Proceedings of Machine Learning Research 288:1–18, 2025 2nd International Conference on Neuro-symbolic Systems (NeuS)

Neurosymbolic Finite and Pushdown Automata: Improved Multimodal Reasoning versus Vision Language Models (VLMs)

Samuel Sasaki SAMUEL.SASAKI@VANDERBILT.EDU
Diego Manzanas Lopez DIEGO.MANZANAS.LOPEZ@VANDERBILT.EDU
Taylor T. Johnson TAYLOR.JOHNSON@VANDERBILT.EDU
Vanderbilt University, Nashville TN 37235, USA

Editors: G. Pappas, P. Ravikumar, S. A. Seshia

Abstract

Multimodal large language models (LLMs), such as vision language models (VLMs), are powerful reasoning tools that have been shown to be capable of solving non-trivial tasks such as image and video reasoning, translation, and text generation. Alternatively, LLMs have also regularly been shown to have difficulties with trivial tasks like performing elementary mathematical reasoning problems and arithmetic. Significant effort has since gone towards directly addressing these tasks in order to show reasoning in LLMs. Despite an extensive training regimen that includes state-of-the-art hardware support and copious amounts of data, it is still straightforward to modify a, now, relatively trivial task such that the LLM again experiences a great deal of difficulty in solving the task. In this work, we focus on two tasks, image-based string acceptance and image-based arithmetic evaluation, for VLMs to solve that involve non-trivial multimodal reasoning and introduce a new neurosymbolic-based model of computation that can significantly outperform VLMs on them. We define two classes of neurosymbolic automata to address this problem, namely neurosymbolic finite automata (NSFA) and neurosymbolic pushdown automata (NSPDA). These neurosymbolic automata are able to model the image-based string acceptance and arithmetic evaluation tasks well, given their derivation from finite and pushdown automata for string acceptance and arithmetic evaluation. We show that state-of-the-art LLMs with multimodal reasoning capabilities are not only outperformed by neurosymbolic automata, but often fail to reason about the tasks altogether with the VLMs getting zero correct in the arithmetic evaluation task while the NSPDA demonstrates 88% accuracy with 2 operands and a steady decline as the complexity of the expressions increased, as expected.

Keywords: automaton, multimodal, mathematical reasoning

1. Introduction

Recent works showcasing the reasoning capabilities of large language models (LLMs) have motivated a paradigm shift in deep learning towards multimodal reasoning Zhang et al. (2024); Wang et al. (2024); Mitra et al. (2024). This, in turn, has led to the development of numerous benchmarks, training techniques, and architectures for improving and evaluating multimodal models on complex tasks such as visual question answering Antol et al. (2015); Li et al. (2022), image captioning Zhou et al. (2020); Radford et al. (2021); Nguyen et al. (2023), image-text retrieval Cao et al. (2022), and text-to-image generation Ramesh et al. (2021); Tan et al. (2025). Despite showing remarkable performance on difficult tasks, LLMs have also been thought to be mimicking critical thinking, an argument that has been defended by numerous examples of LLMs hallucinating or failing at tasks that are trivial in comparison to the previously mentioned tasks where they have demonstrated remarkable success Bender et al. (2021); Gendron et al. (2024). Mathematical reasoning is one area

© 2025 S. Sasaki, D. Manzanas Lopez & T.T. Johnson.

in particular where there have been clear demonstrations of LLMs' inability to reason Yuan et al. (2023); Panas et al. (2024); Mirzadeh et al. (2024). However, in recent times, clever engineering and an emphasis on addressing the weaknesses in LLMs' reasoning capabilities, specifically mathematically, have seemingly made it impossible to trick industry-backed models–such as OpenAI's ChatGPT–with a simple arithmetic evaluation or mathematical reasoning problem OpenAI (a); Lightman et al. (2024); OpenAI (b). Indeed, early evaluation of LLMs highlighted arithmetic accuracy as functions of model size and digit length Brown et al. (2020), as part of the basis of claims for being few-shot learners, and subsequent approaches like chain-of-thought reasoning improved these Wei et al. (2022). While there has been some consideration of regular language acceptance and related tasks with various neural architectures Weiss et al. (2018); Bhattamishra et al. (2020); Merrill et al. (2021), this has not arisen as a significant evaluation task yet for LLMs or VLMs. However, we hypothesize that these limitations are still present for multimodal tasks.

In this work, we direct attention toward the reasoning capabilities of multimodal models, specifically vision language models (VLMs). We modify the regular language acceptance and arithmetic evaluation tasks to take as input images (or sequences thereof) instead of text or numerical values. In doing so, the tasks become multimodal, with the VLM being first responsible for correctly mapping from the image space (either as a sequence or a single image) to the space of strings defined over the alphabet (i.e., all combinations of individual valid characters) via classification, optical character recognition (OCR), or some type of visual reasoning on the image(s). Explicitly, these two tasks are image classification and basic reasoning in the form of pattern matching and simple arithmetic–two individual tasks that LLMs have been shown capable of solving with a considerable degree of success. However, we show that this small change to the input space drastically affects the performance of the models. In response to this, we propose neurosymbolic finite and pushdown automaton. The traditional analog to the conceptualized multimodal reasoning tasks can be trivially solved with finite and pushdown automata, respectively. We show that transferring the structure of finite and pushdown automaton to create a neurosymbolic framework for these tasks outperforms the VLMs by a significant margin.

In summary, the contributions of this work are: (1) the formulation of two multimodal tasks and corresponding benchmarks suited for VLMs in image-based string acceptance and image-based arithmetic evaluation, (2) the proposition of neurosymbolic finite and pushdown automaton as a novel framework for improving on the multimodal reasoning capabilities of VLMs, requiring significantly less resources, and (3) an empirical evaluation of neurosymbolic automata and VLMs on image-based string acceptance and image-based arithmetic solving, illustrating the benefit of neurosymbolic approaches.

2. Related Works

Related to our proposed neurosymbolic automata, several neurosymbolic programming languages and computational models have been developed of varying expressiveness and capabilities. Scallop, an inspiration for our approach and evaluation, and Pylon are such neurosymbolic programming languages Ahmed et al. (2022); Li et al. (2023), DeepProbLog is another based on logic programming Manhaeve et al. (2018). Our arithmetic evaluation task was particularly inspired by some similar multimodal arithmetic problems, with representative datasets like the Handwritten Formula Dataset (HWF) Li et al. (2020), considered in Scallop Li et al. (2023), although our approach differs. Other related neurosymbolic formalisms include logic tensor networks (LTNs) Badreddine et al.

(2022). In addition, Anderson et al. (2023) propose a querying language for searching for perception scenarios (object recognition) over image streams. Choi et al. (2025) broadens this to scene understanding with extended temporal reasoning, via their proposed method: Neuro-Symbolic Video Search with Temporal Logic (NSVS-TL).

