ESANN 2026 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium) and online event, 22-24 April 2026, i6doc.com publ., ISBN 9782875870964. Available from http://www.i6doc.com/en/.

Code-Guided Reasoning in Vision-Language Models for Complex Diagram Understanding

Daniel Steinigen[1,2,4], Lucie Flek[1,2,3] and 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. Understanding complex structured diagrams, such as circuit schematics, molecular structures, musical notation, or business process models, requires precise symbolic, spatial, and relational reasoning. Current vision-language models (VLMs) struggle with such tasks because they lack access to the underlying symbolic structure that governs these diagrams. We introduce a training paradigm in which VLMs explicitly learn to reason through an intermediate symbolic representation of the image that is expressed in code. We generate a large synthetic dataset covering 21 diagram types across 7 domains by prompting large language models to generate code in specific formal representation languages (FRLs) and rendering them into paired code-image samples. During VLM training, the FRL code is provided along with the image, enabling the model to incorporate the symbolic representation during reasoning. Experiments show that models capable of producing valid code benefit from this symbolic intermediate layer, yielding improved accuracy on diagram understanding tasks. Our results demonstrate that integrating symbolic code into VLM training offers a promising direction for VLM design to handle complex visual data by bridging diagram perception with symbolic reasoning.

1 Introduction

Vision-language models (VLMs) have recently achieved strong performance across diverse multimodal tasks, yet they continue to struggle with structured diagrams such as circuit schematics, molecular structures, musical notation, business process models, or class diagrams [1]. These visualizations encode information using precise symbolic primitives, compositional patterns, and domain-specific relational constraints. Understanding them requires more than perceptual recognition: it demands symbolic reasoning, structural interpretation, and knowledge of formal domain rules. Current VLMs lack explicit access to this underlying symbolic structure, which limits their ability to generalize to complex visual formats. Prior efforts to strengthen relational reasoning, including scene graph-based approaches [2, 3], which explicitly decompose images into entities and relations, while program-aided prompting methods like PAL [4] and Program-of-Thought (PoT) [5] show that generating and executing code can substantially improve logical inference. In multimodal settings, systems such as VisProg [6] and ViperGPT [7] extend this idea by having LLMs produce Python programs that coordinate visual modules to solve complex perception tasks. Yang et al. [8]

ESANN 2026 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium) and online event, 22-24 April 2026, i6doc.com publ., ISBN 9782875870964. Available from http://www.i6doc.com/en/.

present CoSyn, a framework for generating synthetic multimodal data based on LLM-generated code to improve VLM reasoning on charts and documents.

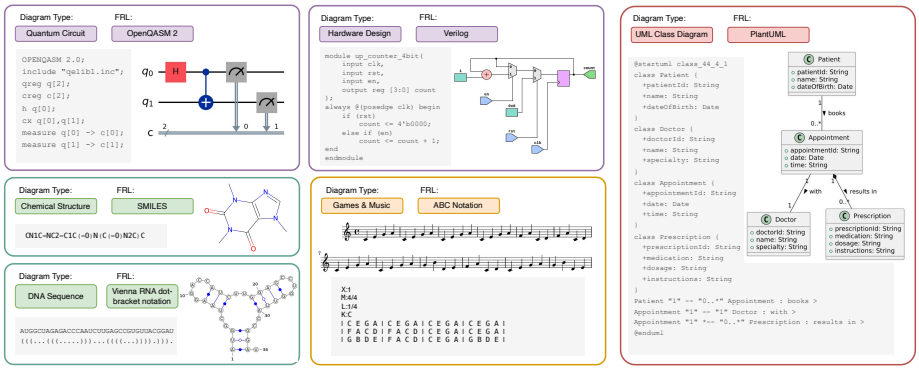

Fig. 1: Examples of rendered diagrams and their FRL code representation.

