Neurosymbolic Programming for Science

Jennifer J. Sun[*] Caltech
Megan Tjandrasuwita[*] MIT CSAIL
Atharva Sehgal[*] UT Austin
Armando Solar-Lezama MIT CSAIL
Swarat Chaudhuri UT Austin
Yisong Yue Caltech
Omar Costilla-Reyes [†] MIT CSAIL

Abstract

Neurosymbolic Programming (NP) techniques have the potential to accelerate scientific discovery These models combine neural and symbolic components to learn complex patterns and representations from data, using high-level concepts or known constraints. As a result, NP techniques can interface with symbolic domain knowledge from scientists, such as prior knowledge and experimental context, to produce interpretable outputs. We identify opportunities and challenges between current NP models and scientific workflows, with real-world examples from behavior analysis in science to enable its use broadly for workflows across the natural and social sciences.

1 Introduction

One of the grand challenges in the artificial intelligence and scientific communities is to find an AI scientist: an artificial agent that can automatically design, test, and infer scientific hypotheses from data. This application poses several distinct challenges for existing learning techniques because of the need to ensure that new theories are consistent with prior scientific knowledge, as well as to enable scientists to reason about the implications of new hypotheses and experimental designs.

The distinct requirements of scientific discovery have pushed the community to explore expressive yet symbolically interpretable techniques such as symbolic regression [Cranmer, 2020], interpretable machine learning [Ustun and Rudin, 2017, Doshi-Velez and Kim, 2017, Kleinberg et al., 2018, McGrath et al., 2021], as well as program synthesis [Koksal et al., 2013, Ellis et al., 2022]. These techniques have helped the community make significant progress in a number of applications, as discussed in Goodwin et al. [2022] and Sapoval et al. [2022], but we are still far from solving the grand challenge.

We focus on the opportunities and challenges behind an important class of learning techniques based on *Neurosymbolic Programming* (NP) [Chaudhuri et al., 2021]. These techniques combine neural and symbolic reasoning to build expressive models that incorporate prior expert knowledge and strong constraints on model behavior and structure. NP is capable of producing symbolic representations of theories that can be analyzed and manipulated to answer rich counterfactuals.

NP empowers a new line of attack on the grand AI scientist challenge: represent scientific hypotheses as programs in a *Domain Specific Language* (DSL) and use neurosymbolic program synthesis to automatically discover these programs (Figure 1). Users can incorporate complex prior knowledge (e.g., known features and constraints) into the design of the DSL. The NP learning algorithms can then follow classic scientific reasoning principles to find predictive programs. Also, models learned this way are often similar to code that human domain experts write during manual scientific modeling.

[*]Equal contribution
[†]Corresponding author: costilla@mit.edu

NeurIPS 2022 AI for Science Workshop.

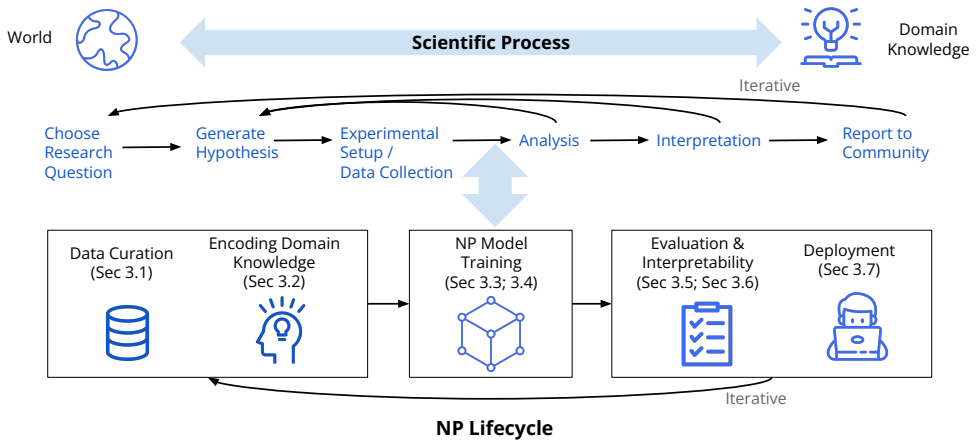

Figure 1: Synergy between the scientific and neurosymbolic programming workflow.

Collectively, these characteristics enable a transparent and interactive process where an AI system and a human expert collaborate on evidence-based reasoning and the discovery of new scientific facts.

Here, we use behavior analysis as a concrete, illustrative example. We start with an introduction to NP (Section 2), then outline challenges and opportunities for future research (Section 3).

