




# DeepVIS: Bridging Natural Language and Data Visualization Through Step-wise Reasoning

Zhihao Shuai , Boyan Li , Siyu Yan , Yuyu Luo , and Weikai Yang 

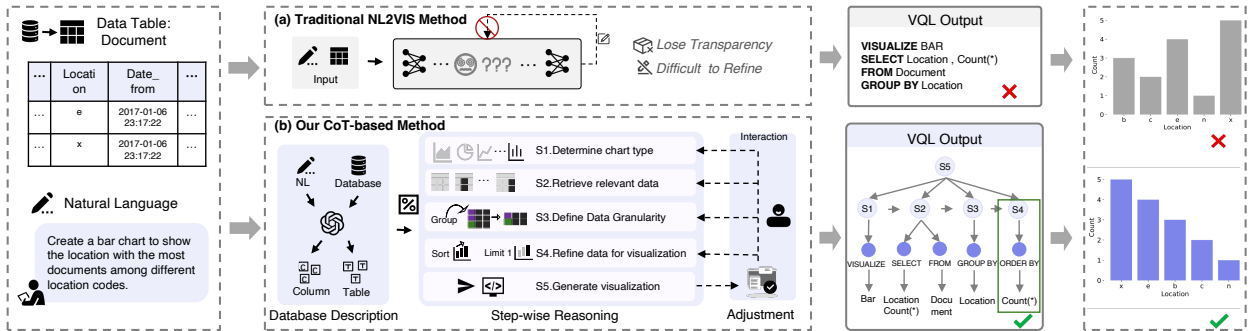


Fig. 1: Comparing traditional NL2VIS methods with our method: (a) Traditional methods often function as black boxes, making it difficult for users to interpret design rationales or refine suboptimal results. (b) Our method leverages CoT reasoning, which enhances both model performance and transparency, enabling users to better understand the model decision and interactively refine results.

**Abstract**—Although data visualization is powerful for revealing patterns and communicating insights, creating effective visualizations requires familiarity with authoring tools and often disrupts the analysis flow. While large language models show promise for automatically converting analysis intent into visualizations, existing methods function as black boxes without transparent reasoning processes, which prevents users from understanding design rationales and refining suboptimal outputs. To bridge this gap, we propose integrating Chain-of-Thought (CoT) reasoning into the Natural Language to Visualization (NL2VIS) pipeline. First, we design a comprehensive CoT reasoning process for NL2VIS and develop an automatic pipeline to equip existing datasets with structured reasoning steps. Second, we introduce *nvBench-CoT*, a specialized dataset capturing detailed step-by-step reasoning from ambiguous natural language descriptions to finalized visualizations, which enables state-of-the-art performance when used for model fine-tuning. Third, we develop *DeepVIS*, an interactive visual interface that tightly integrates with the CoT reasoning process, allowing users to inspect reasoning steps, identify errors, and make targeted adjustments to improve visualization outcomes. Quantitative benchmark evaluations, two use cases, and a user study collectively demonstrate that our CoT framework effectively enhances NL2VIS quality while providing insightful reasoning steps to users.

**Index Terms**—Data visualization, automatic visualization, large language models

## 1 INTRODUCTION

Data visualization serves as a powerful tool in the data analysis pipeline, enabling effective exploration, pattern recognition, and the communication of insights [21, 25, 27, 31, 68]. Despite its critical importance, creating high-quality visualizations remains a challenging task that typically requires both specialized knowledge of visualization principles and proficiency with complex authoring tools [30, 38, 46]. This technical expertise requirement presents a barrier for many data analysts, particularly those without formal training in visualization design [58]. Even for experienced analysts, the process of switching between analysis environments and authoring tools forces them to mentally translate their analysis intent into the specific commands or parameters required by authoring tools, which disrupts analysis flow and negatively affects their productivity [59].

- Zhihao Shuai, Boyan Li, Siyu Yan, Yuyu Luo, Weikai Yang are with the Hong Kong University of Science and Technology (Guangzhou), Guangzhou, China. Yuyu Luo and Weikai Yang are also with the Hong Kong University of Science and Technology, Hong Kong SAR, China. Zhihao Shuai and Boyan Li are joint first authors. Weikai Yang is the corresponding author. E-mail: {zhihaoshuai@, bli303@connect., syan195@connect., yuyuluo@, weikaiyang@}hkust-gz.edu.cn

Manuscript received xx xxx. 201x; accepted xx xxx. 201x. Date of Publication xx xxx. 201x; date of current version xx xxx. 201x. For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org. Digital Object Identifier: xx.xxx/TVCG.201x.xxxxxxx

Given these challenges, there is a compelling need for systems that can automatically transform natural language descriptions of analysis intent into appropriate visualizations. Such systems would enable analysts to remain focused on their primary task of data exploration and insight discovery, rather than becoming distracted by the mechanism of visualization creation. Recent advancements in large language models (LLMs) have demonstrated considerable promise for the Natural Language to Visualization (NL2VIS) task, exhibiting remarkable capabilities to interpret user intent and generate corresponding visualization specifications [23, 58]. Such systems would enable analysts to remain focused on their primary task of data exploration and insight discovery, rather than becoming distracted by the mechanics of visualization creation [38]. However, despite their potential, current methods function largely as black boxes, processing natural language inputs and producing visualization outputs without exposing the intermediate reasoning steps (Fig. 1(a)). This opacity creates several critical limitations: First, users cannot understand how or why specific visualization choices were made, which reduces their trust in the model outputs. Second, when faced with suboptimal outputs, users lack visibility into specific points of failure within the reasoning process, which hinders their ability to pinpoint and rectify problems. Third, users miss the valuable chance to learn from the decision-making process of the model, which could enhance their own visualization expertise.

To address these limitations, we propose integrating Chain-of-Thought (CoT) reasoning into the NL2VIS pipeline (Fig. 1(b)), which encourages LLMs to break down complex problems into intermediate steps [4, 53]. First, we design a comprehensive CoT reasoning process for the NL2VIS task and develop an automatic pipeline to equip ex-

isting NL2VIS datasets with structured CoT reasoning steps. Based on this, we introduce a new dataset, *nvBench-CoT*, which captures the detailed step-by-step reasoning processes that connect ambiguous natural language descriptions to finalized visualizations, mimicking the design process of experienced analysts.

We then train an NL2VIS model on this dataset that explicitly incorporates CoT reasoning. Unlike existing black-box methods, our model exposes the intermediate reasoning steps that guide the transformation from natural language input to visualization output. This transparency not only enables the model to achieve state-of-the-art performance but also enables users to understand the rationale underlying visualization choices and identify potential areas for improvement. We also develop DeepVIS, an interactive visual interface that tightly integrates with the CoT reasoning process. This interface empowers users to inspect each step of the reasoning chain, identify potential errors or suboptimal decisions, and strategically intervene by modifying specific reasoning components to enhance the final visualization. By facilitating this form of human-AI collaboration, our system effectively combines the efficiency of automated visualization generation with the judgment of human analysts.

We evaluate our method through quantitative benchmark evaluation, two use cases, and a user study. The results demonstrate that our CoT framework significantly improves the quality and accuracy of NL2VIS transformations compared to black-box methods. Furthermore, user feedback indicates that the transparent reasoning process substantially enhances trust in the model outputs while providing valuable learning opportunities that enable users to refine their own visualization expertise. To summarize, our contributions are threefold:

- We design a comprehensive CoT reasoning process for the NL2VIS task, which is used to guide the process of enhancing NL2VIS datasets with structured reasoning steps.
- We introduce and curate *nvBench-CoT*, an NL2VIS dataset with detailed reasoning steps, and achieve state-of-the-art performance by fine-tuning models on this dataset.
- We develop a visual interface that tightly integrates with the CoT reasoning process, which allows users to understand the underlying reasoning and make targeted adjustments.

## 2 RELATED WORK

### 2.1 Chain-of-Thought (CoT)

While LLMs have demonstrated impressive performance, they often face challenges when processing complex problems. The CoT technique addresses this by guiding LLMs to decompose complex problems into a series of intermediate steps, thereby improving reasoning and transparency [11, 53]. The effectiveness of CoT is significantly influenced by the design of these intermediate steps. Therefore, several strategies have been developed to generate high-quality reasoning steps, such as zero-shot CoT that instructs the model to generate reasoning steps [20, 22], few-shot CoT that provides the model with a few examples of similar problems with reasoning steps [45, 53], Tree-of-Thought that enables exploration of multiple reasoning paths and dynamic adjustment through backtracking [64], and interactive construction and involve human in crafting the chain [56, 57]. For a comprehensive overview, we refer readers to recent surveys [4, 8]. In this work, we systematically analyze NL2VIS and design a tailored CoT reasoning pipeline, which effectively guides LLMs through the complex stages of visualization generation and improves performance.

### 2.2 NL2VIS

Based on their underlying principles and implementation mechanisms, existing NL2VIS methods can be classified into three categories: Rule-based methods, translation-based methods, and LLM-based methods.

The early efforts in NL2VIS are rule-based methods, which utilize predefined rules to analyze natural language queries and transform them into a set of predefined visualization templates [7, 13, 28, 34, 39, 42, 44, 65]. Articulate [44] is one of the pioneering works, which utilizes a parser to tag each word and then classify the whole query into different analysis tasks. A suitable chart is then generated based on the analysis

tasks and the data to be visualized. Later efforts focus on improving performance in handling ambiguity in natural language input. For example, NL4DV [34] simultaneously considers syntactic and semantic similarity to identify data attributes referenced in the query. While these methods provided an initial framework for NL2VIS, they were often constrained in their ability to handle complex or unforeseen queries and required substantial manual effort in defining the rules and templates.

