A Self-Correcting Neuro-Symbolic AI Reasoning Framework

Ben Wooding*, Kiersten Brennan*, Anne M Tumlin,
Hongchao Zhang and Taylor T Johnson

Institute for Software Integrated Systems, Vanderbilt University, Nashville, TN, USA
{ben.wooding, kiersten.brennan, anne.m.tumlin, hongchao.zhang,
taylor.johnson}@vanderbilt.edu

Abstract. Vision-Language Models (VLMs) struggle with explainability and tasks requiring step-by-step reasoning. This paper proposes a neuro-symbolic framework that leverages both logic and neural networks for interpretable results; chaining together convolutional neural networks (CNNs), computer vision techniques, and Satisfiability Modulo Theories (SMT). We introduce a challenging benchmark suite of KenKen and Sudoku puzzles, both NP-complete in the general case. Provided with an unsolved board image, these puzzles require image decomposition, constraint evaluation, and error detection/correction, making it a relevant benchmark suite for VLMs. We compare our neuro-symbolic framework against five VLMs: Gemini-2.5-Pro, GPT-4o, GPT-4o-mini, Claude Sonnet 4.0, and Qwen2.5-VL-7B-Instruct (these were the state-of-the-art at submission time). These VLMs have significant difficulty solving Sudoku or KenKen puzzles beyond 4×4 grids. The neuro-symbolic framework achieves 100% accuracy on computer-generated characters for all classes. For handwritten characters, we demonstrate the neuro-symbolic framework can correct image decomposition errors, improving solve rate. Overall, the benchmark is used to demonstrate the neuro-symbolic framework outperforms VLMs while providing explainable reasoning to enable the self-correction of error. The benchmark suite consists of 2200 board images: seven classes of KenKen puzzle and four classes of Sudoku, for both computer-generated and handwritten characters.

Keywords: Neuro-Symbolic AI · Vision-Language Models · KenKen · Sudoku · Formal Methods

1 Introduction

Vision-Language Models (VLMs) are multimodal artificial intelligence (AI) systems combining computer vision and language processing to understand, interpret and generate content using images and text. Modern VLMs have transitioned to leveraging Large Language Models (LLMs) as text decoders to perform Visual Question Answering (VQA) tasks [25]; this has rapidly improved their reasoning and visual comprehension capabilities, particularly for image captioning, classification, and segmentation. However, hallucinations are still prevalent,

particularly for reasoning tasks described by [22] as reasoning tasks moving from System 1 (simple recall) to System 2 (step-by-step deductions).

Custom neuro-symbolic frameworks can tackle complex reasoning tasks that cause VLMs to struggle, *e.g.*, computing arithmetic expressions from images [34]. In addition, neuro-symbolic artificial intelligence (AI) enables intelligible, transparently generated content with verifiably high accuracy and confidence by integrating symbolic reasoning with neural networks (NNs). Outsourcing the reasoning to established symbolic tools enables deductive step-by-step logic, as well as explainable reasoning for failures.

In this work, we introduce a novel benchmark suite of solvable Sudoku and Kenken board images. We use it to compare a neuro-symbolic framework with modern VLMs. Both Sudoku and KenKen build upon Latin Squares, where a grid has all digits $1, \ldots, n$ appearing uniquely in every row and column. Sudokus are partially-filled Latin Squares, and each digit $1, \ldots, n$ must appear once in each of n non-overlapping boxes. KenKens are empty Latin Squares with no boxes. Instead cages, with a mathematical operation and target value, constrain which digits can appear in which cells. Latin Squares in the general case are NP-complete [8,45], meaning the solution is easy to verify but challenging to find (assuming P$\neq$ NP). All benchmark Sudokus and KenKens have unique solutions, therefore, unsolvable benchmarks will be the cause of image decomposition errors. In addition, Sudokus and KenKens are highly sensitive, a single error renders the entire puzzle unsolvable. This makes them a challenging benchmark suite to test the performance of VLMs for step-by-step reasoning.

We further benchmark the neuro-symbolic framework on its ability to detect and self-correct its errors, and one of the key novelties of our work. It is not uncommon to misclassify handwritten digits, no matter how refined a digit-classification CNN might be. Even humans have stereotypes about the difficulty of reading the handwriting of different professions. However, in some settings, a reasoning-based framework should be able to backtrack and deduce when classification errors have occurred, which we call self-correction. There are two types of error, totally unsolvable board images and solvable but incorrect board images, the latter requires an oracle to inform the solver of the incorrect solution. We propose and evaluate two correction methodologies.

Constraint-based corrections identifies which set of constraints are conflicting within the board image. For KenKens this could be impossible cages, invalid targets/operations for the cage size, or simply rows or columns that have duplicate numbers. This is effective unsolvable board images, and significantly reduces the search space of solutions.

Confidence-based corrections identifies from the CNN, which digits had the lowest classification confidence of being correct. This approach is based on the assumption that if a digit is misclassified, the correct digit should appear second, or third in the list of candidates for a well-formed CNN. This approach can self-correct puzzles where digits are classified incorrectly, but the board image was solvable (just with an incorrect solution).

Both these contribute to the explainable AI conceptualization. Enabling a line of reasoning to be provided as to the reason a solution was incorrect on the first attempt, and the reason for success in a later attempt.

1.1 Contributions

The key contributions of this paper are as follows:

- A benchmark dataset of 2200 KenKen and Sudoku board images. The KenKens, a particular novelty in this work, range from 3×3 to 9×9, and Sudokus from 4×4 to 16×16. Two variant 16×16 grid styles are used for digits over 9; numeric $(10-16)$ and hexadecimal $(A-G)$. All board images are constructed for both computer-generated characters and handwritten characters;
- We benchmark VLMs on the dataset; Gemini-2.5-Pro, GPT-4o, GPT-4o-mini, Claude Sonnet 4.0, and Qwen2.5-VL-7B-Instruct (each state-of-the-art when submitted). No VLM solves a 5×5 KenKen puzzle, and there is limited success beyond 4×4 Sudokus;
- We present a neuro-symbolic framework that achieves 100% accuracy on all benchmarks using computer-generated characters, and significantly outperform the VLMs for board images with handwritten characters;
- For handwritten characters, we demonstrate how the neuro-symbolic framework can detect constraint violations caused by image decomposition errors. The neuro-symbolic framework can then self-correct, increasing the number of solved KenKen puzzles by 22%;
- The benchmark is available:

`https://github.com/Kiguli/KenKen-Sudoku-VLM-Benchmarks`

1.2 Organization

Section 2 presents the theoretical background. Section 3 introduces details about the benchmark suite. Section 4 describes the experimental setup used to benchmark the VLMs and describes the design of the neuro-symbolic framework. The benchmark results are presented in Section 5. In Section 5.1, the VLMs and the neuro-symbolic framework are compared using board images with computer-generated characters. In Section 5.3, the neuro-symbolic framework is compared against the two best VLMs from Section 5.1, testing them on board images with handwritten characters and focusing on self-correction and explainability for the neuro-symbolic framework. We present related works in Section 6. Conclusions and future work are presented in Section 7.