Behavior analysis as running example. We chose behavior analysis as an example use case for several reasons. Behavioral data is spatiotemporal, which is a common data type across the sciences. Correspondingly, underlying challenges are shared in other domains, from monitoring vital signs, to modeling physical systems, to studying dynamics of chemical reactions. Additionally, behavioral data illustrates common challenges with scientific data. These datasets often contain rare behaviors with noisy and imperfect data, and can vary significantly in relevant time-scales (e.g., milliseconds vs hours). Datasets also vary across labs, organisms/systems, and experimental setups. Finally, automatic behavior quantification is becoming increasingly crucial in many fields, such as neuroscience, ecology, biology, and healthcare. As computational behavior analysis and neurosymbolic learning are both developing research areas, there are many exciting opportunities to explore at their intersection.

Background on behavior analysis. An important objective of behavior analysis is to quantify behavior from video using continuous or discrete representations. We focus on the case of animal behavior analysis in science [Anderson and Perona, 2014, Datta et al., 2019], where there are diverse organisms and naturalistic behaviors. A common approach is first to perform animal pose tracking from video [Mathis et al., 2018, Pereira et al., 2022], then categorize behaviors of interest from animal pose [Segalin et al., 2021] (as discussed later in Figure 4). From a NP perspective, this approach can be viewed as learning a symbolically interpretable intermediate representation (tracked keypoints).

Existing challenges in behavior analysis. Similar to other scientific fields, data collection and annotation are expensive for behavioral experiments. Analyzing data is also time-consuming and expensive since specialized domain expertise is required for identifying behaviors of interest and extracting knowledge. Models need to interface efficiently with scientists and data at both the inputs and outputs from the scientific process (Figure 1). For NP models, leveraging domain expertise in the form of behavioral attributes has been demonstrated to improve data efficiency [Sun et al., 2021] and interpretability [Tjandrasuwita et al., 2021].

Beyond existing works, there is a variety of domain expertise that requires new algorithmic designs to integrate into the NP workflow, such as experimental context, existing ethograms, and scientific spatiotemporal constraints. Incorporating such domain knowledge has the potential to enable NP models to be more robust to noisy and imperfect data, and enable new scientific inquiries that were too expensive to study previously. Furthermore, when black-box models are used for studying behavior, it is difficult to diagnose errors and explain model outputs [Rudin, 2019]. NP models have the potential to produce symbolic descriptions of behavior (Figure 2), which enables experts to connect model interpretations with other parts of the behavior analysis workflow, e.g., describing behavioral differences across different strains of mice. Finally, to enable the use of NP models in real-world science workflows, these models must be scalable and produce robustly reproducible interpretations.

$$\alpha \quad ::= \quad x \mid c \mid \oplus(\alpha_1, \ldots, \alpha_k) \mid \oplus_\theta(\alpha_1, \ldots, \alpha_k) \mid \textbf{if } \alpha_1 \textbf{ then } \alpha_2 \textbf{ else } \alpha_3 \mid \textbf{sel}_S\ x$$
$$\textbf{map } (\textbf{fun } x_1.\alpha_1)\ x \mid \textbf{fold } (\textbf{fun } x_1.\alpha_1)\ c\ x \mid \textbf{mapprefix } (\textbf{fun } x_1.\alpha_1)\ x$$

(a) Example of a domain-specific language for neurosymbolic programming.

$$\begin{aligned}
&\textbf{map } (\textbf{fun } x_t. \\
&\quad \textbf{if } \mathit{DistAffine}_{[.0217];-.2785}(x_t) \\
&\quad \textbf{then } \mathit{AccAffine}_{[-.0007,.0055,.0051,-.0025];3.7426}(x_t) \textbf{ else } \mathit{DistAffine}_{[-.2143];1.822)}(x_t))\ x
\end{aligned}$$

(b) A neurosymbolic program.

Figure 2: Examples of NP for learning programs in mouse social behavior [Shah et al., 2020].

2 Neurosymbolic Programming Techniques

Neurosymbolic programs incorporate latent representations from neural networks and symbols that explicitly capture pre-existing human knowledge, and connect these elements using rich architectures that mirror classic models of computation. The programs, assumed to belong to a DSL, are learned using a combination of gradient-based optimization, probabilistic methods, and symbolic techniques.

Anatomy of a Neurosymbolic Program. In general, a neurosymbolic program comprises its discrete architecture and continuous parameters. Consider the neurosymbolic program in Figure 2b, obtained from DSL 2a, which comprises logical symbolic operations such as "if" statements, as well as functions with continuous parameters. The architecture includes all the discrete symbolic choices that form the structure of the program (such as whether to have an "if" statement and where to place it relative to other operations), and "programming" this architecture is analogous to architecture design in neural networks (e.g., whether to use convolutions, recurrent units, attention, etc.).

