving as an index to retrieve the corresponding embedding vector. These embedding vectors are then fed into the transformer layer [54] for feature propagation.

$$z_p : \{z_{t_0}, z_{t_1}, \cdots, z_{t_s}\} = \text{LLM-backbone}(t_0, t_1, \cdots, t_s)$$

Each token t_i is encoded as a feature representation z_{t_i}. For the prompt text corresponding to node v_i (or hyperedge e_i), we use the representation of the last token as the initial feature $z^0_{v_i}$ (or $z^0_{e_i}$, respectively). For numerical data, following previous works [13, 40], we directly encode the input as its value to avoid unnecessary computational overhead. For categorical features, we employ label encoding, which assigns unique integers to each category, aligning with recent works [22, 40, 57, 58, 74]. Encoded numerical and categorical data are padded into the same latent dimension.

3.5 Bidirectional High-order Message Passing

Motivation. In tabular data, interactions often involve multiple entities simultaneously. For instance, multiple cells within the same row and column exhibit high-order interactions. To capture these complexities, Hypergraph Neural Networks (HGNNs) (e.g., [19]) can be employed for message passing, but they typically come with high computational costs. While a hypergraph-structure-aware transformer was proposed in [8], it not only faces efficiency challenges but also tends to lose fine-grained local information due to its maximal invariant properties. A simplified and fast HGNN [52] can accelerate training, but it only handles node information, overlooking hyperedges. Therefore, we propose BiHMP, a novel Bidirectional High-order Message Passing network that operates bidirectionally between nodes and hyperedges in the constructed hypergraph. BiHMP comprises two key components: 1) Node-to-Hyperedge layer, which propagates information from each node to its associated hyperedges through a linear layer. 2) Hyperedge-to-Node layer, which propagates information from hyperedges back to their constituent nodes through a linear layer. Through the iterative application of these two layers, BiHMP captures high-order and multi-hop relationships within the data, enhancing its ability to model complex interactions.

Node-to-Hyperedge. For each hyperedge $e_j \in \mathcal{E}$ and node $v_i \in e_j$, with their respective embeddings $z^l_{e_j}$ and $z^l_{v_i}$ in l-th Node-to-Hyperedge layer, we use the following equation to learn to update the representation of hyperedge:

$$\begin{aligned} z^{temp}_{e_j} &= \frac{1}{|e_j|} \sum_{v_i \in e_j} \sigma\left(f^l_1(z^l_{v_i})\right) \\ z^{l+1}_{e_j} &= \sigma\left(f^l_2\left(\text{CONCAT}\left(z^l_{e_j}, z^{temp}_{e_j}\right)\right)\right) \end{aligned} \tag{2}$$

where $f^l_1(\cdot)$ and $f^l_2(\cdot)$ represents learnable linear transformation functions. $\sigma(\cdot)$ is a non-linear activation function (ReLU in UnIMP). $\text{CONCAT}(\cdot)$ is the concatenation function that stacks multiple inputs into a single, longer vector. For each node in the hyperedge, we apply a learnable transformation followed by a non-linear activation $\phi(\cdot)$. We then compute the mean of these transformed node features to obtain $z^{temp}_{e_i}$, which captures the average information from all nodes in the hyperedge. This temporary representation is concatenated with the representation of the hyperedge from the previous layer, $z^l_{e_j}$. Finally, we apply another transformation layer and non-linear activation layer to this concatenated vector to obtain the updated hyperedge representation $z^{l+1}_{e_j}$.

Hyperedge-to-Node. The Hyperedge-to-Node layer updates node representations by aggregating information from their incident hyperedges. For each node $v_i \in \mathcal{V}$ with its two incident hyperedges i.e., the row-level hyperedge $e^r_{v_i}$ and column-level hyperedge $e^c_{v_i}$, we update the node representation by the following equation in the l-th Hyperedge-to-Node layer:

$$z^{l+1}_{v_i} = \sigma\left(f^l_3\left(\text{CONCAT}\left[z^l_{v_i}, z^l_{e^c_{v_i}}, z^l_{e^r_{v_i}}\right]\right)\right) \tag{3}$$

where $f^l_3(\cdot)$ is a learnable linear function. The Hyperedge-to-Node layer first concatenates the embedding of the node $z^l_{v_i}$ with the embeddings of all hyperedges that contain the node, i.e., $z^l_{e^c_{v_i}}$ and $z^l_{e^r_{v_i}}$. The concatenated representation is then processed through a learnable linear transformation function $f^l_3(\cdot)$. Finally, a non-linear activation function $\sigma(\cdot)$ is applied to obtain the updated node representation in $(l+1)$-th layer $z^{l+1}_{v_i}$.

3.6 XFusion Block and Projection Head

XFusion block. After encoding the prompt and aggregating local and global information by BiHMP, it is crucial to effectively integrate these features for accurate imputation. We designed XFusion to synthesize the rich information derived from the prompt feature (denoted as $z_p = \{z_{t_0}, z_{t_1}, \cdots, z_{t_s}\}$) with the features extracted from the hypergraph representation (denoted as Z_g where $z_g \in Z_g$ is a concatenation of $z^{l_{max}}_{e^c_{v_i}}$ and $z^{l_{max}}_{e^r_{v_i}}$ for imputing node v_i). XFusion is based on the attention mechanism [54], a critical component of LLMs. The attention is computed as follows:

$$\text{Attn}(Q, K, V; w) = w \cdot \text{Softmax}\left(\frac{QK^T}{\sqrt{d_K}}\right)V \tag{4}$$

where Q, K and V are the input feature matrices, w is a learnable weight matrix, and d_K is the dimensionality of the vectors in K.

Building on this attention mechanism, our XFusion module leverages both token embeddings and hypergraph-derived embeddings to construct a rich, context-aware representation. The forward process of XFusion is as follows:

$$\begin{aligned} z_p^{temp} &= \text{Attn}(z_p W_Q, z_p W_K, z_p W_V; w_1) \\ z_{output} &= \text{Attn}(z_p^{temp} W_Q, z_g W_K, z_g W_V; w_2) \end{aligned} \tag{5}$$

Here, W_Q, W_K and W_V are learnable weight matrices. We first calculate the self-attention of the prompt embedding. We then use the obtained prompt feature h_p^{temp} as the query and the graph feature h_g as both the key and value matrices. The attention mechanism between prompt embeddings and graph embeddings allows the model to effectively align and integrate the prompt information with the local and global information from the graph embeddings.

When predicting the name of the president, as shown in Figure 2, the LLM uses its knowledge to infer the name, while the hypergraph provides contextual details, such as name patterns. Xfusion combines both to enhance imputation accuracy.

Projection head. The obtained $z_{output} \in \mathbb{R}^{(s+1)\times d_h}$ is in a high-dimensional space, and the projection head is a key component that maps this high-dimensional embedding to a target space. It is modeled as a linear layer $f^{head} : z_{output} \mapsto o \in \mathbb{R}^{(s+1)\times d_o}$. For numerical and categorical data, $d_o = 1$, as it requires only a single value as the imputation. In contrast, for text data, the task involves predicting a token ID, making it a classification task. Hence, $d_o = |\text{Vocab}|$, where $|\text{Vocab}|$ denotes the size of the vocabulary.

With the above introduction of key components in UnIMP, we now show that the UnIMP captures the key intrinsic characteristics analyzed in Section 3.1. The proof is in Section 4.

Theorem 3.4: *The UnIMP framework captures the global-local information, high-order relationship, inter-column heterogeneity and intra-column homogeneity for imputation.*

4 TRAINING AND ANALYSIS

4.1 Training Objectives

Motivation. The training objective guides the model towards desired behaviors and outputs during the learning process. For the text data, it may contain an arbitrary length of tokens. Therefore, it is often modeled as the classification of the token ID of the next token and is processed in an auto-regressive manner. For the numerical data and the label-encoded categorical data, the imputation is often modeled as the task of regression.