2 Background

2.1 Vision-Language Models

VLMs consist of a visual and textual encoder trained together with learning frameworks like Contrastive Language-Image Pre-training (CLIP) [33]. For VLMs

using CLIP, the visual encoders typically consist of a Convolutional Neural Network (CNN) like ResNet [17] or the now dominant Vision Transformer [12]. Textual encoders use the classic transformer architecture [40]. Trained on matching pairs of images and text, CLIP uses contrastive loss to teach the vision and text encoders to maximize the correlation between correct images and labels. When combined with a decoder, CLIP serves as the backbone for VLMs such as Llava [26]. As an example, Qwen2-VL [41] uses a frozen visual encoder from the CLIP family and a projector that uses cross attention to extract map relevant information from the image to the text embedding space. The text decoder uses pretrained weights from Qwen2 [38] to generate a response based on the prompt and visual information. Recent developments suggest that model families like GPT [20], Claude [2], and Gemini [9] combine visual and textual tokens in a single input stream with special delimiter tokens to represent modality. This process is referred to as transfusion [47] and allows a single transformer architecture to learn visual and textual relationships without explicit modular layers. Although the training and architecture of these closed source models are undisclosed, the uniform multimodal transformer represents a possible shift in VLM development [25].

2.2 Neuro-Symbolic AI

With unprecedented advances in AI, there are genuine concerns around trust, interpretability, and accountability. Neuro-symbolic AI seeks to bring together robust learning in neural networks with reasoning and explainability by offering symbolic representations for neural models [16]. Neuro-symbolic AI aims to integrate neural approaches (neural networks, deep learning, *etc.*) with symbolic approaches (logic, rules, explicit reasoning, *etc.*). The two are complementary, with neural approaches offering scalable generalization and symbolic approaches adding strong reasoning, interpretability, and robustness.

Various applications of this concept have been developed. The authors of [15] develop a tool that encourages LLMs to develop logical step-by-step programs that are then run by a Python interpreter to avoid solution errors prevalent in LLMs. The work [30] built a model that learns visual concepts and words without supervision, simply by looking at images and reading paired questions and answers. neuro-symbolic frameworks have also been applied to formal automata [29,34], anomaly prediction [37], and advanced air mobility [1].

Z3 is a Microsoft Research SMT solver [11] that combines Boolean logic with theory solvers and a Boolean Satisfiability Problem (SAT) solver to search the solution space and return a concrete model. The SAT solver attempts to generate a satisfying solution with validation from the theory solvers and outputs a concrete model if one exists, returning `sat`. If every branch for the SAT solver is pruned due to logical contradictions, then Z3 determines the constraints unsatisfiable, returning `unsat`. Within the context of neuro-symbolic AI, Z3 is an effective tool for the symbolic reasoning component of larger models. One study leveraged LLMs to generate reasoning for structured financial budgeting plans and employed Z3 to ensure the logical validity of the LLM output [35].

2.3 Explainable AI (XAI)

The integration of human logic can not only improve model performance but also promote explainability in AI applications. With the growing employment of large transformer-based models [40] that exhibit black box characteristics, XAI has become an increasingly relevant field of research [13]. One avenue involves investing in inherently explainable deep learning models such as decision trees, rule-based systems, and probabilistic machine learning techniques like Bayesian models [31,14]. Common XAI techniques can be extended to multimodal applications, leveraging attention mechanisms and joint feature attribution to explain relationships between multimodal input and output [21]. Furthermore, prompting techniques such as Chain-of-Thought (CoT) with few-shot prompting encourage large models to explain the reasoning behind their responses [43]. All of these XAI techniques improve our understanding of how such models approach decision making, however, transformer-based models remain non-deterministic and exceptionally large-scale. For sensitive tasks where explainability and reliability are particularly important, turning to a neuro-symbolic framework can be advantageous.

3 Benchmark Details

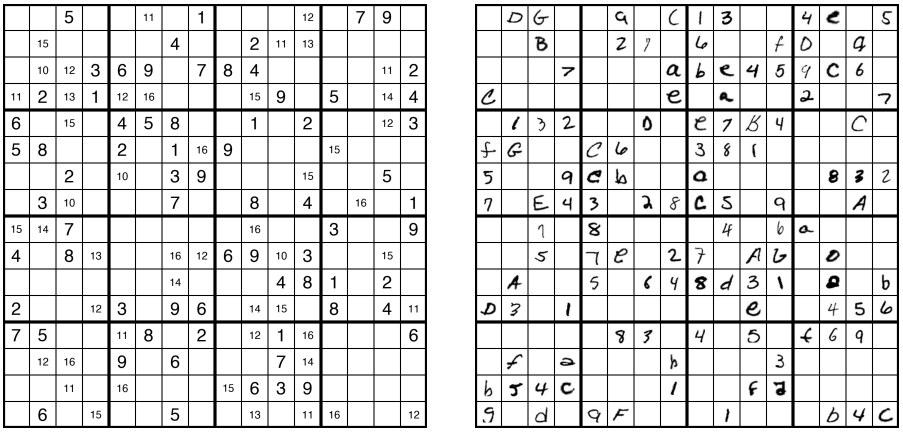
3.1 Sudoku

A Sudoku is a famous logic-based number-placement puzzle played on an $n \times n$ grid, divided into n non-overlapping "boxes", where the goal is to fill every empty cell with numbers $1, \ldots, n$, ensuring each number appears uniquely in every row, column, and box. Sudokus are presented partially completed, with the goal of finding the unique final solution.

We consider in this benchmark four classes of Sudoku puzzles each consisting of 100 board images: $n = 4$, $n = 9$, and two $n = 16$ variants; one using a hexadecimal convention $(1, \ldots, 9, A, \ldots, G)$, and the other, double digit numerical values $(1, \ldots, 16)$. Board images are generated for these puzzles with two further variants; computer-generated characters and handwritten characters, totaling 800 images in the dataset. Visual examples for $n = 16$ are presented in Figure 1.

3.2 KenKen

KenKen puzzles are an interesting visual AI challenge due to their sparse board image and reliance on mathematical operations. KenKen puzzles were invented by Tetsuya Miyamoto, to help improve students' problem-solving skills and logical reasoning, and have since featured in several newspapers and puzzle books. KenKens are grid puzzles similar to Sudoku, where for an $n \times n$ grid each row and column must contain the numbers $1, \ldots, n$. Instead of boxes, KenKen's have heavily outlined "cages" containing a collection of empty cells, with a target value and a mathematical operation $(+, -, \times, \div)$ in the top left corner of each cage.



(a) Computer-generated numeric. (b) Handwritten hexadecimal.

Fig. 1: Example Sudoku variants for $n = 16$.

All numbers contained by a cage must use the designated operation to achieve the target value, *e.g.*, if a cage contains two cells with target number 3 and the operation +, then only the digit pair 1 and 2 could appear in this cage, as only $1 + 2 = 3$. A singleton cage contains strictly one cell and has no operation, the target value is therefore the cell value. Figure 2 provides an example KenKen with its solution. While several other works in the literature have considered Sudoku solving, our work is distinctive in considering KenKen.

We consider in this benchmark seven classes of KenKen puzzles, $n = 3, \ldots, 9$. Each class consists of 100 board images, with a total of 1400 board images when considering variants for both computer-generated and handwritten characters.

3.3 Dataset Generation

Creating KenKen puzzles of different sizes requires two different stages: (a) generating a solvable symbolic representation of a puzzle, and (b) crafting the image of the board.