In this work, we want to address this limitation by linking the visual content to formal representation languages (FRLs), which provide a structured code-based syntax to encode domain-specific rules containing symbols, their relationships, and spatial structures. Therefore, we propose a code-augmented training paradigm in which VLMs learn to reason about diagrams through an intermediate symbolic code representation, a dimension of code-guided reasoning that has not yet been examined in prior work to the best of our knowledge. We construct a large synthetic dataset *StructVis* spanning 21 diagram types across 7 domains by prompting large language models to generate code in diverse FRLs, such as *circuit netlists*, *SMILES*[1], *ABC-Notation*[2], *BPMN*[3] or *PlantUML*[4], and rendering them into images. This pipeline provides paired visual-symbolic samples with exact structural correspondence, examples of which are shown in Figure 1.

During training, the underlying FRL code is embedded into the model's reasoning trace via structured thinking tags, enabling the VLM to internalize and utilize a symbolic intermediate space. Our experiments show that, for models capable of producing syntactically valid code, this symbolic representation consistently improves diagram understanding and reasoning performance.

Our contributions are:

1. A code-augmented training paradigm that exposes VLMs to the FRL code underlying the diagram using reasoning tags.
2. An evaluation demonstrating that intermediate code representations significantly improve reasoning performance on complex diagram tasks, given the model possesses sufficient code generation ability.

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

ESANN 2026 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium) and online event, 22-24 April 2026, i6doc.com publ., ISBN 9782875870964. Available from http://www.i6doc.com/en/.

2 Code-Guided Diagram Generation and Reasoning

For enabling models reasoning on complex structured diagrams, we construct a dataset that couples diagrams with their FRL, and we introduce a paradigm that trains VLMs to use this symbolic code as an intermediate reasoning layer.

2.1 Synthetic Dataset Generation Pipeline

We design a fully automated pipeline that generates a large vision-language dataset covering a diverse set of diagram types. For each diagram type, we have a domain-appropriate FRL capable of describing its structure. Examples include circuit netlists for electronic schematics, SMILES for molecular structures, ABC notation for musical scores, BPMN for business process flows, and PlantUML for class diagrams. Given the diagram type, the large language model (LLM), specifically *Qwen3-Coder-480B-A35B-Instruct*[5], 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 natural language question-answer pair associated with the code snippet. By including unique persona descriptions we ensure a diverse dataset that reflects real-world domain problems.

As next step, each code snippet is rendered into a diagram using a comprehensive rendering toolkit consisting of 35 different rendering tools, comprising various open source Python packages, e.g. RDKit[6], JavaScript libraries, e.g. DBML Cli[7], and Linux command-line tools, e.g. Yosys[8]. Random variation between tool settings and schemes increases the diversity of the generated diagrams. The result is a rich dataset covering 21 diagram types across 7 domains, including *Electrical Engineering, Computer Science & AI, Software Engineering & System Modeling, Biology, Chemistry, Business & Process Management, Games & Music*. Because every image comes from a programmatically generated symbolic description, its visual structure maps exactly and unambiguously to the symbolic content. Finally each rendered diagram undergoes a multi-stage filtering process to obtain a high quality dataset. This includes sorting out samples with invalid code, that are overly or insufficiently complex, for which images have an unusual aspect ratio, or for which images have a low variance. 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 and employ a LLM-as-a-judge (LLMaaJ) approach using the LLM *GPT-OSS-120B*[9] to validate the correctness and the logical consistency of the generated samples.

With an almost equal distribution across diagram types and question types covering both closed-ended and open-ended questions our final training dataset *StructVis* (Structured Visualizations) contains 218K samples.

[5] `https://huggingface.co/Qwen/Qwen3-Coder-480B-A35B-Instruct`
[6] `https://github.com/rdkit/rdkit`
[7] `https://dbml.dbdiagram.io/cli`
[8] `https://github.com/YosysHQ/yosys`
[9] `https://huggingface.co/openai/gpt-oss-120b`

ESANN 2026 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium) and online event, 22-24 April 2026, i6doc.com publ., ISBN 9782875870964.
Available from http://www.i6doc.com/en/.