The learning process for text data is handled using an autoregressive approach. For a sequence of tokens $(t_0, t_1, \cdots, t_s)$, an autoregressive model tries to predict the next element in the sequence, given all the previous tokens. This objective is based on Cross Entropy (CE) [10, 56] as follows:

$$\begin{aligned} &\mathcal{L}(T;\theta) = P(t_0) \cdot P(t_1|t_0) \cdots P(t_s|t_0, t_1, \cdots, t_{s-1}) \\ &= \prod_{i=0}^{s} \text{CE}(t_i, \theta(t_0 \cdots t_{i-1})) = \prod_{i=0}^{s} \sum_{j=0}^{d_o} -t_i[j] \cdot \log(\theta(t_0 \cdots t_{i-1})[j]) \end{aligned} \tag{6}$$

Algorithm 1: Pre-Training Pipeline

Input: The tabular datasets $\mathcal{T}$ with its masks $\mathcal{M}$, chunk size c_{size}, batchsize b_{size}, model parameters θ_{UnIMP}, learning rate φ, Epoch num $Epochs$, mask rate κ.

```
{X^c, M^c} ← Split 𝒳, ℳ into batched chunks by c_size & b_size
Initialize optimizer opt_θ with learning rate φ
for Epoch ∈ {0, 1, ··· , Epochs} do
    'M^c ← mask (κ + Epcoh/Epochs × 0.30) of M^c
    for each X^c ∈ {X^c} do
        X̃^c ← Algorithm 2(X^c, 'M^c, θ_UnIMP)
        for each x_ij ∈ T^c and M^c_ij = 0 do
            if x_ij being numerical or categorical then
                ℒ ← Compute loss by Eqn 7
            else
                ℒ ← Compute loss by Eqn 6
        Update θ_UnIMP by opt_θ with loss ℒ/|X_c|.
Return θ_UnIMP
```

Where $t_i[j]$ and $\theta(\cdot)[j]$ are the probabilities of corresponding values belonging to j-th class.

The learning objective for numerical and categorical data is formulated using the Huber Loss [24]. For a true value x_{ij} and predicted value $\tilde{x}_{ij}$, Huber Loss is defined:

$$\mathcal{L}(x_{ij}, \tilde{x}_{ij}) = \begin{cases} \frac{1}{2}(x_{ij} - \tilde{x}_{ij})^2 & \text{for } |x_{ij} - \tilde{x}_{ij}| \le \delta \\ \delta|x_{ij} - \tilde{x}_{ij}| - \frac{1}{2}\delta^2 & \text{otherwise} \end{cases} \tag{7}$$

where δ is a positive hyper-parameter. Compared to other loss functions, Huber Loss offers the following advantages: 1) For errors smaller than δ, Huber Loss behaves like Mean-Squared-Error (MSE), being sensitive to small errors; 2) For errors larger than δ, it behaves like Mean-Absolute-Error (MAE), being less sensitive to potential outliers. Moreover, Huber Loss is differentiable at all points, which is beneficial for optimization. By adjusting the value of δ, it is flexible to balance between MSE and MAE, adapting to different contexts. We set $\delta = 1$ suggested by the experiment in Section 5.

4.2 Training Pipeline

Motivation. Previous works in imputation [57, 68, 69, 74] typically train a separate model for each dataset. While effective, this practice significantly increases computational costs, as training a model can be time-intensive. Additionally, models trained on individual datasets often struggle to generalize to other datasets. A common alternative in the field of LLM is the "pre-train and finetune" strategy. However, the variability in table sizes and the propensity of neural models to overfit present challenges. In this work, we follow the "pre-train and finetune" paradigm for imputation, introducing two plug-and-play optimizations: the chunking technique and the progressive masking technique.

Optimization 1: Chunking technique. We split the entire dataset into smaller, uniformly sized chunks (512 rows in each chunk in our experiments). This approach ensures that all chunks are processed efficiently and in parallel, enabling batch training and reducing the

heavy IO time associated with handling large and irregularly sized datasets. By transforming the dataset into manageable chunks, we also facilitate more effective memory utilization, preventing bottlenecks during model training. Moreover, in this way, the cardinality of column-level hyperedges becomes closer to that of row-level hyperedges, ensuring a more balanced edge distribution across chunks compared to previous works. This mitigates skewness and ultimately enhances performance.

Optimization 2: Progressive masking technique. Starting with the raw dataset, we gradually increase the proportion of masked data. The newly masked ratio is set from κ to $\kappa + 30\%$ where κ is hyperparameter and is set to 35% suggested by the experiments in Section 5. As training progresses, we mask more data, gradually exposing the model to more challenging imputation scenarios. This progressive masking strategy allows the model to incrementally learn patterns from simpler to more complex missing data scenarios and thus improve the generalization ability.

We summarize the overall pre-training pipeline in Algorithm 1. Following the SFT, the fine-tuning process has a similar procedure to the pre-training. The pre-training stage utilizes all available datasets, while fine-tuning focuses on specific datasets. The pipeline inputs training datasets, chunk size, batch size, initial model parameters, learning rate and the number of epochs. It first splits the training datasets into batched chunks (Line 1) and initializes the optimizer with the given learning rate (Line 2). The model is trained for *Epochs* epochs (Lines 3–12). During each epoch, we mask more data, and for each batch, the algorithm performs forward propagation (as described in Algorithm 2 in the full version [16]) to compute the imputation results (Line 6). For numerical or categorical data, the loss is calculated using Eqn 7, while for text data, the loss is computed using Eqn 6. After processing all cells in a chunk, the model parameters are updated using the optimizer and the loss (Line 12). Finally, the learned parameters are returned (Line 13).

4.3 Complexity Analysis

The LLM inference needs $C_{\text{prompt}} = O(l_{\text{trans}} \cdot t_{\text{max}}^2 \cdot d_{\text{LLM}})$ for inferencing a prompt, where l_{trans} is the number of transformer layers, t_{max} is maximum number of tokens, and d_{LLM} is the dimension of embeddings in LLM. The feature encoding process requires encoding $|\mathcal{V}| + |\mathcal{E}|$ prompts. Therefore, the total time complexity of feature encoding is $O\big((|\mathcal{V}| + |\mathcal{E}|) \cdot C_{\text{prompt}}\big)$.

The message passing of l_{max} layers of Node-to-Hyperedge and Hyperedge-to-Node operations is with time complexity of $C_{\text{msg}} = O(l_{max} \times (d_{LLM} \times |\mathcal{V}| + 2 \times d_{LLM} \times |\mathcal{E}|)$. The attention-based fusion has a time complexity of $C_{\text{fusion}} = (t_{max}^2 \times d_{LLM})$, and the projection has $C_{\text{proj}} = (d_{LLM})$. Therefore, the total time complexity of training *Epochs* epochs is $O(Epochs \times t_{\text{max}} \times (C_{\text{msg}} + C_{\text{fusion}} + C_{\text{proj}}))$.

4.4 Theoretical Proofs

In this part, we present the proofs for the Theorems. We first introduce the concepts of entropy and mutual information, which are the basis of the proof.

Definition 4.1: $H(X) = -\sum_{x \in X} p(x) \log p(x)$ represents the entropy that quantifies the average level of information associated with X. $I(X;Y) = H(X) - H(X|Y)$ is the mutual information between the X and Y.