All dataset board images are in PNG grayscale format. Computer-generated characters were extracted from TMNIST [28] with the Noto Sans Regular font. Handwritten characters were extracted from MNIST [23] for digits and EMNIST [7] for letters A-G. To increase the benchmark difficulty, no restrictions on lower and upper case letters are imposed. KenKen operation characters were extracted from [39]. Characters are resized as appropriate for differing cell sizes and puzzle classes.

Sudoku. A symbolic representation of a Sudoku puzzle is in JSON format with puzzles grouped by grid size (e.g., ‘‘`4`’’ for $n = 4$). Each puzzle contains ‘‘`puzzle`’’ and ‘‘`solution`’’. ‘‘`puzzle`’’ is a 2D array representing

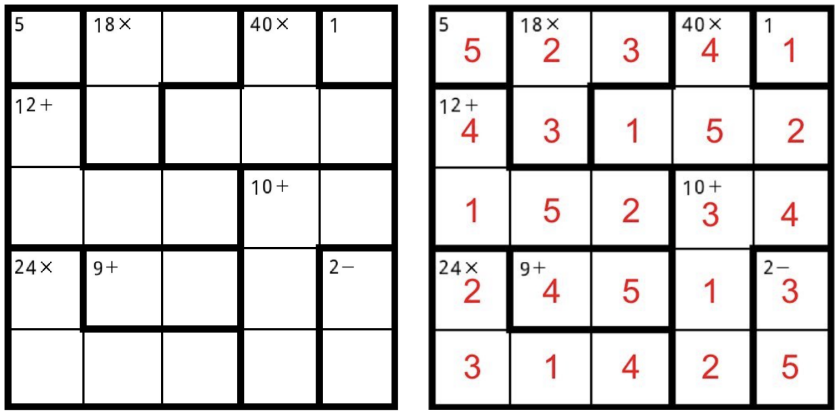


Fig. 2: A KenKen board image ($n = 5$) and its solution.

the grid where each inner array is a row. Non-zero integers ($1, \ldots, n$ for an $n \times n$ grid) represent given/pre-filled cells, and 0 represents empty cells to be solved. ‘‘solution’’ is a 2D array of the same dimensions containing the complete solved grid. Sudokus are generated with 100×100 pixel cells, with the exception of class $n = 4$. Therefore the $n = 9$ board images are 900×900 pixels, all $n = 16$ board images are 1600×1600 pixels. The $n = 4$ board images are also 900×900 pixels, meaning 225×225 pixels per cell. Characters are placed centrally in each cell, with double-digit characters appearing slightly smaller than single characters.

KenKen. A symbolic representation of a KenKen puzzle is in JSON format with puzzles grouped by grid size (e.g., ‘‘3’’ for $n = 3$) containing a list of cages, where each cage has ‘‘cells’’, ‘‘op’’, and ‘‘target’’. ‘‘cells’’ is a list of tuples with the grid coordinates of the cells contained in the cage. ‘‘op’’ is a string representing the operation of the cage, either ‘‘add’’, ‘‘sub’’, ‘‘mul’’, ‘‘div’’; or ‘‘’’ for singleton cells. ‘‘target’’ is an integer representing the target value for the cage. KenKens are generated with 300×300 pixel cells. These board image cells are larger than in Sudoku to give enough pixel space to fit the small target value and operation in the top corner of each cage. KenKen images therefore span between 900×900 pixels and 2700×2700 pixels. Cages are randomly generated for each puzzle, with the following restrictions: (a) maximum two singleton cages per puzzle, (b) maximum cage size is four cells for $n \geq 5$ and three cells for $n \leq 4$, and (c) $\div$ and $-$ can only occur in cages of size two, as these are non-associative operations.

Correctness. The solution correctness and uniqueness for all symbolic representations were verified using the Z3 SMT solver [11] before the board images were generated. As the grid size increases, it becomes exponentially more challenging to design cages that maintain unique solutions.

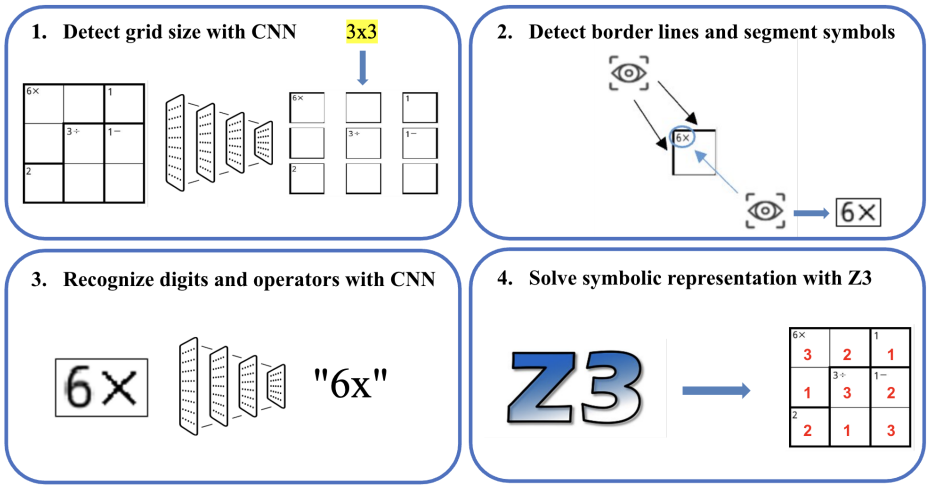


Fig. 3: A breakdown of the four main stages of the neuro-symbolic framework for KenKen puzzles.

4 Experimental Setup

We present in this section the neuro-symbolic framework that will be benchmarked against modern VLMs. Although Sudokus and KenKens are similar, the pipeline required for KenKen puzzles is significantly more challenging due to variable cage structures and mathematical operations. Therefore, this section primarily considers KenKen classes, with the pipeline given in Figure 3.

Firstly, an input image is selected from the benchmark dataset and passed to the neuro-symbolic solver. The input image is received by a Convolutional Neural Network (CNN) used to detect the KenKen grid size. Secondly, computer vision techniques are required to decompose the grid into its elements; cage detection and character extraction. Once the characters are extracted, another CNN is used for character recognition. With the cages defined and the characters recognized, a symbolic representation of the KenKen can be formed and passed to a symbolic solver, *e.g.* the Z3 SMT solver. The symbolic solver then attempts to find a solution, returning it if one is found, else triggering error correction.

4.1 Detect Grid Size

A CNN is trained using grids of varying sizes ($n = 3, \ldots, 9$) with data augmentation techniques to determine the grid size of the image. The CNN had two convolutional layers with ReLU activations, one max pooling layer, one dropout layer (dropout = 0.25), and two linear layers with ReLU activations, trained on 128x128 images for 20 epochs. The CNN consists of 33.6 million parameters and achieved 100% training accuracy. We can ignore overfitting concerns, or even encourage overfitting, as we want high accuracy on a specific known task.

4.2 Detect Borders and Segment Symbols

After determining the size of the grid, Canny edge detection [6] combined with the Hough Line transform [18] is used to detect the grid lines and border edges of cages. Cage detection is calculated via a flood-filling algorithm for cage grouping using breadth first search (BFS). Next, OpenCV is used for image segmentation to extract the individual symbols from the cells in the grid. For KenKens the height factor is used to determine how many characters appear in cells, while the $n = 16$ Sudoku class with double digits uses the ink density ratio of a cell to determine if a cell contains two digits, single digits have higher ink density in the cell center (separating ratio = 1.7). Detected symbols are then normalized to a target size of 28×28 in preparation for character recognition.