Space of Neurosymbolic Programs. The range of NP methods vary in the degree to which they use neural versus symbolic reasoning (Figure 3). The two ends of the spectrum correspond to purely neural (a 1D convolutional network) and purely symbolic (a human written program) models, respectively. The techniques close to the center are neurosymbolic: the model in the center-left is a neurosymbolic encoder [Zhan et al., 2021], while the model in the center-right is a program with differentiable parameters for behavior analysis [Tjandrasuwita et al., 2021]. From a definitional perspective, purely neural and purely symbolic programs can be considered special cases of neurosymbolic programs, although we typically do not refer to those as neurosymbolic programs for practical purposes.

To illustrate the strengths and weaknesses of each model in Figure 3, assume that we have a scientific hypothesis to test on a dataset. On the right side, the fully symbolic model would involve an expert-written program that encodes the hypothesis in a general programming language. This program requires no learnable parameters, is fully interpretable and, if needed, can be iteratively improved. However, this method is also brittle, and the program must be engineered to handle all the dynamics of the dataset. This is intractable for models with complex dynamics. On the left side, the purely neural model would model the hypothesis directly using the dataset. Such models generalize well to the dataset but offer limited interpretability and control over the generated hypothesis (which can

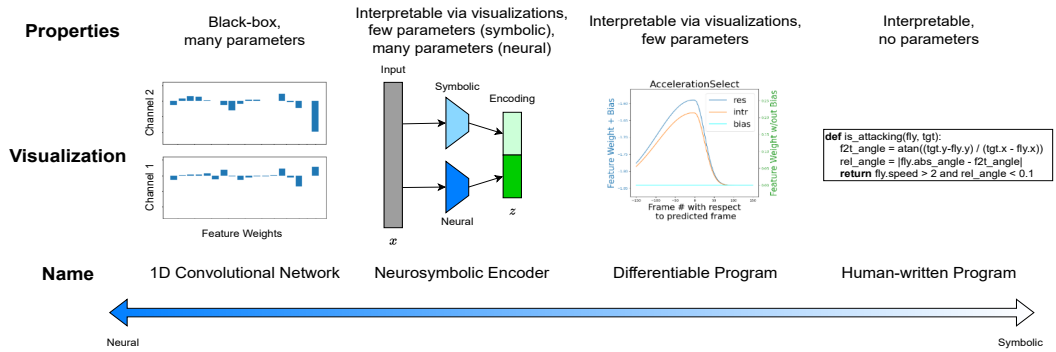

Figure 3: Space of neurosymbolic programming models in behavior analysis, including purely neural (left), purely symbolic (right), and neurosymbolic (two in middle)

make them prone to overfitting and limit generalization). In the middle, neurosymbolic approaches offer a mixture of both neural and symbolic components, and if designed well can inherit both the flexibility of neural networks as well as the structured semantics of symbolic models.

Desiderata for Effective Neurosymbolic Programming. To a first approximation, there are two main requirements for effective neurosymbolic programming: having a good DSL, and scalable learning algorithms. The DSL effectively corresponds to the component building blocks from which one can construct a neurosymbolic architecture, and is one of the primary ways that experts inject domain knowledge (i.e., inductive bias) into the learning process. Learning has two aspects: 1) searching for a the best neurosymbolic architecture, and 2) optimizing the parameters within a fixed architecture. The former is is analogous to neural architecture search Elsken et al. [2019], and the latter is analogous to standard parameter optimization in deep learning. We discuss these issues in detail in Section 3, and conclude this section with a discussion on differentiable programs that enable differentiable parameter optimization within discrete symbolic architectures.

Differentiable programs. On a high level, a differentiable program for a programming language is defined as a composition of functions such that the parameters of the function are differentiable. A differentiable program follows the syntax defined by a DSL which can consist of parametric functions (multi-layered perceptrons, linear transformations), algebraic functions (`add`, `multiply`), and programming languages higher-order functions (`map`, `fold`). Furthermore, the composition of these differentiable functions is also differentiable through the chain rule. This property enables resulting programs to be fully differentiable.

NP programs may be difficult to interpret by domain experts [Tjandrasuwita et al., 2021] focuses on explaining the difference in behavior expert annotations. They replace generic higher-order functions over recursive data structures, e.g. `map` and `fold`, with a differentiable temporal filter operation, the Morlet Filter. The filter models temporal information in a highly data-efficient manner and can be interpreted as a human's impulse response to a given behavioral feature for classification.

3 Opportunities and Challenges at the Intersection of Neurosymbolic Learning and Science

We have defined and outlined benefits of the NP framework. However, gaps remain between current NP approaches and practical use cases in science (Figure 1). We draw attention to these challenges to encourage the research community to collaborate in the development of new NP methods to increase the synergy with scientific workflows to accelerate scientific discovery.