Lemma 4.1: *$H(X|Y) \geq H(X|Y,Z))$ and the equation get if the information of Z is encoded in Y.*

Proof. $H(X, Z \mid Y) = H(Z \mid Y) + H(X \mid Y, Z)$. Therefore $H(X \mid Y, Z) = H(X, Z \mid Y) - H(Z \mid Y)$. As $H(Z \mid Y) \geq 0$, we can obtain the inequality. □

Proof of Theorem 3.1. The imputation error reflects how well the model trained on the training dataset generalizes to the unobserved data. According to the information bottleneck theory [28, 61], $\Psi(\theta_{X_{obs}}, X_{miss})$ is proven to scale as $\tilde{O}(\sqrt{\frac{I(X_{miss};Z_{miss})+1}{n_{obs}}})$ where and n_{obs} is the number of training observed data. For model θ^{g+l}, its mutual information is $I^{g+l}(X_{miss}; Z_{miss}) = I(X_{miss}; Z_{miss}|B^{g+l}) = I(X_{miss}; Z_{miss}|B^g, B^l)$ where B^g is the global information, B^l is the local information, B^{g+l} is the fused global and local information. For model θ^l, its mutual information is $I^l(X_{miss}; Z_{miss}) = I(X_{miss}; Z_{miss}|B^l)$. Then we have

$$
\begin{aligned}
&I(X_{miss}; Z_{miss}|B^l) = I(X_{miss}; Z_{miss}, B^l) - I(X_{miss}; B^l) \\
&= H(X_{miss}) - H(X_{miss}|Z_{miss}, B^l) - H(X_{miss}) + H(X_{miss}|B^l) \\
&= H(X_{miss}|B^l) - H(X_{miss}|Z_{miss}, B^l) \\
&\approx H(X_{miss}|B^l) - H(X_{miss}|Z_{miss}, B^l, B^g) \\
&\geq H(X_{miss}|B^l, B^g) - H(X_{miss}|Z_{miss}, B^l, B^g) \\
&= I(X_{miss}; Z_{miss}|B^l, B^g)
\end{aligned}
$$

The first line is obtained by the definition of conditional mutual information. The second line is obtained by the definition of mutual information. The fourth line and fifth line are based on Lemma 4.1. The fourth line is obtained as Z_{miss} contains the information in B^g, adding this term will not increase the entropy. The sixth line can be obtained by the reversal of the first three lines. Therefore, we have $\Psi(\theta^{g+l}, X_{miss}) \leq \Psi(\theta^l, X_{miss})$. Similarly, we can have $\Psi(\theta^{g+l}, X_{miss}) \leq \Psi(\theta^g, X_{miss})$.

Proofs of Theorems 3.2 and 3.3. The proofs of Theorem 3.2 and Theorem 3.3 are similar to that of Theorem 3.1. Given $\theta^{[0:r]}$, its mutual information can be represented as $I^{[0:r]}(X_{miss}; Z_{miss}) = I(X_{miss}; Z_{miss}|B^{[0:r]}) = I(X_{miss}; Z_{miss}|B^{[0:s]}, B^{[s+1:r]})$. Here, we use $B^{[0:r]}$ to represent the captured interactions from order 0 to order r. For model $\theta^{[0:s]}$, its mutual information is $I^{g+l}(X_{miss}; Z_{miss}) = I(X_{miss}; Z_{miss}|B^{[0:s]})$ where $B^{[0:s]}$ is the captured interactions from order 0 to order s and $r \geq s$. Then, we can get $\Psi(\theta^{[0:r]}, X_{miss}) \leq \Psi(\theta^{[0:s]}, X_{miss})$ by replacing B^{g+l} and B^l in the proof of Theorem 3.1 with $B^{[0:r]}$ and $B^{[0:s]}$, respectively. Similarly, given a model θ^{cp}, its mutual information can be calculated as $I^{cp}(X_{miss}; Z_{miss}) = I(X_{miss}; Z_{miss}|B^{cp}, B^{base})$ where B^{cp} is the captured column patterns beside the base pattern B^{base}. Then, we can get the inequality $\Psi(\theta^{cp}, X_{miss}) \leq \Psi(\theta, X_{miss})$ by replacing B^{g+l} and B^l in the proof of Theorem 3.1 with B^{cp} and B^{base}, respectively.

Proof of Theorem 3.4. *1): Global-local information.* We prove that any two cells x_{ij} and x_{rs} can interact via a two-hop path in the hypergraph. Since v_{ij} and v_{is} both lie in the same row hyperedge, there is a direct connection: $d(v_{ij}, v_{is}) = 1$. Next, since v_{is} and v_{rs} share the same column hyperedge E_s^c, they are directly connected: $d(v_{is}, v_{rs}) = 1$. Thus, the overall distance is $d(v_{ij}, v_{rs}) \leq d(v_{ij}, v_{is}) +$

Table 3: The profiles of datasets

Dataset	Alias	Domain	n	d_{num}	d_{cate}	d_{text}
Blogger	BG	Social	100	0	6	0
ZOO	ZO	Biology	101	0	17	0
Parkinsons	PK	Healthcare	195	22	1	0
Bike	BK	Trans.	8,760	12	1	0
Chess	CS	Game	28,055	3	4	0
Shuttle	ST	Physics	43,500	0	10	0
Power	PW	Energy	2,049,280	6	0	0
Buy	BY	Retail	651	1	1	2
Restaurant	RR	Service	864	0	2	3
Walmart	WM	Location	4,654	0	3	2

$d(v_{is}, v_{rs}) = 2$. Moreover, the BiHMP has 3 layers (as in Section 5.1), allowing it to aggregate information from the whole table.

2): High-order information. The message passing of Node-to-Hyperedge follows $z_{e_j}^{temp} = \frac{1}{|e_j|} \sum_{v_i \in e_j} \sigma\left(f_1^l(z_{v_i}^l)\right)$. It aggregates information over a set of cells at each layer, thereby capturing high-order relationships among cells.

3): Inter-column heterogeneity and intra-column homogeneity. Each cell v_{ij} is represented as a distinct node and is connected to a specific column hyperedge e_j. According to the Node-to-Hyperedge in Equation 2, the network to learn distinct transformations for different columns, thus capturing heterogeneity between columns. Moreover, according to the Hyperedge-to-Node in Equation 3, cells in the same column are learned with the same column representation, thus capturing intra-column homogeneity.

5 EXPERIMENTAL EVALUATION

5.1 Experiment Setup

Datasets description. The evaluation uses 10 real-world datasets from UCI [14] and Kaggle [27], following the previous works [13, 39, 40, 57]. The profiles of these datasets are presented in Table 3.

Missing mechanisms. Original data are fully observed, we thus follow previous works [40, 74] to generate the mask matrix. Three mechanisms are utilized, i.e., missing completely at random (MCAR), missing at random (MAR) and missing not at random (MNAR).

Baseline methods. We include 13 baselines, including state-of-the-art methods, for a comprehensive evaluation. Specifically, we incorporate 10 baselines for numerical and categorical data, including MEAN [25], KNNI [73], MICE [60], VGAIN [40], GAIN [68], GINN [50], GRAPE [69], IRGM [75], TDM [74], NOMI [57] and ReMasker [13]. Additionally, we compare our method to three LLM-based approaches that handle multi-type data imputation, including DFMs [45], Table-GPT [31] and Jellyfish [71].

For our model, we include two variants, i.e., UnIMP which is a generalized model trained on all datasets using one unified parameter set and UnIMP-ft which is fine-tuned on specific datasets. The pre-training phase uses only raw incomplete datasets. Moreover, these datasets are distinct and from different domains as shown in Table 3, ensuring no overlap to avoid data leakage.

Metrics. For numerical and categorical data, we adopt Root-Mean-Square-Error (RMSE) and Mean-Absolute-Error (MAE) as metrics, following previous works [40, 58, 68], to assess accuracy. For text data, we employ two metrics to assess accuracy: Recall-Oriented Understudy for Gisting Evaluation (ROUGE-1) [44] and Cosine Similarity (Cos-Sim) [47], covering both lexical-wise and semantic-wise approaches. The detailed definitions are in online full version [16].

For the RMSE and MAE, a lower value indicates a better imputation. For the ROUGE-1_{F_1} and Cos-Sim, a higher value indicates a better imputation. The first seven datasets (BG, ZO, PK, BK, CS, ST, PW), which with only numerical and categorical features, are assessed using RMSE and MAE. The last three datasets (BY, RR, WM) that include text are evaluated by ROUGE-1_{F1} and Cos-Sim.

Implementation details. The missing rate is 20% and the missing mechanism is MCAR by default. The default LLM for all baselines is Llama2-7B [53], used out-of-the-box without parameter modifications. We use Jellyfish-7B [71] to keep the same size of LLM. As the model in [31] is not released, we fine-tune Llama2-7B using the provided datasets. The number of layers is 3 in BiHMP. For the numerical and categorical data (*resp.* text data), we set the chunk size as 512 (*resp.* 32) with a batch size of 64 (*resp.* 2) and set the latent dimension as 64 ((*resp.* 4096). We trained for 4000 (*resp.* 100) epochs and fine-tuned with 1000 (*resp.* 20) epochs. We conduct experiments on a server with an Intel 4314 CPU, 503GB system memory, and 2 Nvidia A5000 GPUs, each with 24GB GPU memory [33].

