Generalizing Numerical Reasoning in Table Data through Operation Sketches and Self-Supervised Learning

Hanjun Cho[1] and **Gahyun Yoo**[1] and **Hanseong Kim**[2] and **Jay-Yoon Lee**[1]*
[1]Seoul National University [2]Soongsil University
{gkswns0531, padme0421, lee.jayyoon}@snu.ac.kr
gkstjd1201@soongsil.ac.kr

Abstract

Numerical reasoning over expert-domain tables often exhibits high in-domain accuracy but limited robustness to domain shift. Models trained with supervised fine-tuning (SFT) on specific datasets tend to rely on header–operation shortcuts rather than structural reasoning. We introduce **TaNOS**, a continual pretraining framework comprising three components: (i) header anonymization to reduce lexical memorization, (ii) operation sketches that provide minimal structural cues, and (iii) self-supervised pretraining that constructs correctness-guaranteed program–question pairs from given tables in a program-first manner. By decoupling domain semantics and numerical operation structure, TaNOS improves the transferability of numerical reasoning. Applied to an 8B instruction-tuned model, TaNOS achieves 80.13% execution accuracy on FinQA with only 10% train data, outperforming SFT baseline (73.97%) with full train data and proprietary models such as GPT-5, Gemini-2.5-Pro. Furthermore, in the domain-shift experiments, TaNOS displays nearly-negligible cross-domain gap (<2pp) when standard SFT shows over 10pp gap. These results suggest that structural guidance with operation sketches, header-agnostic representations, and correctness-guaranteed self-supervision can improve the robustness of numerical reasoning across diverse expert-domain tables.

1 Introduction

Numerical reasoning over tables plays an important role in expert domains such as finance, engineering, and biology, yet remains challenging for language models (Liu et al., 2023). Unlike textual reasoning, it requires models to integrate arithmetic competence with a structural understanding of table schema, and domain-specific semantics.

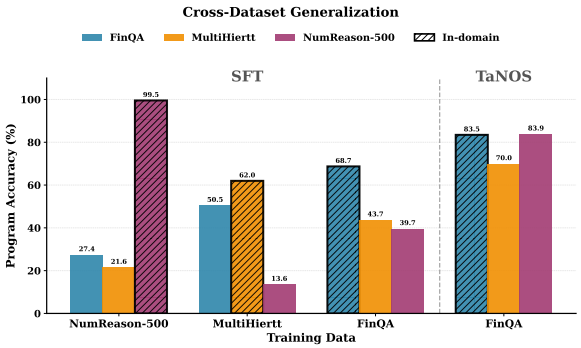

Figure 1: Program Accuracy % on **Cross-Dataset Generalization**. Each group denotes training data; bar colors indicate test data. Hatched bars indicate in-domain evaluation; solid bars denote cross-dataset transfer. SFT degrades under subtle distribution shift. **TaNOS**, pretrained on anonymized NumReason-500 and fine-tuned on FinQA, surpasses in-domain SFT on unseen MultiHiertt when provided with sketch guidance—indicating robust generalization.

A common approach is to fine-tune models on domain-specific datasets (Chen et al., 2021; Zhu et al., 2021; Zhao et al., 2022; Chen et al., 2022b). While this improves in-domain accuracy, it can lead models to rely on surface-level patterns—most notably, direct associations between table headers and operations. As a result, performance often drops under shifts in schema, terminology, or question style.

Figure 1 illustrates this brittleness across three financial datasets. Despite sharing a common source (SEC reports) and exhibiting similar structures and operation types, models trained on one dataset show limited transfer to others. For example, a model trained on FinQA achieves **68.72**% in-domain program accuracy but drops to **43.74**% on MultiHiertt and **39.66**% on NumReason-500

*Corresponding author.

Transactions of the Association for Computational Linguistics, vol. 14, pp. 1993–2011, 2026. https://doi.org/10.1162/TACL.a.788
Action Editor: Roi Reichart. Submission batch: 4/2025; Revision batch: 9/2025; Published 8/2026.

(Appendix Table 8).

Through cross-domain analysis and header-shift experiments, we observe three failure modes that limit the robustness of numerical reasoning:

1. **Reasoning Inefficiency.** Supervision signals allocate disproportionate capacity to trivial arithmetic, enabling strong performance on surface-level patterns, while generalization to unseen schemas and more complex reasoning remains limited.

2. **Data Scarcity for Logical Supervision.** Numerical reasoning requires training data with consistent logical and arithmetic structure, yet standard LLM-based data generation often yields spurious or inconsistent programs, which reduces the amount of useful signal for learning more general reasoning patterns.

3. **Header Dependency.** Models memorize brittle header–operation associations instead of learning relational structures. When headers change, these correlations degrade, exposing substantial lexical overfitting and limited schema generalization.

These observations motivate the use of an **intermediate reasoning abstraction layer** that separates domain-specific lexical cues from the underlying computational structure. Our goal is to design such a layer that exposes structural supervision signals while reducing direct reliance on lexical patterns, with the aim of encouraging more domain-invariant numerical reasoning behavior.

To address this challenge, we introduce **T**able **N**umerical Reasoning with **O**peration **S**ketches (**TaNOS**), a unified framework with three complementary mechanisms designed to disentangle lexical surface forms from structural reasoning:

1. **Operation Sketches.** We introduce minimal symbolic sketches that encode the core computational structure of a query, such as simple compositions of abstract concepts and operators (e.g. % change of x: (x-x)/x). We assume that such sketches are available at test time, as the query itself implicitly specifies this structural information. By making these lightweight cues explicit, the model is relieved from inferring trivial arithmetic details, allowing it to devote its capacity to higher-level contextual and relational reasoning.

2. **Self-Supervised Learning.** We adopt a program-first self-supervised strategy that constructs correctness-guaranteed program–question pairs from unlabeled tables. TaNOS first generates executable programs and answers, and then uses an LLM only to express them as natural-language questions, which helps maintain semantic consistency and allows training without manual annotation at scale.

3. **Header Anonymization.** We propose an *instance-level* header anonymization scheme that assigns unique bijective token mappings to each table. This dynamic mapping preserves structural integrity while weakening header–operation correlations.

Together, these mechanisms are designed to address the three failure modes: operation sketches can reduce reasoning inefficiency by encouraging the model to focus more on contextual reasoning rather than obvious operation structure, self-supervised learning alleviates data scarcity for logical supervision, and instance-level header anonymization mitigates header dependency.

TaNOS unifies these mechanisms to achieve both data efficiency and robust cross-domain generalization. On a Financial TableQA benchmark, TaNOS applied to an 8B instruction-tuned model attains **85.38**% execution accuracy with full supervision, improving over a fully supervised baseline (**73.97**%) and substantially larger proprietary LLMs. With only 10% of labeled data, TaNOS still reaches **80.13**% accuracy, still outperforming the fully supervised and API models. In the domain-shift experiments, TaNOS maintains stable performance with negligible gap **2pp** between domains, compared to over **10pp** performance gap under SFT.

Contributions.

- We analyze how language models struggle with numerical reasoning over expert-domain tables, identifying three failure modes: reasoning inefficiency, data scarcity for logical supervision, and header dependency.

- We introduce *operation sketches*, minimal structural cues that help the model focus less on surface-level arithmetic and more on contextual reasoning, and integrate them with header anonymization and self-supervised

pretraining into **TaNOS**, a unified framework achieving strong in-domain accuracy and robust cross-domain generalization.

- TaNOS maintains competitive performance under limited supervision, suggesting practical utility for expert domains where labeled data is scarce.

2 Related Work

Numerical Reasoning. Numerical reasoning involves locating supporting information from text or tables and composing operations to derive numerical answers. Early work primarily addressed text-based numerical reasoning with datasets such as DROP (Dua et al., 2019) and MathQA (Amini et al., 2019), where models extract numerical evidence directly from passages. HybridQA (Chen et al., 2020) extends this setup by requiring reasoning over both textual and tabular contexts. To better model the structured nature of tables, encoders such as TaBERT (Yin et al., 2020) and TAPAS (Herzig et al., 2020) jointly encode cell values and headers through relational embeddings. More recently, large language models (LLMs) (Chen, 2022; Li et al., 2023; Chen et al., 2022a) have improved general-purpose reasoning capabilities, yet numerical reasoning remains challenging (Ran et al., 2019; Liu et al., 2023). Most prior work focuses on general-domain settings and does not explicitly address the structural biases that arise in expert-domain tables. In contrast, our approach introduces a reasoning-level abstraction that explicitly separates surface lexical patterns from structural reasoning signals, encouraging the learning of domain-agnostic reasoning behavior.