3.1 Dealing with raw, noisy, and imperfect data

Data found in scientific domains provides an opportunity to study NP models with imperfect data in real-world conditions, such as with missing data, experiment noise, and distribution shifts. By incorporating prior knowledge and known constraints in NP models, they have the potential to perform well in the presence of imperfect data. For example, for behavior analysis, neurosymbolic models can automatically learn weak labels from a small amount of annotated examples and apply these trained models to generate weak supervision for a full dataset [Tseng et al., 2022].

These types of imperfect data exist throughout science: missing data in neural recordings due to hardware issues, noise in pose estimators for tracking animal movements, and distribution shifts. An additional source of noise in data is the considerable variability that exists in the labeling generation process, such as annotator subjectivity and ambiguity in category definitions. Furthermore, scientists are often interested in studying rare categories, such as behaviors that may occur in less than 1% of a dataset. NP research [Shah et al., 2020, Cui and Zhu, 2021] leverages the flexibility of neural networks with symbolic domain knowledge; however, there remain challenges in improving model scalability that we have outlined in this section.

Structural discovery. In many scientific workflows, meaningful categories, and structures in raw data may not be clear ahead of time and requires unsupervised or self-supervised learning from data. For example, there are many tools for discovering new behavior categories from data without expert supervision [Pereira et al., 2020]. Zhan et al. [2021] demonstrated that integrating domain knowledge in an NP workflow results in more meaningful discovered categories compared to fully neural methods. In addition to the algorithmic challenges discussed in previous sections, future

research work needs to be robust to variations in experimental noise and produce interpretations of discovered structures in the data that are useful in the context of science.

Distribution Shifts. Distribution shifts are common in real-world applications [Koh et al., 2021]. For typical black-box machine learning models, it is difficult to diagnose and address these errors. NP approaches generally learn interpretable and modular programs, which have the promise to tackle this challenge. For example, in behavior analysis, when the physical behavioral area changes in size, the relative size of mice also changes. This causes errors in behavior classifiers trained in a previously known area, but NP programs can be scaled accordingly to adjust to the new task.

3.2 Encoding and Learning Domain Knowledge

The success of NP techniques often depends on how a DSL is defined. However, it is not always clear how to handcraft domain-specific components that work best in a scientific context, and this can be a time-consuming process. *Library learning* proposes algorithms that consolidate common patterns in successful programs add them iteratively to the current DSL, enabling the program search to discover high-performing programs with little effort.

Library learning for science. In behavior analysis, humans are capable of writing short programs that can improve model learning, such as by designing features and heuristics [Segalin et al., 2021, Tseng et al., 2022, Eyjolfsdottir et al., 2014]. However, these programs are greatly limited by their simplicity and may not capture complex behavior. Library learning has the potential to augment human feature design, by synthesizing interpretable programs and inducing high-level DSL's, given low-level, generic primitives. For example, Dreamcoder [Ellis et al., 2021] is a library learning system that has been applied to physics equation discovery. Library learning has also been studied for generative modeling in molecular chemistry [Guo et al., 2021], which was demonstrated to be able to handle data-limited settings often found in science.

Challenges of library learning for science. In general, it is unclear how library learning can scale to more complex real-world data scientific domains, such as behavior analysis, which often consists of thousands of video frames with noisy data. In addition, it is highly expensive to collect behavior annotations across up to hundreds of behaviors, which is needed to perform traditional library learning. In contrast, current library learning methods have been applied to contexts where each task consists of a few examples, not exceeding hundreds of data points. In addition, the labels are noiseless, as opposed to real-world situations found in behavior analysis [Leng et al., 2020, Segalin et al., 2021].

Another challenge is that domain experts still need to interpret solutions generated by the NP library learning system. One promising approach leverages natural language to impose a stronger prior on the program search and the library learning [Wong et al., 2021], resulting in a more human-interpretable DSL. Additionally, building a smooth interface between expertise in science, program synthesis, neural networks and probabilistic library learning methods, found in NP, would likely require significant engineering and research efforts (Section 3.7).

Representing informal scientific theories. Beyond specific use cases, there is a vast body of knowledge that has been accumulated throughout the span of a given scientific field. Such informal knowledge may not be explicitly represented as a DSL; for instance, behavioral neuroscientists have collected ethograms [Garner], or natural language descriptions of the functions of species-specific behavior. Other examples include causal relations between phenomenon or interventions in an experimental setting. Past work has proposed logical languages capable of representing intuitive theories of causalities [Goodman et al., 2011]. However, capturing all informal and formal knowledge with a single DSL and searching over this space of programs would likely be intractable. Rather, ongoing research in NP focus on identifying the domain knowledge relevant to a specific subset of scientific problems and distilling such theories into a DSL.