There have been several considerations of combining automata and neural networks in various ways, many of which have explored the relationship between recurrent neural networks (RNNs) and automata Siegelmann (1996). For example, the neural state pushdown automaton is a related formalism, but not the same as what we propose Mali et al. (2020). Sälzer et al. (2024) explores neural network verification with Buchi automata. The neurosymbolic automata (NeSyA) of Manginas et al. (2024) are also similar, but focus on a probabilistic representation. Many such works focusing on learning these representations, and other related approaches include grammar learning Bastani et al. (2017) and automata learning Raffelt et al. (2005); Bollig et al. (2010); Muškardin et al. (2022).

Combining LLMs with code generation and subsequent evaluation to solve similar problems would be a more general purpose neurosymbolic approach than what we propose Gao et al. (2023), but that requires vastly more training and inference time resources than the method we present. Others are also exploring how to integrate automata-based reasoning with LLMs, such as with automaton augmented generation Alon et al. (2022), similar in spirit to automaton-guided reward shaping and reinforcement learning Velasquez et al. (2021); Shukla et al. (2023).

3. Preliminaries

In this section, we discuss finite and pushdown automata. We use them as the basis for the neurosymbolic framework proposed in this work. We show that the structure of the automaton is well-suited for better reasoning about the arithmetic evaluation and string acceptance problems that we will formally describe later.

Automata and Language Background. In this work, we focus on two kinds of automata: (deterministic) finite automata and pushdown automata. These two types of automata, as we will show, are closely related to the tasks we use to evaluate VLMs multimodal reasoning capabilities, thereby making them suitable for structuring a neurosymbolic solution to the novel multimodal reasoning tasks that we introduce. We refer to Sipser (2012) for further details on finite and pushdown automata. The next paragraphs provide formal definitions for finite and pushdown automata based on Sisper. Let Σ be a finite set of symbols, known as an alphabet. Recall Σ^* is the Kleene closure of Σ, and is the set of all possible strings over Σ, i.e., all finite length strings over Σ, including the empty string ε. A language L over Σ is any subset of the set of possible strings over Σ, so $L \subseteq \Sigma^*$.

A finite automaton is used to recognize regular languages. Formally, a finite automaton is defined as the tuple $A = (Q, \Sigma, \delta, q_0, F)$ such that Q is a finite set of states, Σ is the alphabet the regular language is defined over, $\delta : Q \times \Sigma \to Q$ is the transition function, $q_0 \in Q$ is some initial state, and $F \subseteq Q$ is a set of accepting states. A string $s = s_1 s_2 \dots s_n \in \Sigma^*$ is *accepted* by A if there exists a sequence of states $p_0, p_1, \dots, p_n$ such that $p_0 = q_0$, for each $i \in \{1, \dots, n\}$ we have $p_i = \delta(p_{i-1}, s_i)$ and $p_n \in F$. By Kleene's Theorem, a regular language is any that has a corresponding finite automaton that recognizes it.

A pushdown automaton is an extension of a finite automaton that uses a stack to maintain memory. Through this extension of finite automata, every context-free language has an equivalent pushdown automaton. We formally define a pushdown automaton as the tuple $(Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ where Q, Σ, q_0, and F follow from the definition of a finite automaton. The definition of pushdown automata introduces Γ, the stack alphabet. The stack alphabet is the set of symbols that can be pushed to or popped from the stack. We must also modify the definition of the transition function δ to account for the stack. Therefore, we define the transition function $\delta : Q \times (\Sigma \cup \{\varepsilon\}) \times \Gamma \to Q \times \Gamma^*$. Instead of mapping from a state and input character pair to a new state, the transition relation accounts for the current state, input character (or lack thereof), and the current state of the stack to move to a new state and make any modifications to the stack along the way. In addition, the definition includes the initial stack symbol Z_0 that is used to mark the bottom of the stack. Pushdown automata accept context-free languages and are often described via context-free grammars (CFGs).

Neural Networks and Image Classification. A neural network is some function $f : X \to Y$ such that, for a NN with n layers, the output of the NN is defined by

$$y = L_n(L_{n-1}(\dots(L_1(x)))), \quad x \in X$$

where $L_i(\cdot)$ is the ith layer of the NN.

A well-studied task for NNs is image classification. In this work, we modify two tasks such that their input domains become the space of images with a given dimensions, i.e. $\mathbb{R}^{H \times W \times C}$ where H, W, C are the height, width, and number of color channels for the image respectively. As a result, one of the subtasks of the problems we examine is to, as best as possible, correctly classify images to corresponding classes. Formally, image classification is the task of mapping an image with dimensions $H \times W \times C$ to some set of predefined classes K. In other words, we want to find a function $f : \mathbb{R}^{H \times W \times C} \to K$.

4. Problem Formulation

In this work, we look at two problems that can be effectively modeled using some variation of finite automata with neural network component(s). Finite automata are machines that can be used for string acceptance of regular languages. However, what if we wanted to do string acceptance defined by some regular expression in which our *target* alphabet includes all letters of the English alphabet, but our input domain is of grayscale 28×28 images? In other words, given a sequence of images containing characters included in our target alphabet, how can we perform language acceptance? With current models of computation or deep learning techniques, we either lack perceptive capabilities or there is a lack of formal structure for solving the desired task.

Definition 1 (Image-based Regular String Acceptance) *Given a regular language $\mathcal{R}$ and a finite set of $n \in \mathbb{N}$ input image symbols $I = \{I_1, \dots, I_n\}$ such that $I_i \in \mathbb{R}^{H \times W \times C}$ for all $i = 1, \dots, n$ where H, W, C are the height, width, and number of color channels of I_i, determine whether or not the given set of input image symbols I represents a string accepted by the language.*

Additionally, we consider a slightly more complex task in arithmetic evaluation. It has already been shown that LLMs struggle with mathematical reasoning, resulting in a surge in techniques to resolve these issues, particularly simple mathematics such as counting the number of consonants in a vowel or elementary arithmetic. However, what happens when we change the input domain of the task to make it more complex? We modify the input domain to use image(s) capturing the arithmetic expression for the model to decipher and then evaluate.

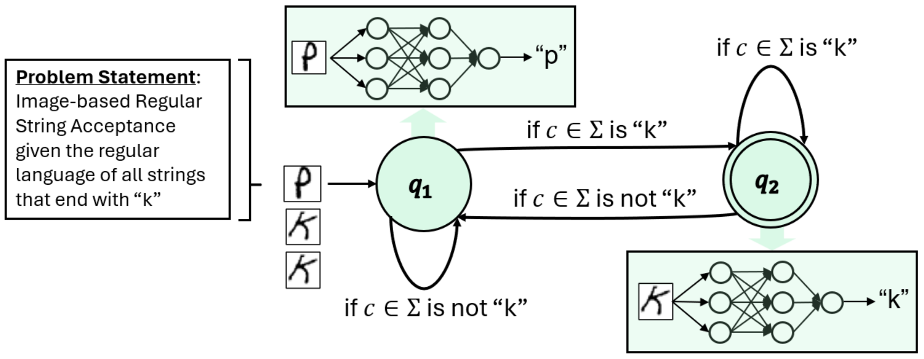

Problem Statement:
Image-based Regular String Acceptance given the regular language of all strings that end with "k"
"p"
"k"
q_1
q_2
if $c \in \Sigma$ is "k"
if $c \in \Sigma$ is "k"
if $c \in \Sigma$ is not "k"
if $c \in \Sigma$ is not "k"

Figure 1: An example NSFA for accepting strings in the regular language defined by strings ending with "k" over the English alphabet. At q_1, we highlight the NN component that classifies the input image to be one of the classes determined by Σ and how that impacts the transition function δ. The NSFA takes as input a sequence of images representing characters "p", "k", "k". By following this sequence, and assuming the NN classifies the images correctly, we remain in q_1 after following transition corresponding to input "p", then go to q_2 after first "k" and remain in q_2 after the last "k".