Financial Numerical Reasoning. In the financial domain, several benchmarks evaluate reasoning over text–table pairs. FinQA (Chen et al., 2021; Zhang and Moshfeghi, 2022) and TATQA (Zhu et al., 2021) require reasoning over textual context and aligned financial tables, while MultiHiertt (Zhao et al., 2022) extends this to documents with multiple hierarchical tables. Conversational variants such as ConvFinQA (Chen et al., 2022b) incorporate dialogue history to model multi-turn financial analysis. Although these datasets have contributed to domain-specific numerical reasoning, they typically assume fixed schemas and stable header semantics, which may limit cross-dataset transfer. Prior methods primarily rely on supervised fine-tuning within a single dataset, which can inadvertently couple domain terminology with reasoning logic. TaNOS is intended to mitigate this limitation at the representational level by anonymizing headers and introducing operation sketches that provide abstract computational scaffolds for reasoning across schemas.

Self-Supervised Learning. Self-supervised learning (SSL) has been studied as a means of augmenting reasoning data or injecting symbolic structure into pretrained models. Geva et al. (2020) synthesize paired textual and numerical examples to enhance mathematical reasoning, while Liu et al. (2024) employ SQL-based augmentation to retrieve missing information. However, most SSL approaches focus on textual or symbolic tasks and pay limited attention to structured numerical data. Our method differs in two respects: (i) it extends self-supervision to table-based reasoning by constructing correctness-guaranteed programs prior to question generation, and (ii) it uses LLMs only for controlled linguistic paraphrasing. This design combines symbolic accuracy with natural-language diversity, allowing scalable pretraining that helps preserve reasoning validity and improves generalization across domains.

In summary, prior work has improved architectures and datasets for numerical reasoning, but comparatively less attention has been given to abstractions that enable generalization across expert-domain tables. TaNOS contributes to this direction by integrating header anonymization, operation sketches, and correctness-guaranteed self-supervised pretraining into a unified framework.

3 Table Numerical Reasoning with Operation Sketches (TaNOS)

We propose the **T**able **N**umerical Reasoning with **O**peration **S**ketches (**TaNOS**) framework, which is designed to improve numerical reasoning in expert-domain tables through three main components: (1) **operation sketches** that provide minimal computational guidance toward structural understanding, (2) **header anonymization** that reduces lexical dependencies on domain-specific terms, and (3) **self-supervised learning (SSL)** that allows large-scale pretraining without human annotation. Each component is associated with a distinct failure mode—sketches are intended to encourage more contextual reasoning,

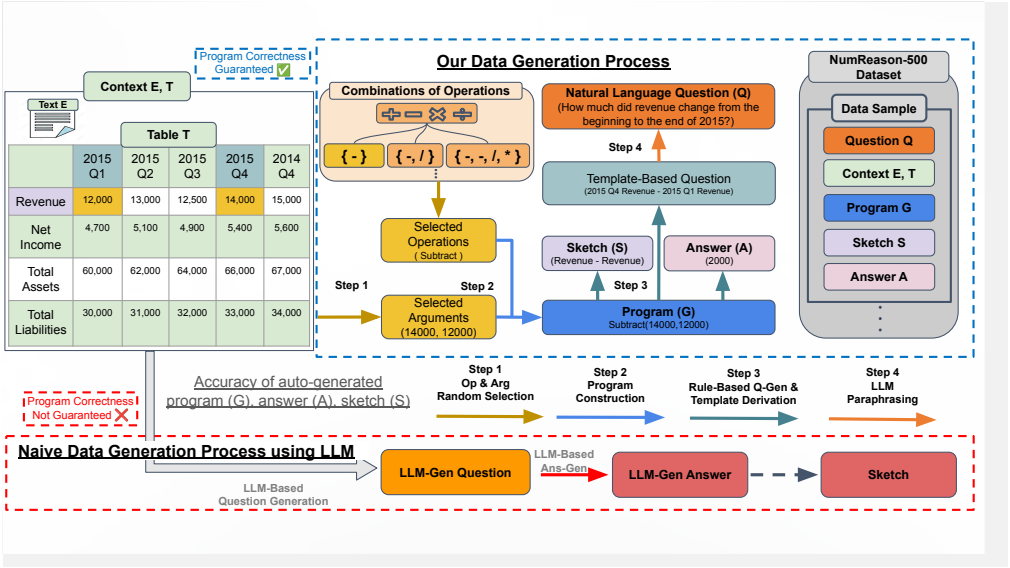

Figure 2: **Automatic data generation process of TaNOS.** Numbered steps correspond to the indicators in the figure: **Step 1: Operation and Argument Sampling.** Sample a sequence of arithmetic operations and randomly select argument cells from the table. **Step 2: Program Construction.** Combine the sampled operation sequence and arguments to form a program G. **Step 3: Program Execution and Template Derivation.** Execute G over (E, T) to obtain the answer A. Then extract row headers of the selected argument cells to build the operation sketch S, and use their row/column headers together with the operation sequence to generate a template-based question Q. **Step 4: LLM Paraphrasing.** Convert the template-based question into a fluent natural-language question using an LLM for paraphrasing. This reverse workflow (program $\rightarrow$ answer $\rightarrow$ question) guarantees internal consistency among G, A, and S by construction (blue dashed box), unlike forward-generation pipelines where correctness of the generated program, answer, and sketch is not guaranteed (red dashed box).

anonymization is intended to reduce lexical bias, and SSL provides additional supervision without labels—and their combination is intended to encourage more structural, rather than surface-level, reasoning. Figure 2 and Figure 3 illustrate the overall process.

3.1 Operation Sketches

Motivation. Operation sketches are minimal yet interpretable computational cues that help models focus on more structural and contextual reasoning rather than memorizing specific operation patterns. They act as lightweight scaffolds that are intended to reduce the need for the model to infer trivial arithmetic relations, and to help the model focus more on complex reasoning processes—such as contextual alignment, temporal comparison, or entity-level inference.

In TaNOS, sketches take two complementary forms: (i) *jargon-defining sketches*, which decompose domain-specific terms into explicit symbolic operations derived from rule-based templates—for example, a question asking for *return on equity* corresponds to a sketch *Net Income / Total Stockholders' Equity*, making the implicit formula explicit; and (ii) *computation-hint sketches*, which correspond to simple computational directions that users may naturally express when posing a question—for instance, *What is the percentage change in operating cash flow from Q1 to Q2?* naturally aligns with a sketch *(Operating Cash Flow - Operating Cash Flow) / Operating Cash Flow*. These hints provide minimal guidance about the intended computation, helping the model focus on higher-level contextual reasoning rather than low-level arithmetic inference.

Sketches do not disclose the final answer or full reasoning path. Instead, they serve as interpretable cues that are intended to help the model focus less on trivial calculation and more on abstract reasoning. (Appendix B. Tables 6 and 7)

Form. A sketch S is represented as a compact symbolic sequence of abstract concepts and operators:

