STRUDEL: Unrolling a Benchmark for Evaluating Vision-Language Models on Structured Diagram Understanding Across Domains

Daniel Steinigen[1,2,4], **Lucie Flek**[1,2,3], **Sebastian Houben**[1,4]

1- Fraunhofer IAIS, Sankt Augustin, Germany
2- Lamarr Institute for Machine Learning and Artificial Intelligence, Germany
3- Bonn-Aachen International Center for IT, University of Bonn, Germany
4- Bonn-Rhein-Sieg University of Applied Sciences, Sankt Augustin, Germany

Abstract

Vision-Language Models (VLMs) have achieved impressive progress across diverse multimodal tasks, yet their ability to interpret structured diagrams, such as circuit schematics, molecular structures, musical notation, business process flow charts or class diagrams, which are central to scientific and engineering communication, remains underexplored. We introduce STRUDEL (STRUctured Diagram EvaLuation), a benchmark for evaluating VLMs on structured diagram understanding across 8 domains and 20 image categories. STRUDEL leverages Large-Language Models (LLMs) to synthesize code in domain-specific formal representation languages (FRLs) (e.g. circuit netlists, SMILES[1], ABC-Notation[2], BPMN[3] or PlantUML[4]), which are rendered into valid diagrams and paired with generated tasks, functional descriptions, and captions. A multi-stage pipeline filters invalid, cluttered, or redundant samples and employs LLM-as-a-judge scoring to ensure correctness. Through targeted experiments, we evaluate the ability of LLMs to generate valid code in distinct FRLs, demonstrating their capability to successfully perform this task. The resulting benchmark comprises diverse task types covering identification, quantification, structural analysis, image-text association, and image-to-code translation. Evaluating 35 VLMs using STRUDEL reveals that models excel at association tasks, demonstrating strong visual-textual alignment, yet struggle with quantification and identification, where precise structural understanding is required. Performance varies markedly in image-to-code translation, reflecting significant differences in how models connect visual inputs to formal representations. Overall, STRUDEL establishes a scalable foundation for assessing and advancing VLMs torward deeper and more systematic understanding of structured visual information across domains.

Keywords: vision language models, structured visualizations, diagrams, benchmark dataset, visual question answering, image rendering, formal representation language

1. Introduction

Recent advances in Vision-Language Models (VLMs) have demonstrated remarkable progress in integrating visual and textual reasoning on tasks like caption generation, visual question answering, and document understanding, benefiting from large-scale multimodal pretraining. However, their capacity to comprehend structured diagrams—visual representations that encode explicit relationships, constraints, and symbolic structure, remains underexplored. Unlike natural images, structured diagrams present unique challanges to VLMs as they require compositional reasoning, symbol grounding, and domain-specific interpretation (Islam et al., 2024; Yue et al., 2024). Structured diagrams are characterized by the fact that they consist of components or symbols that are spatially arranged in a structure and between which precise, spatially-grounded relations can exist. These diagrams are designed to convey specific technical, scientific, or process-based information. Examples include circuit schematics, molecular structures, musical notation, business process flow charts or UML diagrams. Despite their ubiquity, no existing benchmark systematically evaluates VLMs on such structured visual reasoning across a wide domain range.

To address this, we introduce STRUDEL (STRUctured Diagram EvaLuation), a comprehensive benchmark designed to evaluate VLMs on structured diagram understanding across 8 domains and 20 diverse image categories. For generating the dataset, we take advantage of the fact that many types of structured diagrams can be defined as code by a formal representation language (FRL). These FRLs have a structured code-based syntax to encode domain-specific rules containing symbols, their relationships, and spatial structures. We leverage purely text-based LLMs to synthesize code in domain-specific FRLs, including, *circuit netlists*, *SMILES*[1], *ABC-Notation*[2], *BPMN*[3] or *PlantUML*[4]. Each generated code sample is automatically rendered into an image using a comprehensive toolkit, producing direct mapping between formal code and diagrammatic structure. This pro-

[1] http://opensmiles.org/opensmiles.html
[2] https://abcnotation.com/
[3] https://www.omg.org/spec/BPMN/
[4] https://plantuml.com/

Proceedings of the Fifteenth Language Resources and Evaluation Conference (LREC 2026), pages 11085–11107
11–16 May 2026.

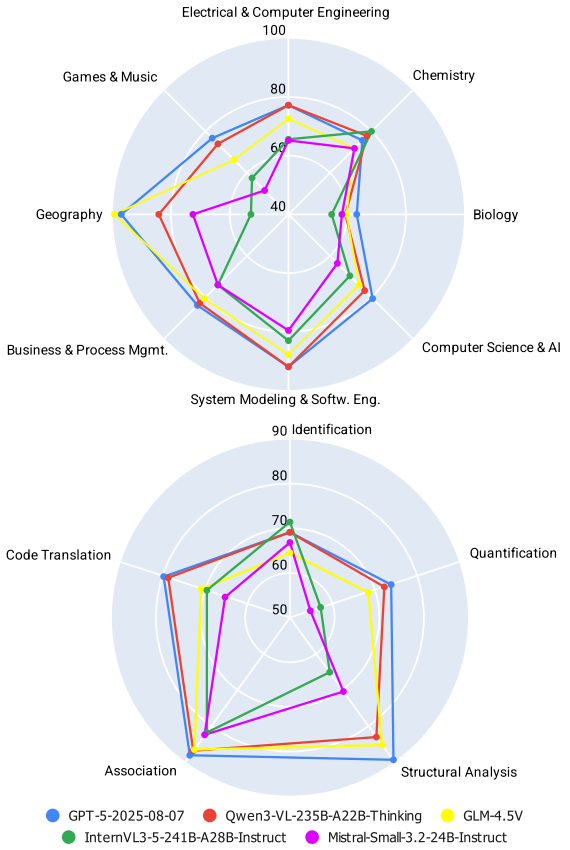

Figure 1: Performance of selected VLMs on STRUDEL benchmark over different domains (top) and task types (bottom), accuracy in %.

cess not only ensures syntactic validity but also enables control over structural complexity, by defining three complexity levels (low, medium, high), as well as diversity by incorporating unique persona descriptions to reflect real-world domain problems.

Beyond code and images, our pipeline incorporates rich context with generated problem statements, solution descriptions, captions, and tasks. The fact that, due to the FRL, the exact underlying structure of each visualization is known provides perfect labels for the content of the visualization to check whether a model's response exactly matches the ground truth. Therefore, we derive a broad set of five different task types for the benchmark, namely identification, quantification, structural analysis, image–text association, and image-to-code translation. A multi-stage filtering and validation pipeline ensures dataset quality, combining heuristic filters (aspect ratio, node thresholds, visual variance) with LLM-as-a-judge validation to assess the correctness of generated samples. The approach for generating STRUDEL is shown in Figure 2 and is explained in detail in Section 3.

We evaluate 35 VLMs on STRUDEL to investigate their ability to reason over structured diagrams. Our results reveal substantial variability across domains and task types as shown in Figure 1: While models perform well at association tasks, demonstrating strong visual-textual alignment, they struggle with quantification and identification, where precise structural understanding is required. Their Performance varies markedly in image-to-code translation, highlighting significant differences in cross-modal reasoning.

In summary, our contributions are threefold:

1. A comparative analysis of current LLMs with respect to their ability to generate code in distinct domain-specific FRLs in terms of code validity and complexity.

2. A cross-domain benchmark (STRUDEL) for evaluating VLMs on understanding structured diagrams, spanning 8 domains with 20 diverse image categories and five different task types, which is generated leveraging LLMs for code synthesis in domain-specific formal representation languages (FRLs), a rendering toolkit, and quality filtering.

3. A comprehensive evaluation of 35 VLMs with varying architectures and parameter scales on structured diagram understanding in different domains using the introduced benchmark.

2. Related Work

The rapid progress of Vision-Language Models (VLMs) has been supported by the availability of public datasets that integrate visual and textual modalities. Foundational datasets such as COCO (Lin et al., 2014), LAION-5B (Schuhmann et al., 2022) or Flickr30k (Young et al., 2014) are mainly focusing on natural images. However, datasets that involve structured information and visualizations that require reasoning with domain-specific knowledge are becoming increasingly important. The MMMU benchmark (Yue et al., 2024) includes 11.5k meticulously collected multimodal questions from college exams, quizzes, and textbooks, covering six core disciplines. Lu et al. (2022) they collect science questions from textbooks and compile multimodal multiple choice questions covering diverse science topics to form ScienceQA. Yu et al. (2024b) gathered 187 images from various online sources and applied 205 questions for 6 core vision language capabilities, generating MM-Vet.

In addition, there are compiled datasets for specific domains. Kembhavi et al. (2016) evaluate science diagrams with structured labels, AI2D. CircuitVQA (Mehta et al., 2024) targets component recognition, value reading and connectivity reasoning in 5,725 circuit diagrams. Li et al. (2025) build a multimodal chemical corpus ChemVLM. Laurençon et al. (2024) created Docmatix dataset with images and QA pairs derived from PDF documents.

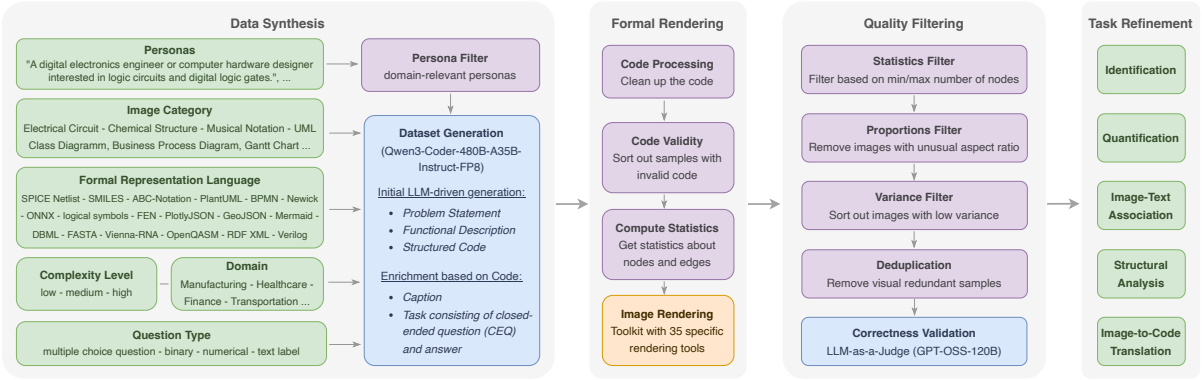

Figure 2: STRUDEL pipeline for synthetically generating a dataset of structured diagrams based on FRLs.

While synthetic data generation is applied extensively for LLMs, such as Self-Instruct (Wang et al., 2022), Magpie (Xu et al., 2024), InstructLab (Sudalairaj et al., 2024) or PersonaHub (Ge et al., 2024), it is not yet as widely used for VLMs. There have been efforts to create vision language datasets partially synthetically. Johnson et al. (2017) introduced CLEVR, 3-D shape scenes with programmatically generated questions targeting compositional reasoning. Particularly for Charts synthetic approaches have been applied, since these can be easily generated from table-like structures. ChartQA (Masry et al., 2022) added a mixture of real and synthetic plots with open-ended questions. FigureQA (Kahou et al., 2017), DVQA (Kafle et al., 2018) and PlotQA (Methani et al., 2020) extended the idea to data visualisations, emphasising numerical comparisons and text-in-image reading. FlowLearn by Pan et al. assesses flow-chart comprehension and reveals severe LVLM weaknesses in path-tracing and node counting, using 10k synthetic flowcharts generated via Mermaid. Yang et al. (2025b) present CoSyn, a framework for generating synthetic multimodal data based on LLM-generated code to improve VLM reasoning on charts and documents. StructEval (Yang et al., 2025a) and SoEval (Liu et al., 2024) show that LLMs are capable in generating structural outputs for common FRLs like JSON, XML, CSV or HTML, while relying on specific pipelines for each FRL, sometimes involving manual intervention.

Recent studies have shown that LLMs are also capable of generating valid code in domain-specific FRLs, for example, for control systems (Koziolek et al., 2023; Fakih et al., 2024) or for chip design (Thakur et al., 2023; Chang et al., 2023; Liu et al., 2023). Nevertheless, no domain-agnostic approach for a comprehensive synthetic generation of diverse multi-domain datasets for VLMs containing automated validity checks, rich context and code-image mapping is currently available. STRUDEL covers 20 structured diagram types from 8 domains, each grounded in valid code of a formal representation language (FRL). Due to our domain-agnostic data generation approach, our benchmark covers a broader spectrum of diagram types than existing benchmarks, including many FRLs not addressed by other works (e.g. SPICE, OpenQASM, Newick, ViennaRNA, FEN, LilyPond, DBML).

3. Dataset Construction

The STRUDEL benchmark is developed to systematically evaluate Vision-Language Models (VLMs) on their ability to understand structured diagrams across a wide range of domains. Its construction follows a multi-stage pipeline (Figure 2) that integrates LLM-driven code synthesis, formal rendering, quality filtering, and task refinement to produce a diverse dataset of structural diagrams.

In order to cover a broad spectrum of diagram categories, we conducted extensive research in various domains to find suitable visualization types that can be described using FRLs. We restricted the selection of FRLs to those that do not require explicit position information or coordinates of the components for the arrangement on the image, as current LLMs do not yet perform reliably in layouting. For example, we ignored typical Computer Aided Design (CAD) languages or formats of technical drawing software. Instead, we focused on those FRLs that can be rendered into an image using a layout engine or another type of arrangement algorithm. We divide the categories into groups of *domain-specific* visualizations and visualizations for *modeling* purposes, which are domain-agnostic.