3.3 Scalability challenge

For an optimization standpoint, compared to conventional deep learning, the main additional challenge is searching over program architectures. Architecture search is in general very challenging, and typically leads to an combinatorial discrete search space.

Inductive synthesis. A large body of works on program synthesis has focused on *inductive synthesis*, or synthesizing programs from examples [Lau and Weld, 1998, Gulwani, 2011, Devlin et al., 2017]. While such a goal is on the surface similar to performing machine learning (ML) with programs

as models, a key difference is that ML approaches depend on defining a clear space of models (i.e. neural networks, support vector machines, decision trees) and generalizing to unseen data. In contrast, much work in inductive synthesis considers an arbitrary space of programs and spends significant effort on sample engineering, treating them as noiseless specifications. As a result, inductive synthesis scales poorly with an increase in program length and number of examples.

Scaling NP in science. To tackle scalability in science, models need to handle large and potentially noisy datasets, high-dimensional input space, and a variety of analysis tasks. Recently, NP research [Shah et al., 2020, Cui and Zhu, 2021] propose frameworks that scale to large datasets given an expressive DSL. These works are instantiated in behavior analysis: learning programs on temporal trajectory data to reproduce expert annotations of behavior that contain noisy labels, similar to other scientific data. These works tackle the challenge of discovering programs with parameters, which can be directly optimized through popular gradient optimization techniques. While NP methods provide a means of scaling inductive synthesis to scientific datasets, these techniques often involve combining a discrete search over an exponential space of programs with continuous optimization.

Challenges for enabling scalability. Scaling up program synthesis for neurosymbolic programming is an active field of research. For instance, differentiable program synthesis methods [Cui and Zhu, 2021] have studied the tradeoff between computation and memory, with heuristics to mitigate memory usage. However, training fully neural models on a GPU is often more efficient than training NP models, which requires searching through an exponentially large space of symbolic architectures on a CPU. Furthermore, scalability has not been broadly explored for different types of scientific data, such as video recordings, which are much higher dimensional than trajectory data. Finally, the effectiveness of program synthesis may still be limited by the expressivity of a DSL, which requires experts to spend time encoding domain knowledge, such as expert-designed behavior attributes [Sun et al., 2021] and temporal filters [Tjandrasuwita et al., 2021] (further discussed in Section 3.2).

Scalability challenges also arise in other work on symbolic regression and interpretable machine learning. For instance, Cranmer et al. [2020] aim to learn exact mathematical relationships between variables by searching a space of mathematical expressions. As another example, Ustun and Rudin [2017] aim to learn optimized risk scores within the same modeling language used by clinicians, which leads to an NP-hard optimization problem that they solve using integer programming techniques.

3.4 Challenges of optimization of discrete and continuous space in neurosymbolic programs

NP relies on techniques from symbolic program synthesis to facilitate interpretable and verifiable searches over the scientific hypothesis space. However, programs are inherently symbolic, owing to their roots in mathematical logic. This makes modeling phenomena in the continuous domain challenging without modifying the way we interpret programs.

For instance, consider a simple program that is modelled by an if-then-else statement (`if condition do expr1 else do expr2`). The possible behaviors of `condition` are partitioned into two sets – True (`1`) or False (`0`). These sets evaluate to either `expr1` or `expr2` respectively. However, an NP approach requires reasoning over a *gradient* of possibilities to be differentiable. Discrete programs are inaccurate models for these applications. Specifically, in behavior classification, modeling the "attack" action using a symbolic if-then-else expression would partition the mouse's aggression into a binary set: either always attacking or not attacking at all. What makes more sense is to model "attacking" as a binomial distribution. This requires *relaxing* our symbolic if-then-else to account for a continuous gradient of probabilities from 0% to 100%.

Continuous relaxations. We approach the continuous program optimization problem of the symbolic domain by changing the semantics of the programming language. Specifically, work on *Smooth Interpretation* [Chaudhuri and Solar-Lezama, 2010] rewrites discrete functions using their closest smooth mathematical functions. Consecutively, an if-then-else statement would be rewritten as a hyperbolic tangent function with a high temperature. This smoothening is not restricted to a one-dimensional input space and specialized functions. In general, in higher dimensions, we can use Gaussian smoothing to smooth discontinuities. Such relaxations, in conjunction with other program analysis tools, allow gradient descent-based optimizers to converge to optimal programmatic models.