$$S = \mathrm{op}(\mathrm{c}_a, \mathrm{c}_b),$$

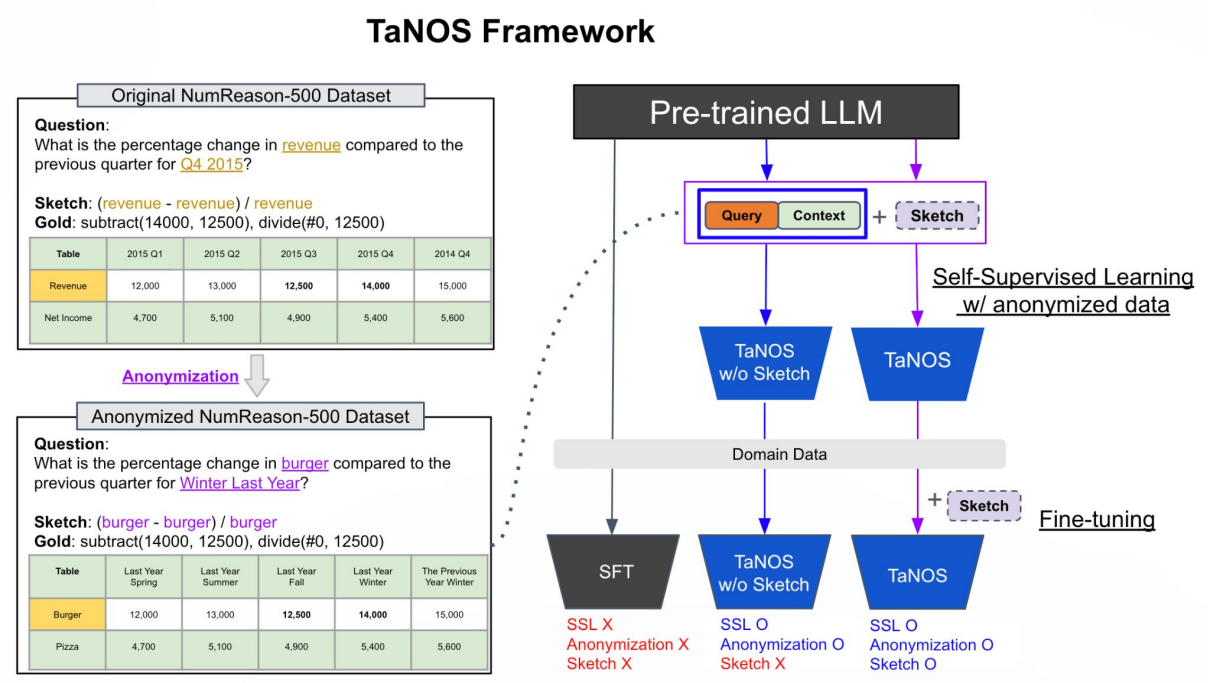

Figure 3: **Overview of TaNOS Framework.** *Left:* Example of instance-level header anonymization, where domain-specific headers (e.g., "revenue") are replaced with arbitrary tokens (e.g., "burger"). *Right:* Training pipeline. Models undergo self-supervised learning on anonymized NumReason-500 with two input formats: `[Query][Context]` or `[Query][Context][Sketch]`. Fine-tuning on labeled domain data yields three system variants: SFT (no SSL), TaNOS w/o Sketch, and TaNOS.

where each `c` denotes a concept token rather than a specific cell. During training, sketches serve as structural blueprints aligning symbolic computation with tabular and linguistic context.

Inference-time availability. At inference time, we assume that a sketch S can be provided as a lightweight specification of the intended operation (e.g., "revenue - revenue"). We expect this to be a relatively mild requirement, as users posing numerical questions over tables often have some sense of the intended computation, such as comparing values or computing ratios. In our experiments, rather than relying on manual annotation, sketches are automatically extracted from the dataset using the same rule-based templates as in training (Section 3.3; see also Figure 2). To assess robustness under more realistic conditions, we additionally evaluate with paraphrased sketches that introduce lexical and syntactic variation (Section 4.5), and find that TaNOS maintains strong performance even with noisy sketch inputs.

Integration into Program Generation. Traditional numerical reasoning models condition program generation on tables T, text E, and questions Q. TaNOS extends this by conditioning on sketches S:

$$P(A \mid T, E, Q, S) = \sum_i P(A \mid G_i, T, E, Q, S) \times P(G_i \mid T, E, Q, S)$$

where G_i represents an executable reasoning program and A denotes the numerical answer. The model is trained to maximize the log-likelihood of gold programs:

$$\mathcal{L}_{\text{prog}} = -\sum_{t=0}^{n} \log P(w_t \mid w_{<t}, T, E, Q, S; \theta)$$

By introducing sketches, TaNOS bridges basic computation and higher-level reasoning through interpretable cues, guiding the model to capture more contextual and abstract reasoning patterns beyond surface-level arithmetic.

3.2 Header Anonymization

Header anonymization is intended to mitigate overfitting to memorized header tokens and to enhance generalization to unseen schemas. Instead of a fixed mapping, TaNOS performs *instance-level anonymization*, assigning distinct anonymized tokens to headers within each training

LLMs (>70B)		LLMs (<10B)				
Model	**Off-the-shelf**	**Model**	**Off-the-shelf**	**SFT (10%)**	**SFT (100%)**	**TaNOS (10%)**	**TaNOS (100%)**
Gemini-2.5-Pro	76.41 (70.00)	*LLaMA-3.2-3B*	10.64 (7.56)	57.56 (52.95)	69.74 (69.74)	70.38 (31.54)	82.31 (31.67)
Claude-4.5-Sonnet	76.79 (73.97)	*Qwen-2.5-3B*	12.44 (13.21)	55.90 (59.23)	68.33 (62.69)	73.97 (43.85)	81.92 (45.64)
GPT-5	**77.82** (71.28)	*Qwen-2.5-7B*	38.08 (35.90)	**72.56** (69.10)	**76.15** (75.26)	79.62 (34.62)	**85.51** (53.21)
Qwen-2.5-72B	60.26 (56.54)	*LLaMA-3.1-8B*	23.59 (21.15)	66.67 (62.44)	73.97 (72.18)	**80.13** (41.15)	85.38 (41.03)

Table 1: **Execution Accuracy (%) on Financial TableQA.** The table reports execution accuracy for proprietary LLMs and instruction-tuned open-source models across five settings: off-the-shelf pretrained LLMs and fine-tuned LLMs(<10B) with SFT, TaNOS using 10% and 100% of the data. By default, sketch guidance is provided at inference time, accuracy in black, and we provide extra result without sketch, indicated in gray. **TaNOS** is trained with sketch, which provides operation sequences, and this training procedure encourages models to concentrate its capacity to higher-level contextual understanding of which values to retrieve from the table. As a result, compact models (3B–8B) achieve better or competitive performance with proprietary LLMs using only 10% of labeled data.

sample. This reduces the tendency of the model to form stable header–operation associations while keeping the table structure intact.

Formally, for each instance we define a bijective mapping $\phi : \mathcal{H} \to \mathcal{V}$ assigning each header $h \in \mathcal{H}$ to a token $v \in \mathcal{V}$ from a reserved vocabulary range:

$$T' = \phi(T), \quad Q' = \phi(Q), \quad S' = \phi(S).$$

Tokens are drawn from the BERT vocabulary (Devlin et al., 2018), excluding special tokens. This approach keeps the table structure intact while reducing stable lexical correlations. Compared to anonymization methods designed primarily for masking, our anonymization specifically targets header–operation correlation biases that can limit cross-domain generalization.

3.3 Automatic Data Generation

To enable scalable self-supervised learning without manual labels, TaNOS automatically constructs reasoning examples (T, E, Q, G, S, A) from unlabeled corporate filings (SEC 10-K reports of S&P 500 companies). Unlike conventional pipelines that generate a question and then compute its answer, TaNOS adopts a reverse-generation strategy—first constructing symbolic programs and answers, then generating corresponding questions (Figure 2). This inversion is designed to leverage the strengths of LLMs in natural-language generation while reducing their weaknesses in arithmetic reasoning, and helps ensure correctness by construction while still allowing for linguistic diversity.

Program and Answer Construction. We construct each program G by sampling operation sequences of length $\ell \in [1, 4]$ from an operator set $\mathcal{O} = \{\texttt{add}, \texttt{sub}, \texttt{mul}, \texttt{div}, \texttt{greater}, \texttt{exp}\}$, assigning operands based on type and unit constraints. Executing G yields an answer A; if execution fails or produces invalid values, we discard the example. This yields (G, A) pairs whose correctness is verified by execution.

Question and Sketch Generation. Given (T, E, G, A), a rule-based generator expresses each operand in G using row and column headers to create canonical templates, which are then paraphrased by an LLM to improve fluency and lexical variation while keeping A fixed. In parallel, a sketch S is automatically generated by abstracting the table structure— specifically, by discarding column headers that carry rich contextual semantics and retaining only row headers that provide simple high-level cues— thereby capturing a high-level description of the operation pattern that is consistent with the anonymized header tokens. All templates are rule-defined and automatically instantiated, and do not require manual effort. After filtering invalid programs, we obtain reasoning examples, forming the **NumReason-500** dataset. (Appendix B)

3.4 Self-Supervised Learning (SSL)

The model is pretrained on the NumReason-500 dataset using self-supervision, treating the automatically generated questions and programs as pseudo-labels. During pretraining, each training example is formatted as either `[Query][Context]` or `[Query][Context][Sketch]`, depending on whether the operation sketch S is included (Figure 3). Accordingly, we employ two pretraining configurations: (i) **TaNOS**, which

incorporates both header anonymization and operation sketches during training; (ii) **TaNOS w/o Sketch**, which performs self-supervised pretraining on anonymized data without sketches.

To encourage active use of sketches, a subset of questions is designed to be difficult to solve without them. Specifically, domain-specific jargon (e.g., "operating margin," "return on equity") is replaced with arbitrary tokens, rendering the question largely semantically opaque; the corresponding sketch thus provides the primary interpretable reasoning signal. This design simulates terminology outside the model's prior knowledge, enabling the model to handle unfamiliar domain-specific concepts through explicit structural guidance rather than memorized associations (Appendix Table 6).

LLMs are used only for natural-language paraphrasing—not for generating reasoning structures—so that the correctness of supervision is determined by the underlying programs rather than by the LLM.