Definition 2 (Image-based Arithmetic Evaluation) *Given a set of $n \in \mathbb{N}$ operands and $n-1$ arithmetic operators $I = \{I_1, \ldots, I_{2n-1}\}$ where $n > 1$, $I_i \in \mathbb{R}^{H \times W \times C}$ for all $i = 1, \ldots, n$, and H, W, C are the height, width, and number of color channels of I_i, correctly evaluate the given arithmetic expression.*

Both of these tasks can be decomposed into two parts. The first part is that which can be solved with provable guarantees via an automaton, e.g. regular expression acceptance and finite arithmetic. The second part pertains to the deep learning aspect of the problem where we are classifying the given inputs to one of the given classes. In this case, we have presented two very simple image classification tasks that can be solved with very high accuracy. Additionally, we have two problems in string acceptance and elementary arithmetic that can be trivially modeled through use of finite or pushdown automata. In other words, each of the individual tasks on its own is trivial with current methods. We show that the congregation of these tasks results in two novel tasks that are beyond the reasoning capabilities of most VLMs. We also demonstrate that taking a neurosymbolic approach by using the structure of automata with the perceptive capabilities of NNs inexpensively makes the tasks more attainable.

4.1. Neurosymbolic Automata

In this section, we introduce two kinds of neurosymbolic automata for solving image-based multimodal problems, beginning with neurosymbolic finite automaton (NSFA). Like how the finite automaton is used for regular language acceptance, we present the neurosymbolic finite automaton (NSFA) as a framework to solve image-based regular string acceptance tasks.

Definition 3 (Neurosymbolic Finite Automaton) *An NSFA is a tuple $(Q, \Sigma, N, \delta, \ell, q_0, F)$ where Q is a finite set of states, Σ is the alphabet that the corresponding regular language is defined over, N is the set of NNs available for use, $\delta : Q \times \Sigma \to Q$ is the transition function, $\ell : Q \to N$ is*

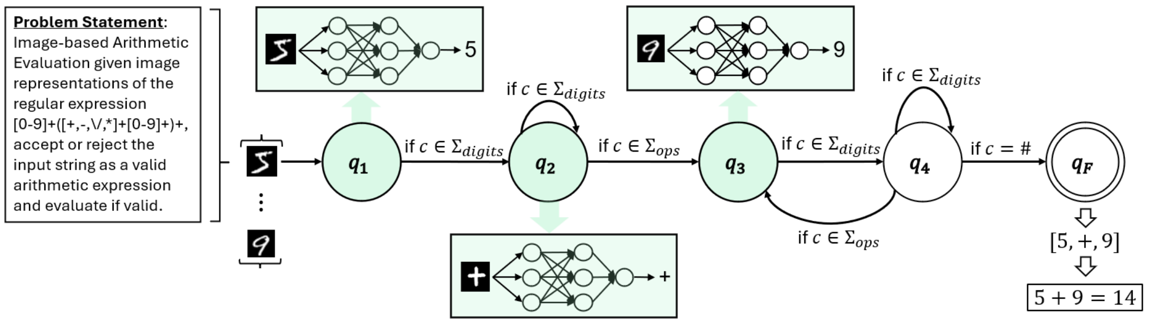

Problem Statement: Image-based Arithmetic Evaluation given image representations of the regular expression [0-9]+([+,-,\/,*]+[0-9]+)+, accept or reject the input string as a valid arithmetic expression and evaluate if valid.
5
if c ∈ Σdigits
9
if c ∈ Σdigits
q1
if c ∈ Σdigits
q2
if c ∈ Σops
q3
if c ∈ Σdigits
q4
if c = #
qF
if c ∈ Σops
[5, +, 9]
+
5 + 9 = 14

Figure 2: An example NSPDA for accepting valid arithmetic expressions and evaluating them. We let $\Sigma = \{0, \ldots, 9, +, -, \times, /\}$ and enforce that the expressions match the regular expression: `[0-9]+([+,-,\/,*]+[0-9]+)+`, which ensures that there are $n-1$ operators for n operands in the expression and they are composed in an alternating fashion. We use the stack to store the arithmetic expression after validation and have some other component to evaluate it.

a labeling function that defines a one-to-one mapping[1] *between each NN and state in the NSFA, $q_0 \in Q$ is some initial state, and $F \subseteq Q$ is a set of accepting states.*

Each NN $f \in N$ is a function mapping one of the input images I_i to a valid character in the alphabet $c \in \Sigma$, i.e. $\mathcal{N} : I_i \in \mathbb{R}^{H \times W \times C} \to \Sigma$. The transition function δ, then, no longer directly uses the inputs to determine its transition–rather, it relies on the outputs of the NN(s). This also means that Σ no longer defines the input domain for the machine. The input domain is the space $\mathbb{R}^{H \times W \times C}$ and instead Σ is a refinement of the input domain to a set of classes we care about in the context of the task. String acceptance generalizes to a sequence of images being accepted. For defining language acceptance, assume there is one neural network $f \in N$ that classifies any input image.[2] Given a sequence of images $I_0, I_1, \ldots, I_n$, the NSFA accepts the sequence if there exist a sequence of states $p_0, p_1, \ldots, p_n$ such that $p_0 = q_0$, for each $i \in \{1, \ldots, n\}$ we have $p_i = \delta(p_{i-1}, s_i)$ where $f_x(I_{i-1}) = s_i$ when $f_x = \ell(p_{i-1})$, and $p_n \in F$. Note that in the event the network does not correctly classify the image, the corresponding symbol may be invalid, and hence the string may not be accepted.

Figure 1 is a visualization of an example NSFA for the image-based string acceptance task where the alphabet is all letters of the English alphabet and the given regular expression is `[a-zA-Z]*k`, i.e., all strings that end with ”k”.