With the advancement of machine learning translation, translation-based methods are proposed to solve NL2VIS by treating it as a translation problem between human languages and visualization languages [26, 29, 30]. Specifically, they often utilize sequence-to-sequence models (e.g., recurrent neural network or transformer) to encode the natural language query into a hidden representation and then decode it into a visualization specification. A representative work in this category is *ncNet* [30], which is a Transformer-based model incorporating visualization-aware optimizations to enhance the translation process and the quality of the generated visualizations. To better address the insufficient accuracy of from-scratch generation methods, RGVisNet [41] leverages a hybrid retrieval-generation method to enhance results by retrieving the most relevant queries from existing data. While these methods offer greater flexibility and the ability to learn visualization-specific knowledge, they sometimes still struggle in understanding complex and ambiguous natural language queries.

The recent surge in the capabilities of LLMs has led to their widespread adoption in NL2VIS. To comprehensively assess the capabilities and limitations of LLMs, researchers have conducted extensive evaluation [3, 16, 19, 35, 37, 47]. For example, Vazquez *et al.* [47] conducted systematic experiments analyzing LLM performance in NL2VIS across different aspects, including chart generation, library adaptation, and visual variable configuration. Chen *et al.* [3] constructed VisEval, a comprehensive NL2VIS benchmark that evaluates multiple LLMs to identify common challenges and provide insights for future research directions. Beyond evaluation, researchers have also explored methods to better harness the strong language understanding and generation abilities of LLMs through techniques like in-context learning [6, 18, 23, 32] and supervised fine-tuning [36, 46, 61]. For example, LLM4VIS [17] employs a few-shot approach to guide the model in recommending appropriate visualization types for the test data, while Prompt4VIS [23] retrieves similar questions and the groundtruth answers as input to enhance model performance. To further help models make correct reasoning, recent work like ChartGPT [46] and V-RECS [36] decompose the visualization generation process into a series of sub-tasks that the LLM addresses sequentially. In contrast to existing approaches, our method provides a detailed reasoning process for each step, which not only effectively boosts model performance but also enhances transparency.

### 2.3 Visualization for Human-LLM Collaboration

Visualization has been proven to be an effective way to help users harness the power of LLMs and achieve different tasks [24, 48, 50, 62, 63, 66]. Based on the purpose of the visualization, existing work can be classified into two categories: visualizations for enhancing LLM inputs and visualizations for enhancing LLM outputs.

A significant body of work has explored how visualization aids users in crafting effective prompts and achieving better performance. For example, Strobelt *et al.* [43] proposed PromptIDE, a tool designed to assist users in constructing prompts for text classification tasks. It allows users to construct multiple prompt variations, compare their performance, and iteratively refine them based on quantitative feedback. For complicated tasks that require complex prompts to unlock the potential of LLMs, PromptChainer [56] and AI Chains [57] help users decompose complicated into smaller, more manageable sub-tasks, thereby simplifying the creation of prompts. In the domain of text-to-image generation, PromptMagician [12] supports users to efficiently explore and compare the prompts and corresponding generated images retrieved from a database. This visual exploration offers valuable guidance and hints for refining user prompts. PromptCharm [52] focuses on iteratively improving generated images through multimodal prompting by allowing users to adjust the attention given to specific keywords within the prompt. PromptMap [1] introduces an intuitive spatial interface for im-

age generation, allowing users to manipulate prompts via interactive 2D layouts (e.g., grids or graphs) instead of text. For multimodal reasoning tasks, POEM [15] visualizes interaction patterns across modalities at different granularities. This multi-level model understanding empowers users to refine prompts in a more interpretable and controllable manner.

Another prominent area of research focuses on facilitating human-LLM collaboration by enabling users to understand and steer the output of LLMs. For example, InsightLens [54] provides a structured and accessible way for users to record, organize, and revisit these insights, enhancing the overall efficiency and value of LLM-powered data analysis workflows. Patchview [10] uses a tangible visual metaphor that enables writers to intuitively guide the LLM in generating story world elements, fostering a more natural and expressive co-creation experience. Tale-Brush [9] employs line-sketching interactions along with a GPT-based language model to support writers in dictating character fortune plots in line with the creative goals of the writers. The most relevant one is WaitGPT [60], which facilitates programmers to understand and verify the code generated by LLMs for data analysis tasks. It transforms the LLM-generated code into an interactive node-link diagram that updates in real time, visually representing data operations and their intermediate states. Users can efficiently monitor the analysis process, understand the logic behind the generated code, and even modify specific operations directly within the visual interface. Our work aligns with the second category, focusing on making the reasoning steps of LLMs more transparent and controllable. However, unlike WaitGPT, which transforms generated code into code in a post-analysis manner, we first design a comprehensive CoT reasoning process tailored for NL2VIS, providing a step-by-step explanation of how natural language descriptions are translated into finalized visualizations. This structured process not only boosts performance in NL2VIS but also provides better transparency.

### 3 DESIGN OF THE CoT PROCESS FOR NL2VIS

The key to incorporating the CoT process into the NL2VIS pipeline lies in understanding how analysts design appropriate visualizations to fulfill their analysis intent. To develop a CoT process that effectively captures visualization design reasoning, we conducted a comprehensive literature review and expert interviews, resulting in a structured five-stage CoT process that is suitable for NL2VIS tasks.

#### 3.1 Literature Review and Expert Interviews

We began by conducting a comprehensive review of existing literature, focusing on frameworks that describe the systematic progression from analysis intent to visualizations [33, 51, 55] and NL2VIS [40, 46]. The nested model proposed by Munzner [33] was particularly influential, which emphasizes progression from domain problem characterization to data abstraction and visual encoding. In addition, prior research [46] has demonstrated decoupling the NL2VIS task from aspects such as data transformation and visualization notably boosts visualization accuracy. To complement our theoretical analysis, we conducted semi-structured interviews with four visualization experts (E1-E4) to understand their real-world visualization authoring workflows. E1 is a senior researcher with 8-year experience in data visualization; E2 is a data analyst familiar with multiple visualization tools; E3 and E4 are Ph.D. students with 4 and 3 years visualization experience, respectively. None of them are co-authors of this work. Using a think-aloud protocol, experts verbalized their thought processes while designing appropriate visualizations. Each interview lasted between 40 and 55 minutes.

#### 3.2 Initial Three-Stage Process

Based on our literature review and expert interviews, we initially identified three key stages in the visualization authoring process:

**Understand analysis intent and determine chart type.** In this initial stage, analysts first understand the visualization goal and select an appropriate chart type. As E1 stated, “I will first quickly examine a few exemplar data and understand the visualization goal, then I can choose the most suitable visualization type to fulfill my analysis need.” E4 further elaborated, “Even if the exact data processing is not immediately clear, I usually start by determining the chart type and identifying the primary axes. It will provide a clear picture of what the visualization

might reveal.” These insights align with findings from Wang *et al.* [51], which noted that analysts preferred a top-down method, starting with determining the chart type and then specifying other configuration details.

**Prepare data for visualization.** After determining the chart type, all experts agreed they would start processing raw data to suit the specific requirements of that chart. As E3 explained, “Once I know I am creating a bar chart to compare categories, I need to make sure my data is aggregated correctly, since there is no point showing individual data points and only the summary statistics matter.” This data processing stage usually involves filtering the data to focus on relevant subsets, aggregating data to summarize key trends or patterns, and transforming the data by calculating new fields. This acts as a crucial bridge between the raw data and the visual representation, ensuring that the data is structured and formatted in a way that is suitable for the selected chart type.

**Convert prepared data into visual elements.** Following data preparation, the final stage involves mapping the data into visual channels appropriate to the selected chart type, such as the height of bars or the area of shapes. E2 described this process as “assigning the right variables to the right visual properties, such as putting the independent variable on the x-axis and the dependent on the y-axis.” E4 also noted that he would usually make some adjustments after examining the results, such as removing bars with very low values to reduce the number of bars. In short, this stage aims to effectively and accurately visualize the prepared data and clearly communicate the intended message to the audience.

#### 3.3 Refined to Five-Stage CoT Process for NL2VIS

After summarizing these three stages, we presented them to our experts for validation and collected their feedback. While they generally agreed with this pipeline, they pointed out that they could be further refined for the NL2VIS task, especially in data preparation. E3 stated, “When analyzing trends over time, I usually experiment with different temporal granularities, like daily, monthly, or yearly. This does not change the underlying data I visualize but only changes the granularity of analysis.” E2 elaborated, “This is essentially the binning operation, which is commonly used in exploratory data analysis and affects how patterns emerge from the data.” Furthermore, E4 suggested splitting the data preparation into three steps: identifying relevant data, determining proper analysis granularity, and implementing necessary modifications to enhance visualization effectiveness. He provided an example: “When visualizing average income across different countries, we first extract data on income and nationality, then calculate the average income per country, and finally select and sort the countries and display the top 10 or 20 countries for visualization. Once these three steps are completed, the data is properly prepared for effective visualization.”

Based on these feedback, we refined the CoT process structure to a five-stage process:

- S1 Determine chart type: Select the most appropriate visualization method based on the data and analysis intent.
- S2 Retrieve relevant data: Identify and extract the specific data attributes required for the visualization.
- S3 Define data granularity: Establish the appropriate granularity for the data visualization.
- S4 Refine data for visualization: Apply transformations such as filtering and sorting to prepare the final data for optimal visualization.
- S5 Generate visualization: Configure and generate the final visualization to effectively communicate the intended insights.