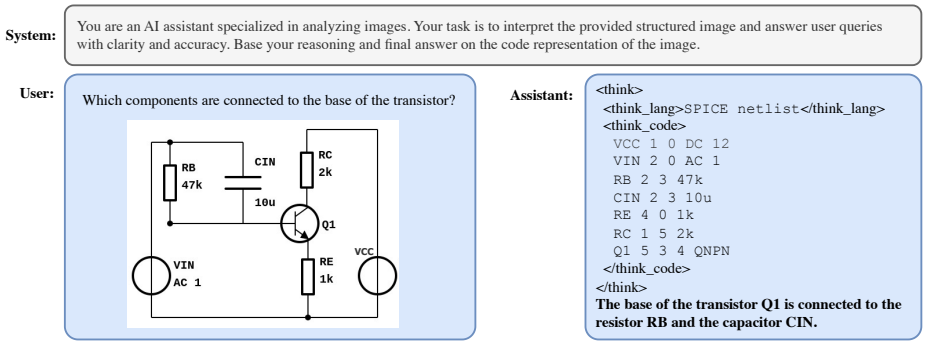

Fig. 2: Training paradigm with incorporated symbolic code representation.

2.2 Code-Augmented Training Paradigm

To leverage the symbolic structure captured in the dataset, we introduce a training paradigm in which the VLM is exposed to the FRL code representation of each image and trained to incorporate this symbolic information during its reasoning process. During supervised fine-tuning, the target output for each sample includes a structured reasoning segment enclosed in special *<think>* tags with the FRL identifier enclosed in *<think_lang>* and the actual symbolic code enclosed in *<think_code>*. An example of the formatting is shown in Figure 2.

This structured annotation provides the model with a symbolic intermediate representation, analogous to a latent code space, through which the model can reason about the depicted structure before producing the final answer. During training the model can learn to infer the symbolic structure from the image and integrate this code representation when solving downstream tasks.

3 Experiments

To assess our proposed code-augmented training paradigm, we fine-tune VLMs of different sizes on our dataset and compare several training configurations.

Table 1: Performance of trained models, and validity of predicted code representations overall, for correct and incorrect predictions, in %.

Model	**Our test set**	**Code-Validity** all	corr.	incorr.
SmolVLM2-2.2B-Instruct_original	32.90	-	-	-
SmolVLM2-2.2B-Instruct_standard	**46.52**	-	-	-
SmolVLM2-2.2B-Instruct_code-augmented	38.86	74.26	81.56	69.62
Gemma-3-4b-it_original	52.56	-	-	-
Gemma-3-4b-it_standard	68.26	-	-	-
Gemma-3-4b-it_code-augmented	**73.18**	89.92	90.32	88.82

ESANN 2026 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium) and online event, 22-24 April 2026, i6doc.com publ., ISBN 9782875870964. Available from http://www.i6doc.com/en/.

3.1 Experimental Setup

We conduct experiments using two VLM architectures that orginate from a different model family than the model used to generate the dataset, *SmolVLM2-2.2B*[10] and *Gemma-3-4b-it*[11]. For the training, we compare two settings: 1) *Standard fine-tuning*, where the model is trained only to produce the final answer, and 2) *Code-augmented fine-tuning*, where the symbolic FRL code is embedded in structured *<think>* tags within the target output.

We split our *StructVis* dataset into 213K samples for training, 2.6K samples for validation and 2.4K samples for test, where the model selection is performed using validation loss, and final results are reported on the test set. We use the TRL library[12] to train multiple models with learning rates of *2e-5*, *5e-5*, and *1e-4* over 3 epochs for a single-run and select the checkpoints with the lowest validation loss report. For evaluation we use the closed-ended questions of the test set and report the exact match between the model's predicted answer and the gold label as metric. Moreover we evaluate on public benchmarks that target structured diagrams using *LMMs-Eval*[13]. For the code-augmented models, we additionally evaluate the validity of the generated FRL code utilizing our toolkit.

Table 2: Performance of trained models on public benchmarks, in %.