4.3 Character Recognition

Computer-generated characters. An additional CNN is trained on a combined dataset of characters. For computer-generated characters training is done with the dataset TMNist and the addition, subtraction, division, and multiplication operators from a handwritten math dataset [39]. This model had the same structure as the grid size detection CNN but is scaled down and trained on 28x28 images. The CNN has $400,000$ parameters. For KenKens the output classifications are $\{1, \ldots, n, +, -, \times, \div\}$. For Sudokus the output classifications are set of alphanumeric characters found in the puzzle, dependent on class. For computer-generated characters, the training accuracy was 100%.

Handwritten characters. The same CNN can be trained using the MNIST and EMNIST datasets. Using a 90-10 training-test split, a CNN training accuracy of 97.8% is acquired. Designing the board images to have handwritten characters unseen by the character recognition CNN can then provide a baseline benchmark of performance. This CNN architecture is named "NSV1" (Neuro-Symbolic Version 1).

As KenKens and Sudokus are incredibly sensitive to even a single image decomposition error, we benchmark a second more complex CNN architecture "NSV2" with the purpose of improving classification accuracy to 99.8%. This CNN has four convolutional layers across two blocks, with BatchNorm after each convolutional layer for stable training. The linear fully-connected layer has twice as many hidden nodes, contributing to the CNN having $850,000$ parameters. Dropout is set at 0.4 compared to 0.25 for NSV1. Data augmentation is considered in the training of NSV2, with elastic deformation, rotation $\pm 20°$, translation ± 4 pixels, Gaussian blur and scale variations. Using confusion-aware pair augmentation, common mistakes can be reduced, such as $3 \leftrightarrow 8$ and $6 \leftrightarrow 8$. In NSV2, the confusion $1 \leftrightarrow 7$ is completely eliminated, while it caused 37% of errors for NSV1.

4.4 Solving Symbolic Representation

After detecting the cages and recognizing the symbols, the model generates a symbolic representation of the puzzle.

The symbolic representation is solved with Z3 [11], a state-of-the-art SMT solver that can efficiently search the solution space of a Boolean logic constraint satisfaction problem and return a concrete model if one exists. The unknown variables for Z3 to determine are stored in a 2D list representing the solution grid. With these constraints, a Z3 Solver can generate a concrete model or determine the constraints unsatisfiable. For unsatisfiable puzzles, the function `unsat_core()` can be used to determine which constraints were violated, highlighting which characters are likely to have been classified wrong. Since the solution space scales exponentially with increased grid size, so does the average time it takes Z3 to determine a solution for a valid puzzle. Some checks can be computed without needing to wait for Z3 to attempt solving the full puzzle, *e.g.* do all cages have a target and operation? For double-digit cells, is the first digit 1?

4.5 Self-Correction

With these constraints, a Z3 Solver can generate a concrete model or determine the constraints unsatisfiable. When unsatisfiable, the neuro-symbolic framework can provide reasons for failure or provide direction to classifications that had low confidence. This ability to explain enables the neuro-symbolic framework to check its own work and self-correct errors. It can also give direction for future development and program updates to the user. We consider the following correction approaches when benchmarking the neuro-symbolic framework.

Constraint-based corrections. Z3 returns `unsat` when symbolic representations are unsolvable. We use Z3's `unsat_core()` to identify conflicting constraints in Sudoku puzzles, then rank the cells or cages that seem most suspicious. This approach can significantly reduce the search space to find the errors, but is less effective on larger grid sizes. We consider a more advanced multi-phase constraint-based correction method for KenKens. Due to the KenKen complexity, the approach analyses the symbolic representation and detects further causes of errors, *e.g.* impossible cages (mathematically invalid) or invalid targets/operations for the cage size. These approaches can filter out invalid candidate characters and reduce the search space for candidates between $10 - 100\times$.

Confidence-based corrections. A simple approach that generates alternate interpretations of the symbolic representation. Run alone, the approach swaps classified characters with low confidence, replacing them with the next most likely (or k-most likely) value instead. This approach can systematically create new interpretations assuming different numbers of values have been misclassified. Choosing alternatives for characters difficult to classify can be a simple but

effective approach. The approach has independent value from constraint-based correction, as "silent errors" could create an incorrect but solvable symbolic representation — in such cases, checking against a ground truth oracle is necessary to determine the solution was incorrect.

Hybrid correction. These approaches can be joined or run sequentially, with constraint-based correction reducing the search space and confidence-based corrections helping order the new symbolic realizations to attempt.

4.6 Vision Language Models (VLMs)

After considering various open and closed source VLMs, the following models were chosen for formal benchmarking (the state-of-the-art when manuscript was submitted): GPT-4o and GPT-4o-mini [20], Claude Sonnet 4.0 [2], Gemini-2.5-Pro [9], and Qwen2.5-VL-7B-Instruct [3]. Initial testing of other VLMs showed poor model efficiency and response even for $n = 3$ puzzles. These models were chosen for their novelty and performance on complex reasoning tasks. Each model is tested on the computer-generated dataset. We assume if a VLM solves no puzzle in a class n, that it will also solve no puzzles in classes $m > n$ (as performance appears monotonic experimentally). Each VLM is given the same prompt, explaining the puzzle's rules, providing an example, and demonstrating the needed output format. For example, the Sudoku prompt is:

Sudoku Prompt. You will be provided a Sudoku puzzle board. Like standard Sudoku, every row, column, and box must contain unique numbers. For a 4×4 puzzle, use numbers $1-4$ with 2×2 boxes. For a 9×9 puzzle, use numbers $1-9$ with 3×3 boxes.

The puzzle shows some pre-filled numbers. Your task is to complete the puzzle by filling in the empty cells.

Format your response as a 2 dimensional list representing the complete solution. An example response for a 4×4 Sudoku is:

$$[[1, 2, 3, 4], [3, 4, 1, 2], [2, 3, 4, 1], [4, 1, 2, 3]]$$

Distinct prompts are provided for KenKens, Sudoku ($n = 4$ and $n = 9$) and each of the variant Sudokus ($n = 16$). The responses of the VLMs were evaluated on their ability to present a correctly formatted valid solution to the puzzle, valid solutions with incorrect reasoning are still considered correct.

5 Results

We show in this section that the neuro-symbolic framework proposed in this work is magnitudes more efficient and accurate than any of the VLMs. We also demonstrate the ability of our neuro-symbolic framework to self-correct errors using logical and probabilistic reasoning as described in Section 4.5.