Each sample in STRUDEL consists of a code representation, its rendered image, and textual context consisting of generated problem statements, solution descriptions, captions, and derived tasks. STRUDEL spans 20 categories of structured diagrams. Each category is associated with a formal representation language (FRL) that defines the syntactic and compositional rules governing its struc-

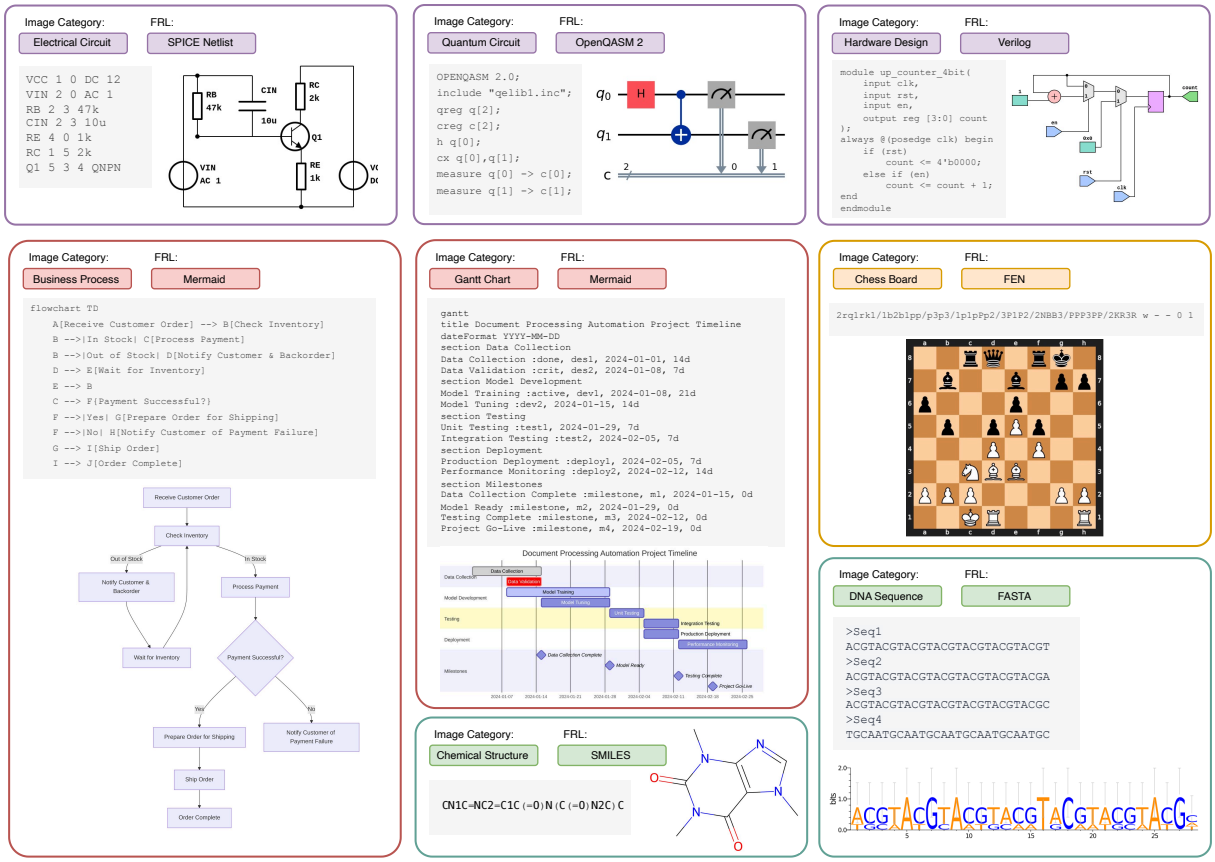

Figure 3: Examples of rendered diagrams and their code representation in different FRLs.

ture. To capture different levels of structural and visual complexity, three levels of complexity, low, medium, and high, are applied corresponding to progressively richer topologies, higher node counts, and more complex relationships. For obtaining a diverse dataset that also reflects real-world domain problem statements, we use domain-specific persona descriptions as seed for the generation process, with a unique persona used for each generated data sample. To this end, we apply the PersonaHub pipeline (Ge et al., 2024) with enhanced filtering to generate a dataset of education and science personas that match our image categories, using the FineWeb-Edu dataset (Lozhkov et al., 2024). More details on the persona pipeline are documented in Appendix A.1.

3.1. LLM-Driven Code Synthesis

The central component of the multi-stage pipeline is the LLM-driven code generation. With the rapid enhancement of LLMs over the last few years, they are now also able to generate valid code in the different FRLs, although some of them are highly application-specific and niche. For this purpose, we evaluate current LLMs with respect to their ability to generate code in distinct domain-specific FRLs and select the best performing model with a permissive license according to section 4 *Qwen3-Coder-480B-A35B-Instruct* (Team, 2025) to generate 600 samples per diagram category. For each diagram category, the model is prompted with the target FRL, a domain-specific persona, and a complexity constraint, in addition to the category itself. From these inputs, the LLM produces a structured code sample written in the target FRL, accompanied by a problem statement that the persona might have and a functional description.

The problem statement is defined as a text that outlines the technical or domain-specific problem. The solution description is defined as a detailed explanation or answer associated with the problem statement. We compare two different prompting approaches. (1) For the `sequential` approach, we first prompt the LLM for generating a problem statement that a persona might have given the input signals, and second, prompt the LLM to generate a textual solution description and a code implementation in the requested FRL while given the problem statement and the input signals. (2) For the `all-in-one` approach we prompt the LLM only once to generate all the outputs mentioned. Since the categories of the modeling group are domain-agnostic, we additionally provide the LLM with a domain as

an input signal by varying the domain across the samples. For some FRLs, we force the LLM to generate only a single line of code when required, e.g. for *FEN* or *SMILES*. We also specify the initial expressions of the code if it is possible for the FRLs, to steer the model to select the appropriate syntax.

3.2. Code-to-Image Rendering

For creating a vision-language dataset from the LLM-generated samples, we utilize domain-specific rendering tools to convert the generated code into structured images. To this end, we build a comprehensive rendering toolkit consisting of 35 different rendering tools, comprising various open source Python packages (e.g. RDKit[5], PySpice[6]), JavaScript libraries (e.g. Mermaid[7], DBML Cli[8]) and Linux command-line tools (e.g. Yosys[9], LilyPond[10]). Before executing the rendering, we apply minor pre-processing to the code, dependent on the FRL. For example, we remove code lines and comments that do not affect the visual outcome.

This rendering step serves two purposes. First, it ensures syntactic and semantic validity, since invalid code simply fails to compile or visualize. The FRLs also include common formats such as *JSON* and *XML*. In these cases, the tools check not only whether the syntax is correct, but also whether the code conforms to the specified schema so that an image can be rendered from it. Second, it provides fine-grained structural metadata, such as node/component and edge counts, and connectivity that can later guide task creation and filtering, e.g. *the number of resistors in a schematic* or the *number of entities in a diagram*.

An overview of rendered diagrams and their corresponding code representations is shown in Figure 3. We publish the toolkit with an easy-to-use interface as a Python library[11].

3.3. Quality Filtering

Since not all generated samples meet the visual or structural quality requirements, each rendered diagram undergoes a multi-stage filtering process in order to obtain a high quality dataset. First, we sort out samples containing invalid code and use statistics to filter based on specific minimal and maximal (overly complex) limit of nodes for each diagram category. Furthermore we remove samples with images that have an unusual aspect ratio of less than `0.25` or greater than `4` and images that have a low variance e.g. are mostly just white. We also apply a deduplication step by removing visual redundant samples with equal code or equal images by calculating a perceptual image hash for each image. For the remaining samples we employ a LLM-as-a-judge (LLMaaJ) (Zheng et al., 2023) approach using the LLM *GPT-OSS-120B*, which has shown strong performance in domain-specific and coding tasks (OpenAI, 2025). We prompt the model to act as a domain expert and validate the correctness as well as the logical consistency of the generated code and the related textual descriptions. Each sample receives a correctness score, and low-quality instances are removed.

3.4. Task Refinement

In order to obtain a more enriched dataset, we enhanced parts of the data samples with further context. We again utilize LLMs to create 1) captions and 2) closed-ended questions for the generated images. To accomplish this, we provide the LLM with the image category, the functional description and the code in the respective FRL as input signals and prompt the model 1) to generate an abstracting caption for an image that will be rendered from the code and 2) to generate a closed-ended question together with the answer for which we randomly provide a question type from multiple choice, binary (yes, no), numerical, or text label.

For STRUDEL we derive a diverse set of benchmarking tasks from the generated dataset. First, we build a task with questions that target the image structure. For these `structural problems`, we use the statistics generated in section 3.2, to generate questions about the number of nodes/components per type, e.g. *How many multiplexers are included in the design?*. For this purpose, we employ a variety of question templates. Second, we generate `image-text association problems` that ask for the association between the different textual descriptions contained in the dataset and the images. For the persona description and the caption we generate multiple choice questions with four options in which the appropriate persona/caption must be assigned to the given image. We also add questions where none of the four options match the image. For the problem statement and the solution description, we generate binary questions in which a problem or solution is presented and it must be determined whether it corresponds to the given image or not. Here we can also increase the difficulty by using false descriptions from the same image category. Furthermore, we incorporate the CEQ generated by the LLM, which cover various tasks, including 1) `Identification` for recognizing key visual or structural features within a diagram; 2) `Quantification` for

[5] https://github.com/rdkit/rdkit
[6] https://github.com/PySpice-org
[7] https://mermaid.js.org/
[8] https://dbml.dbdiagram.io/cli
[9] https://github.com/YosysHQ/yosys
[10] https://lilypond.org/
[11] https://github.com/danielsteinigen/structivize

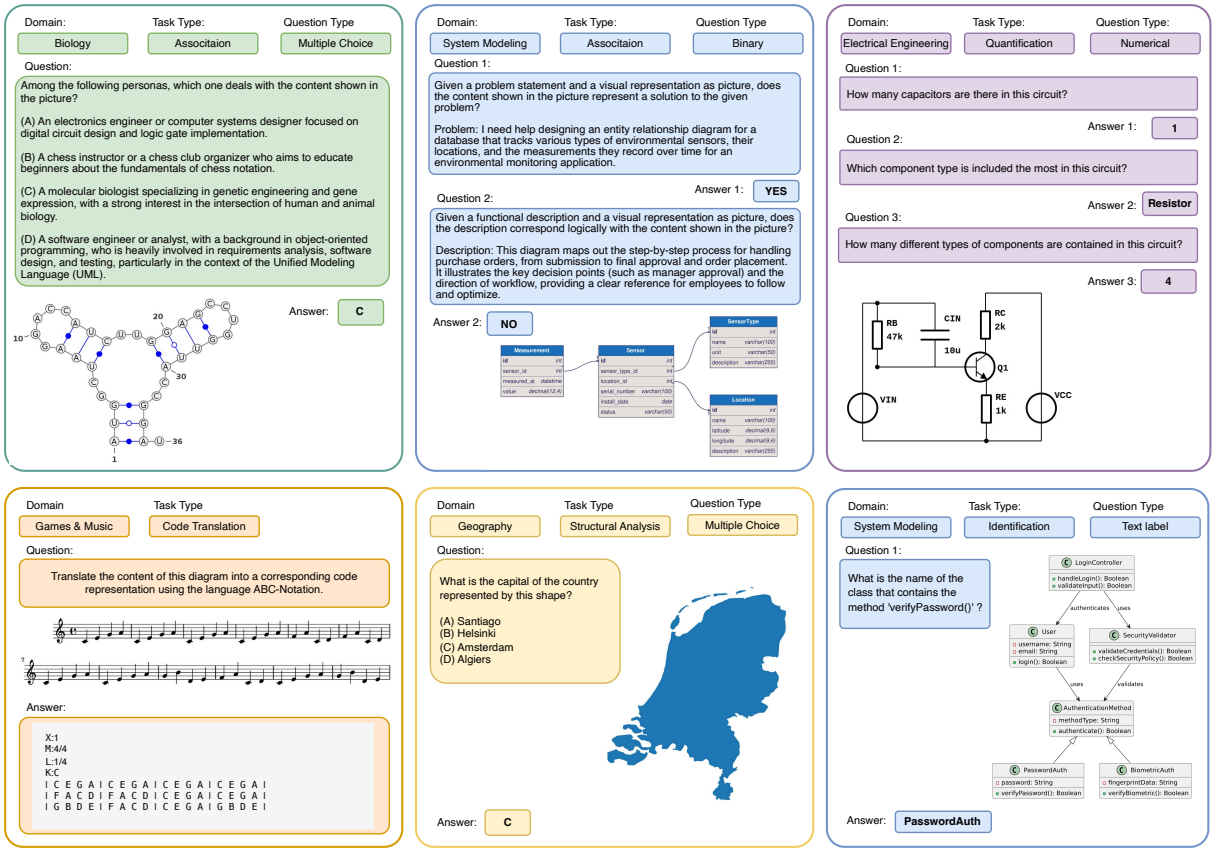

Figure 4: Examples of the benchmark dataset from different task types.

Table 1: Domains covered by the Dataset with image categories and the corresponding [FRLs].

Domain	**Image Categories [with FRLs]**
Electrical & Computer Engineering	Electrical Circuit [SPICE Netlist]; Digital Hardware Design [Verilog]; Quantum Circuit [Open Quantum Assembly Language (OpenQASM) 2]; Propositional Logic [Boolean expressions]
Computer Science & AI	Deep Learning Architecture [Open Neural Network Exchange (ONNX) Graph]; Knowledge Graph [Resource Description Framework (RDF) XML]
Software Engineering & System Modeling	Unified Modeling Language (UML): Class Diagram, Sequence Diagram [PlantUML]; Entity-Relationship (ER) Model [Database Markup Language (DBML)]
Biology	DNA Sequence [FASTA, Vienna RNA dot-bracket notation]; Phylogenetic Tree [Newick]
Chemistry	Chemical Structures & Molecules [Simplified Molecular Input Line Entry System (SMILES), SMILES arbitrary target specification (SMARTS) Reactions]
Business & Process Management	Business Process Model [Business Process Model and Notation (BPMN)]; Gantt Diagram [Mermaid]; Mind Map [Mermaid]; Charts [Plotly json]; Tables [Plotly json]
Games & Music	Chess Board [Forsyth-Edwards-Notation (FEN)]; Musical Notation [ABC]
Geography	geographical map [GeoJSON]

numerical reasoning over structure or duration measurements; 3) `Structural Analysis` for understanding how parts connect, relate, or organize within a structure. Besides these closed-ended questions, we also include an `Image-to-Code Translation` task, where the model needs to generate the corresponding code representation in a given FRL for the provided image.