Formally, let $\mathcal{D}$ denote a pretraining dataset, which depends on the configuration:

- $\mathcal{D}_{\text{TaNOS}} = \{(T', E, Q', S', G)\}$: anonymized data with sketches,
- $\mathcal{D}_{\text{w/o Sketch}} = \{(T', E, Q', G)\}$: anonymized data without sketches,

Here, primed variables (e.g., T', Q') denote anonymized inputs, and S denotes the operation sketch. The training objective is:

$$\mathcal{L}_{\text{SSL}} = \mathbb{E}\left[-\log P(G \mid T, E, Q, S; \theta)\right],$$

For brevity, we write (T, E, Q, S, G) generically to denote either original or anonymized inputs; when anonymization is used, T and Q correspond to T' and Q'. In configurations without sketches, S is omitted.

3.5 Domain-Specific Fine-tuning

After SSL pretraining, models are fine-tuned on labeled datasets (e.g., FinQA, MultiHiertt) containing original headers and text. This reintroduces domain-specific language while aiming to preserve the structurally grounded reasoning learned during SSL.

Summary. TaNOS is designed to separate lexical surface information from structural reasoning. Each component is intended to address a different aspect of the problem—anonymization to reduce lexical bias, sketches to encourage more contextual and high-level reasoning, and SSL to provide additional supervision at scale—and their combination is intended to improve both data efficiency and cross-domain generalization in expert-domain numerical reasoning.

4 Experiments

We evaluate TaNOS across multiple expert-domain numerical reasoning benchmarks to assess its impact on performance, robustness under distribution shift, and data efficiency. Our experiments are designed to answer the following questions:

1. **Overall effectiveness.** Does TaNOS improve numerical reasoning performance on expert-domain tables compared to LLMs? (Table 1)
2. **Component-wise robustness.** How do the components of TaNOS—header anonymization, operation sketches, and self-supervised pretraining—contribute to robustness under distribution shift? (Figure 4, Table 2)
3. **Data efficiency and scalability.** How does TaNOS perform under varying amounts of labeled supervision, and does its data-efficiency behavior transfer across models and table benchmarks with different structural characteristics? (Figure 5, Tables 1, 3, 4, and 5)

Financial TableQA Dataset. To benchmark against prior work, we use a subset of the FinQA dataset (Chen et al., 2021). This subset contains table-only questions for which operation sketches can be automatically derived by our rule-based generator. We extract 3,944 training, 550 development, and 780 test examples.

Domain-Specific Datasets. To evaluate cross-domain robustness, we construct four domain-shifted datasets derived from FinQA by replacing financial headers with terminology from other expert domains. For example, the header *Revenue Growth* is replaced with *Tensile Strength* (Mechanical) or *Cell Viability* (Biology), while maintaining the same tabular structure. This design introduces lexical shifts in domain terminology without altering underlying reasoning patterns, which allows us to isolate the effect of lexical changes. Each domain—**Mechanical Engineering**, **Biology**, **Legal**, and **Scientific**—contains

3,944 training, 550 development, and 780 test examples.

Automatically Generated Dataset. For self-supervised learning, we use the automatically generated **NumReason-500** dataset (Section 3.3), derived from unlabeled SEC 10-K filings of **S&P 500** companies between 1983 and 2023.[1] During the construction of the dataset, we paraphrase all questions from rule-based templates using **GPT-3.5**, restricting the model to surface-form rewriting so that the underlying programs and answers remain unchanged. The corpus contains 100,000 synthetic (T, E, Q, G, S, A) tuples, where each answer A is obtained by executing the program G. We release both *original-header* and *anonymized-header* versions of the dataset. For all experiments, the data are split into 70k/20k/10k train-dev-test splits. (Appendix B) Operation sketches are included for every synthetic instance and are optionally used during fine-tuning depending on the ablation configuration (Figure 4).

Together, these datasets provide a varied and controlled evaluation setting for TaNOS. The **Mechanical Engineering**, **Biology**, **Legal**, and **Scientific** datasets are used to evaluate cross-domain robustness and general numerical reasoning behavior under varying domain semantics. In contrast, the **NumReason-500** dataset supports systematic analysis of TaNOS's three key components—operation sketches, header anonymization, and self-supervised learning—by varying their configurations during pretraining and fine-tuning. This dual-level design makes it possible to assess both the model's overall generalization performance and the specific contribution of each proposed mechanism.

Model Setup. We adopt the retriever–generator pipeline from FinQANet (Chen et al., 2021), where a BERT-based classifier retrieves the top-3 supporting facts from linearized table rows and text sentences, which are then passed to the generator. For the generator, we use **LLaMA-3.1-8B-Instruct** (Dubey et al., 2024) as our **primary backbone**, and unless otherwise specified, all TaNOS variants are instantiated with this model. This model was selected because it is widely used and has been shown to perform reliably on reasoning benchmarks, which facilitates reproducible comparisons. To analyze the impact of model scale and to assess the applicability of TaNOS to different architectures, we also train and evaluate other models, specifically **LLaMA-3.2-3B-Instruct** and the **Qwen-2.5** (Qwen et al., 2025) series (**Qwen-2.5-3B-Instruct** and **Qwen-2.5-7B-Instruct**).

Furthermore, to contextualize TaNOS, we compare against proprietary LLMs (**GPT-5** (Singh et al., 2025), **Claude-4.5-Sonnet**, **Gemini-2.5-Pro** (Comanici et al., 2025)) and a larger open-weight model (**Qwen-2.5-72B-Instruct**, 4-bit quantized). All open-weight models used in our experiments are **instruction-tuned** variants. Implementation details, including hyperparameters and prompts, are provided in Appendix A.

Evaluation Metrics. We assess model performance using two metrics: **Program Accuracy** and **Execution Accuracy**.

Program Accuracy measures the exact match between the predicted and gold programs, requiring identical operation sequences and arguments. This metric directly targets the model's structural reasoning, so that correctness reflects whether the model recovers the intended reasoning path. Since our work focuses on structured reasoning via program generation, this is the primary evaluation metric in most of our experiments.

Execution Accuracy evaluates the numerical correctness of the final derived answer. For direct-answering LLMs, we parse the textual response to extract the final numerical value. To address superficial variations—such as decimal precision, formatting differences, or insignificant rounding discrepancies—we apply standard normalization to both the predicted and gold values. (Appendix A) A prediction is deemed correct if the normalized value is numerically equivalent to the gold answer. For program-generating models (including TaNOS), the generated program is executed to produce a deterministic result, which is then evaluated using the same protocol.

4.1 In-domain Performance

We examine in-domain performance on the *Financial TableQA*. Results are summarized in Table 1.

TaNOS vs. Large Language Models. Across all evaluated systems, **TaNOS (100%)** applied to the Qwen-2.5-7B backbone achieves the highest execution accuracy of **85.51%**. This corresponds to a gap of about 7–9 percentage points compared

[1] SEC filings of S&P 500 companies.

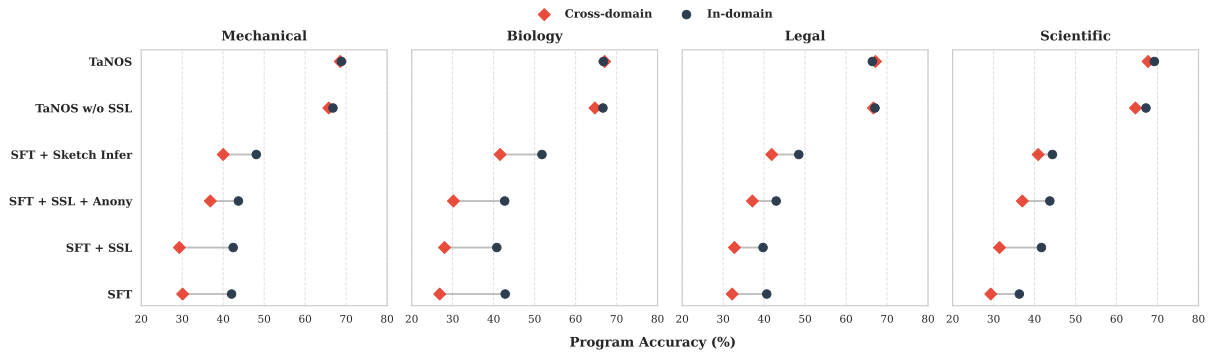

Figure 4: Program accuracy (%) under **Domain Shift**. Models are trained on 10% of in-domain and evaluated in one in-domain (blue) and four out-domains (red, averaged). Here, out-domains share the same table structure with in-domain, but have row, column headers replaced from each out-domain. Horizontal distances between markers indicate robustness to domain shift: **SFT** suffers large cross-domain drops, whereas **TaNOS** improves both absolute accuracy and shrinks the cross-domain gap to a nearly negligible level. Full numerical results are provided in **Appendix Table 11**.