Continuous relaxations enable an approximate interface between neural networks and programming languages, which are essential in the NP framework. For example, in Houdini [Valkov et al., 2018], continuous relaxations enabled the construction of a functional programming language that admits

neural networks and higher-order functions. This construction facilitated high-level transfer of learned concepts across tasks in a life-long learning setting. In NEAR [Shah et al., 2020], the interface between neural networks and differentiable programs allowed measuring the performance of partial programs. This proved to be an $\epsilon-$admissible heuristic for synthesizing differentiable programs in the behavior analysis setting.

Smooth Interpretation allows positing a differentiable *approximation* for a non-differentiable program. This approximation error introduces a tradeoff between the output precision and optimal trainability of the model. Specifically, under-approximating the non-differentiable components might increase the precision of the differentiable program at the cost of retaining discontinuities in the optimization landscape and converging to a suboptimal model, and vice-versa.

3.5 Evaluating Interpretability

The main goal of interpretability is to obtain insights that are understandable and actionable to humans, and to assist scientists in their analysis workflow. The following are commonly described properties of explanations found in machine learning [Murphy, 2023], that have the potential to improve the interpretability and evaluation of NP workflows: **Compactness or sparsity:** Sparsity generally corresponds to some notion of smallness measurement (a few features or a few parameters); **Completeness:** To measure if the explanation includes all the relevant elements, higher-level concepts needed; **Stability:** To measure the extent that there are explanations similar for similar input; **Actionability:** To allow focusing on only aspects of the model that the user might be able to intervene on; **Modularity:** Explanation can be broken down into understandable parts. To study interpretability for NP models for science, we need datasets and benchmarks to quantify these different dimensions in many scientific contexts, which is still an open problem.

An example of an interpretable program is presented Figure 2b. This neurosymbolic program classifies the “sniff" action between two mice. An interpretation is that if the distance between two mice is small, they are doing a “sniff"; otherwise, they are only doing a “sniff" if the accelerations are small. Interpretability can also be expressed as an utility function or a form of abstraction Bowers et al. [2023] which minimizes the size of the corpus of a neurosymnbolic program.

3.6 Cross-Domain Benchmarking

While many individual fields of science have seen some successes through NP, consolidating underlying generalizable and cross-cutting insights remains another significant open challenge for the scientific and machine learning communities. Towards this, we propose to build initial benchmarks around low-dimensional spatiotemporal data, a setting where NP methods have demonstrated potential [Shah et al., 2020, Verma et al., 2018]. We believe that there are several benefits to gain from developing an NP benchmark for the ML and scientific communities: (1) systematic improvements across broad scientific use cases, (2) comprehensive model evaluations, instead of in domain-specific dimensions, (3) increased awareness of important scientific applications that have not received as much attention from the ML community.

Challenges of benchmarking NP for science. Interpreting programmatic structures requires expert domain knowledge, which can be expensive and time-consuming to obtain. In behavior analysis, evaluating learned programmatic structures requires interactions with experts in the behavioral science community. This imposes a major bottleneck on evaluating outputs. A standardized benchmark will make it easier for the community to convene and interact with a panel of experts. We believe that developing a benchmark for NP pipelines is integral to moving the NP field forward.

The space of NP models is broad. Each algorithm presents a unique methodology for encoding expert knowledge into the NP lifecycle. This requires comparing models on multiple evaluation metrics. However, not all NP algorithms can be systematically evaluated on the same set of metrics. For instance, certain classes of models use stochastic search to discover the programmatic structure and the programs found by such an approach may not be reproducible. Additionally, NP models might exhibit properties that do not have concrete evaluation metrics. For instance, classes of NP algorithms that exhibit robust reproducibility. That is, the model's outputs are reproducible with small perturbations to the input data. However, to the best of our knowledge, defining such a metric quantitatively and objectively remains an open challenge.

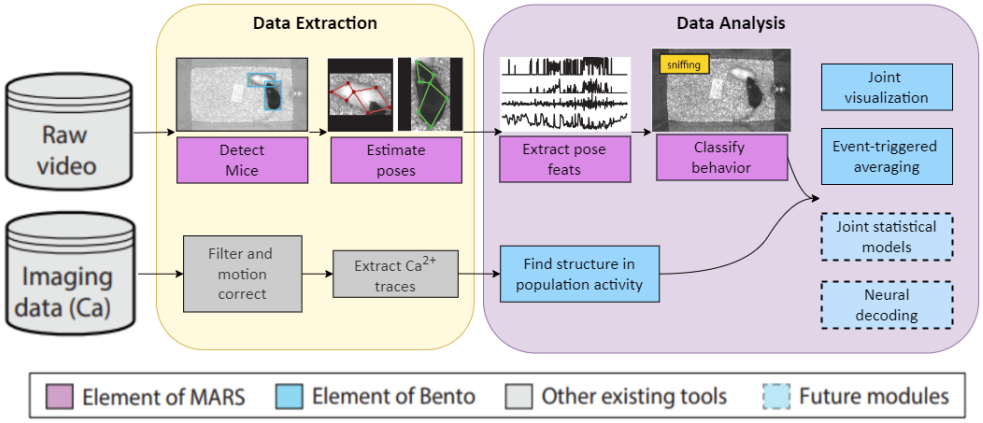

Figure 4: Functionalities of MARS and Bento [Segalin et al., 2021] in the behavior analysis pipeline.