#### 3.4 Validation of the Five-Stage Process

To validate our refined five-stage process, we conducted a follow-up evaluation with the same group of experts. We presented them with five diverse analysis tasks and asked them to describe their visualization design process using our framework. The experts successfully applied the five-stage process to all scenarios, confirming its completeness and flexibility. E4 noted, “This structure is comprehensive yet simple enough to apply across different types of visualization tasks. It captures the key decision points without being overly prescriptive.” E2 added, “What I particularly like about this framework is that it explicitly details decisions often implicitly made by experienced visualizers, which

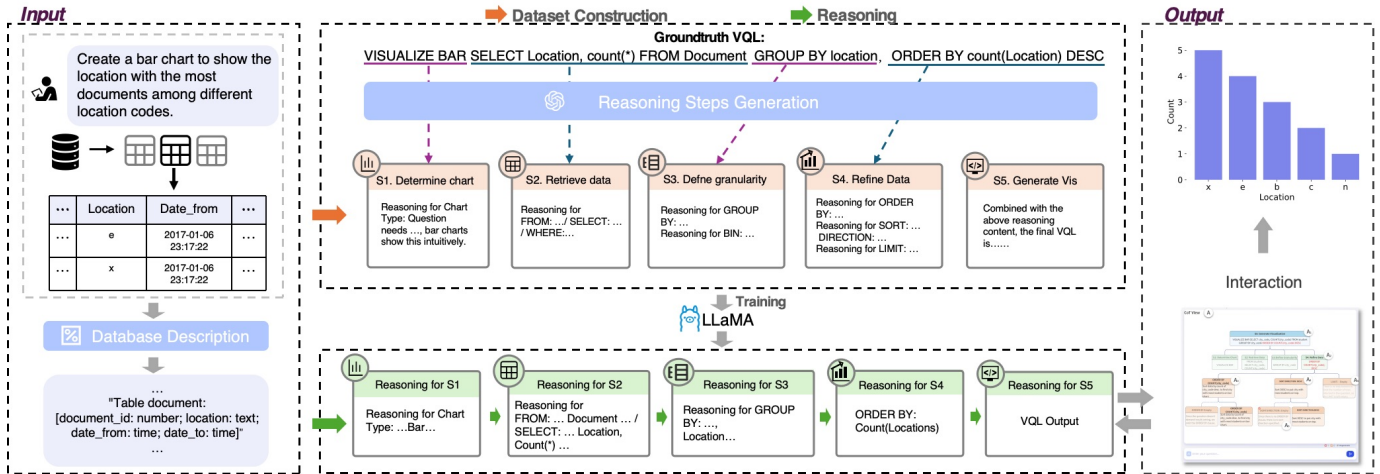


Fig. 2: The pipeline of DeepVIS Framework

could benefit novice training and automation.” The validation confirmed that our five-stage CoT process effectively captures the essential reasoning steps in visualization design while remaining adaptable to diverse analytical scenarios.

## 4 CONSTRUCTION OF NVBENCH-COT

Even with the identified CoT process, we still need sufficient training data to guide models in learning the reasoning steps for NL2VIS. However, it would be prohibitively expensive to manually create the whole reasoning process, which requires an immense amount of time and expert effort. To tackle this issue, we develop an automatic pipeline that augments existing NL2VIS datasets with structured CoT reasoning steps. Unlike the few-shot prompting approach [58], our pipeline significantly enhances the model’s capability to reason about chart-specific patterns and SQL-syntax patterns. Fig. 2 shows our detailed pipeline, which consists of two modules: the database description module and the reasoning steps generation module.

For clarity, we demonstrate our pipeline using nvBench [29], a widely used NL2VIS dataset, but our pipeline can be easily adapted to other NL2VIS datasets. In nvBench, each training sample contains a table (e.g., Faculty), a natural language query (e.g., “Compute the total number of rank across rank as a pie chart”), and the corresponding Visualization expressed in Visualization Query Language (VQL) (e.g., VISUALIZE Pie SELECT Rank, COUNT(Rank) FROM Faculty GROUP BY Rank). This VQL can be executed, transformed to Vega-lite specifications, and rendered to obtain the visualizations.

### 4.1 Database Description Module

The database description module aims to enhance input to provide essential details for subsequent reasoning steps. Our expert interviews revealed that analysts typically scan multiple table rows at the beginning to gain a foundational understanding of data. Our experiments also confirmed that adding comprehensive database descriptions significantly improves reasoning accuracy and reduces errors (Sec. 6.1.2). To mirror this common practice of examining several table rows, we implemented a template-based augmentation method with two key subcomponents: **Schema description.** Given a table, we generate a comprehensible schema that captures essential elements, including table names, column names, and their data types. For instance, when processing the faculty table, our method formats columns using the conventional column name:value type pattern, producing entries such as facid:number, fname:text, and rank:text.

**Value sampling.** In addition to the schema description, it is also necessary to examine sample values for accurate reasoning. For instance, a rank:text column might contain either full names like Associate Professor or abbreviation like AssocProf. Without concrete examples, it is impossible to determine whether WHERE rank=Associate

Professor or WHERE rank=AssocProf is the correct filtering condition. Therefore, we incorporate representative value samples in the input. To ensure inclusion of relevant samples, we use GPT-4o-mini to process both the natural language query and database schema to identify relevant columns and incorporate appropriate samples accordingly.

### 4.2 Reasoning Steps Generation Module

After generating comprehensive database descriptions, our reasoning steps generation module creates structured CoT reasoning steps that systematically guide models through the visualization creation process. Following our identified five-stage CoT process, we decomposed the visualization reasoning into five key steps. We leveraged GPT-4o-mini to complete the reasoning steps using the ground truth VQL and carefully crafted prompts, with full details provided in the supplemental material. We selected GPT-4o-mini due to its strong reasoning capabilities and cost-effectiveness for large-scale data generation.

**Determine chart type (S1).** This initial step analyzes the VISUALIZE clause in the VQL output (e.g., bar, line, pie) to select the most appropriate chart type based on the user analysis intent and database schema. Models are required to justify why the chosen chart type effectively communicates the requested insights and fulfills the analysis intent.

**Retrieve relevant data (S2).** This step carefully identifies the necessary tables, columns, and conditions for the visualization. By reasoning through the FROM, SELECT, and WHERE clauses, only relevant data is extracted for further processing, aligning with the query. In a similar way, models are required to justify each decision they make.

**Define data granularity (S3).** In this step, models are required to determine how to group and aggregate data, using GROUP BY for categorical or numerical grouping and BIN BY for time-based data. It explains the grouping strategy to ensure the visualization accurately reflects trends or summaries in the data.

**Refine data for visualization (S4).** This step focuses on sorting and limiting operations, which correspond to ORDER BY, SORT DIRECTION, and LIMIT clauses in VQL. Models are required to explain why this refinement is needed and how they enhance readability or align with the analysis intent.

**Generate visualization (S5).** The final step synthesizes all previous reasoning results and generates the complete VQL. This step ensures all components work harmoniously to produce a visualization that accurately addresses user needs while maintaining technical correctness.

After building these reasoning steps, we conducted tuning and validation experiments on the Llama3.1-8B-Instruct model and found that sometimes the model will still generate illegal results, such as HISTOGRAM in the VISUALIZE field and unsupported functions like WEEKDAY(Date) in the BIN BY field. Such issues can be addressed by providing explicit constraints in the input prompt. These constraints include limiting visualization types to BAR, PIE, LINE, and SCATTER and restricting column selections to those in the database schema or

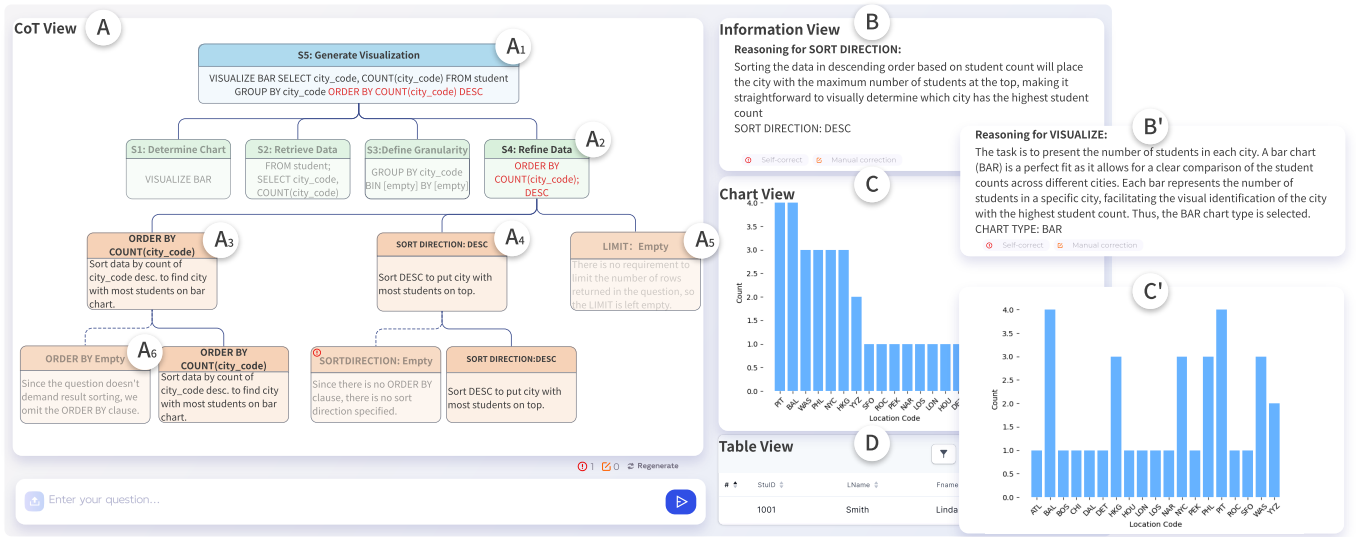


Fig. 3: The interface of DeepVIS. The CoT view (A) provides a structured overview of the reasoning process and allows users to interactively refine it, while three supporting views (B-D) complement the CoT view during the analysis.

valid derivations (e.g., COUNT, AVG, MAX, MIN). Our ablation studies demonstrate that incorporating these constraints as text prompts in our training samples significantly improves model reasoning capabilities and VQL generation accuracy.

### 4.3 nvBench-CoT

Before automatically augmenting nvBench using the database description module and the reasoning steps generation module, we conducted a rule-based filtering and removed 41 problematic samples, including duplicated queries (9), illegal VQLs (26), and empty VQLs (6). Furthermore, we leveraged GPT-4o-mini to identify 1,351 samples exhibiting inconsistency between queries and VQLs. For example, consider the query “What are the dates of the assessment notes, and count them by a bar chart.” The groundtruth VQL is `Visualize BAR SELECT date_of_notes, COUNT(date_of_notes) FROM Assessment_Notes BIN date_of_notes BY WEEKDAY`, which contains an erroneous BIN BY clause that makes it inconsistent with the original query intent. After removing these problematic samples, we conducted a comprehensive quality assurance process by randomly sampling 15% of the augmented data and manually evaluating the appropriateness of the generated reasoning steps. Detailed statistics of the nvBench-CoT are presented in supplemental material.