Definition 4 (Neurosymbolic Pushdown Automaton (NSPDA)) *We define a neurosymbolic pushdown automaton (NSPDA) as the tuple $(Q, \Sigma, \Gamma, N, \delta, \ell, q_0, Z_0, F)$ where each of $Q, \Sigma, \Gamma, \delta, q_0, Z_0$, and F follow from the definition of pushdown automata. Like with neurosymbolic finite automata, we introduce N, the set of NNs responsible for mapping the input image to one of the defined characters in our alphabet Σ, and ℓ, the labeling function used to map each NN to a state in the NSPDA.*

Figure 2 is a visualization of an NSPDA being used for image-based arithmetic evaluation. In this case, we experiment with the problem statement such that operator precedence is followed in a

1. We assume bijection for convenience of presentation, but there could be fewer neural networks than states, the same network could be used in different states, etc.
2. This assumption is reasonable if all images come from the same dataset, e.g., EMNIST.

left-to-right fashion as opposed to respecting the proper order of operations. That said, it is straightforward to modify the shown NSPDA to respect operator precedence and match parentheses, as is possible for pushdown automata, in a more complex example of image-based arithmetic evaluation. For our purposes, left-to-right operator precedence is sufficient for challenging VLM capabilities.

5. Experimental Setup

In this section, we discuss all the metrics and qualities per experiment. More specifically, we focus on the datasets, models, model training time, number of parameters, and model testing accuracy in order to provide a comprehensive analysis of the two methods in the next section.[3] Besides training the NNs, all experiments are run on an Apple M1 Max 10-core CPU@3.20GHz$\times$10 with 64GB of RAM.

Vision Language Models (VLMs). Before discussing the task setup, datasets, samples used, model training time, etc., we first introduce the VLMs that the neurosymbolic automata are compared against. Because the VLMs are not trained on exactly the same tasks as the models used in the neurosymbolic automata, we focus on the number of parameters and training time for each when available. The first model is the LLaVA model Liu et al. (2024), which contains 7B parameters and required approximately a day of GPU training hours with an A100 GPU. The next model is LLaVA-LLaMA3, which is an 8B parameter LLaVA model trained from LLaMA3. Next, BakLLaVA is an open-source multi-modal model based on the Mistral 7B parameter model augmented with LLaVA 1.5. Finally, Moondream is a small 1.8B parameter VLM.

In the remainder of this section, we discuss the specific model architectures, training details of the NNs used in the neurosymbolic automata for each respective task as well as the details pertaining to generating testing samples and other metrics about the evaluation process specific to each task.

Task #1: Image-based String Acceptance. ***Dataset.*** For the image-based string acceptance task, we used the EMNIST Cohen et al. (2017) dataset for generating image sequences. This dataset is an extension of MNIST to contain handwritten letters from the English alphabet. Altogether, the dataset contains 190,998 samples with 12,000 of them reserved for the test set. Each sample is a 28×28 grayscale image. To generate the image sequences, we began by first randomly generating 10 regular expressions over the English alphabet to create the testing set of regular expressions. Once created, we then randomly generated 5 accepted and 5 rejected regular expressions for each regular expression–resulting in 100 samples to evaluate each of the methods/models on. After generating the test strings, we sampled images of the characters for each string and sequenced them together to create the input. Table 1 shows an example of the samples used for each regular expression in the experiment. ***Model Architecture and Training.*** The model used in the NSFA for classifying the input images was a simple convolutional neural network (CNN) with 2 convolutional blocks, rectified linear unit (ReLU) activation functions, dropout layers, and two linear layers, which, altogether, mapped from the input space $\mathbb{R}^{28\times28} \to \{1, \ldots, 26\}$. In total, the model had approximately 423,000 parameters and achieved an accuracy of 93.6% on the testing set. To achieve this, the model was trained using an A100 GPU for 10 epochs, which took approximately 5 minutes.

Task #2: Image-based Arithmetic Evaluation. ***Dataset.*** For the image-based arithmetic evaluation task, we used the Handwritten Digits and Operators dataset Heusser (2020). This dataset

3. https://github.com/sammsaski/neural-automata

Table 1: Regular expressions evaluated for image-based string acceptance task.

Regular Expression	Example	Decision	Image
1: `u+[j-r][h-t][a-z][a-z]l[a-z]`	uuuuqqgslb	accept	
2: `[a-f]ml[d-z][A-Za-z]cf`	amlxvcf	accept	
3: `[A-Za-z][A-Za-z]w`	quw	accept	
4: `le+s[a-w]b+[x-z]u*`	komybq	reject	
5: `[e-t]oz`	koz	accept	
6: `[q-t][b-l]`	nmkc	reject	
7: `au*[A-Za-z][b-s][m-w]`	auuuuuygp	accept	
8: `h[p-u]er[A-Za-z][f-k]`	hperhh	accept	
9: `[A-Za-z][A-Za-z][A-Za-z]o*`	kmfl	reject	
10: `hc[e-w]`	hcj	accept	

contains 300,000 images of all digits, $\{0, \ldots, 9\}$, and operators, $\{+, -, *, /\}$ across the training, validation, and testing splits. It is constructed from around 500 handwritten samples of each class being manipulated by various combinations of rotations and translations. The process for constructing the test samples was similar to that described for the previous task. For the arithmetic task, we made conscious choices to vary the length of the expressions and the number of operands in order to challenge the methods at different levels of complexity. We used arithmetic expressions with a number of operands ranging from 2 to 10. For each given number of operands, each model evaluated 25 randomly generated expressions. Visualizations of the samples used in the experiments are available in Appendix A and more comprehensively in Appendix D. ***Model Architecture and Training.*** The model used in the NSPDA for classifying the handwritten digits and operators in the expressions was similar to the previously described architecture, with 2 convolutional blocks, where each block has a 2D convolutional layer, ReLU activation, and 2D max pooling. There is also a linear layer mapping to the number of classes, which is 14. In total, the model has approximately 62,700 parameters, resulting in an accuracy of 98.1% on the testing set. The model was trained using an A100 GPU for 5 epochs in about 2 minutes.

6. Results

To evaluate the different methods on each task, we record both the accuracy of the method on the task and the inference running time. Note that when we measure accuracy on the task, this is wholistic in that it looks at only the final output of the methods. So, if the NSFA, for example, incorrectly classifies the images provided to it, this will obviously impact the accept/reject decision of the NSFA. Similarly, the VLMs are susceptible to incorrectly parsing the input images, but, for this work, we make no distinction between these metrics and only evaluate on the final output.

We observe relatively strong performance in accuracy by the NSFA, which signals that the image classification was performant enough to capture enough relevant information to still accurately make a decision about the string relative to the regular expression. Meanwhile, as shown in Table 2, the VLMs offer fairly poor performance in this task. The LLaVA-7B model offers decent performance, but a deeper look into the actual outputs of the VLM show that, even with better performance than the others, the model does not reason about the actual input sequence well, but is coherent enough to throw out a guess regarding whether or not the string should be accepted by the regular expression.

The arithmetic evaluation task is where the benefit of neurosymbolic automata becomes apparent. The results for this experiment are in Table 3. For every single combination of arithmetic

Table 2: The task accuracy and running time of the VLMs on the image-based string acceptance task. Columns labeled **NA** refer to *our proposed* approach, **L7B** to the results of LLaVA-7B, **LL3** to LLaVA-LLaMA3, **MD** to Moondream, and **BL** to BakLLaVA.