In addition to the synthetically generated samples, we enrich the dataset by incorporating two existing code-based datasets. Utilizing the *Country Polygons dataset*[12], which provides *geojson*

[12] https://github.com/datasets/geo-countries

polygons for all the world's countries, we rendered shapes of countries and generated questions asking for the corresponding name of the country, capital, currency, continent and dial number. We also incorporate a chess dataset[13] for generating questions asking for the next best move.

The final STRUDEL dataset encompasses 8 domains and contains 7533 questions distributed over 20 categories of structured diagrams and five different task types, as shown in Table 1. This enables a detailed examination of how well VLMs perform in interpreting structured diagrams. Figure 4 shows some examples from this benchmark dataset.

4. Evaluating LLMs for Code Generation in domainspecific FRLs

4.1. Experimental Setup

A prerequisite for automatically generating a synthetic dataset for structured diagrams, as outlined in Section 3, is that the LLM used is capable of generating valid code in distinct domain-specific FRLs. In this experiment, we evaluate and compare current LLMs with respect to this ability.

As personas we draw a random sample of 50 different personas for each image category from our large filtered persona dataset. We take these personas together with the three complexity levels low, medium and high as seed for the generation, resulting in 150 samples per FRL. To compare the performance of the two prompting approaches `sequential` and `all-in-one` described in section 3.1, we run these samples twice applying the different prompts. Thus, for this experiment, we generate 150 * 2 * 62 FRLs (cf. Tables in A.2) = 18,600 data samples with each model. Details about the generation parameters are documented in A.1.

The main metric we assess is the code validity. This defines that the code can be rendered by our toolkit without errors and thereby a valid image is created. We then report the ratio of successful rendered images per language and model. In addition, we assess whether the models follow the specified complexity level. For this, we take the number of lines of code and, for languages where the code is usually shorter than 5 lines, the number of characters as a measure. Next, we check for overall consistency when generating low, medium and high complexity. We verify that code for medium complexity is longer than for low, high is longer than medium, and high is longer than low, considering a significance threshold of at least 10 %. The mean over all samples is reported.

[13] https://huggingface.co/datasets/bonna46/Chess-FEN-and-NL-Format-30K-Dataset

Table 2: Model performance with regard to code validity on different complexity levels and complexity consistency, in %.[15]

Model	Validity overall	Validity per complexity low	medium	high	Comp. Cons.
GPT-4.1-2025-04-14	**89,68**	**94,58**	**89,79**	**84,34**	82,88
Qwen3-Coder-480B-Instr.	84.38	91.17	85.20	76.69	**94.44**
Mistral-Large-Instr.-2411	79.43	83.90	79.44	74.94	71.57
NextCoder-32B	78.35	81.24	77.20	76.60	64.77
GPT-OSS-120B	77.90	87.27	79.74	66.66	88.48
Qwen3-235B-Think.-2507	75.79	83.67	77.45	65.92	84.77
Qwen2.5-72B-Instruct	74.96	78.06	74.58	72.18	70.90
Qwen2.5-Coder-32B	74.24	77.18	73.91	71.56	69.49
GPT-OSS-20B	72.15	82.38	73.18	60.45	89.09
Devstral-Small-2507	69.69	72.84	69.78	66.41	65.18
Nemotron-4-340B-FP8	69.14	73.11	69.93	64.33	64.69
Mistral-Small-3.1-24B	68.61	73.19	68.21	64.37	70.53
Mixtral-8x22B-Instr.	66.08	70.83	65.30	62.06	64.78
Openhands-LM-32B	64.62	71.59	63.90	58.26	68.46
C4AI-Command A	64.49	70.86	63.48	59.12	69.81
Llama-3.3-70B-Instr.	64.29	72.16	63.70	57.00	70.94
QWQ-32B	59.40	64.44	59.94	53.80	69.28
Gemma-3-27B-IT	57.64	62.79	58.19	51.86	80.24
DeepSeek-Qwen-32B	57.25	61.43	56.43	53.87	68.85
Qwen2.5-7B-Instruct	54.18	58.22	54.30	48.87	74.25
DeepCoder-14B	40.16	43.67	39.52	37.29	64.08
Phi-4-Mini-Instruct	37.35	41.50	37.05	33.33	57.31
SmolLM3-3B	21.62	23.65	21.40	19.76	54.06

4.2. Results and Analysis

Table 2 show the results of the evaluation in terms of code validity. It can be seen that *GPT-4.1* clearly dominates in terms of overall performance. The best performing open source model is *Qwen3-Coder-480B-A35B-Instruct-FP8*, followed by *Mistral-Large-Instruct*. It is noteworthy that the code-specific model *NextCoder-32B*, which is based on *Qwen2.5-Coder* performs on a par with the larger models, even though it is a smaller model with 32 billion parameters. The tables in Appendix A.2 indicate which model shows the strongest performance for each FRL and which FRL per category is most suited for an LLM-based generation. In general, it can be derived that LLMs are generally capable of generating valid code in most of the FRLs. Nevertheless, there are notable variations in performance across models when considering specific FRLs. In Yosys netlists, for example, GPT-4.1 can clearly stand out from the other models. Then there are other FRLs where different open source models can compete with or even outperform *GPT-4.1*, e.g. *Mixtral-8x22B* for CircuiTikZ, *Qwen2.5-72B* for VHDL or *Mistral-Large* for Verilog, OpenQASM and FASTA. Furthermore, it is possible to identify which FRLs are best suited for which image categories in terms of synthetic data generation, like the Keras format for neural networks, Verilog for hardware design, FEN for chess board positions or DDL for ER-Diagrams.

In the following, we only report the results of

[15] (Team, 2025; Aggarwal et al., 2025; OpenAI, 2025; Team et al., 2025d; Nvidia et al., 2024; Jiang et al., 2024; Cohere et al., 2025; Grattafiori et al., 2024; Team et al., 2025a; DeepSeek-AI, 2025; Luo et al., 2025; Microsoft et al., 2025; Bakouch et al., 2025)

the all-in-one prompt approach since this beats the sequential approach in terms of code validity and complexity consistency as shown in Apendix A.1. Table 2 shows the performance of the models across all categories for the complexity levels and the metrics for complexity consistency for each model. This shows that increasing the complexity level has a direct effect on the quality of the generated code, as the code validity decreases. Different models also follow the complexity targets to different degrees. *Qwen3-Coder-480B-A35B-Instruct* adheres best to the targets. We also recognized that more advanced models tend to generate more complex problems and tend to generate code in a programming language instead of the desired FRL.

5. Evaluating VLMs on Structured Diagram Understanding

5.1. Experimental Setup

In this experiment, we evaluate 35 Vision-Language Models (VLMs) of varying architectures and parameter scales on structured diagram understanding across five task types and eight domains using the generated benchmark dataset STRUDEL from section 3. As parameters for the VLMs, we use a temperature of 0.2, top-p of 0.95, and a maximum context length of 16,384 (32,768 for reasoning models) and run the inference using the vLLM library[17]. For the evaluation run, we have resized all images to a maximum size of 1024x1024 pixels. The models receive only the image input togther with the text-based task, the related code representations are withheld to ensure the visual reasoning capabilities are evaluated rather than code interpretation.

The model outputs are normalized for scoring through rule-based parsing and string matching. We evaluate the accuracy with which the prediction of the model matches the gold standard answer from the dataset. The gold standard answers can be either *(A)*, *(B)*, *(C)*, *(D)* or *NONE* for the multiple choice questions, *YES* or *NO* for the binary questions, numerical values for quantification and structural questions or textual labels for identification questions. For the Image-to-Code Translation task we calculate the score of the code validity based on the rendering result and the F1-Score for the node/component and edge counts based on the statistics and report the mean of these two metrics. Specifically we calculate statistics about node/component and edge counts from the predicted code and compare to the gold standard code, where True Positives (TP) are the overlap between gold and predicted counts, False Positives (FP) are extra predicted components and false Negatives (FN) are missing components.

5.2. Results and Analysis

Table 3 and Figure 1 show the performance of the individual models for the Benchmark across all domains and task types. Across all models, the strongest results are observed on association tasks, where most systems demonstrate robust visual–textual alignment and contextual grounding. In contrast, quantification and identification tasks expose persistent weaknesses in structural reasoning and numerical understanding. We recognize that the performance of the model is significantly better in visualizations where the nodes or components are labeled, such as the resistors in circuits or the layers in architectural diagrams of neural networks. This can be explained by the fact that the models generally already perform very well on OCR tasks. The code translation task type shows the widest variance among models, highlighting significant differences in cross-modal reasoning. When examined by domain, models perform best in system modeling, business processes, and geographic diagrams, where the visual structure is relatively regular and semantically transparent. However, accuracy declines for abstract symbolic representations such as biological structures, electronic circuits, or musical notation, where semantics are encoded through compact symbols rather than spatial composition.

Among individual models, *GPT-5* achieves the highest overall score (80.19%), leading on three of five task types and six of eight domains. However, *Qwen3-VL-235B-Thinking* closely follows with 78.21%, showing strongest performance of open source (OS) models in quantification tasks and across five domains, particularly electrical engineering and games & music. A consistent pattern emerges across the Qwen model family: “Thinking” variants surpass “Instruct” models in all task types except code translation, where the latter perform better. Notably, *Qwen3-VL-235B-Instruct* delivers the highest scores of OS models in association and code translation tasks, even exceeding GPT-5 on the latter. *GLM-4.5V* ranks among the strongest open-source models, showing exceptional performance in structural and analytical reasoning and dominating the geography domain, though underperforming in code translation. *InternVL3-5-241B-Instruct* demonstrates outstanding ability in identification tasks and the chemistry domain, significantly surpassing GPT-5’s accuracy there. In the smaller model range, *MiniCPM-V-4-5*, *Qwen3-VL-*

[16] (Yu et al., 2025; Lu et al., 2025; Team et al., 2025e; Wang et al., 2024; Team, 2025; Wang et al., 2025; Chen et al., 2024; Team et al., 2025a,c; Dai et al., 2024; Abdin et al., 2024; Microsoft et al., 2025; Wu et al., 2024; Team et al., 2025b; Team, 2024; Laurençon et al., 2024; Marafioti et al., 2025)

[17] https://github.com/vllm-project/vllm

Table 3: Performance of VLMs on Strudel over distinct task types and domains in %.[16]