5.2 Accuracy Comparison

Exp-1: Accuracy over numerical and categorical data. The results are summarized in Table 4, with a missing rate of 20% under the MCAR mechanism. Some results are omitted due to imputation processes exceeding 10 hours (OOT) or running out of memory (OOM). As shown in the table, LLM-based methods demonstrate poor performance, while our method achieves good performance across both data types. Graph-based methods, including GINN, GRAPE, IGRM and UnIMP show an outstanding performance. Furthermore, UnIMP-ft further enhances performance. Specifically, in terms of RMSE, UnIMP-ft shows average performance gains of 12.11%, 28.27%, 28.32% and 29.61% compared to IGRM, MICE, NOMI and ReMasker, respectively. Regarding MAE, UnIMP-ft demonstrates average performance gains of 8.74%, 36.49%, 30.86% and 35.36% over IGRM, MICE, NOMI, and ReMasker, respectively. These results highlight the excellence of UnIMP and UnIMP-ft in imputing numerical and categorical data.

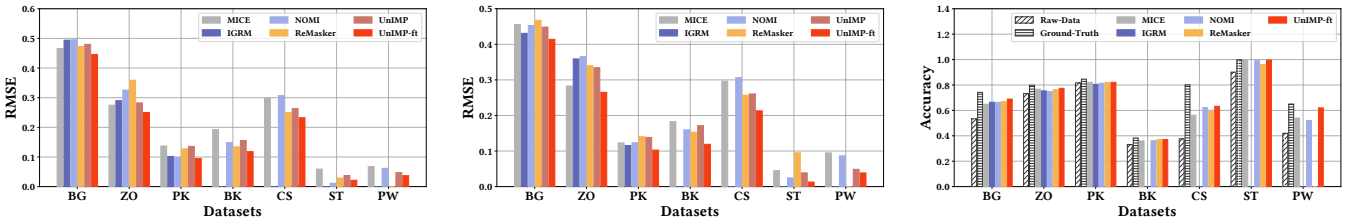
Exp-2: Accuracy over text data. In this experiment, we evaluate the performance of imputing text data, comparing UnIMP with LLM-based methods, as other methods struggle with text data. The results at a 20% missing rate and MCAR are shown in Table 5. Our methods, UnIMP and UnIMP-ft, outperform previous LLM-based methods. Specifically, UnIMPshows a 30.53, 28.04, and 23.47 percentage point (pp) gain in ROUGE-1_{F_1} over DFMs, Table-GPT, and Jellyfish, respectively. For Cos-Sim, it achieves an 8.68 pp, 5.58 pp, and 6.88 pp improvement. UnIMP-ft further boosts UnIMP. These results demonstrate the superiority of UnIMPand UnIMP-ft.

Exp-3: Performance under MAR and MNAR. We evaluate performance under MAR and MNAR, following the missing patterns defined in [40, 57]. Following [40, 57, 68], we model missingness using a logistic model based on randomly selected features, with a 20% missing rate as in Exp-1 and Exp-2. Figure 3 shows the results.

Table 4: Results of missing data imputation (20% MCAR)

Model	RMSE								MAE							
	Blogger	Zoo	Parkinsons	Bike	Chess	Shuttle	Power	Improve%	Blogger	Zoo	Parkinsons	Bike	Chess	Shuttle	Power	Improve%
MEAN	0.4315	0.4260	0.2062	0.2239	0.3079	0.0914	0.0869	45.22%	0.3631	0.3663	0.1473	0.1561	0.2584	0.0467	0.0559	57.24%
KNNI	0.4417	0.2749	0.1891	0.2023	0.3246	0.0456	OOT	35.75%	0.3337	0.1473	0.1207	0.1277	0.2606	0.0192	OOT	41.33%
MICE	0.4134	0.2645	0.1263	0.1796	0.2966	0.0426	0.0650	28.27%	0.3605	0.1731	0.0667	0.1097	0.2453	0.0131	0.0370	36.49%
VGAIN	0.4316	0.4114	0.1913	0.2219	0.2797	0.0786	0.0811	42.48%	0.3643	0.1891	0.1156	0.1363	0.2537	0.0352	0.0509	48.88%
TDM	0.4384	0.2949	0.1862	0.2444	0.3027	0.0769	0.0926	41.48%	0.3229	0.1490	0.0830	0.1499	0.2345	0.0350	0.0600	42.96%
GINN	0.4657	0.2761	0.1466	0.1641	0.2911	0.0734	OOM	32.41%	0.3444	0.1445	0.0884	0.0921	0.2339	0.0434	OOM	36.97%
GRAPE	0.4304	0.3211	0.1064	0.1481	0.2749	0.0242	OOM	20.89%	0.3151	0.1605	0.0535	0.0796	0.2200	0.0073	OOM	21.39%
IGRM	0.4551	0.3063	0.1035	OOM	OOM	OOM	OOM	12.11%	0.3423	0.1621	0.0497	OOM	OOM	OOM	OOM	8.74%
DFMs	0.4413	0.4445	0.2412	0.2483	OOT	OOT	OOT	41.53%	0.3676	0.2934	0.1647	0.1529	OOT	OOT	OOT	49.81%
Table-GPT	0.4237	0.4315	0.2246	0.2547	OOT	OOT	OOT	39.71%	0.3572	0.2713	0.1761	0.1442	OOT	OOT	OOT	48.99%
Jellyfish	0.4133	0.4177	0.2127	0.1935	OOT	OOT	OOT	36.43%	0.3557	0.2719	0.1548	0.1478	OOT	OOT	OOT	47.38%
NOMI	0.4112	0.2576	0.1322	0.1582	0.3042	0.0237	0.0731	28.32%	0.3102	0.1442	0.0710	0.0740	0.2298	0.0071	0.0463	30.86%
ReMasker	0.4068	0.3309	0.1508	0.1277	0.2662	0.1111	OOT	29.61%	0.3293	0.1807	0.0995	0.0655	0.2113	0.0545	OOT	35.36%
UnIMP	0.4171	0.2979	0.1407	0.1730	0.2628	0.0398	0.0485	25.84%	0.3384	0.1822	0.0966	0.1121	0.2050	0.0238	0.0256	36.08%
UnIMP-ft	0.3972	0.2474	0.0990	0.1172	0.2142	0.0134	0.0425	–	0.3082	0.1428	0.0475	0.0602	0.1438	0.0040	0.0225	–

∗ Red text indicates the best result. Blue text indicates the second best result. 'OOT' indicates out of time (with a limit of 10 hours). 'OOM' indicates out of memory.



(a) Results under 20% MAR (b) Results under 20% MNAR (c) Results of downstream classification

Figure 3: Results under different missing mechanisms and results of downstream classification

Table 5: Results of imputation over text data

Model	ROUGE-1_{F_1}			Cos-Sim		
	Buy	Restaurant	Walmart	Buy	Restaurant	Walmart
DFMs	0.1535	0.0822	0.1420	0.8251	0.7609	0.7943
Table-GPT	0.1784	0.1398	0.1344	0.8345	0.8137	0.8254
Jellfish	0.2153	0.1675	0.2067	0.8418	0.8145	0.778
UnIMP	0.3327	0.4017	0.5594	0.8610	0.8774	0.9025
UnIMP-ft	0.4273	0.4326	0.5931	0.8892	0.8923	0.9177

UnIMP-ft achieves superior performance, reducing RMSE under MAR by 27.58%, 10.22%, 4.39%, and 17.86% compared to MICE, IGRM, NOMI, and ReMasker, respectively. Under MNAR, it further reduces RMSE by 30.23%, 10.72%, 27.32%, and 29.57%, highlighting the effectiveness of UnIMP and UnIMP-ft.