## 5 DEEPVIS

In addition to our dataset and pipeline, we also developed DeepVIS, a visual analysis tool that exposes detailed reasoning processes while enabling rich interactive exploration. This tool bridges the gap between NL queries and visualization outputs by making the underlying CoT reasoning accessible and modifiable.

### 5.1 Visualization Design

Fig. 3 presents the interface of DeepVIS. The core component is the CoT view (Fig. 3A), which provides a structured overview of the model’s reasoning process from natural language queries to final visualization results. The detailed reasoning steps are organized as a hierarchical tree following our five-stage CoT process. The root node corresponds to S5, which synthesizes the previous reasoning steps and produces the final VQL result. Placing this stage as the root allows users to quickly grasp the output before exploring the underlying reasoning process. Four second-level nodes correspond to the four stages (S1-S4) in our CoT Process, each displaying the key decisions made by the model at that stage. Their child nodes reveal more detailed reasoning steps, such as GROUP\_BY, and BIN\_BY fields inferences in S3. To facilitate understanding without overwhelming the user with

details, a concise summary of the reasoning is included in each node. To optimize space usage while maintaining context, we implemented a space-tree layout with dynamic expansion controls. This adheres to the “details-on-demand” principle, allowing users to selectively expand nodes of interest while collapsing others. This hierarchical visualization aligns with the top-down analysis workflow users naturally adopt, enabling them to efficiently navigate from high-level visualization to specific implementation details while maintaining contextual awareness throughout the exploration process.

In addition to the CoT view, we provided three supporting views to complement the CoT view during the analysis. The information view (Fig. 3B) provides comprehensive reasoning text for the selected step, which allows users to examine the detailed thought process. The chart view (Fig. 3C) renders the final visualization based on the model-generated VQL. Users can also export the Vega-Lite specification or the SVG file once they confirm their satisfaction with the results. For detailed data exploration, the table view (Fig. 3D) provides a structured presentation of the underlying dataset, enabling users to verify how raw data translates into visual elements throughout the reasoning steps.

### 5.2 Interactions

**Coordinated exploration.** We implemented coordinated view updates to maintain analysis context throughout the exploration process. When users select a node within the CoT view, the information view instantly displays the comprehensive reasoning text associated with that step, and the table view shows the retrieved or transformed data after the selected steps. These coordinated interactions create a cohesive analysis environment that maintains contextual continuity while navigating the complex reasoning chain.

**Interactive refinement.** Beyond passive exploration, our interface allows users to actively refine the reasoning process when they identify potential errors or suboptimal decisions. We offer two complementary refinement mechanisms:

- **Self correction:** This feature prompts the model to automatically reconsider its decisions within the selected reasoning step. By leveraging hints about potential errors in the flagged step, the model can improve results without direct user input.
- **Manual correction:** This allows users to provide specific preferences that steer the regeneration process toward desired results, combining human domain expertise with model capabilities.

The corresponding prompts are provided in the supplemental material.

After making corrections, our tool will intelligently regenerate all subsequent steps to ensure logical consistency throughout the reasoning process. To help users understand the effects of their modifications,

we implement an intuitive comparison feature that clearly identifies differences between original and revised reasoning paths. As shown in Fig. 3A, when a self-correction is applied to the SORT DIRECTION step (Fig. 3A<sub>4</sub>), unchanged nodes appear visually dimmed (e.g., Fig. 3A<sub>5</sub>), while modified nodes are highlighted with affected fields marked in red for immediate identification (Figs. 3A<sub>1</sub>-A<sub>4</sub>). The newly generated reasoning step is appended under the node for SORT DIRECTION (Fig. 3A<sub>6</sub>), allowing users to directly compare the before-and-after reasoning processes and select the more appropriate one.

By making the CoT reasoning both transparent and interactive, our method transforms users from passive consumers of automated visualizations into active collaborators in the design process. This improves both the usability of DeepVIS and the quality of resulting visualizations, ultimately leading to more effective data exploration and analysis.

## 6 EVALUATION

### 6.1 Quantitative Evaluation

#### 6.1.1 Comparison with Baseline Methods

**Baseline methods.** We chose seven representative NL2VIS methods with different architectures for comparison.

- **Seq2Vis** [29]: This method treats the NL2VIS problem as a machine translation problem between natural language and visualization specifications. We included it because it is the pioneering work and establishes the foundational baseline.
- **Transformer**: This is a milestone model and has been widely adopted for various NLP tasks. We included it to evaluate how a general-purpose NLP model performs on the NL2VIS task.
- **ncNet** [30]: This is the state-of-the-art model for NL2VIS based on the Transformer architecture, which introduces several visualization-aware optimizations to better understand user intent and generate specification-compliant outputs. We included it to establish the current performance ceiling for specialized models.
- **RGVisNet** [41]: This is an innovative hybrid retrieval-generation method that first retrieves the most relevant query candidates as prototypes from the VQL codebase and then revises them to produce the desired VQL. We included it to compare with different paradigms in NL2VIS and provide insights into whether our CoT methodology outperforms retrieval-augmented strategies.
- **Llama3.1-8B-SFT**: This baseline employs the identical backbone architecture as our model, yet it adopts end-to-end data without CoT reasoning steps for supervised fine-tuning. We included it to directly validate the effectiveness of our CoT module.
- **General Purpose LLMs**: We evaluate against seven state-of-the-art general purpose LLMs representing diverse architectures and capabilities: Llama3.1-8B (an open-source small-scale model), GPT-4o-mini (which also serves as the source for CoT reasoning steps generation in our method), GPT-o1 and GPT-o3 (OpenAI’s latest reasoning models), Gemini-2.5-Pro (Google’s flagship multimodal model), Claude-3.5-Sonnet (Anthropic’s advanced reasoning model), and DeepSeek-R1 (an open-source reasoning model). We included them because they represent how general users perform NL2VIS using state-of-the-art LLMs.
- **ChartGPT** [61]: This is the most comparable work that also breaks down the chart generation task into multiple steps: column selection, data filtering, data aggregation, chart type determination, and data visualization. We included it to highlight the effectiveness of our specific reasoning chain design.

**Experiment settings.** We adopted the split of the train/dev/test data set following Song *et al.* [41], which achieves a strict separation of databases and ensures that no individual database appears across multiple sets. This prevents potential data leakage and maintains the integrity of the evaluation. We selected Llama3.1-8B-Instruct as our base model due to its stable performance [14], open-source accessibility, and widespread adoption in recent research [49, 67]. The detailed hyperparameter settings are provided in the supplemental material.

**Metrics.** To comprehensively evaluate model performance, we compared the generated VQL with the groundtruth from multiple aspects. Following RGVisNet [41], we measure accuracy in chart type (Chart

Table 1: Performance Comparison.

Method	Chart Acc	Axis Acc	SQL Acc	Data Acc	All Acc
Seq2Vis	93.18%	22.71%	0.80%	0.84%	0.62%
Transformer	97.79%	62.24%	18.59%	18.73%	17.93%
ncNet	<b>98.05%</b>	46.40%	42.59%	43.07%	42.28%
RGVisNet	97.21%	48.34%	53.61%	53.96%	51.70%
Llama3.1-8B	81.27%	68.75%	42.98%	51.62%	46.48%
Llama3.1-8B-SFT	83.83%	77.05%	52.10%	62.52%	59.37%
GPT-4o-mini	91.31%	87.16%	52.35%	75.07%	70.37%
GPT-o1	91.39%	93.36%	59.76%	77.62%	71.63%
GPT-o3	92.70%	91.95%	59.39%	76.02%	71.14%
Gemini-2.5-Pro	92.03%	93.32%	46.57%	75.25%	69.81%
Claude-3.7-Sonnet	92.47%	93.98%	59.72%	77.64%	71.80%
DeepSeek-R1	92.25%	94.36%	51.75%	77.34%	72.60%
ChartGPT	97.34%	94.85%	69.21%	73.84%	73.03%
<b>NL2VIS-CoT</b>	97.52%	<b>95.17%</b>	<b>74.63%</b>	<b>80.74%</b>	<b>77.16%</b>

Acc), x/y axes configuration (Axis Acc), and SQL syntax (SQL Acc), which are the three major components of VQL. However, syntactically different SQL queries can produce identical results, such as the conditions WHERE YEAR IN (1999, 2000) and WHERE YEAR = 1999 OR YEAR = 2000. Therefore, we introduce two additional metrics: Data Acc, which measures whether execution results match regardless of SQL syntax, and All Acc, which indicates when chart type, axes, and data all match correctly. These execution-based metrics complement the syntax-based metrics for a more comprehensive assessment.

**Results analysis.** Table 1 summarizes the results of our proposed NL2VIS-CoT method and the baseline methods on the test set. Traditional models like Seq2Vis and Transformer exhibit high Chart Acc but struggle with SQL Acc and Data Acc. This stems from their inability to effectively handle the hierarchical complex dependencies between natural language semantics and database schema parsing, which is critical for NL2VIS tasks. In contrast, advanced specialized models like ncNet and RGVisNet address this through sophisticated design choices, making them better interpret natural language queries and transform data accurately. Specifically, RGVisNet performs better than ncNet (51.70% vs. 42.28% All Acc), which can be attributed to its hybrid retrieval-generation framework that better utilizes existing VQL prototypes to reduce errors during visualization synthesis. For LLM-based methods, Llama3.1-8B with few-shot prompting achieves 46.48% in All Acc without any training, and Llama3.1-8B-SFT achieves higher performance (59.37%) after supervised fine-tuning. However, the performance is still constrained by model scale limitations. Large-scale LLMs with few-shot prompting demonstrate better performance, and the results are consistent across different models: around 92% in Chart Acc, 93% in Axis Acc, 77% in Data Acc, and 71% in All Acc. However, they lag behind the traditional methods in terms of Chart Acc, revealing a critical limitation of few-shot learning: limited exemplars hinder effective generalization for different cases and may even generate invalid VQLs, such as VISUALIZE HISTOGRAM. Compared with previous LLM-based methods, ChartGPT and our method achieve higher Chart Acc and All Acc, highlighting the great potential of fine-tuning small-scale LLMs with CoT reasoning steps. Notably, our NL2VIS-CoT achieves a leading All Acc of 77.16% while simultaneously performing best in Axis Acc, SQL Acc, and Data Acc.