Model	ChartQA	InfoVQA	DocVQA
SmolVLM2-2.2B-Instruct_original	68.84	37.97	70.51
SmolVLM2-2.2B-Instruct_StructVis	**69.87**	**40.58**	**72.83**
Gemma-3-4b-it_original	50.92	40.39	68.32
Gemma-3-4b-it_StructVis	**60.32**	**43.37**	**74.19**

3.2 Results

Table 1 summarizes the results. Overall, all fine-tuned models significantly outperform their respective base models on our test set. For *SmolVLM2-2.2B*, the code-augmented training paradigm falls short of the standard training variant. This suggests that the reduced capacity likely limits the model's ability to reconstruct valid code and to leverage the symbolic intermediate representation effectively. The model also exhibits degenerative repetition within the code trace for some cases. In contrast, *Gemma-3-4b-it* benefits from the code-augmented paradigm, achieving higher accuracy than its non-code-augmented counterpart. This indicates that incorporating symbolic code representations during training can substantially enhance VLM reasoning over structured diagrams but only when the model is sufficiently capable of generating the underlying code accurately. The code validity results further support this observation: *Gemma-3-4b-it* produces valid FRL code at higher rates than *SmolVLM2-2.2B*, indicating

[10] `https://huggingface.co/HuggingFaceTB/SmolVLM2-2.2B-Instruct`
[11] `https://huggingface.co/google/gemma-3-4b-it`
[12] `https://huggingface.co/docs/trl`
[13] `https://github.com/EvolvingLMMs-Lab/lmms-eval`

ESANN 2026 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium) and online event, 22-24 April 2026, i6doc.com publ., ISBN 9782875870964. Available from http://www.i6doc.com/en/.

that valid code provides a strong inductive signal that improves downstream reasoning. The results on public benchmarks in Table 2 show that fine-tuning VLMs using our *StructVis* dataset improves the model's performance on image types beyond those contained in our dataset, demonstrating generalization.

4 Conclusion

We introduced a symbolic, code-augmented training paradigm that enhances the reasoning capabilities of vision-language models by grounding structured diagrams in their corresponding formal code representations. By generating a large paired dataset of diagram images and FRL code, we enabled VLMs to learn a direct mapping between visual structure and symbolic encoding. Our code-augmented training approach further leverages this mapping by embedding the underlying FRL code into the model's reasoning process.

Experiments on our dataset demonstrate that models with sufficient capacity, benefit from intermediate code representations, achieving higher accuracy and producing more valid symbolic code. Smaller models, however, struggle to utilize symbolic traces effectively, underscoring the importance of code-generation ability in code-guided VLM training. We release our pipelines, toolkit and dataset.[14]

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.

References

[1] Xiang Yue, Yuansheng Ni, Kai Zhang, et al. Mmmu: A massive multi-discipline multimodal understanding and reasoning benchmark for expert agi. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

[2] Drew A. Hudson and Christopher D. Manning. Learning by abstraction: The neural state machine. *arXiv preprint arXiv:1907.03950*, 2019.

[3] Roei Herzig, Alon Mendelson, Leonid Karlinsky, et al. Incorporating structured representations into pretrained vision & language models using scene graphs. In *Conference on Empirical Methods in Natural Language Processing*, December 2023.

[4] Luyu Gao, Aman Madaan, Shuyan Zhou, et al. Pal: Program-aided language models. *arXiv preprint arXiv:2211.10435*, 2022.

[5] Wenhu Chen, Xueguang Ma, Xinyi Wang, and William W. Cohen. Program of thoughts prompting: Disentangling computation from reasoning for numerical reasoning tasks. *Transactions on Machine Learning Research*, 2023.

[6] Tanmay Gupta and Aniruddha Kembhavi. Visual programming: Compositional visual reasoning without training. *ArXiv*, abs/2211.11559, 2022.

[7] Dídac Surís, Sachit Menon, and Carl Vondrick. Vipergpt: Visual inference via python execution for reasoning. *arXiv preprint arXiv:2303.08128*, 2023.

[8] Yue Yang, Ajay Patel, Matt Deitke, et al. 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)*, 2025.

[14] `https://github.com/danielsteinigen/StructVis`