Exp-4: Downstream classification accuracy. We evaluate the downstream classification performance on datasets imputed by MICE, IGRM, NOMI, ReMasker and UnIMP-ft. We also include results from the ground-truth dataset (with no missing data) and the raw dataset (where missing values are replaced with 0, following [57]). We employ Random Forest as our classifier. The datasets are split into 80% training and 20% testing sets. Figure 3(c) presents the results. The results indicate that a higher imputation accuracy generally leads to a higher downstream classification, and datasets imputed by our method achieve superior classification results.

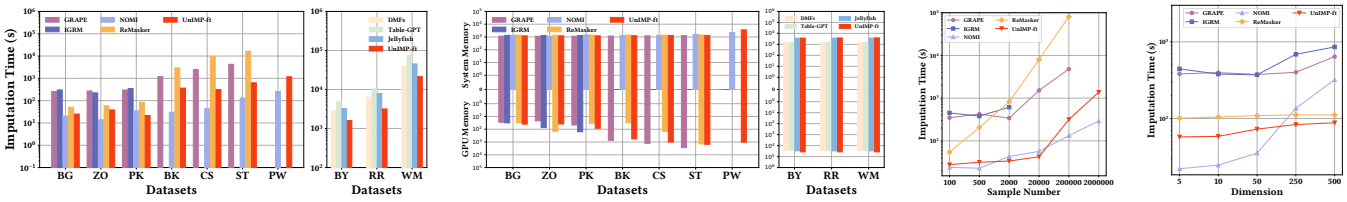
5.3 Efficiency Evaluation

Exp-5: Efficiency and memory consumption. We evaluate imputation efficiency and memory consumption as in Figure 4(a) and (b). We compare UnIMP-ft with graph-based methods (GRAPE, IGRM), LLM-based methods (DFMs, Table-GPT, Jellyfish), and high-accuracy baselines (NOMI, ReMasker). The results demonstrate that UnIMP-ft achieves competitive efficiency. It outperforms graph-based methods with a 3.07× and 5.38× speedup over GRAPE and IGRM, respectively. Additionally, the use of BiHMP as an adapter and fewer tokens for inference improves efficiency compared to other LLM-based methods. Moreover, the results of memory consumption show that graph-based methods require slightly more memory due to graph propagation, while our method demonstrates competitive memory consumption compared to other methods.

Exp-6: Scalability. We evaluate scalability w.r.t. the number of samples and data dimensions. We randomly select 100, 500, 2,000, 20,000, 200,000, and 2,000,000 samples from the Power dataset and generate synthetic datasets with 200 samples and dimensions of 5, 10, 50, 250, and 500. The results in Figure 4(c) and (d) highlight the competitive scalability of UnIMP. For sample scalability, ReMasker has poor scalability as it needs to mask and train each sample, while NOMI scales the slowest. Graph-based methods (GRAPE, IGRM, UnIMP) show similar growth. For dimensional scalability, NOMI has poor scalability due to iterative column-wise enumeration, while ReMasker scales the slowest as it imputes all missing columns simultaneously. Graph-based methods exhibit similar trends.

5.4 Ablation Study

Exp-7: Ablation study. In this experiment, we conduct an ablation study to assess the effectiveness of key components in UnIMP, including hypergraph-aggregated global-local information, the BiHMP



(a) Efficiency evaluation (b) Memory consumption (MB) (c) Scalability vs num (d) Scalability vs dim

Figure 4: Imputation efficiency, memory consumption and scalability evaluation. In the efficiency evaluation, the numerical/categorical data and text data need 20824 seconds and 73221 seconds for pre-training UnIMP, respectively. 45381 seconds are needed for pre-training Table-GPT.

Table 6: Results of the ablation study. Color highlights the percentage point (pp) decline compared to the full model.

	ROUGE-1_{F_1}			Cos-Sim		
Model	Buy	Restaurant	Walmart	Buy	Restaurant	Walmart
UnIMP	0.3327	0.4017	0.5594	0.8610	0.8774	0.9025
w/o Hypergraph	0.2580 (↓7.47pp)	0.3757 (↓2.60pp)	0.4812 (↓7.82pp)	0.8508 (↓1.02pp)	0.8611 (↓1.63pp)	0.8681 (↓3.44pp)
w/o BiHMP	0.3014 (↓3.13pp)	0.3877 (↓1.40pp)	0.5282 (↓3.12pp)	0.8571 (↓0.39pp)	0.8702 (↓0.72pp)	0.8822(↓2.03pp)
w/o Xfusion	0.2867(↓4.60pp)	0.3648(↓3.69pp)	0.5052(↓5.42pp)	0.8517(↓0.93pp)	0.8581(↓1.93pp)	0.8729(↓2.96pp)
w/o P.g. mask	0.3263(↓0.64pp)	0.3908(↓1.09pp)	0.5337(↓2.57pp)	0.8602(↓0.08pp)	0.8654(↓1.20pp)	0.8932(↓0.93pp)

Table 7: RMSE results of inductive generalization

Model	Ionosphere	Breast	Spam	News	Connect	Metro
MICE	0.2487	0.2183	0.0712	0.0237	0.2539	0.1225
IGRM	0.2502	0.2107	0.0571	OOM	OOM	OOM
NOMI	0.2320	0.2115	0.0611	0.0252	0.2473	0.1485
ReMasker	0.2360	0.2058	0.0595	0.0282	OOT	OOT
UnIMP	0.2444	0.2426	0.0626	0.0306	0.2569	0.1308

module, the Xfusion model, and progressive masking. To evaluate hypergraph-aggregated information, we replace it with a zero tensor. For BiHMP, we substitute it with the Set Transformer from [8]. For Xfusion, we replace it with a weighted sum fusion mechanism. To analyze progressive masking, we train the model without masking. The results in Table 6 show that hypergraph information improves ROUGE-1_{F_1} by 5.96 pp and Cos-Sim by 2.03 pp. BiHMP contributes a 2.55 pp increase in ROUGE-1_{F_1} and 1.04 pp in Cos-Sim. Xfusion improves ROUGE-1_{F_1} by 4.57 pp and Cos-Sim by 1.94 pp. Progressive masking results in a 1.43 pp gain in ROUGE-1_{F_1} and 0.74 pp in Cos-Sim. Overall, these components are more important on the Walmart dataset. This is because Walmart is larger and more reliant on Hypergraph, BiHMP and Xfusion to aggregate and fuse information for accurate imputation. Moreover, large datasets have more complex patterns, making progressive masking more critical.

Exp-8: Generalization. In this experiment, we assess the performance on datasets that are not used during pre-training. We use six unseen UCI datasets: Ionosphere, Breast, Spam, News, Connect, and Metro [14]. As shown in Table 7, UnIMP achieves competitive performance despite not being trained on these datasets. Notably, for large datasets like Connect and Metro, its RMSE differs by only ±10% from models trained directly on them.

5.5 Hyper-parameters Analysis

Exp-9: Varying LLMs. We evaluate the performance using several representative LLMs, including LiteLlama [64], TinyLlama, Phi2 [41], Llama2 [53] and Llama3 [15], with the results shown in Figure 5(a). As depicted, more advanced LLM generally leads to better imputation results. We selected Llama2, as it was one of the most outstanding open-source LLMs with extensive resources and support available when conducting experiments.

Exp-10: Chunk sizes. We evaluate RMSE and imputation time with varying chunk sizes, which are critical for scalability and avoiding skewed distributions. The sizes tested are 128, 256, 512, 1024 and 2048. The evaluated datasets are Chess, Shuttle and Power, which are the largest. As in Figure 5(b), RMSE fluctuates with different chunk sizes, while time first decreases and then increases. This may be because very small chunks underutilize the GPU resources, while very large chunks require excessive I/O resources. A chunk size of 512 strikes a good balance between RMSE and time.

Exp-11: Batch sizes. We evaluate RMSE and imputation time with varying batch sizes. The batch sizes tested are 16, 32, 64, 128 and 256. Chess, Shuttle, and Power datasets are used. As illustrated in Figure 5(c), RMSE fluctuates with different batch sizes, while imputation time decreases as batch size increases. A batch size of 64 generally provides a good imputation RMSE and time.