The hardware requirements for learning neural representations and symbolic functions are orthogonal. Neural network training is GPU intensive, while program synthesis is CPU intensive. This increases the cost of computation and imposes a barrier to entry for aspiring NP researchers. NP benchmarks need to take the efficiency and performance of training and inference into account.

3.7 Cross-Domain Analysis Tools for Scientists

Importance of tools in science. User-friendly tools are important for facilitating the integration of ML models in real-world science workflows, but have not been well-explored for NP approaches. For example, numerous tools, based on statistical analysis and ML, have been developed to interface with scientists and facilitate behavior analysis from videos in Pereira et al. [2020]. These tools assist with much of the computational pipeline for behavior classification as outlined in Figure 4, and will often provide visual interfaces that visualize relevant raw data such as video, with model outputs, such as pose data and behavior [Segalin et al., 2021]. Enabling similar tools for NP approaches has the potential to benefit existing scientific workflows. For instance, integrating NEAR into a visual interface could provide scientists with a user-friendly way of generating differentiable programs and means of understanding the programs from NP pipelines. The parameters associated with programmatic primitives are likely to have a much more human interpretation [Tjandrasuwita et al., 2021] than those found in black-box neural networks.

Challenges of building NP tools. Domain expertise in science varies in structure, from behavioral attributes to visual or textual descriptions, to known dynamics of movement, to knowledge graphs, and generally differs across labs and domains. Furthermore, to measure progress, user evaluations are needed that could offer quantitative or qualitative evidence in NP workflows. Taking the first steps to realize and evaluate the effectiveness of NP algorithms through a human-computer interaction approach may not only improve the scientific pipeline, but also yield new algorithmic directions on combining NP with more traditional human-in-the-loop methods, such as active learning.

4 Conclusion

Neurosymbolic programming offers the promise to accelerate scientific discovery and optimize scientific discovery end-to-end. The benefits are in its ability to incorporate prior knowledge and the symbolic nature of the solutions, and is compatible with scientific workflows. However, challenges still remain in scalability and optimization stability of these approaches, comprehensive evaluations, and deployment in the form of tools. In this paper, we have demonstrated the opportunities and challenges of neurosymbolic programming in a concrete scientific application, behavior analysis. A key promise of neurosymbolic programming is to provide a set of unifying principles that connect and generalize solutions in interpretable machine learning and prior scientific literature. We invite the science and computer science communities to adopt these methods in their scientific workflow and to contribute to the research to advance NP techniques for science.

Funding. This project was supported by the National Science Foundation under Grant #1918839 "Understanding the World Through Code" http://www.neurosymbolic.org/

References

David J Anderson and Pietro Perona. Toward a science of computational ethology. *Neuron*, 84(1): 18–31, 2014.

Matthew Bowers, Theo X. Olausson, Catherine Wong, Gabriel Grand, Joshua B. Tenenbaum, Kevin Ellis, and Armando Solar-Lezama. Top-down synthesis for library learning. *Proc. ACM Program. Lang.*, (POPL), 2023.

Swarat Chaudhuri and Armando Solar-Lezama. Smooth interpretation. *ACM Sigplan Notices*, 45(6): 279–291, 2010.

Swarat Chaudhuri, Kevin Ellis, Oleksandr Polozov, Rishabh Singh, Armando Solar-Lezama, Yisong Yue, et al. Neurosymbolic programming. *Foundations and Trends® in Programming Languages*, 7 (3):158–243, 2021.

Miles Cranmer. Pysr: Fast & parallelized symbolic regression in python/julia, 2020.

Miles Cranmer, Alvaro Sanchez Gonzalez, Peter Battaglia, Rui Xu, Kyle Cranmer, David Spergel, and Shirley Ho. Discovering Symbolic Models from Deep Learning with Inductive Biases. In *Advances in Neural Information Processing Systems*, volume 33, pages 17429–17442. Curran Associates, Inc., 2020.

Guofeng Cui and He Zhu. Differentiable synthesis of program architectures. *Advances in Neural Information Processing Systems*, 34:11123–11135, 2021.