In addition to the numerical comparison, we also provide several examples to demonstrate why NL2VIS-CoT performs better than the baseline methods in Fig. 4. Here we select ncNet and GPT-4o-mini as two representative methods for comparison, while the full results are provided in the supplemental material.

#### 6.1.2 Ablation Study

In addition to the benchmark evaluation, we carry out ablation experiments to validate the effectiveness of our constructed nvBench-CoT and the significance of each component. We consider four ablation methods:

- **w/o value sampling**: We removed the column value sampling mechanism from the database description module to evaluate how this mechanism affect model performance.

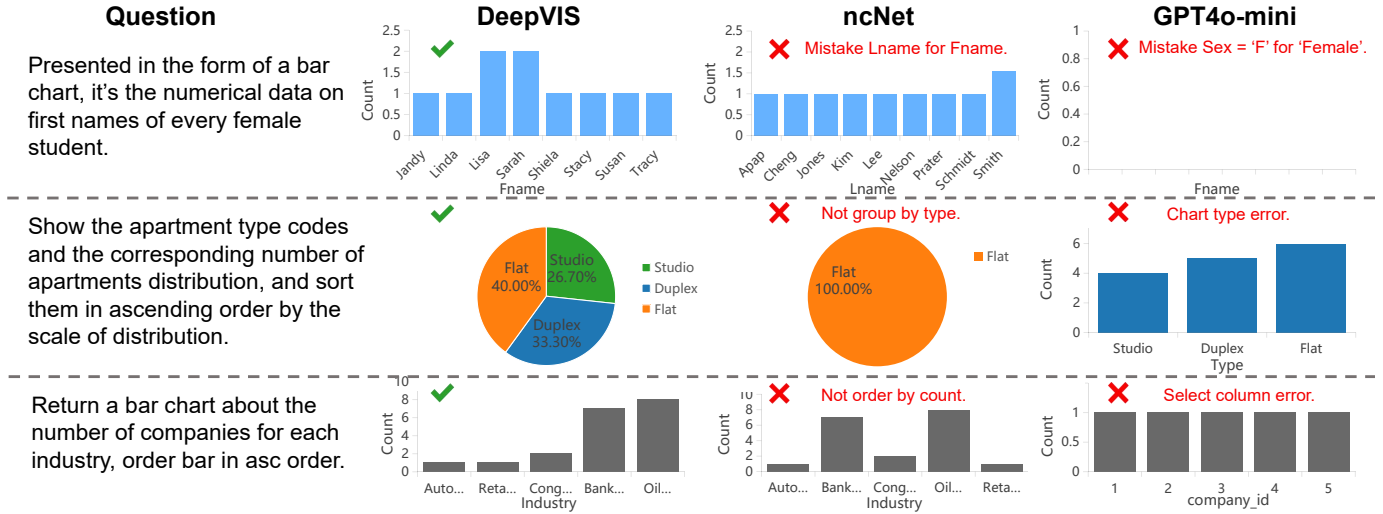


Fig. 4: Comparative analysis of VQLs and generated charts between our method and baseline methods.

- **w/o constraints:** We removed the explicit constraints in the input prompt to understand how these constraints contribute to the model’s behavior.
- **w/o CoT:** We removed the CoT reasoning process to analyze its role and importance in NL2VIS.
- **ChartGPT-pipeline:** ChartGPT also decomposes the generation of VQL into multiple steps but uses a different order from us. To evaluate the comparative effectiveness of reasoning step orders, we reconstructed our dataset using ChartGPT’s proposed order and trained a model under identical experimental settings.

**Results analysis.** Table 2 shows the results of the ablation study. When the value sampling mechanism was eliminated, Data Acc substantially dropped from 80.74% to 72.11%, with All Acc dropping from 77.16% to 69.31%. This is mainly because the model fails to generate correct SQL queries to retrieve relevant data, particularly in the WHERE clause. For example, the model incorrectly generates WHERE rank="Asstant Professor" instead of the correct one WHERE rank="AsstProf", resulting in a query that fails to retrieve any data. The removal of constraints produced a different pattern of performance degradation. While SQL Acc and Data Acc show relatively modest declines compared to removing value sampling, Chart Acc and Axis Acc exhibit more significant drops. This emphasizes the importance of constraints in generating reasonable results. In one example, the model incorrectly uses a function WEEKDAY() instead of the standardized BIN BY syntax, resulting in an illegal VQL. When removing the detailed CoT reasoning steps, all the metrics significantly drop, indicating the critical importance of the reasoning process in our method. Compared with the ChartGPT-pipeline, our method also achieves better performance across all metrics, empirically validating the effectiveness of our order.

Table 2: Ablation Study Results.

Method	Chart Acc	Axis Acc	SQL Acc	Data Acc	All Acc
w/o value sampling	94.99%	88.95%	67.85%	72.11%	69.31%
w/o constraints	94.16%	87.03%	71.81%	78.23%	75.39%
w/o CoT	70.30%	65.79%	48.30%	51.88%	49.31%
ChartGPT pipeline	95.60%	93.08%	71.10%	77.32%	73.54%
<b>NL2VIS-CoT</b>	<b>97.52%</b>	<b>95.17%</b>	<b>74.63%</b>	<b>80.74%</b>	<b>77.16%</b>

### 6.1.3 Error Analysis

To better understand the limitations of our method and identify areas for improvement, we conducted a comprehensive error analysis across our four-step reasoning. Our analysis reveals that S1 (determine chart) had fewer errors (56), while S2 (retrieve data), S3 (define

granularity), and S4 (refine data) exhibited much more errors (183, 202, and 253 samples, respectively). Notably, these included 131 aggregation function errors in S2, 191 GROUP BY errors in S3, and 220 ORDER BY errors in S4. For example, when processing the query “Can you draw the trend of maximal score over the year? rank by the x-axis in descending,” our method generates VISUALIZE LINE SELECT YEAR, MAX(SCORE) FROM WINE ORDER BY YEAR DESC. While the system correctly identified the aggregation function MAX(SCORE) and the sorting requirement, it failed to include the essential GROUP BY YEAR clause necessary for proper aggregation. This demonstrates that while our model effectively handles common queries, there remains room for improvement in complex queries involving data aggregation and transformation logic.

Next, we focused our analysis on S1 due to its foundational role in the reasoning pipeline and its cascading impact on subsequent steps. Our findings reveal significant variations in error rates across different chart types: BAR (1.18%), PIE (2.97%), SCATTER (7.63%), and LINE (9.83%). Notably, despite pie charts representing only 7.84% of the dataset, they maintain a relatively low error rate, suggesting that chart frequency does not directly correlate with prediction accuracy. To better understand the high error rates in line charts and scatter plots, we conducted a deeper investigation into these errors and found that over 95% of them occur in multi-solution scenarios, *i.e.*, multiple chart types can effectively address the same question. For example, when asked “How many documents correspond with each project id,” the ground truth specifies a scatter plot, yet our model’s bar chart response equally solves the problem. This pattern exposes a limitation in the current dataset evaluation framework, which fails to account for multiple valid solutions corresponding to a single natural language query.

## 6.2 Use Cases

We present two use cases to demonstrate how our interactive interface enhances user understanding of the reasoning process and enables targeted adjustments.

### 6.2.1 Case 1: Using Self Correction to Refine Results.

This use case illustrates how users can explore reasoning steps and guide the model to reconsider critical decisions, leading to enhanced visualization outcomes.

Alice analyzes the allergy database, which contains a table student with columns such as stuid, name, city\_code, and age. She aims to examine student distribution across cities and identify the city with the most students. Therefore, she inputs “Please display a bar chart showing all cities and their corresponding number of students to identify the city with the highest student count.” Fig. 3C displays the generated chart, which successfully uses a bar chart to visualize student numbers

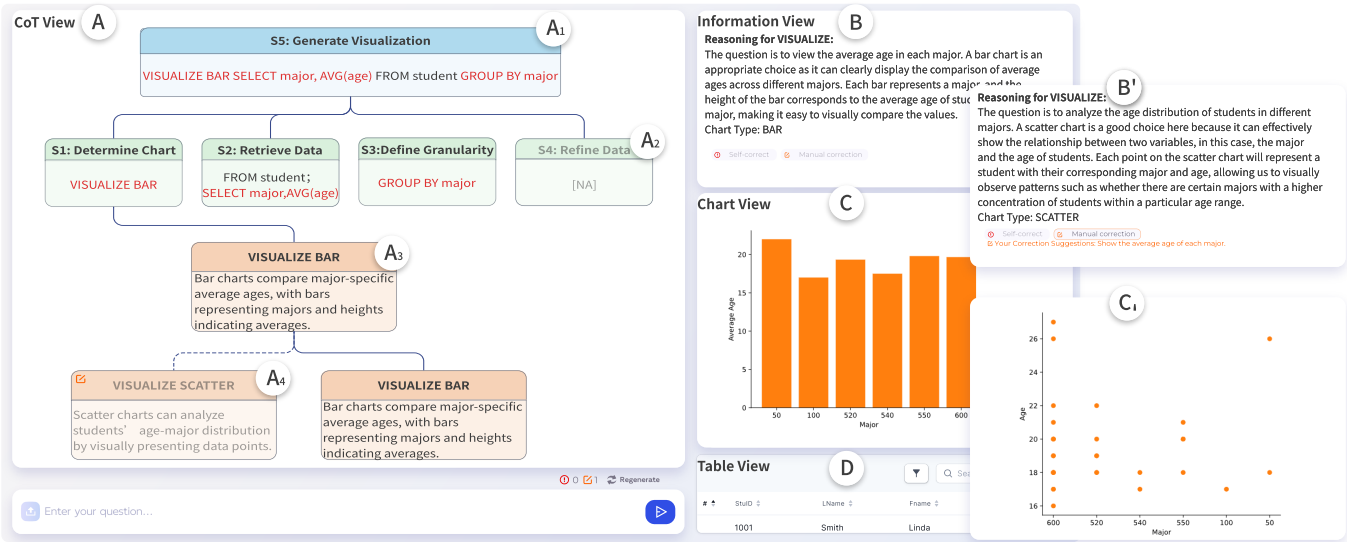


Fig. 5: After using manual correction to specify the preference of the chart type, DeepVIS correctly updates the subsequent reasoning results and generates a proper aggregated visualization.