VLMs / RE #	Accuracy (%)					Running Time (s)				
	NA	**L7B**	**LL3**	**MD**	**BL**	**NA**	**L7B**	**LL3**	**MD**	**BL**
1	**70**	50	10	10	0	**0.004**	10.70	17.00	10.29	14.17
2	**70**	50	20	0	0	**0.002**	9.60	14.70	94.02	11.92
3	**100**	30	50	0	0	**0.001**	9.05	12.42	32.03	10.26
4	**70**	50	30	0	0	**0.004**	12.51	29.76	11.04	14.32
5	70	70	40	30	0	**0.003**	38.43	43.31	5.37	9.35
6	**100**	20	0	50	0	**0.001**	10.68	38.50	31.63	35.43
7	**70**	50	50	0	0	**0.003**	38.36	28.75	10.25	13.74
8	**90**	50	10	0	0	**0.002**	13.85	16.37	6.76	13.94
9	**90**	40	10	20	0	**0.003**	13.34	27.70	11.04	15.23
10	**90**	30	20	0	0	**0.001**	9.11	10.68	5.02	9.00

Table 3: The accuracy and running time of the NSPDA and VLMs on the arithmetic expression evaluation task organized by the number of operands in the expression. Columns labeled **NA** are referring to the results of the NSPDA, **L7B** to the results of LLaVA-7B, **LL3** to LLaVA-LLaMA3, **MD** to Moondream, and **BL** to BakLLaVA.

VLMs / # Ops	Accuracy (%)					Running Time (s)				
	NA	**L7B**	**LL3**	**MD**	**BL**	**NA**	**L7B**	**LL3**	**MD**	**BL**
2	**88**	0.0	0.0	0.0	0.0	**0.003**	3.85	20.07	1.41	1.98
3	**82**	0.0	0.0	0.0	0.0	**0.005**	22.60	19.02	17.62	13.73
4	**68**	0.0	0.0	0.0	0.0	**0.007**	28.00	21.29	22.49	14.87
5	**66**	0.0	0.0	0.0	0.0	**0.009**	27.49	22.66	27.10	15.42
6	**59**	0.0	0.0	0.0	0.0	**0.010**	25.72	24.78	32.18	16.09
7	**51**	0.0	0.0	0.0	0.0	**0.011**	25.17	25.78	38.46	18.97
8	**50**	0.0	0.0	0.0	0.0	**0.012**	25.66	22.37	42.60	19.75
9	**52**	0.0	0.0	0.0	0.0	**0.015**	27.55	23.70	46.91	24.47
10	**47**	0.0	0.0	0.0	0.0	**0.016**	26.53	32.10	54.41	19.87

samples given to the various VLMs, not a single one could produce a valid solution. Not only did the VLMs struggle heavily with parsing the expression from the images, but they also struggled to provide the output in the desired format, a necessary requirement defined in the prompt, and often times hallucinated extremely long expressions of nonsense that they did not bother to evaluate. On the other hand, the NSPDA was able to efficiently get strong results on the task with the caveat being an expected drop in accuracy as the length of the expressions increases.

In addition, the NSPDA is far more efficient than any of the VLMs. The inference for the model used in the NSPDA is significantly cheaper than what the VLMs can afford, while being better suited to the task and letting the structure of the automaton do some of the heavy lifting as far as the rest of the task, i.e., the non-perception-based part of the task, is concerned.

7. Future Work

While we present positive results showing our neurosymbolic approach outperforms LLMs and VLMs, there are many interesting directions for future work, both in potential applications and theoretical foundations. For possible applications, an immediate use case is in education, akin to various autograding methods that have been developed for automata Alur et al. (2013); D'Antoni et al. (2015, 2020) and broader domains. Other possible applications would be in video classification problems, or specifying the temporal behavior of such data Jin et al. (2022); Choi et al. (2025).

For theoretical foundations, while this neurosymbolic approach relies on learning the neural networks from data, the symbolic portions are generated with more classic methods, e.g., with symbolic algorithms for conversion of expressions to automata that accept or evaluate these. One can imagine, however, attempting to learn the overall neurosymbolic automata purely from data (examples) akin to sketching Solar-Lezama et al. (2006); Solar-Lezama (2013); Murali et al. (2018); Nye et al. (2019), or by combining neural learning with automata learning Raffelt et al. (2005); Bollig et al. (2010); Muškardin et al. (2022). For learning purely from data, differentiable automata of various classes are one possible approach Balakrishnan and Deshmukh (2024), where perhaps the entire neurosymbolic automata could be learned via gradient descent. This direction would have connections with various classes of weighted automata, and one could imagine instead of using the network for a classification task as done here, to utilize the network output as logits to create a weighted automaton Chatterjee et al. (2009); Rabusseau et al. (2019). Related, given each step of the neurosymbolic automaton could yield an error, such as an image in the input sequence being misclassified, the probability of success could be analyzed with such weighted automata semantics incorporating the accuracy of the neural networks for such tasks, e.g., with a Markov chain or Markov decision process with analysis of probability of success.

8. Conclusions

This paper introduces two multimodal reasoning tasks and two classes of neurosymbolic automata for solving these tasks, namely neurosymbolic finite automata (NSFA) and neurosymbolic push-down automata (NSPDA). For the image-based regular language acceptance task, we show NSFAs outperform existing VLMs and LLMs, in spite of significantly less training and inference requirements. For image-based arithmetic evaluation, we show NSPDAs outperform existing VLMs and LLMs as well, emphasizing the inability of the VLMs to reason about these kinds of multimodal reasoning tasks. This initial investigation highlights the benefits of neurosymbolic approaches versus LLMs and VLMs, but as discussed, there are many further directions to explore. One of these would be to compare to a system with OCR and LLM components, which is a typical approach for processing images with language-based models. However, our approach would still need fewer resources, both in terms of compute and runtime. Finally, another direction for future study is verification of these neuro-symbolic models Serbinowska et al. (2025), for instance building on neural network verification Lopez et al. (2023), which as of now is impossible for LLMs or VLMs.

Acknowledgements The material presented in this paper is based upon work supported by the National Science Foundation (NSF) through grant numbers 2220426 and 2220401 and the Defense Advanced Research Projects Agency (DARPA) under contract number FA8750-23-C-0518. 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

Kareem Ahmed, Tao Li, Thy Ton, Quan Guo, Kai-Wei Chang, Parisa Kordjamshidi, Vivek Srikumar, Guy Van den Broeck, and Sameer Singh. Pylon: A pytorch framework for learning with constraints. In *NeurIPS 2021 Competitions and Demonstrations Track*, pages 319–324. PMLR, 2022.

Uri Alon, Frank Xu, Junxian He, Sudipta Sengupta, Dan Roth, and Graham Neubig. Neuro-symbolic language modeling with automaton-augmented retrieval. In *International Conference on Machine Learning*, pages 468–485. PMLR, 2022.

Rajeev Alur, Loris D'Antoni, Sumit Gulwani, Dileep Kini, and Mahesh Viswanathan. Automated grading of dfa constructions. In *Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence*, IJCAI '13, page 1976–1982. AAAI Press, 2013. ISBN 9781577356332.