to proprietary models such as GPT-5, Gemini-2.5-Pro, and Claude-4.5-Sonnet, even when they are given operation sketches at inference time. The compact LLaMA-3.2-3B model also reaches **82.31%** with TaNOS, which is comparable to or higher than the proprietary systems in our evaluation. These results suggest that small-scale open models can perform competitively with closed-source LLMs on specialized numerical reasoning tasks. Collectively, these findings indicate that TaNOS can improve structural numerical reasoning over expert-domain tables. (Appendix Table 9)

Sketches as training signals rather than prompts. Operation sketches yield consistent but modest improvements when used as inference prompts for general LLMs. For instance, Qwen-2.5-72B gains about 3.72pp, and proprietary models such as GPT-5 (6.54pp) and Claude-4.5-Sonnet (2.82pp) show similarly limited improvements. The SFT baseline for LLaMA-3.1-8B reaches 73.97% when provided with sketches. In contrast, TaNOS (100%) uses these sketches as structural training signals and achieves 85.38%, about 11 percentage points higher than the SFT baseline. This indicates that our proposed training of TaNOS with sketches allows the model to dedicate its capacity to higher-order contextual reasoning of which values to retrieve from the table, rather than allocating capacity to infer operations from scratch. As a byproduct, TaNOS becomes dependent on sketch guidance at inference time: without it, performance drops to 41.03%. In typical use cases, however, users formulating numerical questions are likely to already have an implicit understanding of the intended operations (e.g., sum, average, difference), making this requirement relatively lightweight.

Data efficiency and utility in privacy-constrained environments. TaNOS improves data efficiency. With 10% of the labeled data, the LLaMA-3.1-8B model reaches 80.13%. This is higher than the **SFT (100%)** baseline (73.97%) and the **GPT-5** (77.82%), even when both are provided with operation sketches. This setting is particularly relevant for privacy-sensitive sectors such as finance, where regulatory constraints can limit the use of external cloud-based APIs and encourage the deployment of local models. These results suggest that TaNOS can provide competitive reasoning performance in such local deployments, using a relatively compact model even when labeled data are limited.

4.2 Component-wise Analysis of Cross-domain Generalization

We evaluate cross-domain generalization on four domain-shifted variants of the Financial TableQA benchmark: Mechanical, Biology, Legal, and Scientific. The overall trends are summarized in Figure 4, and detailed numerical results for all train–test domain pairs are provided in Appendix Table 11. For each training domain, we report average accuracy across all evaluation domains (**Avg.**), accuracy on the training (in-domain) domain (**In-domain**), and the mean over the remaining target domains (**Cross-domain**).

Gaps under standard supervised fine-tuning. Purely **SFT** shows a systematic gap between in-domain and cross-domain performance. Across all training domains, accuracy drops under distribution shift—for example, from 42.05 to 29.77 in Mechanical, from 42.82 to 27.21 in Biology, and from 36.28 to 27.87 in Scientific—corresponding to decreases of roughly 8–16pp across domains. Adding SSL on non-anonymized data (**SFT + SSL**) does not substantially improve cross-domain performance, with scores remaining similar (e.g., Mechanical 29.36, Biology 28.57). These results suggest that conventional SSL, when applied to original headers, may preserve domain-specific lexical associations.

Anonymized SSL improves domain transfer. Introducing header anonymization during SSL yields consistent cross-domain improvements, while in-domain changes remain modest. With **SFT + SSL + Anonymization**, cross-domain accuracy increases across all training domains: from 29.36 to 36.80 in Mechanical, 28.57 to 30.89 in Biology, 31.91 to 36.84 in Legal, and 30.04 to 36.11 in Scientific. This pattern indicates that anonymization reduces reliance on lexical cues specific to the source domain and encourages more domain-agnostic reasoning.

Sketches as structural supervision. To isolate the impact of operation sketches as a training signal, we compare **SFT + Sketch Inference** (sketches as test-time prompts) against **TaNOS w/o SSL** (sketches as training targets). Using sketches for supervision yields clear improvements in both in-domain and cross-domain performance compared to using them only as prompts at inference time. For instance, in the Mechanical domain, **In-domain** accuracy improves from 48.08% to 66.79% (+18.7pp). **Cross-domain** performance shows an even larger gain, from 40.00% to 66.08% (+26.1pp). This dual improvement suggests that explicitly *learning* structural reasoning paths helps the model capture underlying reasoning patterns, leading to better task performance and improved robustness under distribution shift (see Appendix Table 11).

Combined effect: stable cross-domain performance. Combining anonymized SSL with sketch-based supervision yields the most consistent cross-domain performance among our configurations. Across all training domains, **TaNOS** attains the highest average accuracy (e.g., Mechanical 68.48, Biology 66.97, Legal 66.98, Scientific 67.97), while the gap between in-domain and cross-domain performance is reduced to within about 2pp. This pattern suggests that anonymized SSL and sketch-based supervision play complementary roles, with anonymization reducing residual lexical dependencies and sketches providing additional structural guidance, resulting in more stable performance under domain shift.

Taken together, these findings indicate that sketches provide a useful structural learning signal for program-level reasoning, while anonymized SSL encourages more domain-general representations. In combination, these components help TaNOS generalize more reliably across diverse expert domains and outperform SFT baselines under distribution shift. This pattern also extends to a smaller RoBERTa backbone, with even more pronounced gains (Appendix Table 12).

Train Setting	Test Setting	
	Original	Anonymized
Off-the-shelf	15.87	14.44
Original	99.54	83.21
Anonymized	99.42	**99.25**

Table 2: Program accuracy (%) on **NumReason-500 under header anonymization** (inference-only baseline). Original-header training drops under anonymized evaluation, whereas anonymized-header training stays stable across both.

4.3 Effectiveness of Anonymization

Table 2 examines the effect of header anonymization on lexical robustness. To ensure that improvements stem from generalization rather than token overlap, the anonymization tokens used during training and testing are drawn from disjoint sets. Models trained on the original headers achieve near-perfect accuracy on the matched test set (99.54), but their performance drops to 83.21 when evaluated under anonymized headers, suggesting a sensitivity to memorized lexical cues. In contrast, models trained with anonymized headers maintain consistently high accuracy across both evaluation settings (99.42 on original, 99.25 on anonymized). Taken together, these results indicate that anonymization reduces reliance on specific headers and acts as a regularization strategy that improves robustness to lexical variation.

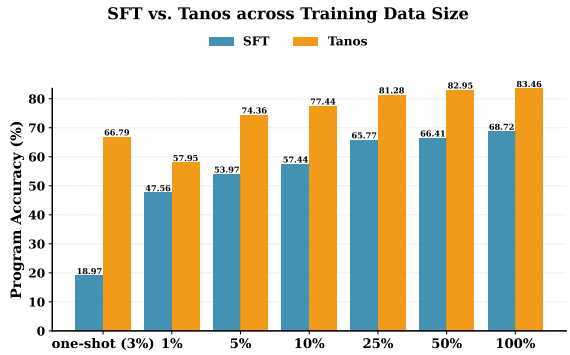

Figure 5: Program accuracy (%) of TaNOS and the SFT baseline across **varying fractions of Financial TableQA** training data. TaNOS consistently surpasses SFT, with particularly strong gains in the **one-shot training** setting, indicating stronger pattern-level generalization.

4.4 Robustness to Training Data Scale

We analyze how TaNOS scales with the amount of supervision by comparing it to the SFT baseline across a wide range of data sizes (Figure 5). We observe three trends across different data regimes.

One-shot Training per Operation Pattern. In this setting, training is constrained to exactly one example for each operation pattern, yielding a **one-shot training** regime at the pattern level. Under this supervision, SFT attains 18.97% accuracy, indicating poor pattern generalization from such limited exposure. In contrast, TaNOS achieves 66.79% in the same setting, surpassing the baseline trained on 50% of the data (66.41%) and approaching the baseline trained on 100% (68.72%), demonstrating robust pattern-level generalization from a single example per pattern.

Low- to mid-resource regimes. With 1–10% of the data, TaNOS attains higher accuracy than the baseline (approximately 10pp at 1%, 20pp at 5%, and 20pp at 10%). This gap indicates improved sample efficiency and the ability to recover strong program accuracy from limited labeled examples.

High-resource regimes. Even with 25–100% of the data, TaNOS maintains an advantage of around 15pp over the SFT. Accuracy increases smoothly as data grows, suggesting that the model continues to benefit from sketch-level structural guidance.

4.5 Robustness to Sketch Variations

To evaluate sensitivity to the surface form of operation sketches, we paraphrased all sketches using GPT-5, yielding noticeable lexical and syntactic variation (BLEU = 0.2691). The results in Table 3 show that accuracy decreases for all configurations, but performance remains relatively high, indicating that the models do not rely solely on the exact phrasing of the sketches.