by city. Alice verifies the reasoning process and confirms it is sound. For example, Fig. 3B' shows that the model appropriately justifies the chart type selection: "A bar chart (BAR) is a perfect fit as it allows for a clear comparison of the student counts across different cities." However, while the chart enables identification of the city with the most students, Alice prefers to sort the data to make the pattern more evident. The model initially overlooked sorting because it reasoned: "Since the question doesn't demand result sorting, we omit the ORDER BY clause" (Fig. 3A6). To address this issue, Alice activates the "Self correct" feature to prompt the model to reconsider this decision. Upon reconsideration, the model now acknowledges that "Sorting the data in descending order based on student count will place the city with the maximum number of students at the top, making it straightforward to visually determine which city has the highest student count" (Fig. 3B), and the node for reasoning SORT DIRECTION also updates accordingly (Fig. 3A4). Fig. 3C shows the improved visualization with sorted bars, which Alice finds satisfactory.

### 6.2.2 Case 2: Use Manual Correction to Refine Results.

This use case illustrates how users can explore reasoning steps and make target refinements to their query.

Bob is tasked with analyzing a university database to explore the age distribution of students across different majors. The database includes a student table with key columns such as `stuid`, `age`, `sex`, `major`, `advisor`, and `city_code`, alongside other tables like `faculty` and `department`. To begin his analysis, he inputs the query: "Analyze the age distribution of students in different majors" into DeepVIS. DeepVIS responds by generating a scatter plot with individual data points for each student's major and age (Fig. 5C'). He finds this scatter plot difficult to interpret and compare the age distribution across different majors. Therefore, he decides to examine the reasoning process to see why DeepVIS select scatter plot. After clicking the node for reasoning the chart type, he finds that DeepVIS thinks a scatter plot effectively displays the relationship between two variables—major and age, revealing patterns like age concentrations within specific majors (Fig. 5A4). He then realizes that his query may not be so accurate and may mislead the model. Therefore, he refines his requirement to "Show the average age of each major" and clicks "Manual correction." After applying this correction, DeepVIS changes the chart type from SCATTER to BAR, selection from `age` to `AVG(age)`, and adds `GROUP BY major` to aggregate the data by major. The resulting bar chart displays each major as a distinct bar, with the height of each bar representing the average age of students in that major (Fig. 5C). This new visualization proves

far clearer than the initial scatter plot, enabling Bob to easily compare the central tendency of ages across different majors. Reflecting on the process, Bob realizes that his original query "Analyze the age distribution" lacks specificity. However, the reasoning steps help him realize it and successfully guide the tool to produce a more suitable output. He also appreciates that DeepVIS automatically adjusted related fields to align with his corrected request, sparing him the effort of modifying each field manually.

## 6.3 User Study

We conducted a user study to evaluate whether presenting reasoning steps in DeepVIS facilitates users in understanding model behavior and generating better visualization.

**Participants.** We recruited 32 participants (P1-P32, 16 males and 16 females) for the experiment, 20 are from the local university and 12 are from the workforce, including 20 from the local university and 12 industry professionals, 20–51 years old ( $M=25.78$ ,  $SD=7.39$ ). They are from diverse majors, including Computer Science (8), Software Engineering (5), Finance (5), Mathematical Statistics (3), Digital Media Design (2), Journalism (2), Civil Engineering (3), Biomedical Engineering (2) and Geographic Information Science (2). All participants have experience using data visualization tools, such as Excel, Matplotlib, ECharts, and Vega-Lite. In addition, 25 of them have tried using LLMs to assist in data visualization.

**Study procedure.** We began the user study by introducing NL2VIS tasks using representative nvBench examples. Then, participants were introduced to four interfaces: ncNet (using the developers' Jupyter notebook), DeepSeek and ChartGPT (using chatbot interfaces integrating the backend models), and our DeepVIS. Following familiarization, participants completed 10 randomly sampled nvBench examples across all interfaces, with the order counterbalanced to mitigate learning and fatigue effects. Upon task completion, participants responded to a five-point Likert-scale questionnaire to evaluate the effectiveness and usability of the interfaces. Finally, we conducted a brief interview with participants to collect detailed feedback and analyzed the interview data using thematic analysis [2], where the first author did the initial coding and revised it with an external second coder.

**Result analysis.** Fig. 6 shows the rating across six perspectives: insights communication, intent reflection, logic comprehension, error identification, refinement efficiency, and workload saving. The detailed questions are provided in the supplemental material. Overall, DeepVIS has consistently received a relatively high proportion of "Strongly Agree" and "Agree" ratings across all six dimensions, which provides





Fig. 6: User study results: DeepVIS has consistently received relatively higher ratings across all six dimensions.

compelling evidence of its effectiveness in enhancing user understanding and facilitating interactive refinement. P7 commented positively regarding the reasoning steps generated by the DeepVIS: “These reasoning steps sound reasonable and justify the choice made by models, making the visualization decisions transparent and understandable.” Other participants also agreed that the detailed reasoning steps helped them “identify the errors in reasoning more easily”, (P2) “allow for quicker iteration” (P8), and “provide useful hints to refine the input query” (P11). Such feedback highlights the benefits of disclosing the reasoning steps in the analysis. With respect to overall workload reduction, the participants consistently acknowledged the effectiveness of DeepVIS. Many expressed sentiments similar to those of P4, who remarked: “It alleviates the tasks of creating visualization by automatically handling data transformation and visual encoding.” P1 further added: “I would like to use this tool in the future for my data analysis projects,” demonstrating the practical usability of DeepVIS.

In comparative evaluations against other baseline methods, DeepVIS exhibited better performance across all perspectives. A particularly notable distinction emerged in refinement efficiency, where our method received only 9% negative rating, substantially lower than ncNet (50%), DeepSeek (38%), and ChartGPT (47%). This performance advantage stems from our transparent interface design, which exposes intermediate reasoning steps and enables targeted interactive refinement. In contrast, other interfaces require users to restart the entire process when modifications are needed. For example, P2 commented on ncNet: “While this package is very easy to use, it only generates the final visualization, and I sometimes need to try different queries to obtain a satisfactory result, which can be time-consuming and frustrating.” Regarding the workload saving, P17 pointed out that she needs to “provide more detailed instructions to steer ncNet and DeepSeek compared to DeepVIS.” This highlights the advantage of DeepVIS in reducing the cognitive burden on users while producing high-quality visualizations that accurately reflect their analysis intent.

## 7 DISCUSSION AND FUTURE WORK

Based on the interview with our experts and the participants in the user study, we discuss several promising directions for future work.

**Integrating visualization feedback.** While our CoT process has shown promising results, it currently lacks direct integration of the final visualization or its underlying data into the reasoning process. This limitation can result in suboptimal visualizations, especially when multiple alternatives could meet user demands. For example, while both bar charts and scatter plots can reveal relationships between variables, the optimal choice depends on data characteristics. Scatter plots work better with fewer data points, while bar charts are preferable when presenting aggregated values. To address this limitation, we suggest integrating feedback from the final visualization into the creation process by adapting our CoT pipeline. We propose two key strategies to achieve this: First, we could explore leveraging multi-modal large language models to analyze generated visualizations and provide actionable insights, such as detecting visual clutter or recommending alternative chart types. Second, tools like VizLinter [5] can automatically pinpoint common visualization flaws. These flaws can be described in natural language and integrated into the reasoning process, allowing models to suggest fixes or apply corrections automatically.

**Enhancing fine-grained control.** Currently, we translate the natural language queries into VQLs, which are then converted into Vega-

Lite specifications for rendering. This design choice abstracts away rendering-specific details, enabling cross-tool compatibility with visualization frameworks such as Vega-Lite, ECharts, and matplotlib. However, this also limits fine-grained control on the chart, such as setting color scales, adjusting layout tuning, and changing mark size. This issue can be addressed by further advancing our CoT framework. On the one hand, we can analyze existing visualization tools, summarize common specifications, and incorporate additional fields, such as COLOR\_BY for color encoding and Mark\_Size controlling mark dimensions. This enhancement would allow users to specify detailed aesthetic preferences while preserving the benefits of cross-tool compatibility. On the other hand, we can explore datasets that map natural language queries directly to Vega-Lite specifications, which supports more diverse chart types and allows more detailed configurations. The dataset can be constructed in a similar way, *i.e.*, prompting the LLMs to generate detailed, step-by-step reasoning traces that connect analysis intent to ground truth specifications. The generated reasoning steps can guide models in learning how to achieve analysis intent using Vega-Lite.

**Extending the CoT framework to broader tasks.** By incorporating CoT reasoning into the NL2VIS pipeline, we have demonstrated how structured reasoning steps can improve both model performance and transparency in NL2VIS. Our method is highly flexible and can be readily adapted to more complex NL2VIS datasets. For example, a preprint dataset nvBench2.0 [26] extends nvBench by providing multiple valid VQLs for identical queries. On the one hand, our method can directly generate intermediate reasoning steps for this dataset without modification due to their similar data format. On the other hand, users can make targeted modifications to the prompts in our reasoning steps generation module, which instructs models to produce multiple valid outputs at each reasoning step and generate diverse VQLs. Furthermore, we would like to highlight that by augmenting existing datasets with reasoning steps, models are able to solve a complex task beyond NL2VIS. For example, in data storytelling, it can help explain the reasoning behind the visual choices so that the model learns how to better connect the stories to visuals. In visualization debugging tasks, the reasoning process provides rich contextual insights into why certain visualizations may fail, allowing models to refine design choices more effectively. Future work can explore how this framework facilitates various visualization tasks and achieves better human-AI collaboration.