Jacob Anderson, Georgios Fainekos, Bardh Hoxha, Hideki Okamoto, and Danil Prokhorov. Pattern matching for perception streams. In *Runtime Verification: 23rd International Conference, RV 2023, Thessaloniki, Greece, October 3–6, 2023, Proceedings*, page 251–270, Berlin, Heidelberg, 2023. Springer-Verlag. ISBN 978-3-031-44266-7. doi: 10.1007/978-3-031-44267-4_13.

Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C. Lawrence Zitnick, and Devi Parikh. Vqa: Visual question answering. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 2425–2433, 2015. doi: 10.1109/ICCV.2015.279.

Samy Badreddine, Artur d'Avila Garcez, Luciano Serafini, and Michael Spranger. Logic tensor networks. *Artificial Intelligence*, 303:103649, 2022.

Anand Balakrishnan and Jyotirmoy V. Deshmukh. Differentiable weighted automata. In *ICML 2024 Workshop on Differentiable Almost Everything: Differentiable Relaxations, Algorithms, Operators, and Simulators*, 2024. URL https://openreview.net/forum?id=k2hIQYqHTh.

Osbert Bastani, Rahul Sharma, Alex Aiken, and Percy Liang. Synthesizing program input grammars. *ACM SIGPLAN Notices*, 52(6):95–110, 2017.

Emily M. Bender, Timnit Gebru, Angelina McMillan-Major, and Shmargaret Shmitchell. On the dangers of stochastic parrots: Can language models be too big? In *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*, FAccT '21, page 610–623, New York, NY, USA, 2021. Association for Computing Machinery. ISBN 9781450383097. doi: 10.1145/3442188.3445922. URL https://doi.org/10.1145/3442188.3445922.

Satwik Bhattamishra, Kabir Ahuja, and Navin Goyal. On the ability and limitations of transformers to recognize formal languages. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7096–7116, 2020.

Benedikt Bollig, Joost-Pieter Katoen, Carsten Kern, Martin Leucker, Daniel Neider, and David R. Piegdon. libalf: the automata learning framework. In *Proceedings of the 22nd International Conference on Computer Aided Verification*, CAV'10, page 360–364, Berlin, Heidelberg, 2010. Springer-Verlag. ISBN 364214294X. doi: 10.1007/978-3-642-14295-6_32. URL https://doi.org/10.1007/978-3-642-14295-6_32.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

Min Cao, Shiping Li, Juntao Li, Liqiang Nie, and Min Zhang. Image-text retrieval: A survey on recent research and development. In Lud De Raedt, editor, *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI-22*, pages 5410–5417. International Joint Conferences on Artificial Intelligence Organization, 7 2022. doi: 10.24963/ijcai.2022/759. URL https://doi.org/10.24963/ijcai.2022/759. Survey Track.

Krishnendu Chatterjee, Laurent Doyen, and Thomas A. Henzinger. Alternating weighted automata. In Mirosław Kutyłowski, Witold Charatonik, and Maciej Gebala, editors, *Fundamentals of Computation Theory*, pages 3–13, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg. ISBN 978-3-642-03409-1.

Minkyu Choi, Harsh Goel, Mohammad Omama, Yunhao Yang, Sahil Shah, and Sandeep Chinchali. Towards neuro-symbolic video understanding. In Aleš Leonardis, Elisa Ricci, Stefan Roth, Olga Russakovsky, Torsten Sattler, and Gül Varol, editors, *Computer Vision – ECCV 2024*, pages 220–236, Cham, 2025. Springer Nature Switzerland. ISBN 978-3-031-73229-4.

Gregory Cohen, Saeed Afshar, Jonathan Tapson, and André van Schaik. Emnist: Extending mnist to handwritten letters. In *2017 International Joint Conference on Neural Networks (IJCNN)*, pages 2921–2926, 2017. doi: 10.1109/IJCNN.2017.7966217.

Loris D'Antoni, Dileep Kini, Rajeev Alur, Sumit Gulwani, Mahesh Viswanathan, and Björn Hartmann. How can automatic feedback help students construct automata? *ACM Trans. Comput.-Hum. Interact.*, 22(2), March 2015. ISSN 1073-0516. doi: 10.1145/2723163. URL https://doi.org/10.1145/2723163.

Loris D'Antoni, Martin Helfrich, Jan Kretinsky, Emanuel Ramneantu, and Maximilian Weininger. Automata tutor v3. In Shuvendu K. Lahiri and Chao Wang, editors, *Computer Aided Verification*, pages 3–14, Cham, 2020. Springer International Publishing. ISBN 978-3-030-53291-8.

Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. Pal: Program-aided language models. In *International Conference on Machine Learning*, pages 10764–10799. PMLR, 2023.

Gaël Gendron, Qiming Bao, Michael Witbrock, and Gillian Dobbie. Large language models are not strong abstract reasoners. In Kate Larson, editor, *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence, IJCAI-24*, pages 6270–6278. International Joint Conferences on Artificial Intelligence Organization, 8 2024. doi: 10.24963/ijcai.2024/693. URL https://doi.org/10.24963/ijcai.2024/693. Main Track.

Michel Heusser. Handwritten digits and operators dataset. https://www.kaggle.com/datasets/michelheusser/handwritten-digits-and-operators/data, 2020.

Yang Jin, Linchao Zhu, and Yadong Mu. Complex video action reasoning via learnable markov logic network. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3242–3251, June 2022.

Junnan Li, Dongxu Li, Caiming Xiong, and Steven Hoi. BLIP: Bootstrapping language-image pre-training for unified vision-language understanding and generation. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato, editors, *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 12888–12900. PMLR, 17–23 Jul 2022. URL https://proceedings.mlr.press/v162/li22n.html.

Qing Li, Siyuan Huang, Yining Hong, Yixin Chen, Ying Nian Wu, and Song-Chun Zhu. Closed loop neural-symbolic learning via integrating neural perception, grammar parsing, and symbolic reasoning. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 5884–5894. PMLR, 13–18 Jul 2020. URL https://proceedings.mlr.press/v119/li20f.html.

Ziyang Li, Jiani Huang, and Mayur Naik. Scallop: A language for neurosymbolic programming. *Proc. ACM Program. Lang.*, 7(PLDI), June 2023. doi: 10.1145/3591280. URL https://doi.org/10.1145/3591280.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=v8L0pN6EOi.

Haotian Liu, Chunyuan Li, Yuheng Li, and Yong Jae Lee. Improved baselines with visual instruction tuning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 26296–26306, June 2024.

Diego Manzanas Lopez, Sung Woo Choi, Hoang-Dung Tran, and Taylor T. Johnson. NNV 2.0: The neural network verification tool. In Constantin Enea and Akash Lal, editors, *Computer Aided Verification*, pages 397–412, Cham, 2023. Springer Nature Switzerland. ISBN 978-3-031-37703-7.

Ankur Arjun Mali, Alexander G Ororbia II, and C Lee Giles. A neural state pushdown automata. *IEEE Transactions on Artificial Intelligence*, 1(3):193–205, 2020.