	KenKen						
VLM	3×3	4×4	5×5	6×6	7×7	8×8	9×9
NSV1	100%	100%	100%	100%	100%	100%	100%
Gemini 2.5 Pro	69%	35%	0%	—	—	—	—
Claude Sonnet 4	41%	6%	0%	—	—	—	—
Qwen 2.5-VL	24%	0%	—	—	—	—	—
GPT-4o	21%	0%	—	—	—	—	—
GPT-4o Mini	5%	0%	—	—	—	—	—

	Sudoku			
VLM	4×4	9×9	16×16 (Hex)	16×16 (Numeric)
NSV1	100%	100%	100%	100%
Gemini 2.5 Pro	99%	0%	—	—
Claude Sonnet 4	73%	0%	—	—
Qwen 2.5-VL	51%	0%	—	—
GPT-4o	75%	8%	0%	0%
GPT-4o Mini	65%	1%	0%	0%

Table 1: VLM comparison results for KenKen and Sudoku benchmarks with computer-generated characters. We assume performance is monotonic, so if a VLM solves 0% for a benchmark class we stop the evaluation for further classes and use — for later entries.

5.1 Computer-Generated Characters

Accuracy. We evaluate the performance of our neuro-symbolic framework by comparing the accuracy of NSV1 against the VLMs: GPT-4o and GPT-4o-mini, Claude Sonnet 4.0, Gemini-2.5-Pro, and Qwen2.5-VL-7B-Instruct. We assume that if any model fails to solve a puzzle for a specific class that it will fail on all classes more complex as well, and so it drops out from participating in further comparisons, *e.g.* GPT-4o mini solved 5% of $n = 3$ KenKens but 0% of $n = 4$, we therefore assume it would score 0% on all classes $n \geq 4$. This assumption is appropriate as there is a clear monotonic behavior exhibited by the VLMs as the grid size increases.

Table 1 demonstrates the performance of NSV1 against the VLMs. The VLMs have significant difficulties solving these step-by-step reasoning problems, with Gemini-2.5-Pro performing best at KenKen and GPT-4o performing best at Sudoku. Conversely, NSV1 has a perfect score for all classes. This is expected as NSV1 also got a 100% training validation score, meaning the CNN perfectly recognizes all computer-generated characters, and so there is no error when creating the symbolic representation of the image.

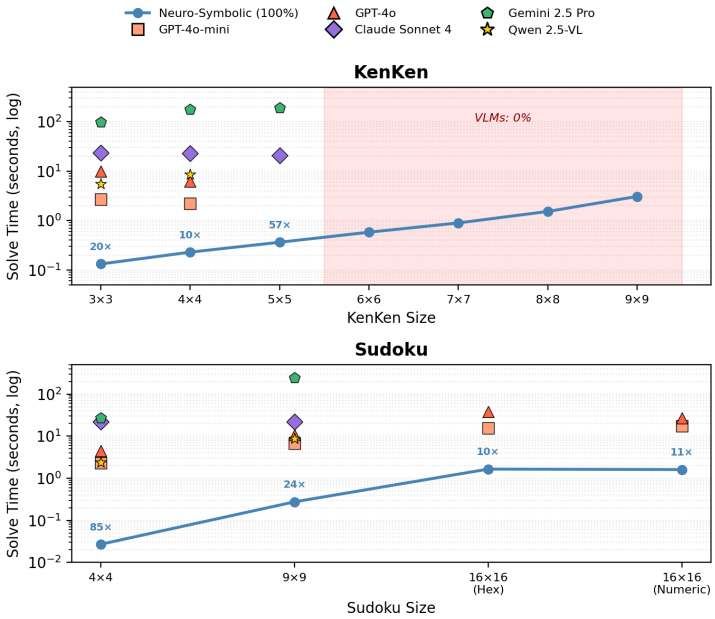


Fig. 4: Average solve time for NSV1 and the VLMs.

Efficiency. Figure 4 compares the average solve time, the total time from receiving the board image (and prompt) to providing an answer, for the VLMs and for NSV1. In addition to having a perfect solve rate for computer-generated characters, NSV1 is at least 10× faster at finding a solution than the fastest VLMs. The efficiency improvement is significant, even NSV1 solving KenKens for $n = 9$ provides faster solve rates than Gemini and Claude solving KenKens for $n = 3$. Considering the solve accuracy for the VLMs, it is noticeable that the most accurate VLM for KenKens are also the slowest.

5.2 API vs Online VLMs

It was posited that there may be some difference in running the results via an API (as we have done) compared with collecting the results by manually uploading each image with the prompt to the web interfaces of the various VLMs, which show clearer agentic behavior. There is also the challenge of keeping up with every new VLM release. Due to the manual effort this would have involved we did not attempt to thoroughly evaluate if such a difference exists. However, through some ad hoc testing of the latest VLMs there was no clear reason to doubt the current results. For example, we manually tested several KenKen puzzles on Gemini 3 Flash, a newer improved VLM, but did not see notable improvement in the results. In fact, it was a surprise to see just how many errors were made, including errors for singleton cells, see Figure 5. While the VLM appeared to

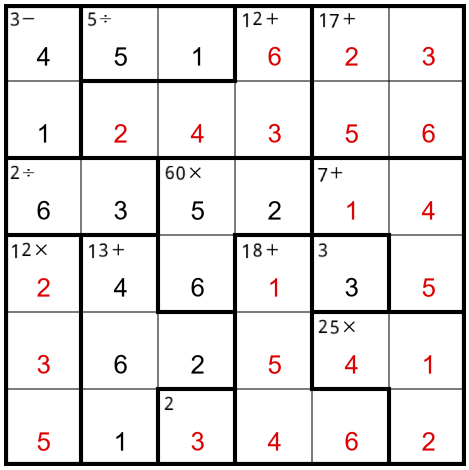


Fig. 5: A 6x6 KenKen puzzle (benchmark image: `board6_0.png`) evaluated using Gemini 3 Flash on the web interface, incorrect cages shown with red digits. Gemini provided a google sheet with a result, which was created after two passes from its own validation that the puzzle is correctly solved. However, there are only 6 of 13 correct boxes. Notably, the singleton box 2 is *incorrect* with a 3 placed inside.

correctly identify the cages in the text output, the subsequent "solved" KenKen had multiple mistakes, a clear reasoning issue.

5.3 Handwritten Characters

Due to the poor performance of the VLMs on the computer-generated character board images, and expecting handwritten-character board images to be more challenging, we consider only two of the VLMs in this section: GPT-4o (most accurate for Sudoku) and Gemini-2.5-Pro (most accurate for KenKen). In this section, we also compare the difference in performance between NSV1 and NSV2, with NSV1 being a simpler CNN model than NSV2. Of note, NSV1 was not trained on any handwritten characters that appear in the board images of the benchmark suite. As a reminder, NSV1 has a training accuracy of 97.8% and NSV2 an accuracy of 99.8%. We are not concerned with overfitting issues as we consider a specific problem setting.