Method	Original	Paraphrased	Exec. Para.
SFT + Sketch w train & infer (10%)	74.49	69.62	72.56
TaNOS w/o Anony (10%)	75.38	71.67	74.36
TaNOS (10%)	77.44	72.95	75.90
TaNOS (100%)	**83.46**	**75.90**	**78.97**

Table 3: Program accuracy (%) on performance under **paraphrased operation sketches** (BLEU = 0.2691). Paraphrases generated by GPT-5 introduce noticeable lexical and syntactic variation. Execution accuracy under paraphrased sketches is also reported.

More extensive supervised training is associated with greater sensitivity to paraphrasing: **TaNOS (Full)** drops from 83.46% to 75.90% (-7.6pp), whereas the 10%-data variant declines more modestly (-4.5pp). This pattern suggests that extensive fine-tuning increases dependence on specific linguistic realizations.

Systems leveraging SSL and header anonymization are more robust to paraphrased operation sketches. Even when sketches are provided during both training and inference, removing both SSL and anonymization (SFT + Sketch) yields the lowest program accuracy (69.62%), while removing only anonymization (TaNOS w/o Anony) improves robustness (71.67%). The full **TaNOS** model achieves the best performance (72.95%), indicating that **SSL** and **header anonymization** play complementary roles in interpreting sketches beyond their surface lexical form.

4.6 Generalization to Human-Curated Biology Data

To evaluate cross-domain robustness beyond synthetic data, we assess human-curated biology dataset containing 122 expert-authored questions (Table 4). These questions introduce new linguistic constructions, reasoning styles, and table layouts that are not present in the training.

Limited effects of anonymization and SSL under structural divergence. SFT yields moderate performance (57.38% with 10% data, 62.30% with 100%), and adding SSL provides no improvement (56.56% and 61.48%). **TaNOS w/o Sketch**, which applies anonymized SSL without

Method	10% Data	100% Data
SFT	57.38	62.30
SFT + SSL	56.56	61.48
TaNOS w/o Sketch	58.20	59.84
TaNOS w/o SSL	62.18	68.85
TaNOS	**63.85**	**70.26**

Table 4: Program accuracy (%) on the **expert-curated Biology dataset**. Models are trained on a finance-domain dataset and evaluated on expert-authored biology questions. TaNOS achieves the highest accuracy under both supervision budgets, indicating improved transfer to non-synthetic data.

Method	10% Data	100% Data
SFT	51.43	61.96
SFT + SSL	55.27	61.96
TaNOS w/o Sketch	54.77	63.44
TaNOS w/o SSL	68.28	82.65
TaNOS	**70.63**	**82.90**

Table 5: Program accuracy (%) on the **MultiHiertt benchmark**, which requires hierarchical multi-table numerical reasoning. TaNOS achieves the highest accuracy under both supervision budgets, indicating that it extends to complex tables.

sketch guidance, also underperforms the baseline at 100% data (59.84% vs. 62.30%). This suggests that when table structures differ substantially from the pretraining distribution, header anonymization and SSL offer only limited benefits by themselves.

Sketches enable high-level contextual reasoning. In contrast, sketch-guided supervision remains effective under substantial structural divergence. **TaNOS w/o SSL** surpasses SFT by 4.8pp at 10% data and 6.6pp at 100%, indicating that sketches guide the model beyond surface-level cues toward higher-level contextual reasoning. The full **TaNOS** model further improves accuracy (by 6.5pp and 8.0pp over SFT), showing that anonymization and SSL become most useful when combined with sketch supervision.

4.7 Generalization to Structurally Complex Multi-Table Reasoning

The preceding experiments focus on relatively simple table structures. To evaluate whether TaNOS extends to more complex layouts, we assess it on MultiHiertt, which requires reasoning across multiple nested tables (Table 5).

Larger gains on complex structures. The trends observed on Financial TableQA carry over to MultiHiertt, but with larger relative gains. SFT achieves 61.96% program accuracy with 100% data. In contrast, **TaNOS** reaches **82.90%**, a gain of roughly 21pp over the baseline, which exceeds the corresponding improvement on Financial TableQA (+14pp; Table 10).

5 Conclusion

We introduced **TaNOS**, a framework for numerical reasoning over expert-domain tables that is designed to decouple surface-level lexical patterns from underlying computational structure. The framework combines header anonymization, operation sketches, and correctness-guaranteed self-supervised pretraining to provide more structurally grounded supervision for program generation. Experiments across multiple table benchmarks indicate that TaNOS can reduce performance degradation under domain shift while maintaining competitive in-domain performance. These results suggest that lightweight structural abstractions, when paired with appropriate pretraining, may offer a practical path toward more generalizable numerical reasoning over expert-domain tables. We view this work as an initial step toward closer integration between symbolic structure and neural models, and hope it can serve as a basis for further investigation into structurally grounded numerical reasoning.

6 Limitations

TaNOS has several limitations. First, the automatic data generation process assumes well-structured tables, and its effectiveness may diminish when applied to highly irregular, noisy, or layout-dependent tabular formats. We do not evaluate such settings in this work. Second, TaNOS currently assumes that, at inference time, users may provide lightweight operation sketches that convey minimal computational intent. This assumption can hold in simple reasoning scenarios, but automating sketch generation—for example, via a small auxiliary model—remains an important direction for future work, especially in settings where user-provided hints are unavailable.

Acknowledgments

This work was supported in part by the National Research Foundation of Korea (NRF) grant (RS-2023-00280883, RS-2023-00222663); by the National Research Foundation, Korea, under project BK21 FOUR (Dept. of Data Science, SNU, No. 5199990914569); by the Korea Institute of Science and Technology Information (KISTI) in 2026 (No. (KISTI)K26L3M1C1), aimed at developing KONI (KISTI Open Neural Intelligence), a large language model specialized in science and technology; and by the Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (RS-2025-02263754, Human-Centric Embodied AI Agents with Autonomous Decision-Making); by grant (25202MFDS003) from Ministry of Food and Drug Safety in 2025; by AI-BIO Research Grant through Seoul National University; Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No. RS-2025-25442149, LG AI STAR Talent Development Program for Leading Large-Scale Generative AI Models in the Physical AI Domain).

References

Aida Amini, Saadia Gabriel, Shanchuan Lin, Rik Koncel-Kedziorski, Yejin Choi, and Hannaneh Hajishirzi. 2019. Mathqa: Towards interpretable math word problem solving with operation-based formalisms. *arXiv preprint arXiv:1905.13319*.

Wenhu Chen. 2022. Large language models are few (1)-shot table reasoners. *arXiv preprint arXiv:2210.06710*.

Wenhu Chen, Xueguang Ma, Xinyi Wang, and William W Cohen. 2022a. Program of thoughts prompting: Disentangling computation from reasoning for numerical reasoning tasks. *arXiv preprint arXiv:2211.12588*.

Wenhu Chen, Hanwen Zha, Zhiyu Chen, Wenhan Xiong, Hong Wang, and William Wang. 2020. Hybridqa: A dataset of multi-hop question answering over tabular and textual data. *arXiv preprint arXiv:2004.07347*.

Zhiyu Chen, Wenhu Chen, Charese Smiley, Sameena Shah, Iana Borova, Dylan Langdon, Reema Moussa, Matt Beane, Ting-Hao Huang, Bryan Routledge, and William Yang Wang. 2021. Finqa: A dataset of numerical reasoning over financial data. *arXiv preprint arXiv:2109.00122*.

Zhiyu Chen, Shiyang Li, Charese Smiley, Zhiqiang Ma, Sameena Shah, and William Yang Wang. 2022b. Convfinqa: Exploring the chain of numerical reasoning in conversational finance question answering. *arXiv preprint arXiv:2210.03849*.

Gheorghe Comanici, Eric Bieber, Mike Schaekermann, Ice Pasupat, Noveen Sachdeva, Inderjit Dhillon, Marcel Blistein, Ori Ram, Dan Zhang, Evan Rosen, et al. 2025. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *arXiv preprint arXiv:2507.06261*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Dheeru Dua, Yizhong Wang, Pradeep Dasigi, Gabriel Stanovsky, Sameer Singh, and Matt Gardner. 2019. Drop: A reading comprehension benchmark requiring discrete reasoning over paragraphs. *arXiv preprint arXiv:1903.00161*.

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 e-prints*, pages arXiv–2407.

Mor Geva, Ankit Gupta, and Jonathan Berant. 2020. Injecting numerical reasoning skills into language models. *arXiv preprint arXiv:2004.04487*.

Jonathan Herzig, Paweł Krzysztof Nowak, Thomas Müller, Francesco Piccinno, and Julian Martin Eisenschlos. 2020. Tapas: Weakly supervised table parsing via pre-training. *arXiv preprint arXiv:2004.02349*.