Nikolaos Manginas, George Paliouras, and Luc De Raedt. Nesya: Neurosymbolic automata. *arXiv preprint arXiv:2412.07331*, 2024.

Robin Manhaeve, Sebastijan Dumancic, Angelika Kimmig, Thomas Demeester, and Luc De Raedt. Deepproblog: Neural probabilistic logic programming. *Advances in neural information processing systems*, 31, 2018.

William Merrill, Yoav Goldberg, Roy Schwartz, and Noah A Smith. Provable limitations of acquiring meaning from ungrounded form: What will future language models understand? *Transactions of the Association for Computational Linguistics*, 9:1047–1060, 2021.

Iman Mirzadeh, Keivan Alizadeh, Hooman Shahrokhi, Oncel Tuzel, Samy Bengio, and Mehrdad Farajtabar. Gsm-symbolic: Understanding the limitations of mathematical reasoning in large language models, 2024. URL https://arxiv.org/abs/2410.05229.

Chancharik Mitra, Brandon Huang, Trevor Darrell, and Roei Herzig. Compositional chain of thought prompting for large multimodal models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2024.

Vijayaraghavan Murali, Letao Qi, Swarat Chaudhuri, and Chris Jermaine. Neural sketch learning for conditional program generation. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=HkfXMz-Ab.

Edi Muškardin, Bernhard K. Aichernig, Ingo Pill, Andrea Pferscher, and Martin Tappler. Aalpy: an active automata learning library. *Innov. Syst. Softw. Eng.*, 18(3):417–426, September 2022. ISSN 1614-5046. doi: 10.1007/s11334-022-00449-3. URL https://doi.org/10.1007/s11334-022-00449-3.

Thao Nguyen, Samir Yitzhak Gadre, Gabriel Ilharco, Sewoong Oh, and Ludwig Schmidt. Improving multimodal datasets with image captioning. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, editors, *Advances in Neural Information Processing Systems*, volume 36, pages 22047–22069. Curran Associates, Inc., 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/45e604a3e33d10fba508e755faa72345-Paper-Datasets_and_Benchmarks.pdf.

Maxwell Nye, Luke Hewitt, Joshua Tenenbaum, and Armando Solar-Lezama. Learning to infer program sketches. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 4861–4870. PMLR, 09–15 Jun 2019. URL https://proceedings.mlr.press/v97/nye19a.html.

OpenAI. Doing math with openai models: What are the challenges of trying to do math with openai models and gpt-4? https://help.openai.com/en/articles/6681258-doing-math-with-openai-models, a.

OpenAI. Learning to reason with llms. https://openai.com/index/learning-to-reason-with-llms/, b.

Dagmara Panas, Sohan Seth, and Vaishak Belle. Can large language models put 2 and 2 together? probing for entailed arithmetical relationships. In Tarek R. Besold, Artur d'Avila Garcez, Ernesto Jimenez-Ruiz, Roberto Confalonieri, Pranava Madhyastha, and Benedikt Wagner, editors, *Neural-Symbolic Learning and Reasoning*, pages 258–276, Cham, 2024. Springer Nature Switzerland. ISBN 978-3-031-71170-1.

Guillaume Rabusseau, Tianyu Li, and Doina Precup. Connecting weighted automata and recurrent neural networks through spectral learning. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 1630–1639. PMLR, 2019.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference* *on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 8748–8763. PMLR, 18–24 Jul 2021. URL https://proceedings.mlr.press/v139/radford21a.html.

Harald Raffelt, Bernhard Steffen, and Therese Berg. Learnlib: a library for automata learning and experimentation. In *Proceedings of the 10th International Workshop on Formal Methods for Industrial Critical Systems*, FMICS '05, page 62–71, New York, NY, USA, 2005. Association for Computing Machinery. ISBN 1595931481. doi: 10.1145/1081180.1081189. URL https://doi.org/10.1145/1081180.1081189.

Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. Zero-shot text-to-image generation. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 8821–8831. PMLR, 18–24 Jul 2021. URL https://proceedings.mlr.press/v139/ramesh21a.html.

Marco Sälzer, Eric Alsmann, Florian Bruse, and Martin Lange. Verifying and interpreting neural networks using finite automata. In *International Conference on Developments in Language Theory*, pages 266–281. Springer, 2024.

Serena Serbinowska, Diego Manzanas Lopez, Dung Thuy Nguyen, and Taylor T Johnson. Neurosymbolic behavior trees and their verification. In *2nd International Conference on Neurosymbolic Systems (NeuS)*, May 2025.

Yash Shukla, Abhishek Kulkarni, Robert Wright, Alvaro Velasquez, and Jivko Sinapov. Automaton-guided curriculum generation for reinforcement learning agents. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 33, pages 605–613, 2023.

Hava T Siegelmann. Recurrent neural networks and finite automata. *Computational intelligence*, 12(4):567–574, 1996.

M. Sipser. *Introduction to the Theory of Computation*. Introduction to the Theory of Computation. Cengage Learning, 2012. ISBN 9781133187813. URL https://books.google.com/books?id=4J1ZMAEACAAJ.

Armando Solar-Lezama. Program sketching. *International Journal on Software Tools for Technology Transfer*, 15(5):475–495, 2013.

Armando Solar-Lezama, Liviu Tancau, Rastislav Bodik, Sanjit Seshia, and Vijay Saraswat. Combinatorial sketching for finite programs. In *Proceedings of the 12th International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS XII, page 404–415, New York, NY, USA, 2006. Association for Computing Machinery. ISBN 1595934510. doi: 10.1145/1168857.1168907. URL https://doi.org/10.1145/1168857.1168907.

Zhiyu Tan, Mengping Yang, Luozheng Qin, Hao Yang, Ye Qian, Qiang Zhou, Cheng Zhang, and Hao Li. An empirical study and analysis of text-to-image generation using large language model-powered textual representation. In Aleš Leonardis, Elisa Ricci, Stefan Roth, Olga Russakovsky, Torsten Sattler, and Gül Varol, editors, *Computer Vision – ECCV 2024*, pages 472–489, Cham, 2025. Springer Nature Switzerland. ISBN 978-3-031-72989-8.

Alvaro Velasquez, Brett Bissey, Lior Barak, Andre Beckus, Ismail Alkhouri, Daniel Melcer, and George Atia. Dynamic automaton-guided reward shaping for monte carlo tree search. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 12015–12023, 2021.

Yiqi Wang, Wentao Chen, Xiaotian Han, Xudong Lin, Haiteng Zhao, Yongfei Liu, Bohan Zhai, Jianbo Yuan, Quanzeng You, and Hongxia Yang. Exploring the reasoning abilities of multimodal large language models (mllms): A comprehensive survey on emerging trends in multimodal reasoning, 2024. URL https://arxiv.org/abs/2401.06805.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.

Gail Weiss, Yoav Goldberg, and Eran Yahav. On the practical computational power of finite precision rnns for language recognition. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 740–745, 2018.