Model	Overall	Task Type					Domain							
		Identification	Quantification	Structural	Association	Code Translation	Electrical & Computer Eng.	Chemistry	Biology	Computer Science & AI	System Modeling & Softw. Eng.	Business & Process Mgmt.	Geography	Games & Music
GPT-5-2025-08-07	**80.19**	69.14	**73.81**	**89.38**	**87.99**	79.84	**77.36**	75.63	**63.43**	**80.71**	**92.01**	**83.80**	96.84	**76.68**
Qwen3-VL-235B-Think.	*78.21*	69.03	*72.17*	83.01	86.71	78.65	*77.02*	78.04	59.20	*76.89*	*91.79*	*82.80*	84.21	*73.87*
GLM-4.5V	75.55	64.41	68.36	*85.13*	86.56	70.76	72.45	71.89	*59.82*	73.73	87.66	80.58	***99.47***	66.25
Qwen3-VL-235B-Instr.	74.11	69.71	62.70	73.41	*87.01*	***79.92***	73.39	77.38	57.65	74.86	89.81	77.97	83.42	59.57
Qwen3-VL-30B-Think.	74.04	68.81	66.84	81.72	84.68	67.06	71.05	72.20	55.93	71.95	90.00	78.36	93.68	64.78
Qwen2.5-VL-72B	70.35	64.41	57.95	71.93	85.33	73.55	69.44	71.83	56.54	70.91	85.64	73.14	91.58	51.87
InternVL3-5-241B-Instr.	68.42	***71.40***	57.17	65.19	81.77	69.55	65.65	***80.14***	54.54	69.78	83.02	73.94	52.63	57.42
Mistral-Small-3.2-24B	67.39	66.78	54.75	70.54	82.36	65.22	65.13	71.85	58.10	63.30	79.52	73.98	72.63	51.39
Qwen3-VL-30B-Instr.	67.06	62.61	52.01	69.44	81.97	74.31	65.11	67.73	47.47	66.83	80.11	74.29	87.11	52.39
MiniCPM-V-4-5	66.76	70.50	55.41	78.76	83.25	46.74	65.25	71.20	57.10	63.09	78.78	66.31	88.16	56.49
Qwen3-VL-8B-Thinking	66.14	61.82	63.85	64.27	81.47	48.33	59.30	64.03	49.92	70.05	85.30	74.72	55.26	57.37
Ovis2.5-9B	66.11	65.54	59.10	70.73	82.61	47.16	64.38	63.46	54.14	70.12	78.76	73.17	79.21	45.20
Gemma-3-27B-it	65.32	61.60	48.20	74.15	81.57	67.70	62.94	69.04	54.90	68.97	73.11	67.92	92.63	48.29
Qwen3-VL-4B-Thinking	64.67	61.49	62.83	69.62	80.48	37.28	56.88	64.19	45.38	68.16	80.57	73.76	80.26	51.29
Kimi-VL-A3B-Thinking	64.47	60.02	55.98	77.29	84.09	38.13	57.57	58.74	52.88	68.44	78.51	70.47	94.74	46.11
Qwen3-VL-8B-Instruct	64.23	62.61	52.05	62.14	80.63	64.45	62.85	65.87	51.85	67.86	76.29	69.09	74.47	47.10
Qwen3-VL-4B-Instruct	61.81	60.92	50.57	63.43	80.48	51.45	60.85	62.05	50.17	62.20	73.93	65.81	75.00	47.05
InternVL3-5-30B	60.65	61.71	47.58	50.05	82.51	59.02	61.73	69.20	48.60	66.61	75.32	68.05	26.58	44.30
Gemma-3-12B-it	59.95	58.78	41.68	71.38	76.38	59.95	56.93	59.09	51.60	62.84	71.25	59.87	91.58	44.32
Qwen2.5-VL-7B	59.79	58.90	50.78	62.60	77.42	45.28	57.11	59.28	47.75	61.49	72.12	63.85	83.95	42.39
InternVL-3-8B	59.67	61.60	49.47	58.73	77.92	48.11	60.37	59.92	50.93	63.11	72.09	65.36	48.16	43.88
Kimi-VL-A3B-Instruct	58.22	59.46	45.53	68.05	70.90	52.36	53.72	54.46	49.87	55.62	69.44	64.82	76.58	44.20
InternVL3-5-8B	56.38	60.02	45.41	45.80	75.15	53.66	57.45	61.04	43.67	63.46	71.48	65.20	18.42	40.79
NVLM-D-72B	56.30	60.92	43.03	65.00	66.55	54.54	50.35	59.97	45.65	57.29	68.05	61.86	68.68	43.26
InternVL3-5-4B	54.98	59.12	45.74	41.27	77.03	45.05	55.99	64.39	49.69	59.08	64.56	60.99	18.16	42.43
DeepSeek-VL2	54.20	66.33	46.68	74.15	59.49	31.72	51.50	56.18	46.40	53.63	66.31	54.58	86.05	36.47
Granite-Vision-3.3-2B	48.51	52.25	44.59	53.37	66.85	15.63	42.21	42.46	41.72	54.48	60.82	56.38	55.53	31.26
Phi-3.5-vision-instr.	46.47	54.95	36.64	59.10	58.84	26.21	47.42	44.59	37.23	50.15	52.99	52.28	50.53	32.22
Phi-4-multimodal-instr.	45.15	51.01	27.38	52.54	68.63	29.36	45.59	42.03	32.60	46.80	54.03	52.05	52.11	31.48
InternVL3-5-2B	42.55	50.23	28.98	41.27	62.35	31.25	43.93	49.05	31.93	44.76	52.44	48.25	25.00	30.43
InternVL3-5-1B	35.75	50.11	25.45	51.89	39.77	23.67	40.50	45.37	32.74	32.90	40.45	35.32	23.16	28.27
Chameleon-30B	30.09	37.39	18.11	47.74	36.12	22.27	30.74	37.14	23.80	27.31	34.05	30.30	38.95	24.43
SmolVLM2-2.2B-Instr.	26.42	31.53	17.62	36.57	38.88	08.88	28.45	30.52	21.93	25.18	29.46	23.51	34.21	25.07
Chameleon-7B	24.58	27.93	13.81	44.78	34.34	07.90	27.41	29.89	22.99	23.90	28.76	24.75	19.74	15.63
Idefics3-8B-Llama3	12.93	16.44	10.61	19.30	12.65	09.44	09.94	14.75	07.99	21.55	17.92	09.91	24.74	09.31

8B-Thinking, and *Ovis2.5-9B* represent the most competitive 8–9B models, maintaining solid performance across domains despite limited capacity.

Overall, these findings illustrate STRUDEL's suitability in revealing both model capability on distinct tasks and domain-specific specialization, providing nuanced insight into how model scale, architecture, and training objective influence structured diagram understanding.

6. Conclusion

Understanding structured diagrams lies at the intersection of vision, language, and formal reasoning. With STRUDEL, we introduce a benchmark that systematically evaluates Vision-Language Models on structured diagram understanding across domains. By combining LLM-driven code synthesis utilizing FRLs and personas, a rendering toolkit, and quality filtering, STRUDEL provides a extensible resource that covers the diversity and complexity of real-world diagrammatic representations.

The large-scale evaluation across 35 Vision-Language Models (VLMs) demonstrates that STRUDEL effectively captures model capability across distinct tasks and specific domains. Most models show strong performance in association tasks but struggle with quantification and identification, revealing persistent gaps in compositional and symbolic reasoning. The wide variance in code translation performance highlights significant differences in how models connect visual input to formal representations for cross-modal reasoning. Across domains, models exhibit reliable competence in system modeling, business processes, and geographic reasoning, but struggle with domains that demand dense symbolic reasoning and specialized visual parsing, such as biology or musical notation.

Overall the synthetic, code-grounded data generation approach enables continuous extension of STRUDEL to new domains, such as architecture, process engineering, or design engineering, and can support fine-tuning experiments for both VLMs and LLMs, guiding the development of VLMs that can truly understand and reason about structured visual information.

We release our pipelines together with the toolkit and the benchmarking dataset STRUDEL[18].

[18] https://github.com/danielsteinigen/STRUDEL

Limitations

As all data in this work is synthetically generated using LLMs, the overall diversity and complexity of the resulting problems are limited by the generative capacity and biases of these models. Therefore, highly irregular or nuanced problem statements may not be fully represented. Furthermore, the rendered visualizations are digitally produced using the toolkit, which omit imperfections commonly found in real-world images such as hand-drawn diagrams, scanned documents, or degraded print figures.

Acknowledgments

This work was partially supported by the BMFTR, the state of North Rhine-Westphalia as part of the Lamarr Institute for Machine Learning and Artificial Intelligence and the National Science Foundation through Grant No. IIS-2143529.

The licensing was carried out with the support of the Rhine-Ruhr Center for Scientific Data Literacy (DKZ.2R) by Alexandra Mielke affiliated with Bonn-Rhein-Sieg University of Applied Sciences.

References

Marah Abdin, Jyoti Aneja, Hany Awadalla, Ahmed Awadallah, Ammar Ahmad Awan, Nguyen Bach, Amit Bahree, Arash Bakhtiari, Jianmin Bao, Harkirat Behl, et al. 2024. Phi-3 technical report: A highly capable language model locally on your phone.

Tushar Aggarwal, Swayam Singh, Abhijeet Awasthi, Aditya Kanade, and Nagarajan Natarajan. 2025. Nextcoder: Robust adaptation of code lms to diverse code edits. In *ICML 2025*.

Elie Bakouch, Loubna Ben Allal, Anton Lozhkov, Nouamane Tazi, Lewis Tunstall, Carlos Miguel Patiño, Edward Beeching, Aymeric Roucher, Aksel Joonas Reedi, Quentin Gallouédec, et al. 2025. SmolLM3: smol, multilingual, long-context reasoner. https://huggingface.co/blog/smollm3.

Kaiyan Chang, Ying Wang, Haimeng Ren, Mengdi Wang, Shengwen Liang, Yinhe Han, Huawei Li, and Xiaowei Li. 2023. Chipgpt: How far are we from natural language hardware design. *arXiv preprint arXiv:2305.14019*.

Zhe Chen, Jiannan Wu, Wenhai Wang, Weijie Su, Guo Chen, Sen Xing, Muyan Zhong, Qinglong Zhang, Xizhou Zhu, Lewei Lu, et al. 2024. Internvl: Scaling up vision foundation models and aligning for generic visual-linguistic tasks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 24185–24198.

Team Cohere, Aakanksha, Arash Ahmadian, Marwan Ahmed, Jay Alammar, Yazeed Alnumay, Sophia Althammer, Arkady Arkhangorodsky, Viraat Aryabumi, Dennis Aumiller, et al. 2025. Command a: An enterprise-ready large language model.

Wenliang Dai, Nayeon Lee, Boxin Wang, Zhuolin Yang, Zihan Liu, Jon Barker, Tuomas Rintamaki, Mohammad Shoeybi, Bryan Catanzaro, and Wei Ping. 2024. Nvlm: Open frontier-class multimodal llms. *arXiv preprint*.

DeepSeek-AI. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning.

Mohamad Fakih, Rahul Dharmaji, Yasamin Moghaddas, Gustavo Quiros, Oluwatosin Ogundare, and Mohammad Abdullah Al Faruque. 2024. Llm4plc: Harnessing large language models for verifiable programming of plcs in industrial control systems. In *International Conference on Software Engineering: Software Engineering in Practice*, pages 192–203.

Tao Ge, Xin Chan, Xiaoyang Wang, Dian Yu, Haitao Mi, and Dong Yu. 2024. Scaling synthetic data creation with 1,000,000,000 personas. *arXiv preprint arXiv:2406.20094*.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. 2024. The llama 3 herd of models.

Mohammed Saidul Islam, Raian Rahman, Ahmed Masry, Md Tahmid Rahman Laskar, Mir Tafseer Nayeem, and Enamul Hoque. 2024. Are large vision language models up to the challenge of chart comprehension and reasoning. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 3334–3368. Association for Computational Linguistics.

Albert Q. Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, et al. 2024. Mixtral of experts.

Justin Johnson, Bharath Hariharan, Laurens Van Der Maaten, Li Fei-Fei, C Lawrence Zitnick, and Ross Girshick. 2017. Clevr: A diagnostic dataset for compositional language and elementary visual reasoning. In *International Conference*

on Computer Vision and Pattern Recognition (CVPR), pages 2901–2910.

Kushal Kafle, Brian Price, Scott Cohen, and Christopher Kanan. 2018. Dvqa: Understanding data visualizations via question answering. In *International Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5648–5656.

Samira Ebrahimi Kahou, Vincent Michalski, Adam Atkinson, Ákos Kádár, Adam Trischler, and Yoshua Bengio. 2017. Figureqa: An annotated figure dataset for visual reasoning. *arXiv preprint arXiv:1710.07300*.

Aniruddha Kembhavi, Mike Salvato, Eric Kolve, Minjoon Seo, Hannaneh Hajishirzi, and Ali Farhadi. 2016. A diagram is worth a dozen images. In *European Conference on Computer Vision (ECCV)*, pages 235–251. Springer.

Heiko Koziolek, Sten Gruener, and Virendra Ashiwal. 2023. Chatgpt for plc/dcs control logic generation. In *International Conference on Emerging Technologies and Factory Automation (ETFA)*, pages 1–8. IEEE.

Hugo Laurençon, Andrés Marafioti, Victor Sanh, and Léo Tronchon. 2024. Building and better understanding vision-language models: insights and future directions. In *Workshop on Responsibly Building the Next Generation of Multimodal Foundational Models*.

Hugo Laurençon, Andrés Marafioti, Victor Sanh, and Léo Tronchon. 2024. Building and better understanding vision-language models: insights and future directions.

Junxian Li, Di Zhang, Xunzhi Wang, Zeying Hao, Jingdi Lei, Qian Tan, Cai Zhou, Wei Liu, Yaotian Yang, Xinrui Xiong, et al. 2025. Chemvlm: Exploring the power of multimodal large language models in chemistry area. In *AAAI Conference on Artificial Intelligence*, volume 39, pages 415–423.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. 2014. Microsoft coco: Common objects in context. In *European Conference on Computer Vision (ECCV)*, pages 740–755. Springer.

Mingjie Liu, Teodor-Dumitru Ene, Robert Kirby, Chris Cheng, Nathaniel Pinckney, Rongjian Liang, Jonah Alben, Himyanshu Anand, Sanmitra Banerjee, Ismet Bayraktaroglu, et al. 2023. Chipnemo: Domain-adapted llms for chip design. *arXiv preprint arXiv:2311.00176*.

Yu Liu, Duantengchuan Li, Kaili Wang, Zhuoran Xiong, Fobo Shi, Jian Wang, Bing Li, and Bo Hang. 2024. Are llms good at structured outputs? a benchmark for evaluating structured output capabilities in llms. *Information Processing & Management*, 61(5):103809.

Anton Lozhkov, Loubna Ben Allal, Leandro von Werra, and Thomas Wolf. 2024. Fineweb-edu: the finest collection of educational content.

Pan Lu, Swaroop Mishra, Tony Xia, Liang Qiu, Kai-Wei Chang, Song-Chun Zhu, Oyvind Tafjord, Peter Clark, and Ashwin Kalyan. 2022. Learn to explain: Multimodal reasoning via thought chains for science question answering. In *Conference on Neural Information Processing Systems (NeurIPS)*.

Shiyin Lu, Yang Li, Yu Xia, Yuwei Hu, Shanshan Zhao, Yanqing Ma, Zhichao Wei, Yinglun Li, Lunhao Duan, Jianshan Zhao, et al. 2025. Ovis2.5 technical report. *arXiv:2508.11737*.

Michael Luo, Sijun Tan, Roy Huang, Ameen Patel, Alpay Ariyak, Qingyang Wu, Xiaoxiang Shi, Rachel Xin, Colin Cai, Maurice Weber, Ce Zhang, Li Erran Li, Raluca Ada Popa, and Ion Stoica. 2025. Deepcoder: A fully open-source 14b coder at o3-mini level. Notion Blog.

Andrés Marafioti, Orr Zohar, Miquel Farré, Merve Noyan, Elie Bakouch, Pedro Cuenca, Cyril Zakka, Loubna Ben Allal, Anton Lozhkov, Nouamane Tazi, et al. 2025. Smolvlm: Redefining small and efficient multimodal models. *arXiv preprint arXiv:2504.05299*.

Ahmed Masry, Xuan Long Do, Jia Qing Tan, Shafiq Joty, and Enamul Hoque. 2022. ChartQA: A benchmark for question answering about charts with visual and logical reasoning. In *Findings of the Association for Computational Linguistics (ACL)*, pages 2263–2279. Association for Computational Linguistics.