Exp-12: Varying δ. We evaluate the model with different δ, which is used in the loss function in Eqn 7. The values tested are 0.1, 0.5, 1, 4 and 10, and the results are in Figure 5(d). As illustrated, RMSE fluctuates with δ, and $\delta = 1$ generally provides a good RMSE.

Exp-13: Varying ratios of progressive masked data, κ. We evaluate imputation RMSE for different values of κ (0.15, 0.25, 0.35, 0.45), which is used to control the newly masked ratio. The results in Figure 5(e) show that RMSE decreases and then increases, with $\kappa = 0.35$ generally providing the best performance.

Exp-14: Missing rates. In Figure 5(f), we experiment with missing rates ranging from 0.1 to 0.8 on the Power dataset which has the most samples. We compare UnIMP with MICE and NOMI, as they show superior performance on Power as in Exp-1. As illustrated, the RMSE increases with higher missing rates, and UnIMP-ft consistently outperforms the baselines across varying missing rates.

Exp-15: Pre-train epochs. We assess the performance across pre-training epochs, varying from 1000 to 4000 at an interval of 1000 in Figure 5(g). As shown, RMSE typically decreases with more epochs, and 4000 epochs are generally adequate for superior performance.

Exp-16: Fine-tuning epochs. We conduct evaluation across fine-tuning epochs, varying from 500 to 2000 at an interval of 500 in Figure 5(h). As shown, RMSE generally decreases with more epochs, and 1000 epochs is generally adequate for an excellent performance.

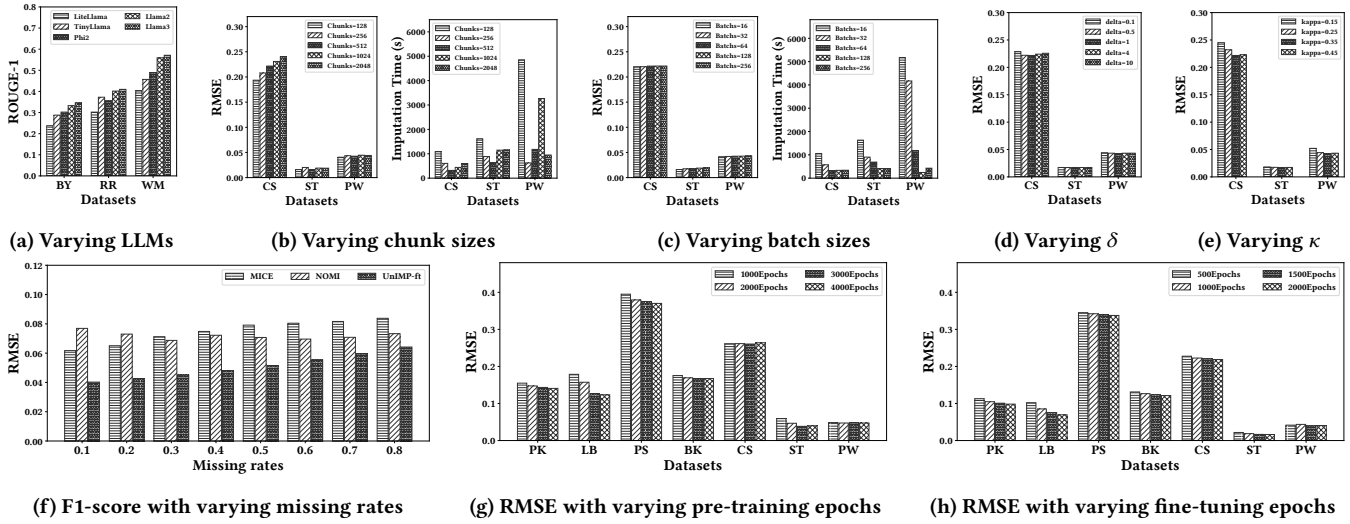


Figure 5: Results of hyper-parameter analysis

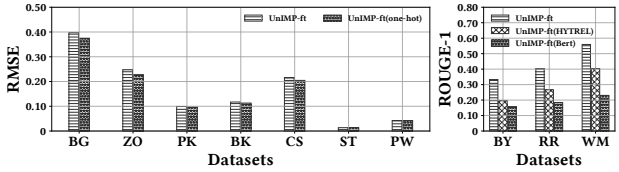


Figure 6: Encoding methods evaluations.

Exp-17: Encoding methods. We evaluate other feature encoding methods, including one-hot for categorical data and HYTREL [8] and BERT [11] for text, in Figure 6. The results indicate that one-hot generally performs better, while HYTREL and BERT perform less effectively, likely because their encoded features are not well aligned with the projection head. Our default setting follows previous works [13, 58] and uses label encoding for categorical data.

6 RELATED WORK

Statistics-based and similarity-based imputation. Statistics-based algorithms impute missing data using statistical values (e.g., mean or mode) for each feature [18, 42]. Similarity-based algorithms impute missing data using closely related data. KNNI [3] selects K nearest neighbors of the sample with missing features for imputation. An iterative method proposed in [73] locates a more reliable local context. NOMI [57] uses similarity to iteratively augment the input and employs an uncertainty-driven network for imputation.

Graph-based models for imputation. Graph structures [32] have been employed to capture complex relationships for data imputation. One popular approach involves using similarity graphs, as demonstrated in GINN [50] and GEDI [7], where edges are constructed between samples to facilitate imputation. Another popular approach leverages bipartite graphs, as seen in GRAPE [69] and IGRM [75], where each row and column is represented as a node, and the cell values are modeled as edges in the graph.

Learning-based imputation. Various AI techniques are employed for imputation. Traditional methods include models like XGBoost [9], random forests in MissForest [51], multi-layer perceptrons [20], and linear regression in MICE [48]. Generative models are also widely employed, with autoencoders being particularly popular, such as deep denoising autoencoders in MIDAE [21] and ReMasker [13], variational autoencoders [38], and importance-weighted autoencoders in MIWAE [37]. GANs are another commonly used backbone, as seen in GAIN [68] and VGAIN [40]. Recently, distribution-matching-based methods are introduced. They align missing values with observed data distributions in the original space (e.g., OTImputer [43]) or in the latent space (e.g., TDM [74]).

LLMs-based imputation. Recently, LLMs have shown a remarkable ability to understand and generate human-like text across a diverse array of tasks. Two main approaches have emerged: in-context learning and fine-tuning. In the in-context-learning-based approach, LLMs are prompted to predict missing data directly with task-related examples, as demonstrated in [45, 46]. Fine-tune-based methods, such as Table-GPT [31] and Jellyfish [71], fine-tune the LLMs to adapt the model to tasks in the field of tabular data.

7 CONCLUSION

We study mixed-type data imputation and propose UnIMP, an LLM-enhanced framework with high-order message passing. We introduce a cell-oriented hypergraph and design BiHMP to aggregate global-local information while capturing column patterns. Xfusion aligns BiHMP with LLMs, serving as an adapter. To train UnIMP, we employ chunking and progressive masking in a pre-training and fine-tuning strategy. Both theoretical and empirical results on 10 real-world datasets demonstrate the superiority of UnIMP.

ACKNOWLEDGMENTS

Kai Wang and Ying Zhang are the joint corresponding authors. Kai Wang is supported by NSFC 62302294. Ying Zhang is supported by the Major Project on Innovation of Research Methods of Zhejiang Philosophy and Social Sciences Laboratory (25SYS06ZD). Wenjie Zhang is supported by ARC DP230101445 and ARC FT210100303.

REFERENCES

[1] 2024. Mastering Mixed Methods Research in 2024: Combining Qualitative and Quantitative Approaches. https://editverse.com/mastering-mixed-methods-research-in-2024/

[2] Nikolich Alexandr, Osliakova Irina, Kudinova Tatyana, Kappusheva Inessa, and Puchkova Arina. 2021. Fine-tuning GPT-3 for Russian text summarization. In *Data Science and Intelligent Systems: Proceedings of 5th Computational Methods in Systems and Software 2021, Vol. 2*. Springer, 748–757.