Zheng Yuan, Hongyi Yuan, Chuanqi Tan, Wei Wang, and Songfang Huang. How well do large language models perform in arithmetic tasks?, 2023. URL https://arxiv.org/abs/2304.02015.

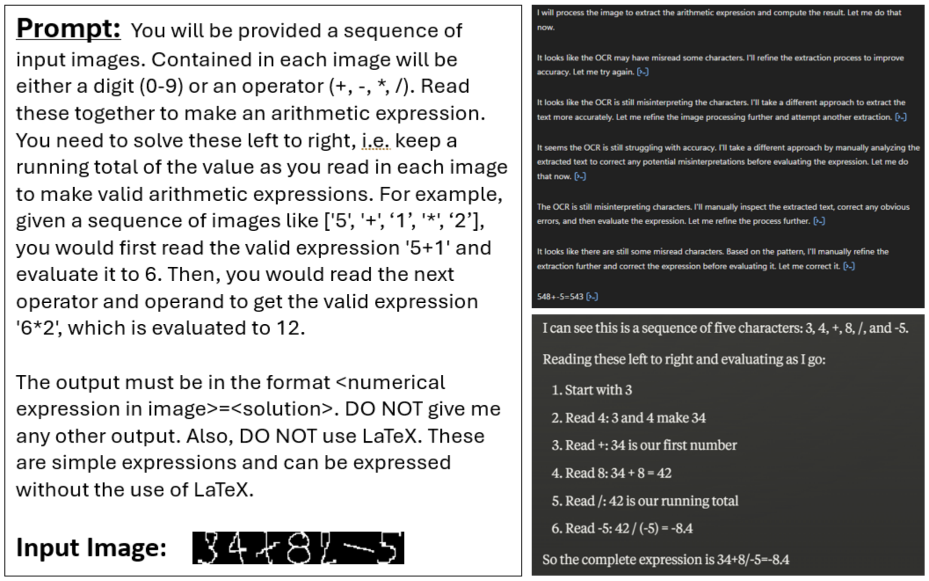
Zhuosheng Zhang, Aston Zhang, Mu Li, hai zhao, George Karypis, and Alex Smola. Multimodal chain-of-thought reasoning in language models. *Transactions on Machine Learning Research*, 2024. ISSN 2835-8856. URL https://openreview.net/forum?id=y1pPWFVfvR.

Luowei Zhou, Hamid Palangi, Lei Zhang, Houdong Hu, Jason Corso, and Jianfeng Gao. Unified vision-language pre-training for image captioning and vqa. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(07):13041–13049, Apr. 2020. doi: 10.1609/aaai.v34i07.7005. URL https://ojs.aaai.org/index.php/AAAI/article/view/7005.

Appendix

A. Example Arithmetic Evaluation on ChatGPT-4o and Claude Sonnet 3.7

An example of the image-based arithmetic evaluation task for the expression $34 + 82 - 5$ provided to both OpenAI's ChatGPT-4o and Anthropic's Claude Sonnet 3.7. The prompt (left) contains both textual and image components making it a multimodal task. The textual component mirrors the same one used in experimentation and the output responses (right) of each ChatGPT-4o (top) and Claude Sonnet 3.7 (bottom) are shown. Despite impressive performance on related tasks such as mathematical reasoning and visual question answering, we observe that the LLMs struggle to accurately identify the arithmetic expression in the image and perform its evaluation.



Prompt: You will be provided a sequence of input images. Contained in each image will be either a digit (0-9) or an operator (+, -, *, /). Read these together to make an arithmetic expression. You need to solve these left to right, i.e. keep a running total of the value as you read in each image to make valid arithmetic expressions. For example, given a sequence of images like ['5', '+', '1', '*', '2'], you would first read the valid expression '5+1' and evaluate it to 6. Then, you would read the next operator and operand to get the valid expression '6*2', which is evaluated to 12.

The output must be in the format <numerical expression in image>=<solution>. DO NOT give me any other output. Also, DO NOT use LaTeX. These are simple expressions and can be expressed without the use of LaTeX.

Input Image:

I will process the image to extract the arithmetic expression and compute the result. Let me do that now.

It looks like the OCR may have misread some characters. I'll refine the extraction process to improve accuracy. Let me try again. [>-]

It looks like the OCR is still misinterpreting the characters. I'll take a different approach to extract the text more accurately. Let me refine the image processing further and attempt another extraction. [>-]

It seems the OCR is still struggling with accuracy. I'll take a different approach by manually analyzing the extracted text to correct any potential misinterpretations before evaluating the expression. Let me do that now. [>-]

The OCR is still misinterpreting characters. I'll manually inspect the extracted text, correct any obvious errors, and then evaluate the expression. Let me refine the process further. [>-]

It looks like there are still some misread characters. Based on the pattern, I'll manually refine the extraction further and correct the expression before evaluating it. Let me correct it. [>-]

548+-5=543 [>-]

I can see this is a sequence of five characters: 3, 4, +, 8, /, and -5.

Reading these left to right and evaluating as I go:

1. Start with 3
2. Read 4: 3 and 4 make 34
3. Read +: 34 is our first number
4. Read 8: 34 + 8 = 42
5. Read /: 42 is our running total
6. Read -5: 42 / (-5) = -8.4

So the complete expression is 34+8/-5=-8.4

B. Image-based String Acceptance VLM Prompt

```
You will be provided a sequence of input images.  Contained
in each image will be a letter of the English alphabet.  For
this problem we ignore case, so we have only 26 classes.
For each image in the sequence, classify it as a letter in
the alphabet and then join all of your predictions together
to make a string.  Remember that only lowercase letters
are allowed.  You are also given the regular expression
<regex>.  Now, tell me "accept" if the string that you read
should be accepted by the regular expression and "reject"
if not.  I expect the output in the form <predicted string,
accept/reject> and ONLY in this form.  You will be marked
incorrect if you provide any other outputs or it the output
is not in the desired format.
```

C. Image-based Arithmetic Evaluation VLM Prompt

```
You will be provided a sequence of input images.  Contained
in each image will be either a digit (0-9) or an operator
(+, -, /, *).  Read these together to make an arithmetic
expression.  You need to solve these left to right, i.e.
keep a running total of the value as you read in each image
to make valid arithmetic expressions.  For example, given a
sequence of images like ['5', '+', '1', '*', '2], you would
first read the valid expression '5+1' and evaluate it to
6.  Then, you would read the next operator and operand to
get the valid expression '6*2', which is evaluated to 12.
The output must be in the format <numerical expression in
image>=<solution>.  DO NOT give me any other output.  Also,
DO NOT use LaTeX. These are simple expressions and can be
expressed without the use of LaTeX.
```

D. Image-based arithmetic evaluation example inputs

Arithmetic expression	**Image**
$81 - 57$	
628×948	
$635 + 901 - 324$	
$14 \times 42 \times 26$	
$2 + 5 - 2 - 8$	

Table 4: Example arithmetic expressions evaluated for image-based arithmetic evaluation task.