Xianzhi Li, Xiaodan Zhu, Zhiqiang Ma, Xiaomo Liu, and Sameena Shah. 2023. Are chatgpt and gpt-4 general-purpose solvers for financial text analytics? an examination on several typical tasks. *arXiv preprint arXiv:2305.05862*.

Hanmeng Liu, Ruoxi Ning, Zhiyang Teng, Jian Liu, Qiji Zhou, and Yue Zhang. 2023. Evaluating the logical reasoning ability of chatgpt and gpt-4. *arXiv preprint arXiv:2304.03439*.

Yujian Liu, Jiabao Ji, Tong Yu, Ryan Rossi, Sungchul Kim, Handong Zhao, Ritwik Sinha, Yang Zhang, and Shiyu Chang. 2024. Augment before you try: Knowledge-enhanced table question answering via table expansion. *arXiv preprint arXiv:2401.15555*.

Qwen, :, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tianyi Tang, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. 2025. Qwen2.5 technical report.

Qiu Ran, Yankai Lin, Peng Li, Jie Zhou, and Zhiyuan Liu. 2019. Numnet: Machine reading comprehension with numerical reasoning. 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)*.

Aaditya Singh, Adam Fry, Adam Perelman, Adam Tart, et al. 2025. OpenAI GPT-5 system card. *arXiv preprint arXiv:2601.03267*.

Pengcheng Yin, Graham Neubig, Wen-tau Yih, and Sebastian Riedel. 2020. Tabert: Pretraining for joint understanding of textual and tabular data. *arXiv preprint arXiv:2005.08314*.

Jiaxin Zhang and Yashar Moshfeghi. 2022. ELASTIC: Numerical reasoning with adaptive symbolic compiler. In *Advances in Neural Information Processing Systems*.

Yilun Zhao, Yunxiang Li, Chenying Li, and Rui Zhang. 2022. Multihiertt: Numerical reasoning over multi hierarchical tabular and textual data. *arXiv preprint arXiv:2206.01347*.

Fengbin Zhu, Wenqiang Lei, Youcheng Huang, Chao Wang, Shuo Zhang, Jiancheng Lv, Fuli Feng, and Tat-Seng Chua. 2021. Tat-qa: A question answering benchmark on a hybrid of tabular and textual content in finance. *arXiv preprint arXiv:2105.07624*.

A Implementation Details

This appendix summarizes the set of implementation details required to reproduce our experiments. All models were fine-tuned using the HuggingFace `transformers` library, and we report only the components that directly affect training dynamics and evaluation.

A.1 Chat Prompt Format

We adopt a chat-style interface for all open-weight models. The system message is fixed as:

- **System**: "You are an assistant that returns only the math program solving the user request. Respond with the program text only."

For each training example, the user turn is constructed by concatenating: (i) the question `qa["question"]`, (ii) the linearized table and textual context `qa["model_input"]`, and (optionally) (iii) the gold sketch `qa["explanation_math"]`, separated by a literal `[SEP]` token:

$$\text{user} = \text{question}\ [\texttt{SEP}]\ \text{context}\ [\texttt{SEP}]\ \text{sketch}.$$

We rely on the built-in `chat_template` of each tokenizer to render the (System, User, Assistant) turns whenever available; otherwise, we fall back to a simple textual prefix ("System:", "User:", "Assistant:").

During supervised training, the gold program is appended as the assistant turn. All tokens preceding the assistant prefix are masked with label -100, so that the model is trained *only* on the program tokens.

A.2 Training Hyperparameters

All supervised fine-tuning experiments (TaNOS, TaNOS w/o SSL, and SFT baselines) use the following configuration:

- Backbone:

  - `LLaMA-3.1-8B-Instruct`
  - `LLaMA-3.2-3B-Instruct`
  - `Qwen-2.5-7B-Instruct`
  - `Qwen-2.5-3B-Instruct`
- Optimizer: AdamW
- Learning rate: 1×10^{-5}
- Epochs: 10
- Effective batch size: 16 (batch size 4, gradient accumulation 4)
- Max sequence length: 1024
- LR schedule: constant LR with 3% warmup
- Precision: `bf16`
- Gradient checkpointing: enabled
- Random seed: 42
- Model selection: unless otherwise specified, we select the checkpoint achieving the highest program accuracy on the dev set and report its test performance.

SSL pretraining. For self-supervised learning (SSL), we use the same configuration except for:

- **Epochs**: 1
- **Scheduler**: cosine LR schedule

A.3 Program Generation and Decoding

At evaluation time (both for development-time model selection and final test evaluation), we generate programs with greedy decoding:

- **Decoding**: `do_sample = False`
- **Max new tokens**: 128
- **Termination**: model EOS token

We always discard the prompt portion of the sequence and keep only tokens generated *after* the final user turn. For batched generation, tokenizers use right padding during training and left padding during evaluation for efficiency, with the pad token tied to the EOS token.

A.4 Metrics: Program and Execution Accuracy

Program accuracy. To compute program accuracy, we post-process the generated text as follows: (i) we take the substring after the last occurrence of the token "assistant" in the decoded text; (ii) we strip leading punctuation and whitespace; and (iii) we select the first non-empty line as the predicted program. We then compute exact string match between this line and the gold program. All reported program accuracies in the paper are based on this strict exact match criterion.

Execution accuracy. For execution accuracy, we evaluate generated programs with a custom interpreter that implements the arithmetic and table operations used in FinQA (e.g., `add`, `subtract`, `multiply`, `divide`, `table_sum`, `table_average`, `table_max`, `table_min`). The interpreter executes the sequence of operations and returns a scalar numeric answer. To decide correctness, we apply a FinQA-style numeric matching rule: we first normalize both the predicted and gold answers by lowercasing, removing commas, currency symbols, and unit markers (e.g., "million", "k"), and stripping extraneous punctuation and whitespace. We then interpret simple numeric forms such as percentages (with or without the "%" sign), ratios and fractions (e.g., "3/4"), and map `yes`/`no`-style outputs to booleans. If both answers contain numeric values, we require them to have the same sign and to agree after rounding to the number of decimal places specified in either the gold answer or the prediction. We do not apply any additional numeric tolerance beyond this rounding-based equivalence.

A.5 Prompt for Non-finetuned Model Evaluation

For all non-finetuned models—including API-based LLMs (e.g., **GPT-5**, **GPT-5-Nano**, **Claude-4.5-Sonnet**, **Claude-4.5-Haiku**, **Gemini-2.5-Pro**, **Gemini-2.5-Flash-Lite**) and open-weight instruction-tuned models evaluated in the inference-only setting (e.g., **LLaMA-3.3-70B-Instruct**, **Qwen-2.5-72B-Instruct**, **LLaMA-3.1-8B-Instruct**, **LLaMA-3.2-3B-Instruct**, **Qwen-2.5-7B-Instruct**, **Qwen-2.5-3B-Instruct**)—we use a shared instruction-style prompt:

```
Based on the provided
context and table, answer
the question.
Context:
{context}
Table:
{table}
Question: {question}
```

```
Important: Provide only the
exact numeric answer. Do not
include any explanation or
additional information.
```

Here, `{context}` is formed by concatenating the pre- and post-text segments from the FinQA instance, and `{table}` is a row-wise linearization of the table. When evaluating the sketch condition, we append an additional block:

```
Sketch:
{explanation_math}
```

Model outputs are treated as numeric answers and evaluated with the same normalization and rounding-based numeric matching rule as described in the *Execution accuracy* subsection.

A.6 Hardware and Quantization

All experiments were executed on a single **NVIDIA H100 SXM (80GB) GPU**. We enable deterministic CUDA behavior (`torch.use_deterministic_algorithms`) and disable cuDNN benchmarking to improve reproducibility.

Large open-weight models (LLaMA-3.3-70B-Instruct and Qwen-2.5-72B-Instruct) are loaded with 4-bit NF4 quantization using the bitsandbytes library, with `bfloat16` computation and `device_map="auto"` for efficient inference. All open-weight models used in our experiments are instruction-tuned variants.

B NumReason-500 Jargon and Examples

Key	Jargon
operation_0	goodwill + intangibleAssets - otherFinancingActivities + minorityInterest
operation_1	commonStock + commonStock
operation_2	const_100 × [numberOfShares - costAndExpenses]

Table 6: Examples of operation-level jargon definitions in NumReason-500.