[3] Naomi S Altman. 1992. An introduction to kernel and nearest-neighbor nonparametric regression. *The American Statistician* 46, 3 (1992), 175–185.

[4] Martin Berglund and Brink van der Merwe. 2023. Formalizing BPE Tokenization. *arXiv preprint arXiv:2309.08715* (2023).

[5] Fabian Biester, Mohamed Abdelaal, and Daniel Del Gaudio. 2024. Llmclean: Context-aware tabular data cleaning via llm-generated ofds. In *European Conference on Advances in Databases and Information Systems*. Springer, 68–78.

[6] Tom B Brown. 2020. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165* (2020).

[7] Katrina Chen, Xiuqin Liang, Zheng Ma, and Zhibin Zhang. 2023. GEDI: A graph-based end-to-end data imputation framework. In *2023 IEEE 35th International Conference on Tools with Artificial Intelligence (ICTAI)*. IEEE, 723–730.

[8] Pei Chen, Soumajyoti Sarkar, Leonard Lausen, Balasubramaniam Srinivasan, Sheng Zha, Ruihong Huang, and George Karypis. 2024. HYTREL: Hypergraph-enhanced tabular data representation learning. *Advances in Neural Information Processing Systems* 36 (2024).

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

[10] Pieter-Tjerk De Boer, Dirk P Kroese, Shie Mannor, and Reuven Y Rubinstein. 2005. A tutorial on the cross-entropy method. *Annals of operations research* 134 (2005), 19–67.

[11] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. 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)*. 4171–4186.

[12] Kounianhua Du, Weinan Zhang, Ruiwen Zhou, Yangkun Wang, Xilong Zhao, Jiarui Jin, Quan Gan, Zheng Zhang, and David P Wipf. 2022. Learning enhanced representation for tabular data via neighborhood propagation. *Advances in Neural Information Processing Systems* 35 (2022), 16373–16384.

[13] Tianyu Du, Luca Melis, and Ting Wang. 2024. ReMasker: Imputing Tabular Data with Masked Autoencoding. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net. https://openreview.net/forum?id=KI9NqjLVDT

[14] Dheeru Dua and Casey Graff. 2017. UCI Machine Learning Repository. http://archive.ics.uci.edu/ml

[15] Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783* (2024).

[16] Jianwei Wang et al. 2025. UnIMP: Full Version. https://github.com/guaiyoui/UnIMP/blob/master/full_version/UnIMP_Full_Version.pdf. Accessed April 2025.

[17] Xi Fang, Weijie Xu, Fiona Anting Tan, Ziqing Hu, Jiani Zhang, Yanjun Qi, Srinivasan H Sengamedu, and Christos Faloutsos. [n.d.]. Large Language Models (LLMs) on Tabular Data: Prediction, Generation, and Understanding-A Survey. *Transactions on Machine Learning Research* ([n. d.]).

[18] Alireza Farhangfar, Lukasz A Kurgan, and Witold Pedrycz. 2007. A novel framework for imputation of missing values in databases. *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans* 37, 5 (2007), 692–709.

[19] Yifan Feng, Haoxuan You, Zizhao Zhang, Rongrong Ji, and Yue Gao. 2019. Hypergraph neural networks. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 33. 3558–3565.

[20] Pedro J García-Laencina, José-Luis Sancho-Gómez, and Aníbal R Figueiras-Vidal. 2010. Pattern classification with missing data: a review. *Neural Computing and Applications* 19, 2 (2010), 263–282.

[21] Lovedeep Gondara and Ke Wang. 2018. Mida: Multiple imputation using denoising autoencoders. In *Advances in Knowledge Discovery and Data Mining: 22nd Pacific-Asia Conference, PAKDD 2018, Melbourne, VIC, Australia, June 3-6, 2018, Proceedings, Part III 22*. Springer, 260–272.

[22] John T Hancock and Taghi M Khoshgoftaar. 2020. Survey on categorical data for neural networks. *Journal of big data* 7, 1 (2020), 28.

[23] Ruidan He, Linlin Liu, Hai Ye, Qingyu Tan, Bosheng Ding, Liying Cheng, Jiawei Low, Lidong Bing, and Luo Si. 2021. On the Effectiveness of Adapter-based Tuning for Pretrained Language Model Adaptation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. 2208–2222.

[24] Peter J Huber. 1992. Robust estimation of a location parameter. In *Breakthroughs in statistics: Methodology and distribution*. Springer, 492–518.

[25] Mortaza Jamshidian and Matthew Mata. 2007. Advances in analysis of mean and covariance structure when data are incomplete. In *Handbook of latent variable and related models*. Elsevier, 21–44.

[26] Alistair EW Johnson, Tom J Pollard, Lu Shen, Li-wei H Lehman, Mengling Feng, Mohammad Ghassemi, Benjamin Moody, Peter Szolovits, Leo Anthony Celi, and Roger G Mark. 2016. MIMIC-III, a freely accessible critical care database. *Scientific data* 3, 1 (2016), 1–9.

[27] Kaggle. 2021. Flipkart Products. https://www.kaggle.com/datasets/PromptCloudHQ/flipkart-products/data

[28] Kenji Kawaguchi, Zhun Deng, Xu Ji, and Jiaoyang Huang. 2023. How does information bottleneck help deep learning?. In *International Conference on Machine Learning*. PMLR, 16049–16096.

[29] T Kudo. 2018. Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. *arXiv preprint arXiv:1808.06226* (2018).

[30] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems* 33 (2020), 9459–9474.

[31] Peng Li, Yeye He, Dror Yashar, Weiwei Cui, Song Ge, Haidong Zhang, Danielle Rifinski Fainman, Dongmei Zhang, and Surajit Chaudhuri. 2024. Table-GPT: Table Fine-tuned GPT for Diverse Table Tasks. *Proceedings of the ACM on Management of Data* 2, 3 (2024), 1–28.

[32] Shunyang Li, Kai Wang, Xuemin Lin, Wenjie Zhang, Yizhang He, and Long Yuan. 2024. Querying Historical Cohesive Subgraphs over Temporal Bipartite Graphs. In *2024 IEEE 40th International Conference on Data Engineering (ICDE)*. IEEE, 2503–2516.

[33] Shunyang Li, Kai Wang, Wenjie Zhang, Xuemin Lin, and Yizhang He. 2025. Efficient Bitruss Decomposition on GPU. *IEEE Transactions on Knowledge and Data Engineering* (2025).

[34] Zongxi Li, Xianming Li, Yuzhang Liu, Haoran Xie, Jing Li, Fu-lee Wang, Qing Li, and Xiaoqin Zhong. 2023. Label supervised llama finetuning. *arXiv preprint arXiv:2310.01208* (2023).

[35] Wei-Chao Lin and Chih-Fong Tsai. 2020. Missing value imputation: a review and analysis of the literature (2006–2017). *Artificial Intelligence Review* 53 (2020), 1487–1509.

[36] Somayya Madakam, Ramya Ramaswamy, and Siddharth Tripathi. 2015. Internet of Things (IoT): A literature review. *Journal of Computer and Communications* 3, 5 (2015), 164–173.

[37] Pierre-Alexandre Mattei and Jes Frellsen. 2019. MIWAE: Deep generative modelling and imputation of incomplete data sets. In *International conference on machine learning*. PMLR, 4413–4423.

[38] John T McCoy, Steve Kroon, and Lidia Auret. 2018. Variational autoencoders for missing data imputation with application to a simulated milling circuit. *IFAC-PapersOnLine* 51, 21 (2018), 141–146.

[39] Yinan Mei, Shaoxu Song, Chenguang Fang, Haifeng Yang, Jingyun Fang, and Jiang Long. 2021. Capturing semantics for imputation with pre-trained language models. In *2021 IEEE 37th International Conference on Data Engineering (ICDE)*. IEEE, 61–72.

[40] Xiaoye Miao, Yangyang Wu, Lu Chen, Yunjun Gao, and Jianwei Yin. 2022. An experimental survey of missing data imputation algorithms. *IEEE Transactions on Knowledge and Data Engineering* 35, 7 (2022), 6630–6650.