Accuracy. Both Gemini-2.5-Pro and GPT-4o showed surprising resilience to changing from computer-generated characters to handwritten characters. KenKen performances were similar: both GPT-4o and Gemini-2.5-Pro performed 1% worse for class $n = 3$ and Gemini-2.5-Pro was 5% worse for class $n = 4$. For $n = 4$ Sudokus, GPT-4o achieved 63% accuracy (12% worse) but Gemini-2.5-Pro improved to 100% accuracy (1% more). For $n = 9$ Sudokus, GPT-4o solved

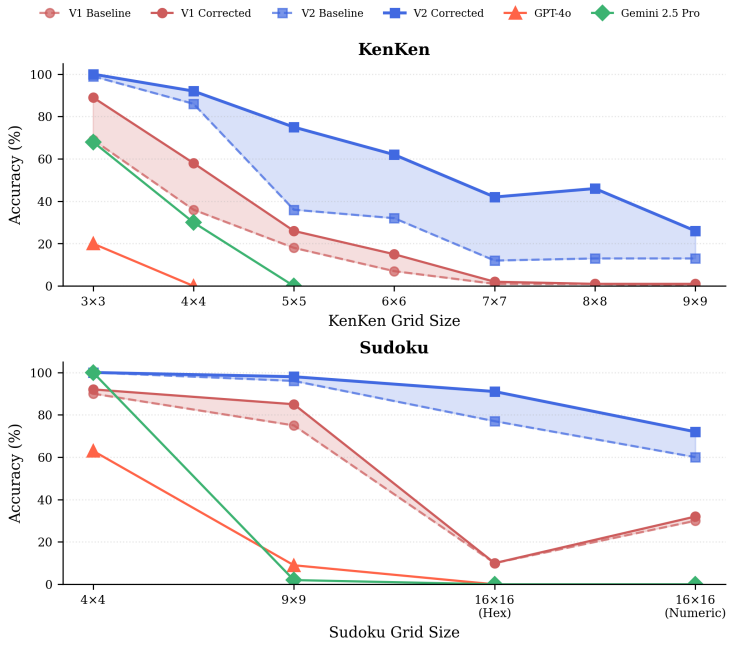


Fig. 6: Benchmarking NSV1, NSV2, Gemini-2.5-Pro and GPT-4o on board images with handwritten characters.

9% (1% more) and Gemini-2.5-Pro 2% (previously 0%) — although the success is low, both scores improved performance compared with computer-generated characters. These benchmark results demonstrate that VLMs exhibit strong Optical Character Recognition (OCR) for identifying digits. Both NSV1 and NSV2 show imperfect performance on handwritten characters, but still outperform the VLMs with their baseline solve (no self-correction) in all cases. The single exception is Sudoku $n = 4$ and Gemini-2.5-Pro which matches the performance of NSV2. The performance results for handwritten benchmarks can be seen in Figure 6.

Although NSV1 performs well on smaller board images, it struggles with the larger ones, showing the value of using complex CNNs, as used by NSV2, with higher classification accuracy demonstrating noticeably improved performance. This is highlighted most clearly in the comparison of NSV1 and NSV2 for both $n = 16$ Sudokus; 16×16 boards have significant complexity with an average of 120 pre-filled cells. Based on training performance, we should expect every puzzle to have at least one misclassified cell when using NSV1, and at least 10% of puzzles when using NSV2. NSV1 has clear struggles when the characters A-G are included in the board images, with a higher solve rate for board images requiring the classification of two digits in one cell. This is the opposite for NSV2

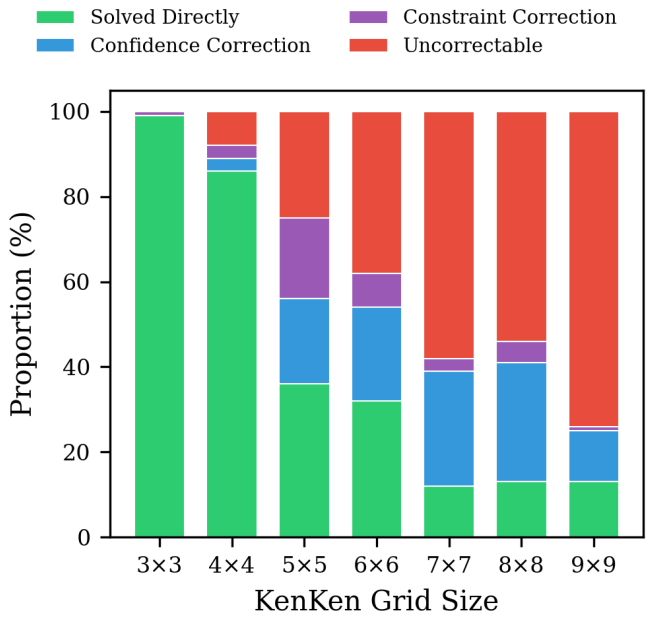


Fig. 7: Self-correction performance of NSV2 on KenKen puzzles.

with its high training accuracy, it performs worse when the board images require the additional step of detecting if a cell has one or two digits.

Self-Correction. From Figure 6, self-correction shows significant improvement over the baseline for both NSV1 and NSV2. This exemplifies the benefit of the neuro-symbolic framework as an XAI technique, understanding likely causes of failure and correcting them. We run the self-correction flow sequentially, with the constraint-based correction used first, followed by the confidence-based correction. Constraint-based methods demonstrate explainability by generating a search space of suspects that violate constraints, confidence-based methods demonstrate explainability by highlighting the model's doubts. Figure 7 shows a breakdown of self-correction method success for NSV2 on KenKens. For small grids the constraint-based approach contributes more self-corrections, while larger board images benefit from confidence-based corrections. Intuitively, as the grid size grows, there is an increased chance of (multiple) errors or "silent errors"; erroneous symbolic representations not returning `unsat`. For KenKens, the NSV2 confidence-based method corrected 112 silent errors. The self-correction results were computed assuming maximum 3 errors and checking top-4 classifications in constraint-based correction, increasing to maximum 4 errors and top-6 predictions for the subsequent confidence-based correction. Only one puzzle was corrected that had 3 errors, with the remaining corrections

having either 1 or 2 errors only. The root cause of uncorrectable puzzles is multiple misclassified digits and larger board sizes, when self-correcting the set of incorrect but solvable grids increases significantly and finding the correct grid could require exhaustive search.

6 Related Work

Benchmarking VLMs is an active field of research. In [19], the authors benchmark VLMs on their ability to conduct compositional reasoning. For a given image, GPT-4V and other VLMs generate a description of said image, GPT-4V is then used to generate questions about the image to test the other VLMs. These VLMs generate answers to the questions and those answers are evaluated by the authors. This process created tougher reasoning questions for the images and highlighted more flaws in VLM 'reasoning' than existing works. The MMMU benchmark [46] evaluates VLMs on academic exams, covering exam questions from engineering, science, art & design, business, humanities and social sciences, and medicine. The benchmark covers heterogeneous images, interleaved text and images, and perception routed in subject knowledge, with VLMs not surpassing a score of more than 60%.

Several benchmarks test VLMs on their use in unique application domains. Geobench-VLM [10] measures VLMs on their geospatial in event detection, scene and temporal understanding, segmentation, object classification, localization, counting and image captioning. AgroBench [36] applied to crops, assessing diseases, weeds, and pests. Images labeled by expert agronomists are assessed across 203 crop categories and 682 diseases.

Neuro-symbolic benchmarks include resbench [5], which observed that for tasks needing learning and reasoning that shortcuts were taken (not associating the correct concept to the data). They show that obtaining high quality concepts from neural models and neuro-symbolic models is still an open problem. The benchmark includes logical tasks and high-stakes traffic scenarios. In a similar vein KANDY [27] is benchmarks both neural and symbolic approaches highlighting the need for a unified neuro-symbolic framework. The benchmark consists of Kandinsky patterns, binary true false puzzles using simple geometric shapes but complex rules. The benchmark relies on identification of both spatial and Gestalt concepts. They also assess Bongard problems, from cognitive psychology, where two groups are given - one satisfying and one failing a rule. The benchmark requires the correct identification of the rule. They also consider Raven matrices and other logical puzzles.