Rahul Mehta, Bhavyajeet Singh, Vasudeva Varma, and Manish Gupta. 2024. Circuitvqa: A visual question answering dataset for electrical circuit images. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 440–460. Springer.

Nitesh Methani, Pritha Ganguly, Mitesh M Khapra, and Pratyush Kumar. 2020. Plotqa: Reasoning over scientific plots. In *Winter Conference on Applications of Computer Vision*, pages 1527–1536.

Microsoft, Abdelrahman Abouelenin, Atabak Ashfaq, Adam Atkinson, Hany Awadalla, Nguyen Bach, Jianmin Bao, Alon Benhaim, Martin Cai,

Vishrav Chaudhary, et al. 2025. Phi-4-mini technical report: Compact yet powerful multimodal language models via mixture-of-loras.

Nvidia, Bo Adler, Niket Agarwal, Ashwath Aithal, Dong H. Anh, Pallab Bhattacharya, Annika Brundyn, Jared Casper, Bryan Catanzaro, Sharon Clay, et al. 2024. Nemotron-4 340b technical report.

OpenAI. 2025. gpt-oss-120b & gpt-oss-20b Model Card.

Huitong Pan, Qi Zhang, Cornelia Caragea, Eduard Dragut, and Longin Jan Latecki. Flowlearn: Evaluating large vision-language models on flowchart understanding, 2024. *arXiv preprint arXiv:2407.05183*.

Christoph Schuhmann, Romain Beaumont, Richard Vencu, Cade Gordon, Ross Wightman, Mehdi Cherti, Theo Coombes, Aarush Katta, Clayton Mullis, Mitchell Wortsman, et al. 2022. Laion-5b: An open large-scale dataset for training next generation image-text models. *Advances in Neural Information Processing Systems (NeurIPS)*, 35:25278–25294.

Shivchander Sudalairaj, Abhishek Bhandwaldar, Aldo Pareja, Kai Xu, David D Cox, and Akash Srivastava. 2024. Lab: Large-scale alignment for chatbots. *arXiv preprint arXiv:2403.01081*.

Chameleon Team. 2024. Chameleon: Mixed-modal early-fusion foundation models. *arXiv preprint arXiv:2405.09818*.

Gemma Team, Aishwarya Kamath, Johan Ferret, Shreya Pathak, Nino Vieillard, Ramona Merhej, Sarah Perrin, Tatiana Matejovicova, Alexandre Ramé, Morgane Rivière, et al. 2025a. Gemma 3 technical report.

Granite Vision Team, Leonid Karlinsky, Assaf Arbelle, Abraham Daniels, Ahmed Nassar, Amit Alfassi, Bo Wu, Eli Schwartz, Dhiraj Joshi, Jovana Kondic, et al. 2025b. Granite vision: a lightweight, open-source multimodal model for enterprise intelligence.

Kimi Team, Angang Du, Bohong Yin, Bowei Xing, Bowen Qu, Bowen Wang, Cheng Chen, Chenlin Zhang, Chenzhuang Du, Chu Wei, et al. 2025c. Kimi-VL technical report.

Qwen Team. 2025. Qwen3 technical report.

Qwen Team, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, et al. 2025d. Qwen2.5 technical report.

V Team, Wenyi Hong, Wenmeng Yu, Xiaotao Gu, Guo Wang, Guobing Gan, Haomiao Tang, Jiale Cheng, Ji Qi, Junhui Ji, Lihang Pan, et al. 2025e. Glm-4.5v and glm-4.1v-thinking: Towards versatile multimodal reasoning with scalable reinforcement learning.

Shailja Thakur, Jason Blocklove, Hammond Pearce, Benjamin Tan, Siddharth Garg, and Ramesh Karri. 2023. Autochip: Automating hdl generation using llm feedback. *arXiv preprint arXiv:2311.04887*.

Peng Wang, Shuai Bai, Sinan Tan, Shijie Wang, Zhihao Fan, Jinze Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, et al. 2024. Qwen2-vl: Enhancing vision-language model's perception of the world at any resolution. *arXiv preprint arXiv:2409.12191*.

Weiyun Wang, Zhangwei Gao, Lixin Gu, Hengjun Pu, Long Cui, Xingguang Wei, Zhaoyang Liu, Linglin Jing, Shenglong Ye, Jie Shao, et al. 2025. Internvl3.5: Advancing open-source multimodal models in versatility, reasoning, and efficiency. *arXiv preprint arXiv:2508.18265*.

Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A Smith, Daniel Khashabi, and Hannaneh Hajishirzi. 2022. Self-instruct: Aligning language models with self-generated instructions. *arXiv preprint arXiv:2212.10560*.

Zhiyu Wu, Xiaokang Chen, Zizheng Pan, Xingchao Liu, Wen Liu, Damai Dai, Huazuo Gao, Yiyang Ma, Chengyue Wu, Bingxuan Wang, et al. 2024. Deepseek-vl2: Mixture-of-experts vision-language models for advanced multimodal understanding.

Zhangchen Xu, Fengqing Jiang, Luyao Niu, Yuntian Deng, Radha Poovendran, Yejin Choi, and Bill Yuchen Lin. 2024. Magpie: Alignment data synthesis from scratch by prompting aligned llms with nothing. *arXiv preprint arXiv:2406.08464*.

Jialin Yang, Dongfu Jiang, Lipeng He, Sherman Siu, Yuxuan Zhang, Disen Liao, Zhuofeng Li, Huaye Zeng, Yiming Jia, Haozhe Wang, et al. 2025a. Structeval: Benchmarking llms' capabilities to generate structural outputs. *arXiv preprint arXiv:2505.20139*.

Yue Yang, Ajay Patel, Matt Deitke, et al. 2025b. Scaling text-rich image understanding via code-guided synthetic multimodal data generation. In *63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.

Peter Young, Alice Lai, Micah Hodosh, and Julia Hockenmaier. 2014. From image descriptions to

visual denotations: New similarity metrics for semantic inference over event descriptions. *Transactions of the Association for Computational Linguistics*, 2:67–78.

Puxuan Yu, Luke Merrick, Gaurav Nuti, and Daniel Campos. 2024a. Arctic-embed 2.0: Multilingual retrieval without compromise.

Tianyu Yu, Zefan Wang, Chongyi Wang, Fuwei Huang, Wenshuo Ma, Zhihui He, Tianchi Cai, Weize Chen, Yuxiang Huang, Yuanqian Zhao, et al. 2025. Minicpm-v 4.5: Cooking efficient mllms via architecture, data, and training recipe.

Weihao Yu, Zhengyuan Yang, Linjie Li, Jianfeng Wang, Kevin Lin, Zicheng Liu, Xinchao Wang, and Lijuan Wang. 2024b. Mm-vet: Evaluating large multimodal models for integrated capabilities. In *International Conference on Machine Learning (ICML)*. PMLR.

Xiang Yue, Yuansheng Ni, Kai Zhang, Tianyu Zheng, Ruoqi Liu, Ge Zhang, Samuel Stevens, Dongfu Jiang, Weiming Ren, Yuxuan Sun, et al. 2024. Mmmu: A massive multi-discipline multimodal understanding and reasoning benchmark for expert agi. In *International Conference on Computer Vision and Pattern Recognition (CVPR)*.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. 2023. Judging llm-as-a-judge with mt-bench and chatbot arena.

A. Appendix

A.1. Dataset Generation

For obtaining a large set of persona descriptions we follow the PersonaHub pipeline (Ge et al., 2024) and generate personas based on the 10BT split of FineWeb-Edu dataset (Lozhkov et al., 2024). The dataset contains texts mainly from the education and scientific domain, which fits perfectly for our purpose. For the generation we use *Qwen3-235B-A22B-Instruct* (Team, 2025) since it facilitates a permissive license for the generated data, yielding to a dataset of 9 M educational personas after applying MinHash deduplication. Next we apply *Snowflake-arctic-embed-l-v2.0* (Yu et al., 2024a) for embedding and filtering the dataset. We filter the dataset by running a semantic search with top-k of `25,000` on `5` search queries per image category and take only those that has at least `3` hits or a cosine similarity score above `0.7`, limiting to 25,000 personas per image category.

Table 4 compares the performance of the two prompt approaches. Across all models, we see that the *all-in-one* approach beats the *sequential* Approach in terms of code validity and complexity consistency.

For the dataet generation we use as parameters for the LLMs a temperature of 0.6, top-p of 0.95, and a maximum context length of 16,384 (32,768 for reasoning models, 4,096 for Nemotron-340B) and run the inference using the vLLM library[19].

The generated dataset is comprised, for each sample, of one (or more) structured diagram(s) as image(s), a code representation in the specific FRL, a label of the image category, a problem statement, a solution description, a persona description, a caption, a complexity level, statistics of the structural details and metadata, e.g. the name of the FRL, the rendering tool used and its chosen parameters, the log output of the tool and image properties, such as width and height.

Table 4: Performance per Prompt-Template over all models, in %.

Metric	Sequential	All-In-One
Code Validity	60.93	**64.43**
Complexity Consistency	61.26	**69.56**

A.1.1. Dataset Examples

Figures 5 and 6 illustrate samples of the synthetic generated generated problem-solution pairs.

A.1.2. Toolkit

The rendering of images from FRL-specific code is achieved using a comprehensive toolkit consisting of 54 different tools. Table 5 shows the compilation of the tools with the corresponding licenses. The specific settings for each tool are stored as default values in the published source code.

A.1.3. Prompt templates

The prompt templates used for dataset generation, correctness scoring and dataset enrichment are provided in Figures 7 to 10.

A.2. Evaluate LLMS for Code generation

Tables 6 to 8 provide detailed evaluation results of 36 LLMs on generating valid code in the various FRLs. These tables can be used to determine which LLM is best suited for which FRL.

A.3. Artifacts and Licenses

For generating persona descriptions we used the 10BT split of FineWeb-Edu dataset (Lozhkov et al., 2024), which has ODC-By license, using *Qwen3-235B-A22B-Instruct* (Team, 2025). For embedding and filtering the persona dataset we use *Snowflake-arctic-embed-l-v2.0* (Yu et al., 2024a). These models facilitate a permissive license Apache-2.0, allowing our generated persona dataset to be published under permissive license as well. For the generation of STRUDEL we use *Qwen3-Coder-480B-A35B-Instruct* (Team, 2025) and *GPT-OSS-120B* (OpenAI, 2025), where both models are under Apache-2.0. The used tools for image rendering are documented in Table 5 together with the corresponding permissive licenses, allowing our toolkit to be released under permissive license, as it assembles these tools without applying any changes to the source code.

[19] https://github.com/vllm-project/vllm

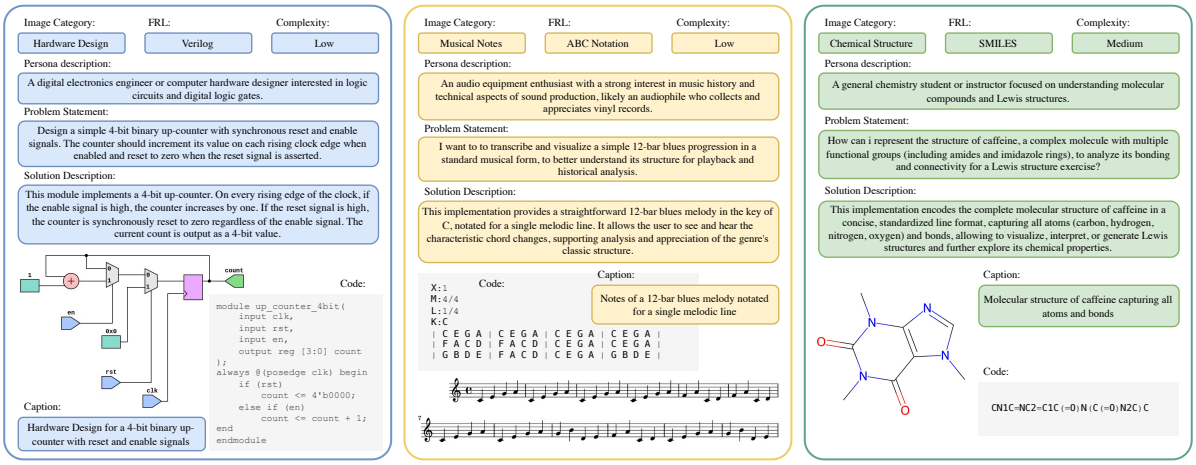

Figure 5: Examples of the synthetically generated dataset from various domains.