Examples
Question: What is the *combined amount of* other financing activities in Q3 and the full-year operating cash flow? **Sketch**: Other Financing Activities + Operating Cash Flow **Program**: `add(-3000000.0, 1697000000.0)` **Answer**: 1694000000.0
Question: What is the *percentage change* in operating cash flow from Q1 to Q2? **Sketch**: (Operating Cash Flow − Operating Cash Flow)/Operating Cash Flow **Program**: `subtract(85900000.0, 82100000.0), divide(#0, 82100000), multiply(#1, 100)` **Answer**: 4.63
Question: What is DVN's Q4 2021 operation_2? **Sketch**: const_100 × [Number Of Shares - Cost And Expenses] **Program**: `subtract(671000, 257600), multiply(const_100, #0)` **Answer**: 41340000.0
Question: What is SWKS's 2016 operation_1 from Q3 to Q4? **Sketch**: Common Stock + Common Stock **Program**: `add(46900000, 46900000)` **Answer**: 93800000.0
Question: What is DLTR's Q4 2007 Return On Equity? **Sketch**: Net Income / Total Stockholders Equity **Program**: `divide(97600000, 1167700000)` **Answer**: 0.0835

Table 7: Illustrative examples from the NumReason-500 dataset before anonymization. Operation identifiers (jargon) are highlighted in teal, and row headers are shown in magenta. The last example corresponds to a standard financial ratio (*return on equity*, ROE). After anonymization, the question text alone no longer reveals this underlying financial concept, so the model must rely on the sketch to recover the correct program.

C Full Experiment Results

C.1 Cross-Dataset Generalization

	Test Data		
Train Data	FinQA	MultiHiertt	NumReason-500
FinQA (SFT)	<u>68.72</u>	43.74	39.66
MultiHiertt (SFT)	50.51	<u>61.96</u>	13.60
NumReason-500 (SFT)	27.44	21.56	<u>99.54</u>
FinQA (TaNOS)	**<u>83.46</u>**	**70.01**	**83.89**

Table 8: Program accuracy (%) on **Cross-Dataset Generalization**. Rows denote training data, columns test data. In-domain results are underlined. SFT degrades under subtle distribution shift. **TaNOS**, pre-trained on anonymized NumReason-500 and fine-tuned on FinQA, surpasses in-domain SFT on unseen MultiHiertt when provided with sketch guidance—indicating robust generalization.

C.2 Execution Accuracy of Models

Model	**Execution Accuracy (%)**	
	✗	✓
GPT-5	71.28	77.82
Claude-4.5-Sonnet	73.97	76.79
Gemini-2.5-Pro	70.00	76.41
GPT-5-Nano	69.49	71.41
Claude-4.5-Haiku	68.33	71.54
Gemini-2.5-Flash-Lite	42.05	41.03
Qwen-2.5-72B-Instruct (4-bit)	56.54	60.26
LLaMA-3.3-70B-Instruct (4-bit)	50.77	51.92
LLaMA-3.1-8B-Instruct	21.15	23.59
Qwen-2.5-7B-Instruct	35.90	38.08
Qwen-2.5-3B-Instruct	13.21	12.44
LLaMA-3.2-3B-Instruct	7.56	10.64

Table 9: Execution accuracy (%) on **Financial TableQA for API-based and open-weight LLMs.** Columns report execution accuracy with (✓) and without (✗) sketch guidance at inference time.

C.3 Program Accuracy of Small- and Mid-Scale Models

Small- and mid-scale models	*LLaMA-3.2-3B*		*Qwen-2.5-3B*		*LLaMA-3.1-8B*		*Qwen-2.5-7B*	
Sketch	✗	✓	✗	✓	✗	✓	✗	✓
SFT (10%)	47.69	52.05	50.90	50.26	57.44	62.82	58.59	**63.97**
SFT (100%)	62.05	62.82	57.05	61.67	68.72	**69.87**	66.28	66.54
TaNOS (10%)	28.85	67.31	23.21	71.28	38.08	**77.44**	27.95	76.28
TaNOS (100%)	28.08	79.87	36.28	79.36	37.05	**83.46**	44.87	82.82

Table 10: Program Accuracy (%) on **Financial TableQA for small- and mid-scale models.**

C.4 Full Cross-Domain Experiment Results

Train Domain	Method	Data	Avg.	In-domain	Cross-domain	Mechanical	Biology	Legal	Scientific	Anonymized
Mechanical	SFT	10%	31.82	42.05	29.77	42.05	34.10	26.79	31.54	27.95
	SFT + SSL	10%	31.54	42.44	29.36	42.44	31.92	27.05	28.85	29.36
	SFT + SSL + Anonymization	10%	37.95	43.72	36.80	43.72	37.69	34.87	37.95	36.92
	SFT + Sketch Inference	10%	41.62	48.08	40.00	48.08	40.64	37.44	41.79	40.13
	TaNOS w/o SSL	10%	66.20	66.79	66.08	66.79	66.15	64.23	66.54	66.15
	TaNOS	10%	**68.48**	**68.85**	**68.41**	**68.85**	**69.36**	**67.69**	**69.10**	**68.08**
Biology	SFT	10%	29.81	42.82	27.21	26.79	42.82	22.31	29.10	29.10
	SFT + SSL	10%	30.60	40.77	28.57	29.23	40.77	25.00	28.85	28.97
	SFT + SSL + Anonymization	10%	32.86	42.69	30.89	28.21	42.69	26.28	30.13	36.15
	SFT + Sketch Inference	10%	43.61	51.79	41.57	41.92	51.79	36.92	43.85	43.59
	TaNOS w/o SSL	10%	65.15	66.67	64.85	65.51	66.67	64.10	64.36	64.87
	TaNOS	10%	**66.97**	**66.79**	**67.01**	**67.56**	**66.79**	**66.03**	**67.44**	**67.31**
Legal	SFT	10%	32.93	40.64	30.72	31.67	34.36	40.64	30.64	32.18
	SFT + SSL	10%	33.29	39.74	31.91	31.41	33.21	39.74	32.69	33.72
	SFT + SSL + Anonymization	10%	37.76	42.95	36.84	36.79	38.59	42.95	36.41	36.79
	SFT + Sketch Inference	10%	43.18	48.46	41.86	39.23	43.46	48.46	41.67	43.08
	TaNOS w/o SSL	10%	66.69	**67.05**	66.40	66.41	**67.56**	**67.05**	65.38	**67.31**
	TaNOS	10%	**66.98**	66.41	**67.08**	**67.69**	67.05	66.41	**67.18**	66.79
Scientific	SFT	10%	30.49	36.28	27.87	31.28	31.54	23.72	36.28	30.64
	SFT + SSL	10%	33.46	41.67	30.04	31.28	34.74	29.36	41.67	30.38
	SFT + SSL + Anonymization	10%	37.82	43.72	36.11	35.77	38.59	34.62	43.72	39.10
	SFT + Sketch Inference	10%	41.57	44.36	40.87	42.31	41.03	37.82	44.36	42.31
	TaNOS w/o SSL	10%	65.24	67.18	64.48	65.77	65.51	62.18	67.18	65.00
	TaNOS	10%	**67.97**	**69.23**	**67.34**	**67.95**	**68.21**	**66.67**	**69.23**	**67.95**

Table 11: Program accuracy (%) on **full cross-domain performance** when training on 10% data and evaluating on various domain-shifted variants. The columns show the average performance (**Avg.**), performance on the source domain (**In-domain**), the average on all other domains (**Cross-domain**), and the specific accuracy for each evaluation domain.

C.5 RoBERTa-based Cross-Domain Results

Train Domain	Method	Data	Avg.	In-domain	Cross-domain	Mechanical	Biology	Anonymized
Mechanical	SFT	10%	26.82	31.01	24.72	31.01	28.77	20.67
	SFT + SSL + Anonymization	10%	29.28	35.05	26.39	35.05	29.47	23.32
	TaNOS w/o SSL	10%	49.25	54.61	46.58	54.61	52.93	40.22
	TaNOS	10%	**65.78**	**67.32**	**65.02**	**67.32**	**63.83**	**66.20**
Biology	SFT	10%	18.81	23.74	16.34	20.11	23.74	12.57
	SFT + SSL + Anonymization	10%	30.96	34.78	29.05	31.70	34.78	26.40
	TaNOS w/o SSL	10%	37.76	43.16	35.05	42.04	43.16	28.07
	TaNOS	10%	**63.45**	**62.71**	**63.83**	**65.50**	**62.71**	**62.15**

Table 12: Program accuracy (%) on **RoBERTa-based cross-domain performance** when training on 10% data and evaluating on various domain-shifted variants. **Avg.** is the mean over the three evaluation domains, **In-domain** is accuracy on the train domain, and **Cross-domain** is the average over the remaining two domains. While SSL and anonymization provide only incremental gains on the larger LLaMA-3.1-8B-Instruct backbone, they yield much more pronounced improvements for the smaller RoBERTa model. Despite differences in architecture and model scale, the final TaNOS performance differs by at most 4 pp between the two settings, in sharp contrast to the 17–27 pp gap observed for TaNOS w/o SSL.