[41] Microsoft. 2024. microsoft/phi-2. https://huggingface.co/microsoft/phi-2

[42] Karel GM Moons, Rogier ART Donders, Theo Stijnen, and Frank E Harrell Jr. 2006. Using the outcome for imputation of missing predictor values was preferred. *Journal of clinical epidemiology* 59, 10 (2006), 1092–1101.

[43] Boris Muzellec, Julie Josse, Claire Boyer, and Marco Cuturi. 2020. Missing data imputation using optimal transport. In *International Conference on Machine Learning*. PMLR, 7130–7140.

[44] Ramesh Nallapati, Bowen Zhou, Cícero Nogueira dos Santos, Çaglar Gülçehre, and Bing Xiang. 2016. Abstractive Text Summarization using Sequence-to-sequence RNNs and Beyond. In *Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning, CoNLL 2016, Berlin, Germany, August 11-12, 2016*, Yoav Goldberg and Stefan Riezler (Eds.). ACL, 280–290. https://doi.org/10.18653/V1/K16-1028

[45] Avanika Narayan, Ines Chami, Laurel Orr, and Christopher Ré. 2022. Can Foundation Models Wrangle Your Data? *Proceedings of the VLDB Endowment* 16, 4 (2022), 738–746.

[46] Yichen Qian, Yongyi He, Rong Zhu, Jintao Huang, Zhijian Ma, Haibin Wang, Yaohua Wang, Xiuyu Sun, Defu Lian, Bolin Ding, et al. 2024. UniDM: A Unified Framework for Data Manipulation with Large Language Models. *Proceedings of Machine Learning and Systems* 6 (2024), 465–482.

[47] Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. 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 2019, Hong Kong, China, November 3-7, 2019*, Kentaro Inui, Jing Jiang, Vincent Ng, and Xiaojun Wan (Eds.). Association for Computational Linguistics, 3980–3990. https://doi.org/10.18653/V1/D19-1410

[48] Patrick Royston and Ian R White. 2011. Multiple imputation by chained equations (MICE): implementation in Stata. *Journal of statistical software* 45 (2011), 1–20.

[49] Rajat Singh and Srikanta Bedathur. 2023. Embeddings for Tabular Data: A Survey. *arXiv preprint arXiv:2302.11777* (2023).

[50] Indro Spinelli, Simone Scardapane, and Aurelio Uncini. 2020. Missing data imputation with adversarially-trained graph convolutional networks. *Neural Networks* 129 (2020), 249–260.

[51] Daniel J Stekhoven and Peter Bühlmann. 2012. MissForest—non-parametric missing value imputation for mixed-type data. *Bioinformatics* 28, 1 (2012), 112–118.

[52] Bohan Tang, Zexi Liu, Keyue Jiang, Siheng Chen, and Xiaowen Dong. 2024. Simplifying Hypergraph Neural Networks. https://api.semanticscholar.org/CorpusID:267547514

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

[54] A Vaswani. 2017. Attention is all you need. *Advances in Neural Information Processing Systems* (2017).

[55] Jianwei Wang, Kai Wang, Xuemin Lin, Wenjie Zhang, and Ying Zhang. 2024. Efficient Unsupervised Community Search with Pre-trained Graph Transformer. *Proc. VLDB Endow.* 17, 9 (2024), 2227–2240. https://doi.org/10.14778/3665844.3665853

[56] Jianwei Wang, Kai Wang, Xuemin Lin, Wenjie Zhang, and Ying Zhang. 2024. Neural Attributed Community Search at Billion Scale. *Proceedings of the ACM on Management of Data* 1, 4 (2024), 1–25.

[57] Jianwei Wang, Ying Zhang, Kai Wang, Xuemin Lin, and Wenjie Zhang. 2024. Missing Data Imputation with Uncertainty-Driven Network. *Proceedings of the ACM on Management of Data* 2, 3 (2024), 1–25.

[58] Wei Wang, Yimeng Chai, and Yue Li. 2022. A Global to Local Guiding Network for Missing Data Imputation. In *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 4058–4062.

[59] Sharada Prasad Wasti, Padam Simkhada, Edwin R van Teijlingen, Brijesh Sathian, and Indrajit Banerjee. 2022. The growing importance of mixed-methods research in health. *Nepal journal of epidemiology* 12, 1 (2022), 1175.

[60] Ian R White, Patrick Royston, and Angela M Wood. 2011. Multiple imputation using chained equations: issues and guidance for practice. *Statistics in medicine* 30, 4 (2011), 377–399.

[61] Wikipedia contributors. 2024. Information bottleneck method — Wikipedia, The Free Encyclopedia. https://en.wikipedia.org/w/index.php?title=Information_bottleneck_method&oldid=1260066467 [Online; accessed 8-December-2024].

[62] Yangyang Wu, Xiaoye Miao, Zi-ang Nan, Jinshan Zhang, Jianhu He, and Jianwei Yin. 2024. Effective and Efficient Multi-View Imputation With Optimal Transport. *IEEE Trans. Knowl. Data Eng.* 36, 11 (2024), 6029–6041. https://doi.org/10.1109/TKDE.2024.3387439

[63] Yangyang Wu, Jun Wang, Xiaoye Miao, Wenjia Wang, and Jianwei Yin. 2024. Differentiable and Scalable Generative Adversarial Models for Data Imputation. *IEEE Trans. Knowl. Data Eng.* 36, 2 (2024), 490–503. https://doi.org/10.1109/TKDE.2023.3293129

[64] Xiaotian Han, Xia Ben Hu. 2024. ahxt/LiteLlama-460M-1T. https://huggingface.co/ahxt/LiteLlama-460M-1T

[65] Shuangyin Xie. 2019. *Sentiment Analysis using machine learning algorithms: online women clothing reviews.* Ph.D. Dissertation. Dublin, National College of Ireland.

[66] Yang Yang, Yizhou Sun, Jie Tang, Bo Ma, and Juanzi Li. 2015. Entity matching across heterogeneous sources. In *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*. 1395–1404.

[67] Antonio Jimeno Yepes, Yao You, Jan Milczek, Sebastian Laverde, and Leah Li. 2024. Financial Report Chunking for Effective Retrieval Augmented Generation. *arXiv preprint arXiv:2402.05131* (2024).

[68] Jinsung Yoon, James Jordon, and Mihaela Schaar. 2018. Gain: Missing data imputation using generative adversarial nets. In *International conference on machine learning*. PMLR, 5689–5698.

[69] Jiaxuan You, Xiaobai Ma, Yi Ding, Mykel J Kochenderfer, and Jure Leskovec. 2020. Handling missing data with graph representation learning. *Advances in Neural Information Processing Systems* 33 (2020), 19075–19087.

[70] Haochen Zhang, Yuyang Dong, Chuan Xiao, and Masafumi Oyamada. 2023. Large language models as data preprocessors. *arXiv preprint arXiv:2308.16361* (2023).

[71] Haochen Zhang, Yuyang Dong, Chuan Xiao, and Masafumi Oyamada. 2024. Jellyfish: Instruction-Tuning Local Large Language Models for Data Preprocessing. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*. 8754–8782.

[72] Renrui Zhang, Jiaming Han, Chris Liu, Peng Gao, Aojun Zhou, Xiangfei Hu, Shilin Yan, Pan Lu, Hongsheng Li, and Yu Qiao. 2023. Llama-adapter: Efficient fine-tuning of language models with zero-init attention. *arXiv preprint arXiv:2303.16199* (2023).

[73] Shichao Zhang. 2012. Nearest neighbor selection for iteratively kNN imputation. *Journal of Systems and Software* 85, 11 (2012), 2541–2552.

[74] He Zhao, Ke Sun, Amir Dezfouli, and Edwin V Bonilla. 2023. Transformed distribution matching for missing value imputation. In *International Conference on Machine Learning*. PMLR, 42159–42186.

[75] Jiajun Zhong, Ning Gui, and Weiwei Ye. 2023. Data imputation with iterative graph reconstruction. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 37. 11399–11407.