## 8 CONCLUSION

To tackle the challenges that existing NL2VIS methods suffer from a lack of transparency and are challenging to refine due to their black-box designs, we propose the integration of the CoT process into the NL2VIS pipeline. Our work delivers three key contributions: 1) designing a comprehensive CoT reasoning process for NL2VIS, 2) introducing the nvBench-CoT dataset with detailed reasoning steps to achieve state-of-the-art performance, and 3) developing an interactive visual interface that lets users inspect and tweak the reasoning behind visualizations. Quantitative benchmark evaluation and qualitative case studies demonstrate that our method outperforms traditional methods, with users appreciating the inherent transparency that fosters trust and expertise. Furthermore, our method suggests the broader potential of CoT reasoning to enhance model performance and foster effective human-AI collaboration across diverse visualization tasks.

## SUPPLEMENTAL MATERIALS

All the supplemental materials are available at the website <https://github.com/Bvivib-shuai/DeepVIS>, including: 1) a PDF file including the prompts for the reasoning steps generation module, exemplar natural language queries, VQLs, and visualizations generated by baseline methods and our methods, and the questionnaire for the user study, 2) a video demonstrating the interface and the two use cases, 3) the nvBench-CoT dataset, 4) the code for training data and deploying DeepVIS, 5) the prompt of interactive refinement, and 6) implementation details.

## ACKNOWLEDGMENTS

This work is partly supported by the NSF of China (62402409), the Guangdong Basic and Applied Basic Research Foundation (2023A1515110545), the Guangzhou Basic and Applied Basic Research Foundation (2025A04J3935), and the Guangzhou-HKUST(GZ) Joint Funding Program (2025A03J3714).

## REFERENCES

- [1] K. Adamkiewicz, P. W. Woźniak, J. Dominiak, A. Romanowski, J. Karolus, and S. Frolov. PromptMap: An alternative interaction style for ai-based image generation. In *International Conference on Intelligent User Interfaces*, IUI '25, 15 pages, p. 1162–1176. Association for Computing Machinery, New York, NY, USA, 2025. doi: 10.1145/3708359.3712150
- [2] V. Braun and V. Clarke. *Thematic analysis*. American Psychological Association, 2012. 8
- [3] N. Chen, Y. Zhang, J. Xu, K. Ren, and Y. Yang. VisEval: A benchmark for data visualization in the era of large language models. *IEEE Transactions on Visualization and Computer Graphics*, 31(1):1301–1311, 2025. doi: 10.1109/TVCG.2024.3456320
- [4] Q. Chen, L. Qin, J. Liu, D. Peng, J. Guan, P. Wang, M. Hu, Y. Zhou, T. Gao, and W. Che. Towards reasoning era: A survey of long chain-of-thought for reasoning large language models. *arXiv preprint arXiv:2503.09567*, 2025. 1, 2
- [5] Q. Chen, F. Sun, X. Xu, Z. Chen, J. Wang, and N. Cao. VizLinter: A linter and fixer framework for data visualization. *IEEE Transactions on Visualization and Computer Graphics*, 28(1):206–216, 2022. doi: 10.1109/TVCG.2021.3114804
- [6] Y. Chen, R. Li, A. Mac, T. Xie, T. Yu, and E. Wu. Nl2interface: Interactive visualization interface generation from natural language queries. *arXiv preprint arXiv:2209.08834*, 2022. 2
- [7] Y. Chen and E. Wu. Pi2: End-to-end interactive visualization interface generation from queries. In *Proceedings of the 2022 International Conference on Management of Data*, SIGMOD '22, 15 pages, p. 1711–1725. Association for Computing Machinery, New York, NY, USA, 2022. doi: 10.1145/3514221.3526166
- [8] Z. Chu, J. Chen, Q. Chen, W. Yu, T. He, H. Wang, W. Peng, M. Liu, B. Qin, and T. Liu. Navigate through enigmatic labyrinth a survey of chain of thought reasoning: Advances, frontiers and future. *arXiv preprint arXiv:2309.15402*, 2023. 2
- [9] J. J. Y. Chung, W. Kim, K. M. Yoo, H. Lee, E. Adar, and M. Chang. TaleBrush: Sketching stories with generative pretrained language models. In *CHI Conference on Human Factors in Computing Systems*, CHI '22, article no. 209, 19 pages. Association for Computing Machinery, 2022. doi: 10.1145/3491102.3501819
- [10] J. J. Y. Chung and M. Kreminski. Patchview: Llm-powered worldbuilding with generative dust and magnet visualization. In *Symposium on User Interface Software and Technology*, pp. 1–19, 2024. doi: 10.1145/3654777.3676352
- [11] G. Feng, B. Zhang, Y. Gu, H. Ye, D. He, and L. Wang. Towards revealing the mystery behind chain of thought: a theoretical perspective. *Advances in Neural Information Processing Systems*, 36:70757–70798, 2023. 2
- [12] Y. Feng, X. Wang, K. K. Wong, S. Wang, Y. Lu, M. Zhu, B. Wang, and W. Chen. PromptMagician: Interactive prompt engineering for text-to-image creation. *IEEE Transactions on Visualization and Computer Graphics*, 30(1):295–305, 2024. doi: 10.1109/TVCG.2023.3327168
- [13] T. Gao, M. Dontcheva, E. Adar, Z. Liu, and K. G. Karahalios. DataTone: Managing ambiguity in natural language interfaces for data visualization. In *Symposium on User Interface Software and Technology*. Association for Computing Machinery, 2015. doi: 10.1145/2807442.2807478
- [14] A. Grattafiori, A. Dubey, A. Jauhri, A. Pandey, A. Kadian, A. Al-Dahle, A. Letman, A. Mathur, A. Schelten, A. Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024. 6
- [15] J. He, X. Wang, S. Liu, G. Wu, C. Silva, and H. Qu. POEM: Interactive prompt optimization for enhancing multimodal reasoning of large language models. *arXiv preprint arXiv:2406.03843*, 2024. 3
- [16] J. Hong, C. Seto, A. Fan, and R. Maciejewski. Do llms have visualization literacy? an evaluation on modified visualizations to test generalization in data interpretation. *IEEE Transactions on Visualization and Computer Graphics*, 2025. 2
- [17] M. Kahng, I. Tenney, M. Pushkarna, M. X. Liu, J. Wexler, E. Reif, K. Kallarakal, M. Chang, M. Terry, and L. Dixon. Llm comparator: Visual analytics for side-by-side evaluation of large language models. In *Extended Abstracts of the CHI Conference on Human Factors in Computing Systems*, pp. 1–7, 2024. 2
- [18] E. Kavaz, A. Puig, and I. Rodríguez. Chatbot-based natural language interfaces for data visualisation: A scoping review. *Applied Sciences*, 13(12):7025, 2023. 2
- [19] N. W. Kim, G. Myers, and B. Bach. How good is chatgpt in giving advice on your visualization design? *arXiv preprint arXiv:2310.09617*, 2023. 2
- [20] T. Kojima, S. S. Gu, M. Reid, Y. Matsuo, and Y. Iwasawa. Large language models are zero-shot reasoners. *Advances in neural information processing systems*, 35:22199–22213, 2022. 2
- [21] B. Li, Y. Luo, C. Chai, G. Li, and N. Tang. The dawn of natural language to SQL: are we fully ready? *Proc. VLDB Endow.*, 17(11):3318–3331, 2024. 1
- [22] B. Li, J. Zhang, J. Fan, Y. Xu, C. Chen, N. Tang, and Y. Luo. Alpha-sql: Zero-shot text-to-sql using monte carlo tree search. In *Forty-Second International Conference on Machine Learning, ICML 2025, Vancouver, Canada, July 13-19, 2025*. OpenReview.net, 2025. 2
- [23] S. Li, X. Chen, Y. Song, Y. Song, and C. Zhang. Prompt4vis: Prompting large language models with example mining and schema filtering for tabular data visualization. *arXiv preprint arXiv:2402.07909*, 2024. 1, 2
- [24] S. Liu, W. Yang, J. Wang, and J. Yuan. *Visualization for Artificial Intelligence*. Springer Nature Switzerland, 2025. doi: 10.1007/978-3-031-75340-4
- [25] X. Liu, S. Shen, B. Li, P. Ma, R. Jiang, Y. Zhang, J. Fan, G. Li, N. Tang, and Y. Luo. A survey of text-to-sql in the era of llms: Where are we, and where are we going?, 2025. 1
- [26] T. Luo, C. Huang, L. Shen, B. Li, S. Shen, W. Zeng, N. Tang, and Y. Luo. nvbench 2.0: Resolving ambiguity in text-to-visualization through stepwise reasoning, 2025. 2, 9
- [27] Y. Luo, X. Qin, C. Chai, N. Tang, G. Li, and W. Li. Steerable self-driving data visualization. *IEEE Trans. Knowl. Data Eng.*, 34(1):475–490, 2022. 1
- [28] Y. Luo, X. Qin, N. Tang, and G. Li. DeepEye: Towards automatic data visualization. In *International Conference on Data Engineering (ICDE)*, pp. 101–112. IEEE, 2018. doi: 10.1109/ICDE.2018.00019
- [29] Y. Luo, N. Tang, G. Li, C. Chai, W. Li, and X. Qin. Synthesizing natural language to visualization (nl2vis) benchmarks from nl2sql benchmarks. In *International Conference on Management of Data*, SIGMOD '21, 13 pages, p. 1235–1247. Association for Computing Machinery, New York, NY, USA, 2021. doi: 10.1145/3448016.3457261
- [30] Y. Luo, N. Tang, G. Li, J. Tang, C. Chai, and X. Qin. Natural language to visualization by neural machine translation. *IEEE Transactions on Visualization and Computer Graphics*, 28(1):217–226, 2022. doi: 10.1109/TVCG.2021.3114848
- [31] Y. Luo, Y. Zhou, N. Tang, G. Li, C. Chai, and L. Shen. Learned data-aware image representations of line charts for similarity search. *Proc. ACM Manag. Data*, 1(1):88:1–88:29, 2023. 1
- [32] P. Maddigan and T. Susnjak. Chat2VIS: Generating data visualizations via natural language using chatgpt, codex and gpt-3 large language models. *IEEE Access*, 11:45181–45193, 2023. doi: 10.1109/ACCESS.2023.3274199
- [33] T. Munzner. A nested model for visualization design and validation. *IEEE Transactions on Visualization and Computer Graphics*, 15(6):921–928, 2009. doi: 10.1109/TVCG.2009.111
- [34] A. Narechania, A. Srinivasan, and J. Stasko. NL4DV: A toolkit for generating analytic specifications for data visualization from natural language queries. *IEEE Transactions on Visualization and Computer Graphics*, 27(2):369–379, 2021. doi: 10.1109/TVCG.2020.3030378
- [35] S. Pandey and A. Ottley. Benchmarking visual language models on standardized visualization literacy tests. In *Computer Graphics Forum*, p.