Similar work to ours include the work [24]. They develop a visual neuro-symbolic Sudoku solver using recurrent neural networks (RNNs) for image decomposition and Z3 with relaxed logic constraints for symbolic reasoning. The work considers only 9×9 Sudokus, while we include two variants of more complex 16×16 Sudokus as well as seven classes of KenKen (both computer-generated and handwritten characters). In addition, we demonstrate how a neuro-symbolic VLM can reason to determine where it has made errors and to correct them.

The work NeurASP [44] extends Answer Set Programming (ASP) to include pre-trained neural networks in symbolic computations. They apply their approach to tasks that need both perception and reasoning, including relations between perceived objects in images and Sudoku puzzles from images. They include some simple additional constraints including 'anti-knight moves', 'sudoku-X' and 'Offset Sudoku'. A novelty of our work is the benchmarking of KenKen puzzles, a more challenging logical puzzle than considered in their work. We also achieved 100% accuracy for computer-generated digits, while NeurASP had a maximum of 66.5% (with a caveat of a different training approach).

In SATNet [42], the authors incorporate deep learning and logical reasoning from a SAT solver to learn how to solve 9×9 Sudoku puzzles solely from examples. Similarly in [32] a benchmark of over 200,000 computer-digit Sudokus are used to train the Sudoku solvers. Both papers have over 90% success when tested on Sudokus not included in the training. For SATNet this drops to 63% for 'visual' Sudokus including MNIST images. Overall, our approach has more success in solving Sudoku puzzles with 100% accuracy on 9×9 Sudokus and almost the same for handwritten digits. Neither [42] or [32] consider KenKen puzzles.

KenKen puzzles are considered alongside Sudoku puzzles in [4] as part of set selection dynamical system neural networks. The approach uses partial memory to capture the state of some neurons in the network. The work is assessed on 15 Sudoku puzzles, 60 6×6 KenKen puzzles, and a 4×4 KenKen with all boxes having to 12+. Although solving 100% of cases via neural network, the number of experiments is light compared with our 2200 board images, we consider handwritten characters as well as computer-generated ones, and we consider more complex puzzles (16×16 Sudokus and 9×9 KenKens).

7 Conclusion and Future Work

This work presented a benchmark suite of 2200 board images of Sudoku and KenKen puzzles, both for computer-generated and handwritten characters. This benchmark suite provides an interesting challenge for Vision-Language Models (VLMs), as the suite requires both accurate image decomposition and deductive reasoning to get solutions. Both Sudoku and KenKen are NP-complete in the general case, meaning a single error can render the problems unsolvable. We also present a neuro-symbolic framework that significantly outperformed the VLMs, and demonstrate how it can leverage explainability. By highlighting when constraints are violated or when characters are classified with low confidence, the neuro-symbolic framework can reassess and self-correct image decomposition errors, increasing the number of solvable puzzles. Future work will consider safety-critical applications, including uncertainty quantification of foundation models in robotics.

Acknowledgments. B. Wooding and K. Brennan are co-first authors.

The material presented in this paper is based upon work supported by the National Science Foundation (NSF) through grant number 2220401, the Defense Advanced Research Projects Agency (DARPA) under contract number FA8750-23-C-0518, and the U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computing Research, under Award Number DE-SC0025528. Any opinions, findings, and conclusions or recommendations expressed in this paper are those of the authors and do not necessarily reflect the views of DARPA or NSF.

References

1. Acharya, K., Sharifi, I., Lad, M., Sun, L., Song, H.: Integrating Neurosymbolic AI in Advanced Air Mobility: A Comprehensive Survey. In: Kwok, J. (ed.) Proceedings of the Thirty-Fourth International Joint Conference on Artificial Intelligence, IJCAI-25. pp. 10362–10370. International Joint Conferences on Artificial Intelligence Organization (8 2025). https://doi.org/10.24963/ijcai.2025/1151, https://doi.org/10.24963/ijcai.2025/1151, survey Track
2. Anthropic: Claude 4 Sonnet (2025), https://www.anthropic.com/news/claude-4
3. Bai, S., Chen, K., Liu, X., Wang, J., Ge, W., Song, S., Dang, K., Wang, P., Wang, S., Tang, J., et al.: Qwen2.5-VL Technical Report. arXiv preprint arXiv:2502.13923 (2025)
4. Boreland, B., Clement, G., Kunze, H.: Set selection dynamical system neural networks with partial memories, with applications to sudoku and kenken puzzles. Neural Networks **68**, 46–51 (2015)
5. Bortolotti, S., Marconato, E., Carraro, T., Morettin, P., van Krieken, E., Vergari, A., Teso, S., Passerini, A.: A neuro-symbolic benchmark suite for concept quality and reasoning shortcuts. Advances in neural information processing systems **37**, 115861–115905 (2024)
6. Canny, J.: A computational approach to edge detection. IEEE Transactions on pattern analysis and machine intelligence (6), 679–698 (1986)
7. Cohen, G., Afshar, S., Tapson, J., Van Schaik, A.: EMNIST: Extending MNIST to handwritten letters. In: 2017 international joint conference on neural networks (IJCNN). pp. 2921–2926. IEEE (2017)
8. Colbourn, C.J.: The complexity of completing partial latin squares. Discrete Applied Mathematics **8**(1), 25–30 (1984)
9. Comanici, G., Bieber, E., Schaekermann, M., Pasupat, I., Sachdeva, N., Dhillon, I., Blistein, M., Ram, O., Zhang, D., Rosen, E., et al.: Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. arXiv preprint arXiv:2507.06261 (2025)
10. Danish, M., Munir, M.A., Shah, S.R.A., Kuckreja, K., Khan, F.S., Fraccaro, P., Lacoste, A., Khan, S.: Geobench-vlm: Benchmarking vision-language models for geospatial tasks. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 7132–7142 (2025)
11. De Moura, L., Bjørner, N.: Z3: An efficient SMT solver. In: International conference on Tools and Algorithms for the Construction and Analysis of Systems. pp. 337–340. Springer (2008)
12. Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., Houlsby, N.: An image is worth 16x16 words: Transformers for image recognition at scale. In: International Conference on Learning Representations (2021), https://openreview.net/forum?id=YicbFdNTTy