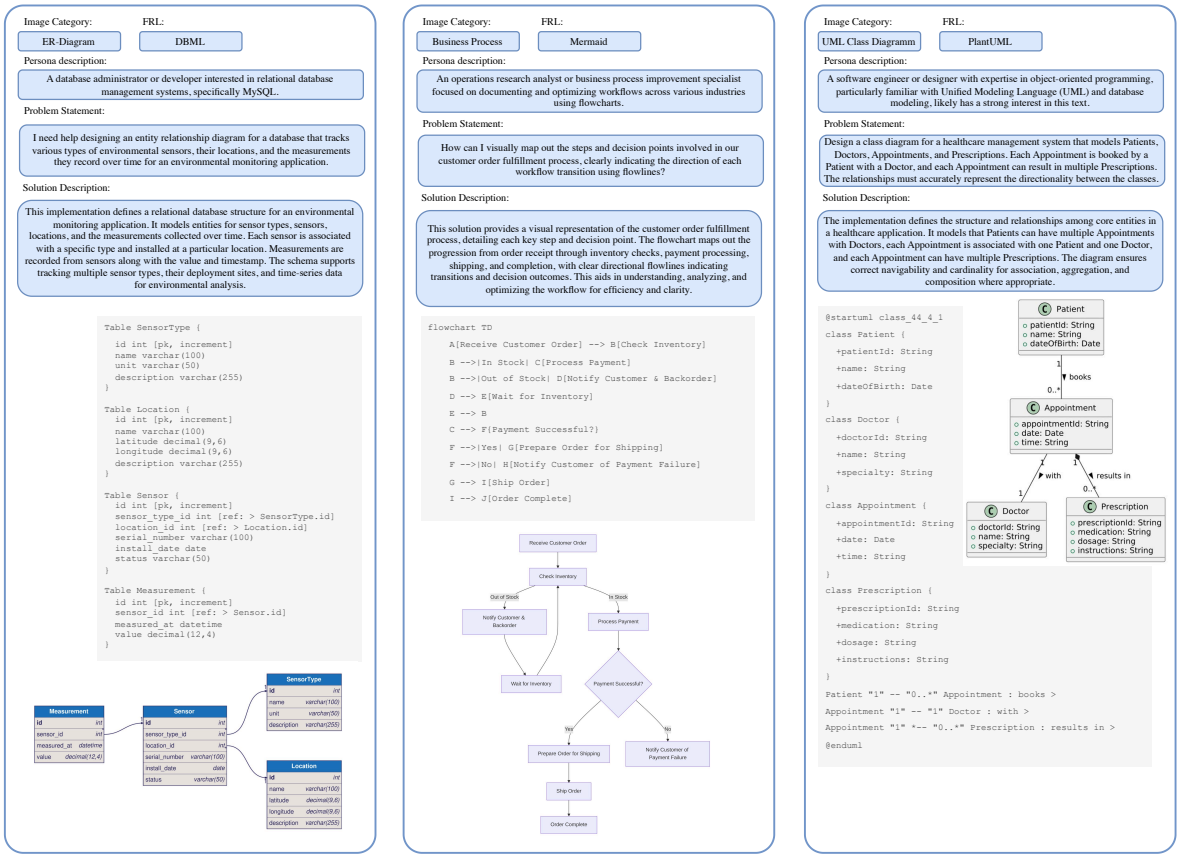

Figure 6: Examples of the synthetically generated dataset from modeling categories.

Table 5: Tools covered by the toolkit.

Environment	Library	Link	License
Python	Matplotlib	https://matplotlib.org/	PSF/BSD
Python	Logomaker	https://logomaker.readthedocs.io/	MIT
Python	Biopython	https://biopython.org/	Biopython/BSD
Python	ETE Toolkit	https://etetoolkit.org/	GPL-3
Python	forgi	https://viennarna.github.io/forgi/index.html	GPL-3
Python	Graphviz Python	https://graphviz.readthedocs.io/	MIT
Python	Pydot	https://github.com/pydot/pydot	MIT
Python	NetworkX	https://networkx.org/	BSD-3
Python	RDFLib	https://rdflib.readthedocs.io/	BSD-3
Python	bpmn_python	https://bpmn-python.readthedocs.io/en/latest/	GPL-3
Python	Sismic	https://github.com/AlexandreDecan/sismic	LGPL-3
Python	pyhtml2pdf	https://github.com/kumaF/pyhtml2pdf	MIT
Python	Plotly	https://plotly.com/python/	MIT
Python	Dash	https://dash.plotly.com/	MIT
Python	Vega-Altair	https://altair-viz.github.io/	BSD-3
Python	RDKit	https://github.com/rdkit/rdkit	BSD-3
Python	python-chess	https://python-chess.readthedocs.io/en/latest/	GPL-3
Python	Lcapy	https://lcapy.readthedocs.io/	LGPL-2.1
Python	Schemdraw	https://schemdraw.readthedocs.io/	MIT
Python	PySpice	https://pyspice.fabrice-salvaire.fr/pages/documentation.html	GPL-3
Python	SKiDL	https://devbisme.github.io/skidl/	MIT
Python	PyLaTeX	https://jeltef.github.io/PyLaTeX/current/	MIT
Python	Qiskit	https://www.ibm.com/quantum/qiskit	Apache 2.0
Python	Cirq	https://quantumai.google/cirq	Apache 2.0
Python	Keras	https://keras.io/	Apache 2.0
Python	SBMLDiagrams	https://sys-bio.github.io/SBMLDiagrams/	MIT
Python	SBMLNetwork	https://sbmlnetwork.readthedocs.io/	MIT
Python	Netron Export	https://github.com/raphael-prevost/netron-export	MIT
Python	Playwright	https://playwright.dev/python/	Apache 2.0
Javascript	dbml-renderer	https://github.com/softwaretechnik-berlin/dbml-renderer/	none provided
Javascript	prisma-dbml-generator	https://github.com/notiz-dev/prisma-dbml-generator	MIT
Javascript	DBML Cli	https://dbml.dbdiagram.io/cli/convert–a-sql-file-to-dbml	Apache 2.0
Javascript	Cytoscape.js	https://js.cytoscape.org/	MIT
Javascript	netlistsvg	https://github.com/nturley/netlistsvg	MIT
Javascript	State Machine Cat	https://state-machine-cat.js.org/	MIT
Javascript	Vega Cli	https://vega.github.io/vega/usage/	BSD-3
Javascript	Netron	https://github.com/lutzroeder/netron	MIT
Javascript	Mermaid	https://mermaid.js.org/	MIT
Command-Line	PlantUML	https://plantuml.com/	MIT
Command-Line	D2	https://d2lang.com/	MPL-2
Command-Line	VARNA	https://varna.lisn.upsaclay.fr/	GPL-3
Command-Line	R2DT	https://docs.r2dt.bio/	Apache 2.0
Command-Line	Open Babel	https://openbabel.org/index.html	GPL-2
Command-Line	WebLogo	https://weblogo.threeplusone.com/	MIT
Command-Line	ditaa	https://ditaa.sourceforge.net/	GPL-2
Command-Line	Asymptote	https://asymptote.sourceforge.io/	LGPL-3
Command-Line	GHDL	https://ghdl.github.io/ghdl/	GPL-2
Command-Line	Yosys	https://yosyshq.readthedocs.io/projects/yosys	ISC
Command-Line	LilyPond	https://lilypond.org/index.de.html	GPL + OFL
Command-Line	abcm2ps	http://moinejf.free.fr/abcm2ps-doc/index.html	GPL-3
Command-Line	MuseScore 3	https://musescore.org/en/handbook/3	GPL-2
Command-Line	TikZ	https://tikz.dev/	GPL-2 or LPPL
Command-Line	CircuiTikZ	https://github.com/circuitikz/circuitikz	GPL-3 or LPPL
Command-Line	CarioSVG	https://cairosvg.org/	LGPL-3
Command-Line	pdfcrop	https://ctan.org/pkg/pdfcrop	LPPL
Command-Line	pdf2svg	https://wiki.ubuntuusers.de/pdf2svg/	GPL 2
Command-Line	ImageMagick	https://imagemagick.org/script/convert.php	ImageMagick
Command-Line	markmap	https://markmap.js.org/docs/packages–markmap-cli	MIT
Command-Line	Inkscape	https://inkscape.org/doc/inkscape-man.html	GPL-2

Dataset Generation with problem-solution pairs and code as soulution (all-in-one)

System Prompt:
You are an helpful AI assistant expert in generating solutions with functional descriptions for the problem statements of users.

User Prompt:
Given a persona, generate a prompt with one problem statement that this persona might give to an AI assistant. The problem should deal with a `{category}` and should lead to a solution using the code language `{lang}`. Consider that the problem should be of `{complexity}` complexity.
Then generate a code implementation as solution to the problem statement using the specific defined code language.`{code_instruct}` Don't use Python or any other programming language. Don't add any comments in the code. The code should be syntactically correct and compile and run without errors.
Also provide a functional description of the solution implementation. Do not mention and not use the code language in the problem statement or in the functional description.`{phrasing}`
Finally, generate an answer with which the AI assistent will respond to the problem statement described in the persona's prompt.

Persona: `{persona}`

Now, please output your results in the format below by filling in the placeholders in <...>:

Problem:
<problem statement>

Functionality:
<functional description>

Code:
```
<code implementation>
```

Answer:
<assistant answer>

Figure 7: Prompt for dataset generation with problem-solution pairs (all-in-one).

Dataset Generation sequential with generating problem statement as first step

System Prompt:
You are an AI assistant expert at simulating user interactions.

User Prompt:
Generate a prompt with one problem statement that the following persona might ask to an AI assistant. Consider that the problem should be `{complexity}` complex and lead to a solution using a `{category}` with the Code Language {lang}. Do not mention and not use the Code Language in the problem statement.

Persona: `{persona}`

Now, please output your results in the format below by filling in the placeholders in <...>:
Problem: <problem statement>

Dataset Generation sequential with generating solution as second step

System Prompt:
You are an helpful AI assistant expert in generating solutions with functional descriptions for the problem statements of users.

User Prompt:
Generate a code implementation as solution for the following problem statement of the given persona. Also provide a functional description of the implementation. Use the code language `{lang}`. Don't use Python or any other programming language.
Don't add any comments in the code. The code should be syntactically correct and compile and run without errors. Do not mention the code language in the description.

Persona: `{persona}`

Problem: `{problem}`

Now, please output your results in the format below by filling in the placeholders in <...>:

Functionality:
<functional description>

Code:
```
<code implementation>
```

Figure 8: Prompts for sequential dataset generation, first problem statement and second solution.

Dataset LLM-as-a-Judge correctness scoring for problem-solution pairs

System Prompt:
You are a professional expert in `{category}`, with deep experience in reviewing code and solution specifications.

You will be given:
- A problem statement that describes a task
- The programming language used to implement a solution
- The actual code implementation
- A functional description of what the code is supposed to do

Your task is to judge whether
1. the code and the functional description are factually, technically and logically correct,
2. the functional description correctly reflect what the code does.

Use your domain expertise in `{category}` for the assessment. Don't assume missing context; do not run code; judge from text only.

Output your result as binary value "correct".
Output "true" when all is correct by tolerating minor deviations or irrelevant inconsistencies.
Output "false" only when you detect clear, material errors or contradictions.
Otherwise output "null" for an unsure/indeterminate result.

Also provide a short factual explanation for your decision, free of speculation, of max. 30 words.

User Prompt:
Given the following problem statement, code language, code and its functional description, assess the correctness.

Problem Statement:
`{problem}`

Code Language: `{language}`

Code:
```
{code}
```

Functional Description:
`{description}`

Now, please output your results in the format below by filling in the placeholders in <...>:

correct: <true|false|null>
explanation: <brief factual explanation>

Figure 9: Prompt for scoring correctness for problem-solution pairs.

Dataset Enrichment with captions

System Prompt:
You are an helpful AI assistant expert in summarizing, enriching and expanding data samples.

User Prompt:
Given a code representation in the code language `{language}` that implements a `{category}`, generate a caption for a picture which will be rendered from this code.
The picture will represent exactly what is contained in the code. The caption should be abstracting and contain only the most relevant information.
Consider that in the end only the rendered picture and the caption are visible, not the code. Therefore, the coding language itself or anything invisible should not be mentioned in the caption.

Code:
```
{code}
```

Now, please output your results in the format below by filling in the placeholders in <...>:

Caption: <caption>

Dataset Enrichment with closed-ended questions

System Prompt:
You are an helpful AI assistant expert in generating realistic question-answer pairs.

User Prompt:
Given a code representation in the code language `{language}`, generate a natural user question that refers to the image of a `{category}` which will be rendered from this code.
The image will represent exactly what is contained in the code. Use the code internally as ground truth to understand the contents and visual structure of the image.
Also generate an assistant response that gives the correct answer, derived directly from the code. The answer must match exactly what is in the image.
The code will **not** be present in the final dataset, so the question and answer must sound like they refer to the image itself, not the code. Do not mention, quote, or reference the code or the code language in the question or answer. Derive every detail of the question and answer only from what is unambiguously implied by the code. Do not invent content.
Frame the question such that it requires visual interpretation of the rendered image. `{question_type}`

Code:
```
{code}
```

Now, please output your results in the format below by filling in the placeholders in <...>:

User: <user question>
Assistant: <assistant answer>

Figure 10: Prompt for dataset enrichment with captions and closed-ended questions.