- e70137. Wiley Online Library, 2025. 2
- [36] L. Podo, M. Angelini, and P. Velardi. V-recs, a low-cost llm4vis recommender with explanations, captioning and suggestions. *arXiv preprint arXiv:2406.15259*, 2024. 2
- [37] L. Podo, M. Ishmal, and M. Angelini. Toward a Structured Theoretical Framework for the Evaluation of Generative AI-based Visualizations. In M. El-Assady and H.-J. Schulz, eds., *EuroVis Workshop on Visual Analytics (EuroVA)*. The Eurographics Association, 2024. doi: 10.2312/eurova.20241118 2
- [38] L. Podo, B. Prenkaj, and P. Velardi. Agnostic visual recommendation systems: Open challenges and future directions. *IEEE Transactions on Visualization and Computer Graphics*, 2024. 1
- [39] V. Setlur, S. E. Battersby, M. Tory, R. Gossweiler, and A. X. Chang. Eviza: A natural language interface for visual analysis. In *Proceedings of the 29th annual symposium on user interface software and technology*, pp. 365–377, 2016. 2
- [40] L. Shen, E. Shen, Y. Luo, X. Yang, X. Hu, X. Zhang, Z. Tai, and J. Wang. Towards natural language interfaces for data visualization: A survey. *IEEE Transactions on Visualization and Computer Graphics*, 29(6):3121–3144, 2023. doi: 10.1109/TVCG.2022.3148007 3
- [41] Y. Song, X. Zhao, R. C.-W. Wong, and D. Jiang. RGVisNet: A hybrid retrieval-generation neural framework towards automatic data visualization generation. In *ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, KDD '22, 10 pages, p. 1646–1655. Association for Computing Machinery, New York, NY, USA, 2022. doi: 10.1145/3534678.3539330 2, 6
- [42] A. Srinivasan and V. Setlur. BOLT: A Natural Language Interface for Dashboard Authoring. In T. Hoell, W. Aigner, and B. Wang, eds., *EuroVis 2023 - Short Papers*. The Eurographics Association, 2023. doi: 10.2312/evs.20231035 2
- [43] H. Strobel, A. Webson, V. Sanh, B. Hoover, J. Beyer, H. Pfister, and A. M. Rush. Interactive and visual prompt engineering for ad-hoc task adaptation with large language models. *IEEE Transactions on Visualization and Computer Graphics*, 29(1):1146–1156, 2023. doi: 10.1109/TVCG.2022.3209479 2
- [44] Y. Sun, J. Leigh, A. Johnson, and S. Lee. Articulate: A semi-automated model for translating natural language queries into meaningful visualizations. In *Smart Graphics*, pp. 184–195. Springer, 2010. doi: 10.1007/978-3-642-13544-6\_18 2
- [45] F. Teng, Z. Yu, Q. Shi, J. Zhang, C. Wu, and Y. Luo. Atom of thoughts for markov LLM test-time scaling. *CoRR*, abs/2502.12018, 2025. 2
- [46] Y. Tian, W. Cui, D. Deng, X. Yi, Y. Yang, H. Zhang, and Y. Wu. ChartGPT: Leveraging llms to generate charts from abstract natural language. *IEEE Transactions on Visualization and Computer Graphics*, 31(3):1731–1745, 2025. doi: 10.1109/TVCG.2024.3368621 1, 2, 3
- [47] P.-P. Vázquez. Are llms ready for visualization? In *2024 IEEE 17th Pacific Visualization Conference (PacificVis)*, pp. 343–352. IEEE, 2024. 2
- [48] J. Wang, S. Liu, and W. Zhang. Visual analytics for machine learning: A data perspective survey. *IEEE transactions on visualization and computer graphics*, 30(12):7637–7656, 2024. 2
- [49] J. Wang, D. Paliotta, A. May, A. M. Rush, and T. Dao. The mamba in the llama: Distilling and accelerating hybrid models. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang, eds., *Advances in Neural Information Processing Systems*, vol. 37, pp. 62432–62457. Curran Associates, Inc., 2024. 6
- [50] Q. Wang, Z. Chen, Y. Wang, and H. Qu. A survey on ml4vis: Applying machine learning advances to data visualization. *IEEE Transactions on Visualization and Computer Graphics*, 28(12):5134–5153, 2022. 2
- [51] Y. Wang, Z. Hou, L. Shen, T. Wu, J. Wang, H. Huang, H. Zhang, and D. Zhang. Towards natural language-based visualization authoring. *IEEE Transactions on Visualization and Computer Graphics*, 29(1):1222–1232, 2023. doi: 10.1109/TVCG.2022.3209357 3
- [52] Z. Wang, Y. Huang, D. Song, L. Ma, and T. Zhang. PromptCharm: Text-to-image generation through multi-modal prompting and refinement. In *CHI Conference on Human Factors in Computing Systems*, pp. 1–21, 2024. doi: 10.1145/3613904.3642803 2
- [53] J. Wei, X. Wang, D. Schuurmans, M. Bosma, F. Xia, E. Chi, Q. V. Le, D. Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022. 1, 2
- [54] L. Weng, X. Wang, J. Lu, Y. Feng, Y. Liu, H. Feng, D. Huang, and W. Chen. InsightLens: Augmenting llm-powered data analysis with interactive insight management and navigation. *arXiv preprint arXiv:2404.01644*, 2024. 3
- [55] L. Wilkinson. The grammar of graphics. In *Handbook of computational statistics: Concepts and methods*, pp. 375–414. Springer, 2011. 3
- [56] T. Wu, E. Jiang, A. Donsbach, J. Gray, A. Molina, M. Terry, and C. J. Cai. Promptchainer: Chaining large language model prompts through visual programming. In *CHI Conference on Human Factors in Computing Systems Extended Abstracts*, pp. 1–10, 2022. doi: 10.1145/3491101.3519729 2
- [57] T. Wu, M. Terry, and C. J. Cai. Ai chains: Transparent and controllable human-ai interaction by chaining large language model prompts. In *CHI Conference on Human Factors in Computing Systems*, pp. 1–22, 2022. doi: 10.1145/3491102.3517582 2
- [58] Y. Wu, Y. Wan, H. Zhang, Y. Sui, W. Wei, W. Zhao, G. Xu, and H. Jin. Automated data visualization from natural language via large language models: An exploratory study. *Proceedings of the ACM on Management of Data*, 2(3):1–28, 2024. 1, 4
- [59] Z. Wu, V. Le, A. Tiwari, S. Gulwani, A. Radhakrishna, I. Radiček, G. Soares, X. Wang, Z. Li, and T. Xie. NI2viz: natural language to visualization via constrained syntax-guided synthesis. In *Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, pp. 972–983, 2022. 1
- [60] L. Xie, C. Zheng, H. Xia, H. Qu, and C. Zhu-Tian. Waitgpt: Monitoring and steering conversational llm agent in data analysis with on-the-fly code visualization. In *Symposium on User Interface Software and Technology*, pp. 1–14, 2024. doi: 10.1145/3654777.3676374 3
- [61] Y. Xie, Y. Luo, G. Li, and N. Tang. Haichart: Human and ai paired visualization system. *Proc. VLDB Endow.*, 17(11):3178–3191, 14 pages, July 2024. doi: 10.14778/3681954.3681992 2, 6
- [62] W. Yang, C. Chen, J. Zhu, L. Li, P. Liu, and S. Liu. A survey of visual analytics research for improving training data quality. *Journal of Computer-Aided Design & Computer Graphics*, 35(11):1629–1642, 2023. doi: 10.3724/SP.J.1089.2023.2023-00321 2
- [63] W. Yang, M. Liu, Z. Wang, and S. Liu. Foundation models meet visualizations: Challenges and opportunities. *Computational Visual Media*, 10(3):399–424, May 2024. doi: 10.1007/s41095-023-0393-x 2
- [64] S. Yao, D. Yu, J. Zhao, I. Shafran, T. Griffiths, Y. Cao, and K. Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. *Advances in neural information processing systems*, 36:11809–11822, 2023. 2
- [65] B. Yu and C. T. Silva. FlowSense: A natural language interface for visual data exploration within a dataflow system. *IEEE Transactions on Visualization and Computer Graphics*, 26(1):1–11, 2020. doi: 10.1109/TVCG.2019.2934668 2
- [66] J. Yuan, C. Chen, W. Yang, M. Liu, J. Xia, and S. Liu. A survey of visual analytics techniques for machine learning. *Computational Visual Media*, 7(1):3–36, 2021. 2
- [67] D. Zhang, Z. Hu, S. Zhoubian, Z. Du, K. Yang, Z. Wang, Y. Yue, Y. Dong, and J. Tang. Sciinstruct: a self-reflective instruction annotated dataset for training scientific language models. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang, eds., *Advances in Neural Information Processing Systems*, vol. 37, pp. 1443–1473. Curran Associates, Inc., 2024. 6
- [68] Y. Zhu, S. Du, B. Li, Y. Luo, and N. Tang. Are large language models good statisticians? In *NeurIPS*, 2024. 1