13. Dwivedi, R., Dave, D., Naik, H., Singhal, S., Omer, R., Patel, P., Qian, B., Wen, Z., Shah, T., Morgan, G., et al.: Explainable AI (XAI): Core ideas, techniques, and solutions. ACM computing surveys **55**(9), 1–33 (2023)
14. Ellis, K.: Human-like few-shot learning via bayesian reasoning over natural language. In: Oh, A., Naumann, T., Globerson, A., Saenko, K., Hardt, M., Levine, S. (eds.) Advances in Neural Information Processing Systems. vol. 36, pp. 13149–13178. Curran Associates, Inc. (2023), https://proceedings.neurips.cc/paper_files/paper/2023/file/2aa9b18b9ab37b0ab1fdaae46fb781d4-Paper-Conference.pdf
15. Gao, L., Madaan, A., Zhou, S., Alon, U., Liu, P., Yang, Y., Callan, J., Neubig, G.: Pal: Program-aided language models. In: International Conference on Machine Learning. pp. 10764–10799. PMLR (2023)
16. Garcez, A.d., Lamb, L.C.: Neurosymbolic AI: The 3rd wave. Artificial Intelligence Review **56**(11), 12387–12406 (2023)
17. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 770–778 (2016)
18. Hough, P.V.: Method and means for recognizing complex patterns (Dec 18 1962), uS Patent 3,069,654
19. Huang, I., Lin, W., Mirza, M.J., Hansen, J., Doveh, S., Butoi, V., Herzig, R., Arbelle, A., Kuehne, H., Darrell, T., et al.: Conme: Rethinking evaluation of compositional reasoning for modern vlms. Advances in Neural Information Processing Systems **37**, 22927–22946 (2024)
20. Hurst, A., Lerer, A., Goucher, A.P., Perelman, A., Ramesh, A., Clark, A., Ostrow, A., Welihinda, A., Hayes, A., Radford, A., et al.: Gpt-4o system card (2024), https://arxiv.org/abs/2410.21276
21. Joshi, G., Walambe, R., Kotecha, K.: A review on explainability in multimodal deep neural nets. IEEE Access **9**, 59800–59821 (2021)
22. Kahneman, D.: Thinking, Fast and Slow. Macmillan (2011)
23. LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. Proceedings of the IEEE **86**(11), 2278–2324 (2002)
24. Li, Z., Huang, Y., Li, Z., Yao, Y., Xu, J., Chen, T., Ma, X., Lu, J.: Neuro-symbolic learning yielding logical constraints. Advances in Neural Information Processing Systems **36**, 21635–21657 (2023)
25. Li, Z., Wu, X., Du, H., Liu, F., Nghiem, H., Shi, G.: A survey of state of the art large vision language models: Benchmark evaluations and challenges. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops. pp. 1587–1606 (June 2025)
26. Liu, H., Li, C., Wu, Q., Lee, Y.J.: Visual instruction tuning. In: Oh, A., Naumann, T., Globerson, A., Saenko, K., Hardt, M., Levine, S. (eds.) Advances in Neural Information Processing Systems. vol. 36, pp. 34892–34916. Curran Associates, Inc. (2023), https://proceedings.neurips.cc/paper_files/paper/2023/file/6dcf277ea32ce3288914faf369fe6de0-Paper-Conference.pdf
27. Lorello, L.S., Lippi, M., Melacci, S.: The KANDY benchmark: Incremental neuro-symbolic learning and reasoning with Kandinsky patterns. Machine Learning **114**(7), 161 (2025)
28. Magre, N., Brown, N.: Typography-MNIST (TMNIST): an MNIST-Style Image Dataset to Categorize Glyphs and Font-Styles (2022)
29. Manginas, N., Paliouras, G., De Raedt, L.: NeSyA: Neurosymbolic Automata. In: Kwok, J. (ed.) Proceedings of the Thirty-Fourth International Joint Conference on

Artificial Intelligence, IJCAI-25. pp. 5950–5958. International Joint Conferences on Artificial Intelligence Organization (8 2025). https://doi.org/10.24963/ijcai.2025/662, https://doi.org/10.24963/ijcai.2025/662, main Track
30. Mao, J., Gan, C., Kohli, P., Tenenbaum, J.B., Wu, J.: The Neuro-Symbolic Concept Learner: Interpreting Scenes, Words, and Sentences From Natural Supervision. In: International Conference on Learning Representations (2019)
31. Murphy, K.P.: Machine learning : a probabilistic perspective. MIT Press (2012)
32. Palm, R., Paquet, U., Winther, O.: Recurrent relational networks. Advances in neural information processing systems **31** (2018)
33. Radford, A., Kim, J.W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., et al.: Learning transferable visual models from natural language supervision. In: International conference on machine learning. pp. 8748–8763. PmLR (2021)
34. Sasaki, S., Lopez, D.M., Johnson, T.T.: Neurosymbolic Finite and Pushdown Automata: Improved Multimodal Reasoning versus Vision Language Models (VLMs). Proceedings of Machine Learning Research vol vvv **1**, 18 (2025)
35. Sheth, A., Khandelwal, V., Roy, K., Pallagani, V., Chakraborty, M.: Neurosymbolic knowledge-grounded planning and reasoning in artificial intelligence systems. IEEE Intelligent Systems **40**(2), 27–34 (2025)
36. Shinoda, R., Inoue, N., Kataoka, H., Onishi, M., Ushiku, Y.: Agrobench: Vision-language model benchmark in agriculture. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 7634–7644 (2025)
37. Shyalika, C., Prasad, R., El Kalach, F., Venkataramanan, R., Zand, R., Harik, R., Sheth, A.: NSF-MAP: Neurosymbolic Multimodal Fusion for Robust and Interpretable Anomaly Prediction in Assembly Pipelines. In: Kwok, J. (ed.) Proceedings of the Thirty-Fourth International Joint Conference on Artificial Intelligence, IJCAI-25. pp. 9330–9338. International Joint Conferences on Artificial Intelligence Organization (8 2025). https://doi.org/10.24963/ijcai.2025/1037, https://doi.org/10.24963/ijcai.2025/1037
38. Team, Q.: Qwen2 technical report (2024), https://arxiv.org/abs/2407.10671
39. Thapa, S.: Handwritten math symbols (2021), https://www.kaggle.com/datasets/sagyamthapa/handwritten-math-symbols/data
40. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. Advances in neural information processing systems **30** (2017)
41. Wang, P., Bai, S., Tan, S., Wang, S., Fan, Z., Bai, J., Chen, K., Liu, X., Wang, J., Ge, W., et al.: Qwen2-vl: Enhancing vision-language model's perception of the world at any resolution (2024)
42. Wang, P.W., Donti, P., Wilder, B., Kolter, Z.: Satnet: Bridging deep learning and logical reasoning using a differentiable satisfiability solver. In: International Conference on Machine Learning. pp. 6545–6554. PMLR (2019)
43. Wei, J., Wang, X., Schuurmans, D., Bosma, M., Xia, F., Chi, E., Le, Q.V., Zhou, D., et al.: Chain-of-thought prompting elicits reasoning in large language models. Advances in neural information processing systems **35**, 24824–24837 (2022)
44. Yang, Z., Ishay, A., Lee, J.: Neurasp: embracing neural networks into answer set programming. In: Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence. IJCAI'20 (2021)
45. Yato, T., Seta, T.: Complexity and completeness of finding another solution and its application to puzzles. IEICE transactions on fundamentals of electronics, communications and computer sciences **86**(5), 1052–1060 (2003)

46. Yue, X., Ni, Y., Zhang, K., Zheng, T., Liu, R., Zhang, G., Stevens, S., Jiang, D., Ren, W., Sun, Y., et al.: Mmmu: A massive multi-discipline multimodal understanding and reasoning benchmark for expert agi. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 9556–9567 (2024)
47. Zhou, C., YU, L., Babu, A., Tirumala, K., Yasunaga, M., Shamis, L., Kahn, J., Ma, X., Zettlemoyer, L., Levy, O.: Transfusion: Predict the next token and diffuse images with one multi-modal model. In: The Thirteenth International Conference on Learning Representations (2025), https://openreview.net/forum?id=SI2hI0frk6