Table 6: Model performance on distinct FRLs with regard to code validity in %. (Part 1)

Category	FRL	GPT-4.1	Qwen3-Coder-480B-A35B-Instruct-FP8	Mistral-Large-Instruct-2411	Next Coder-32B	GPT-OSS-120B	Qwen3-235B-A22B-Thinking-2507	Qwen-2.5-72B-Instruct	Qwen3-235B-A22B-Instruct-2507	Qwen-2.5-Coder-32B-Instruct	GPT-OSS-20B	Devstral-Small-2507	Qwen3-Coder-30B-A3B-Instruct
Electrical Circuit	SPICE Netlist	88.00	91.33	79.33	86.58	80.00	75.68	86.00	88.00	86.67	83.22	57.05	78.33
Electrical Circuit	CircuiTikZ	57.86	57.33	71.33	54.61	30.67	60.00	55.33	65.33	56.00	25.17	58.00	49.11
Hardware Design	VHDL	62.75	74.00	76.00	84.67	77.33	81.12	84.00	68.00	73.33	65.75	75.84	67.79
Hardware Design	Verilog	80.33	90.67	96.67	98.00	93.33	88.11	93.33	89.33	90.67	92.31	89.33	90.84
Hardware Design	Yosys	97.66	91.16	0.00	93.33	55.70	74.38	74.00	26.00	15.33	2.88	0.00	0.68
Logic	Boolean expr.	92.67	99.33	99.33	96.00	100.00	98.67	98.00	99.33	90.00	99.33	92.00	93.24
Logic	Logical symb.	100.00	94.00	77.33	90.67	98.67	95.95	96.67	93.33	98.00	89.93	96.00	91.95
PLC	Instruction List	84.67	99.33	52.67	44.00	88.00	78.00	16.00	92.67	14.67	49.66	20.67	94.96
Quantum Circuit	OpenQASM 2	86.33	87.33	93.33	87.33	65.33	88.67	78.67	82.00	0.00	74.83	8.00	66.00
Quantum Circuit	Quirk	51.00	72.30	58.67	73.33	2.00	12.00	64.67	53.02	64.00	0.00	0.00	39.36
Phylogenetic Tree	Newick	91.33	84.00	90.00	96.00	94.00	90.91	88.67	90.67	94.67	87.25	92.31	67.05
Phylogenetic Tree	PhyloXML	77.33	86.58	70.67	64.00	69.33	31.54	62.67	48.00	58.00	24.16	43.45	45.95
DNA Sequences	FASTA	72.07	84.83	78.00	46.97	92.67	85.00	52.67	80.77	42.00	90.41	51.19	83.87
DNA Sequences	Vienna RNA	94.63	58.54	86.67	87.97	66.00	82.22	53.33	97.22	66.67	64.71	81.03	57.75
Chemical Struct.	SMILES	91.64	91.16	90.67	52.41	83.33	95.95	72.00	88.67	64.00	82.73	88.73	54.08
Chemical Struct.	SMARTS	85.00	66.00	80.00	74.67	92.67	85.71	86.67	80.67	78.00	91.95	90.54	59.66
Chemical Struct.	SMARTS Reac.	47.33	67.11	28.00	40.67	78.67	18.67	20.67	4.67	33.33	79.73	9.46	25.83
Chemical Struct.	PDB	74.58	67.13	68.67	70.07	74.00	59.84	77.33	74.32	66.00	58.11	60.27	39.10
Chemical Struct.	CML	99.00	97.33	97.33	92.67	98.00	94.00	97.33	90.67	98.00	95.95	96.67	89.33
Metab. Pathway	SBML	69.00	79.33	68.67	30.67	77.33	80.00	62.00	59.33	8.67	56.38	0.67	56.00
Metab. Pathway	BioPAX	81.33	98.64	76.00	60.00	91.33	62.00	66.00	67.33	67.33	88.00	62.00	88.67
Deep Learning	ONNX Graph	94.67	99.32	36.00	51.68	71.72	23.29	46.67	44.37	28.67	18.02	28.38	91.20
Deep Learning	Keras Config.	97.33	100.00	90.67	98.00	87.33	98.67	97.33	99.33	98.67	94.59	88.67	97.33
Deep Learning	ProtoBuf	95.67	97.33	15.33	82.55	84.46	82.64	0.00	85.33	72.00	68.84	0.00	70.27
Chess Board	FEN	99.00	95.33	96.00	76.67	84.96	97.20	98.00	98.67	82.67	71.32	95.30	7.25
Chess Board	PGN	78.23	67.33	72.67	80.14	76.00	93.86	85.33	88.97	75.33	34.00	91.34	77.78
Musical Notation	ABC	85.28	82.43	86.00	74.00	81.51	76.51	84.67	89.33	68.00	59.15	84.25	72.93
Musical Notation	LilyPond	73.67	80.00	82.67	75.33	79.19	75.33	75.33	71.33	75.33	64.14	66.00	79.33
Musical Notation	MusicXML	90.67	95.21	96.67	92.67	93.33	92.00	95.33	96.00	94.00	88.89	79.33	94.67
Class Diagram	PlantUML	99.67	100.00	100.00	100.00	98.66	95.27	96.67	96.67	100.00	96.64	98.00	92.00
Class Diagram	Mermaid	97.67	100.00	96.67	100.00	88.00	94.00	86.00	96.67	98.67	80.67	98.00	99.33
Sequence Diag.	PlantUML	99.00	98.00	100.00	98.67	94.00	96.64	99.33	86.00	98.67	89.93	98.67	98.67
Sequence Diag.	Mermaid	99.33	98.00	98.67	100.00	94.00	90.00	96.00	90.67	98.67	97.97	100.00	98.67
Activity Diag.	PlantUML	81.67	73.33	93.33	94.00	68.67	80.67	72.00	49.33	96.67	65.77	92.67	51.33
Activity Diag.	Mermaid	96.33	98.00	98.00	98.67	96.67	89.26	99.33	94.67	99.33	98.65	97.33	97.30
Component Diag.	PlantUML	98.67	87.33	92.00	88.00	96.00	89.26	94.67	84.00	83.33	98.00	96.67	57.33
Usecase Diag.	PlantUML	96.00	98.00	95.33	90.00	96.67	97.30	98.67	83.33	90.67	92.67	94.67	90.60
State Chart	PlantUML	92.00	94.00	90.67	91.33	85.33	68.67	94.67	80.67	97.33	85.91	82.67	86.58
State Chart	Mermaid	97.00	96.00	97.33	96.00	98.00	88.59	98.67	89.33	97.33	87.25	93.33	84.46
State Chart	SCXML	80.33	86.00	98.00	97.33	84.67	70.00	96.00	88.00	98.00	78.67	93.33	92.67
Business Process	BPMN	26.00	98.00	46.67	19.33	67.38	66.00	32.67	74.00	16.00	89.19	70.00	73.33
Business Process	PlantUML	87.67	64.67	96.00	94.00	77.33	86.67	90.00	28.00	95.33	74.00	90.00	36.67
Business Process	Mermaid	95.67	98.00	99.33	98.67	93.33	97.33	99.33	92.67	98.67	97.32	98.67	98.65
Business Process	D2	82.67	87.33	44.00	86.67	70.67	53.33	27.33	18.67	86.67	37.33	6.00	0.67
Business Process	DOT	99.00	100.00	100.00	98.67	100.00	100.00	100.00	97.33	98.00	99.33	96.67	96.67
Gantt Diagram	PlantUML	38.67	31.33	48.67	32.89	7.33	4.00	12.00	34.67	21.33	10.00	12.67	0.00
Gantt Diagram	Mermaid	84.00	87.33	93.33	78.67	77.33	62.42	72.67	69.33	69.33	66.00	82.67	51.33
ER-Diagram	Mermaid	97.33	94.00	33.33	96.67	89.33	92.00	74.67	92.67	98.67	82.55	84.00	80.00
ER-Diagram	PlantUML	8.33	25.33	97.33	94.00	0.00	13.42	24.67	0.00	95.33	6.00	78.67	63.33
ER-Diagram	MySQL DDL	95.67	98.67	96.67	98.67	96.00	86.00	97.33	91.33	98.67	90.00	94.67	88.67
ER-Diagram	Postgres DDL	99.67	99.33	100.00	97.33	90.00	76.00	98.00	94.67	98.67	87.25	98.67	88.00
ER-Diagram	DBML	74.00	78.67	66.00	82.00	34.67	76.00	26.00	54.67	69.33	53.02	69.33	45.33
ER-Diagram	PSL	81.33	73.33	74.00	66.67	74.00	80.67	68.67	64.67	69.33	69.80	56.67	34.67
Mind Map	Mermaid	99.33	97.33	100.00	94.67	100.00	93.33	98.67	99.33	98.00	97.33	100.00	100.00
Mind Map	PlantUML	100.00	98.00	98.67	97.33	100.00	92.67	100.00	100.00	92.67	98.66	99.33	100.00
Mind Map	Markmap	100.00	100.00	99.33	100.00	100.00	98.00	100.00	99.33	100.00	100.00	99.33	100.00
Graph	GraphML	85.00	93.33	95.33	94.67	94.67	93.33	82.00	96.00	93.33	96.00	54.00	92.67
Graph	GML	94.67	95.33	93.33	94.67	86.00	93.96	88.67	86.67	90.00	95.27	79.33	87.92
Knowledge Graph	RDF XML	96.33	96.67	99.33	93.33	97.33	91.33	98.00	86.67	92.00	92.62	84.67	91.33
Knowledge Graph	RDF Turtle	90.67	94.00	80.67	88.00	90.00	90.67	94.67	79.33	92.00	95.97	86.67	74.00
Knowledge Graph	OWL XML	33.67	90.00	98.67	0.67	82.00	74.67	66.67	76.00	4.00	78.52	76.67	0.00
Knowledge Graph	OWL Turtle	95.67	95.33	97.33	82.00	90.67	93.33	80.00	84.00	86.67	93.29	89.33	61.33
Overall		89.68	84.38	79.43	78.35	77.90	75.79	74.96	74.57	74.24	72.15	69.69	69.19

Table 7: Model performance on distinct FRLs with regard to code validity in %. (Part 2)

Category	FRL	Nemotron-4-340B-Instruct-FP8	Mistral-Small-3.1-24B	XBai-o4	Mixtral-8x22B-Instruct	Open-hands-LM-32B-v0.1	C4AI-Comm-and A	Llama-3.3-70B-Instruct	Qwen-2.5-14B-Instruct	Qwen3-30B-A3B-Instruct-2507	Phi-4-reasoning	Deep Seek-Coder-V2-Lite	QWQ-32B
Electrical Circuit	SPICE Netlist	79.33	52.67	82.55	84.67	77.33	87.33	80.67	89.51	81.68	79.02	84.56	82.67
Electrical Circuit	CircuiTikZ	71.33	70.00	26.00	74.00	29.33	40.00	68.67	66.21	27.83	63.50	77.55	43.33
Hardware Design	VHDL	62.67	54.67	66.67	45.33	60.67	52.67	34.00	68.67	43.36	43.57	58.00	65.33
Hardware Design	Verilog	84.67	84.67	90.67	84.00	86.67	86.67	78.67	85.81	81.75	87.41	91.33	88.67
Hardware Design	Yosys	0.00	0.00	0.00	0.00	12.67	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Logic	Boolean expr.	94.67	96.00	100.00	66.67	85.33	98.00	94.00	94.63	77.18	96.67	85.03	96.00
Logic	Logical symb.	91.33	96.67	99.33	82.67	94.67	89.33	98.00	96.67	91.89	8.94	85.23	98.67
PLC	Instruction List	68.67	81.33	64.00	38.67	52.00	59.33	85.33	82.55	97.32	36.62	41.61	52.67
Quantum Circuit	OpenQASM 2	20.00	64.67	77.33	72.67	37.33	14.67	89.33	69.33	71.33	48.67	46.67	74.00
Quantum Circuit	Quirk	0.00	0.00	2.67	0.00	41.33	7.33	44.67	0.00	9.89	0.00	0.00	6.67
Phylogenetic Tree	Newick	76.67	76.67	93.96	80.00	87.33	89.33	74.00	94.67	88.00	87.76	97.20	83.69
Phylogenetic Tree	PhyloXML	36.00	37.33	18.79	40.67	47.33	29.33	68.67	44.67	22.82	18.18	47.97	8.84
DNA Sequences	FASTA	28.67	48.67	83.69	0.00	34.00	14.67	50.00	2.07	79.38	50.40	40.00	83.81
DNA Sequences	Vienna RNA	34.67	59.33	78.38	33.33	38.67	39.33	68.00	59.76	55.38	66.41	57.75	72.26
Chemical Struct.	SMILES	74.67	80.67	82.43	53.33	53.33	84.67	86.00	61.49	83.19	85.33	19.58	74.31
Chemical Struct.	SMARTS	57.33	72.00	81.08	65.33	72.00	53.33	80.00	68.49	58.70	83.78	22.45	87.92
Chemical Struct.	SMARTS Reac.	2.00	16.67	16.67	4.00	16.00	1.33	21.33	30.20	8.47	45.64	7.64	16.11
Chemical Struct.	PDB	39.33	56.67	56.00	41.33	69.33	68.67	69.33	72.11	55.63	38.10	17.33	64.67
Chemical Struct.	CML	93.33	96.00	74.00	95.33	95.33	97.33	99.33	96.00	90.67	88.67	99.33	85.33
Metab. Pathway	SBML	40.00	2.67	6.67	34.00	16.00	2.67	50.00	20.67	37.33	38.00	40.67	45.33
Metab. Pathway	BioPAX	79.33	57.33	40.00	69.33	59.33	51.33	12.67	55.33	64.67	58.00	42.00	74.67
Deep Learning	ONNX Graph	68.00	24.67	14.00	76.00	44.67	42.00	1.33	33.10	79.41	16.38	40.54	21.23
Deep Learning	Keras Config.	84.00	87.33	98.67	85.33	86.00	93.33	94.67	76.00	91.33	81.16	0.67	96.00
Deep Learning	ProtoBuf	59.33	1.33	8.00	42.67	46.67	32.00	19.33	12.67	1.35	12.32	23.81	6.04
Chess Board	FEN	95.33	72.00	90.00	28.00	81.33	74.00	76.67	29.03	1.69	83.80	73.21	87.25
Chess Board	PGN	46.67	90.00	93.15	66.67	85.33	84.67	100.00	78.10	80.52	79.07	2.74	87.50
Musical Notation	ABC	53.33	76.67	69.33	95.33	58.67	80.00	46.67	89.93	65.69	58.39	82.54	62.00
Musical Notation	LilyPond	74.00	76.67	75.33	68.00	63.33	72.67	93.33	78.00	56.67	68.00	76.67	68.67
Musical Notation	MusicXML	78.00	91.33	80.00	78.00	68.67	92.67	41.33	89.33	90.00	60.67	89.33	70.00
Class Diagram	PlantUML	100.00	93.33	96.67	98.00	86.00	95.33	85.33	96.67	95.97	96.83	98.67	57.33
Class Diagram	Mermaid	96.67	98.67	97.33	99.33	98.00	94.67	95.33	86.00	96.64	75.00	97.33	55.33
Sequence Diag.	PlantUML	98.67	99.33	96.00	94.00	87.33	97.33	94.00	98.00	98.67	94.78	98.67	84.67
Sequence Diag.	Mermaid	98.67	98.67	95.33	100.00	100.00	98.67	98.67	98.67	92.00	84.14	97.33	94.00
Activity Diag.	PlantUML	81.33	89.33	65.33	85.33	73.33	14.00	43.33	88.00	38.00	58.14	93.29	38.67
Activity Diag.	Mermaid	98.67	98.67	80.67	95.33	97.33	100.00	48.00	88.67	86.99	91.95	92.00	85.33
Component Diag.	PlantUML	96.67	81.33	96.67	82.00	60.00	90.67	96.67	61.33	75.33	90.62	59.33	92.00
Usecase Diag.	PlantUML	98.00	93.33	94.00	90.00	60.00	86.00	83.33	90.67	96.64	89.71	84.67	58.00
State Chart	PlantUML	92.67	72.67	85.33	85.33	54.67	94.00	94.67	34.00	67.57	68.89	73.33	39.33
State Chart	Mermaid	89.33	71.33	91.33	96.67	76.67	94.00	93.33	94.00	76.67	86.71	98.00	66.67
State Chart	SCXML	96.67	87.33	80.00	93.33	89.33	94.67	95.33	98.00	84.00	62.67	97.33	91.33
Business Process	BPMN	30.00	36.67	41.33	52.00	26.67	48.67	72.67	61.33	36.67	61.33	26.67	39.33
Business Process	PlantUML	73.33	76.00	73.15	88.67	77.33	18.67	72.00	85.33	53.02	63.71	94.63	46.67
Business Process	Mermaid	100.00	99.33	92.00	99.33	97.33	98.00	47.33	96.67	89.86	94.41	97.99	85.33
Business Process	D2	9.33	14.00	59.33	1.33	49.33	46.00	14.00	21.33	11.41	10.27	7.33	3.33
Business Process	DOT	100.00	98.00	98.66	100.00	99.33	100.00	96.00	100.00	97.33	99.19	98.00	93.96
Gantt Diagram	PlantUML	15.33	10.67	0.00	8.67	9.33	4.00	42.00	0.67	11.33	0.00	0.00	3.33
Gantt Diagram	Mermaid	60.00	78.00	57.05	42.67	40.00	58.00	62.67	77.33	50.34	52.05	90.67	18.00
ER-Diagram	Mermaid	48.67	48.67	90.67	95.33	90.00	94.00	92.67	60.00	82.07	53.74	83.33	45.33
ER-Diagram	PlantUML	54.00	72.00	10.00	88.67	63.33	89.33	92.67	4.00	0.67	12.00	86.00	27.33
ER-Diagram	MySQL DDL	94.67	98.67	84.67	98.67	92.00	99.33	100.00	96.00	85.33	89.44	96.00	96.67
ER-Diagram	Postgres DDL	95.33	98.67	84.00	95.33	94.00	94.00	99.33	94.00	76.67	83.82	98.67	97.33
ER-Diagram	DBML	74.67	89.33	63.33	65.33	36.67	60.67	28.67	18.00	14.67	31.21	35.33	26.00
ER-Diagram	PSL	73.33	0.00	56.00	56.67	52.00	62.67	48.67	45.33	33.33	36.00	26.00	58.67
Mind Map	Mermaid	92.67	99.33	96.00	88.67	88.67	84.00	54.00	36.67	98.00	78.77	93.33	32.00
Mind Map	PlantUML	88.00	92.00	95.33	70.00	81.33	95.33	12.67	63.33	100.00	81.51	97.33	97.33
Mind Map	Markmap	96.00	93.33	99.33	98.00	92.67	96.67	100.00	95.33	100.00	100.00	92.62	94.67
Graph	GraphML	92.67	90.00	94.00	53.33	75.33	94.67	28.67	79.33	92.67	90.00	96.00	89.33
Graph	GML	64.00	88.00	88.00	76.00	72.00	49.33	15.33	86.00	56.76	89.71	65.33	80.41
Knowledge Graph	RDF XML	78.00	98.00	96.00	70.00	75.33	78.67	92.67	89.33	76.00	80.00	76.67	87.33
Knowledge Graph	RDF Turtle	86.67	88.00	72.00	81.33	75.33	80.67	66.67	82.00	66.00	77.33	51.33	61.33
Knowledge Graph	OWL XML	56.67	64.67	72.67	39.33	0.67	0.00	51.33	1.33	0.00	63.33	2.67	48.00
Knowledge Graph	OWL Turtle	90.00	88.67	49.33	91.33	64.00	64.00	66.67	76.00	65.33	76.67	56.67	57.33
Overall		69.14	68.61	67.48	66.08	64.62	64.49	64.29	63.47	61.59	61.45	61.32	60.10

Table 8: Model performance on distinct FRLs with regard to code validity in %. (Part 3)

Category	FRL	Magistral-Small-2507	Gemma-3-27B-IT	DeepSeek-R1-Distill-Qwen-32B	Star-Coder2-15B-Instruct	Qwen-2.5-7B-Instruct	Gemma-3-12B-IT	Mixtral-8x7B-Instruct	Phi-3.5-MoE-instruct	Deep-Coder-14B	Code-Gemma-7b	Phi-4-Mini-Instruct	SmolLM3-3B
Electrical Circuit	SPICE Netlist	70.95	56.67	81.63	71.33	72.03	62.00	71.94	85.52	66.44	65.73	84.09	52.08
Electrical Circuit	CircuiTikZ	41.50	28.67	35.17	84.72	60.84	10.74	43.94	66.90	8.78	33.57	31.43	50.00
Hardware Design	VHDL	38.00	39.33	53.33	63.09	38.00	23.33	47.12	60.00	28.67	17.33	82.67	13.42
Hardware Design	Verilog	68.67	82.67	92.57	70.00	78.67	78.00	89.52	89.33	51.33	71.33	89.26	23.13
Hardware Design	Yosys	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.68
Logic	Boolean expr.	98.61	76.67	89.93	18.12	76.19	98.00	70.50	92.00	89.19	56.00	80.43	80.67
Logic	Logical symb.	96.38	92.00	79.02	18.67	88.73	88.67	62.68	98.00	81.08	62.00	91.67	60.16
PLC	Instruction List	49.66	72.67	55.10	10.07	43.24	69.33	1.08	0.67	65.31	1.34	3.03	2.29
Quantum Circuit	OpenQASM 2	20.67	0.00	49.33	66.67	55.33	0.00	60.67	78.67	10.00	41.33	57.33	0.00
Quantum Circuit	Quirk	0.00	0.00	11.68	16.28	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Phylogenetic Tree	Newick	74.40	80.00	84.67	67.35	81.88	86.39	81.62	51.77	72.48	56.38	51.28	9.70
Phylogenetic Tree	PhyloXML	46.31	0.00	6.00	53.47	0.00	14.00	6.20	15.65	9.33	1.41	17.24	0.00
DNA Sequences	FASTA	44.22	2.00	34.92	21.43	41.91	10.84	2.42	1.53	15.17	23.68	12.50	0.00
DNA Sequences	Vienna RNA	75.89	28.00	72.79	25.18	40.30	23.91	32.17	62.32	37.40	67.74	54.05	0.70
Chemical Struct.	SMILES	55.04	54.67	54.79	54.07	64.79	45.89	56.52	40.50	64.63	30.71	29.03	3.79
Chemical Struct.	SMARTS	71.88	44.00	64.43	30.07	60.00	24.83	52.99	48.04	66.67	35.61	32.80	16.30
Chemical Struct.	SMARTS Reac.	3.17	22.00	5.41	10.29	22.38	3.36	9.17	4.59	0.68	5.19	8.57	0.00
Chemical Struct.	PDB	42.47	54.00	59.33	25.52	11.35	39.60	12.39	1.35	5.33	15.00	7.56	0.70
Chemical Struct.	CML	80.67	96.67	86.00	47.33	45.71	84.00	78.67	44.00	85.33	85.33	50.00	18.00
Metab. Pathway	SBML	12.00	0.67	8.00	2.67	0.00	0.00	4.67	13.33	0.00	0.00	0.00	0.00
Metab. Pathway	BioPAX	18.00	38.00	83.33	82.00	27.89	24.00	32.00	7.33	60.67	37.33	14.67	10.67
Deep Learning	ONNX Graph	10.67	1.33	11.33	42.86	11.19	0.00	1.00	1.42	0.67	0.00	0.00	0.00
Deep Learning	Keras Config.	34.00	98.67	84.00	69.11	68.71	53.69	58.14	16.11	68.00	29.73	2.04	0.00
Deep Learning	ProtoBuf	0.68	2.00	19.86	4.69	2.99	4.67	17.42	18.24	0.67	2.07	0.00	0.00
Chess Board	FEN	73.91	62.67	78.03	80.99	76.87	80.95	11.82	0.00	46.88	0.00	0.00	0.00
Chess Board	PGN	69.18	58.00	74.83	44.30	61.94	81.12	68.85	72.60	61.33	29.63	9.56	0.72
Musical Notation	ABC	73.83	86.67	57.34	56.08	53.85	84.67	83.61	77.70	41.89	68.28	33.33	23.08
Musical Notation	LilyPond	60.67	36.00	42.67	90.00	72.81	38.00	45.33	50.00	30.00	16.67	26.67	6.67
Musical Notation	MusicXML	58.00	70.00	77.33	60.67	74.15	45.33	48.67	80.00	26.00	9.33	0.67	0.00
Class Diagram	PlantUML	80.41	96.00	95.33	70.47	74.67	98.67	96.97	97.33	74.67	60.67	79.19	36.36
Class Diagram	Mermaid	90.00	94.00	88.67	41.22	71.33	55.33	87.22	86.67	74.00	36.24	45.58	20.14
Sequence Diag.	PlantUML	93.29	89.33	86.67	76.03	84.67	89.33	96.25	94.00	67.33	72.67	78.52	45.52
Sequence Diag.	Mermaid	94.67	86.00	98.66	62.00	98.67	77.33	91.97	82.00	83.33	93.33	68.46	68.46
Activity Diag.	PlantUML	81.33	41.33	42.67	69.01	72.00	50.67	72.50	15.33	18.67	84.56	33.78	40.41
Activity Diag.	Mermaid	96.00	75.33	90.67	51.45	97.33	76.00	93.16	91.33	80.67	72.67	74.15	34.25
Component Diag.	PlantUML	91.33	89.33	70.00	78.77	80.67	95.33	47.06	64.67	40.00	87.33	58.39	24.49
Usecase Diag.	PlantUML	93.33	90.00	74.00	77.08	80.00	95.33	57.14	52.00	40.00	76.67	48.00	24.83
State Chart	PlantUML	65.33	87.33	40.27	76.67	53.33	76.00	12.33	52.67	28.86	58.78	48.32	25.74
State Chart	Mermaid	98.00	91.33	44.00	63.76	90.67	82.67	90.00	88.67	82.67	86.58	38.41	58.78
State Chart	SCXML	94.00	92.67	84.67	84.00	87.33	90.67	78.67	82.67	83.33	86.67	93.33	12.67
Business Process	BPMN	19.33	39.33	36.00	54.67	63.76	51.33	29.33	0.00	3.33	33.33	47.33	0.00
Business Process	PlantUML	73.33	58.67	38.67	84.09	72.00	52.00	55.32	15.07	31.76	81.21	32.19	41.55
Business Process	Mermaid	97.33	80.00	94.67	64.19	98.67	54.67	90.68	70.55	86.00	78.00	82.43	41.26
Business Process	D2	36.91	2.00	12.08	12.33	54.42	3.33	7.69	3.60	3.36	2.68	14.29	0.00
Business Process	DOT	99.33	98.67	98.00	82.73	99.33	100.00	98.45	73.13	91.22	92.57	93.24	79.29
Gantt Diagram	PlantUML	4.67	9.33	4.03	53.10	0.00	0.00	16.09	15.33	14.00	13.51	11.41	11.94
Gantt Diagram	Mermaid	57.33	33.33	22.67	81.33	23.65	18.00	25.00	16.11	7.33	1.33	7.64	0.00
ER-Diagram	Mermaid	62.67	45.33	27.33	41.33	51.68	67.33	80.62	7.33	0.00	2.00	21.50	0.70
ER-Diagram	PlantUML	23.33	68.00	74.00	85.81	62.00	15.33	4.65	67.33	41.61	77.70	31.29	0.74
ER-Diagram	MySQL DDL	96.00	97.33	93.33	75.33	97.33	96.67	87.50	92.00	79.33	82.67	92.00	86.67
ER-Diagram	Postgres DDL	94.00	100.00	86.58	72.67	94.67	97.33	88.89	82.67	83.33	81.33	89.33	89.33
ER-Diagram	DBML	6.71	21.33	19.59	61.70	2.19	34.69	26.83	0.00	0.00	5.56	0.00	0.00
ER-Diagram	PSL	46.00	63.33	26.00	78.67	44.96	36.67	37.33	32.67	1.33	6.67	17.33	0.00
Mind Map	Mermaid	80.00	91.33	92.00	0.69	62.67	56.00	81.89	72.92	26.67	61.49	23.78	28.67
Mind Map	PlantUML	73.33	95.33	85.91	78.62	7.33	77.33	38.81	36.91	17.57	36.05	16.30	6.21
Mind Map	Markmap	100.00	98.67	98.00	100.00	100.00	100.00	94.21	78.00	97.33	98.18	98.64	95.80
Graph	GraphML	48.00	97.33	42.67	58.00	25.50	87.33	68.00	40.00	28.00	29.33	46.00	60.00
Graph	GML	50.00	24.67	71.33	78.52	44.59	4.67	31.65	56.00	47.33	45.27	39.37	11.76
Knowledge Graph	RDF XML	85.33	77.33	82.67	54.00	76.67	85.33	40.67	42.00	52.00	32.00	2.67	6.00
Knowledge Graph	RDF Turtle	84.00	73.33	74.67	80.00	66.44	66.67	59.33	40.67	62.00	4.00	24.00	16.00
Knowledge Graph	OWL XML	54.67	60.67	58.00	69.33	25.34	55.33	8.00	20.00	22.67	8.67	1.33	1.33
Knowledge Graph	OWL Turtle	73.33	70.00	62.67	83.33	46.00	57.33	44.67	37.33	55.33	8.00	12.00	16.00
Overall		58.44	57.64	57.42	54.98	54.18	51.89	49.09	45.87	40.99	40.04	37.35	21.62