Sandeep Robert Datta, David J Anderson, Kristin Branson, Pietro Perona, and Andrew Leifer. Computational neuroethology: a call to action. *Neuron*, 104(1):11–24, 2019.

Jacob Devlin, Jonathan Uesato, Surya Bhupatiraju, Rishabh Singh, Abdel-rahman Mohamed, and Pushmeet Kohli. RobustFill: Neural Program Learning under Noisy I/O. In *Proceedings of the 34th International Conference on Machine Learning*, pages 990–998. PMLR, July 2017. ISSN: 2640-3498.

Finale Doshi-Velez and Been Kim. Towards a rigorous science of interpretable machine learning. *arXiv preprint arXiv:1702.08608*, 2017.

Kevin Ellis, Catherine Wong, Maxwell Nye, Mathias Sablé-Meyer, Lucas Morales, Luke Hewitt, Luc Cary, Armando Solar-Lezama, and Joshua B. Tenenbaum. DreamCoder: bootstrapping inductive program synthesis with wake-sleep library learning. In *Proceedings of the 42nd ACM SIGPLAN International Conference on Programming Language Design and Implementation*, PLDI 2021, pages 835–850, New York, NY, USA, June 2021. Association for Computing Machinery. ISBN 978-1-4503-8391-2.

Kevin Ellis, Adam Albright, Armando Solar-Lezama, Joshua B Tenenbaum, and Timothy J O'Donnell. Synthesizing theories of human language with bayesian program induction. *Nature communications*, 13(1):1–13, 2022.

Thomas Elsken, Jan Hendrik Metzen, and Frank Hutter. Neural architecture search: A survey. *The Journal of Machine Learning Research*, 20(1):1997–2017, 2019.

Eyrun Eyjolfsdottir, Steve Branson, Xavier P. Burgos-Artizzu, Eric D. Hoopfer, Jonathan Schor, David J. Anderson, and Pietro Perona. Detecting Social Actions of Fruit Flies. In David Fleet, Tomas Pajdla, Bernt Schiele, and Tinne Tuytelaars, editors, *Computer Vision – ECCV 2014*, Lecture Notes in Computer Science, pages 772–787, Cham, 2014. Springer International Publishing. ISBN 978-3-319-10605-2. doi: 10.1007/978-3-319-10605-2_50.

Joseph Garner. Mouse Ethogram – Stanford School of Medicine. URL `https://mousebehavior.org/`.

Noah D. Goodman, Tomer D. Ullman, and Joshua B. Tenenbaum. Learning a theory of causality. *Psychological Review*, 118(1):110–119, January 2011. ISSN 1939-1471. doi: 10.1037/a0021336.

Nastacia L Goodwin, Simon RO Nilsson, Jia Jie Choong, and Sam A Golden. Toward the explainability, transparency, and universality of machine learning for behavioral classification in neuroscience. *Current Opinion in Neurobiology*, 73:102544, 2022.

Sumit Gulwani. Automating String Processing in Spreadsheets using Input-Output Examples. In *PoPL'11, January 26-28, 2011, Austin, Texas, USA*, January 2011.

Minghao Guo, Veronika Thost, Beichen Li, Payel Das, Jie Chen, and Wojciech Matusik. Data-efficient graph grammar learning for molecular generation. In *International Conference on Learning Representations*, 2021.

Jon Kleinberg, Himabindu Lakkaraju, Jure Leskovec, Jens Ludwig, and Sendhil Mullainathan. Human decisions and machine predictions. *The quarterly journal of economics*, 133(1):237–293, 2018.

Pang Wei Koh, Shiori Sagawa, Henrik Marklund, Sang Michael Xie, Marvin Zhang, Akshay Balsubramani, Weihua Hu, Michihiro Yasunaga, Richard Lanas Phillips, Irena Gao, et al. Wilds: A benchmark of in-the-wild distribution shifts. In *International Conference on Machine Learning*, pages 5637–5664. PMLR, 2021.

Ali Sinan Koksal, Yewen Pu, Saurabh Srivastava, Rastislav Bodik, Jasmin Fisher, and Nir Piterman. Synthesis of biological models from mutation experiments. In *Proceedings of the 40th annual ACM SIGPLAN-SIGACT symposium on Principles of programming languages*, pages 469–482, 2013.

Tessa A Lau and Daniel S Weld. Programming by demonstration: An inductive learning formulation. In *Proceedings of the 4th international conference on Intelligent user interfaces*, pages 145–152, 1998.

Xubo Leng, Margot Wohl, Kenichi Ishii, Pavan Nayak, and Kenta Asahina. Quantifying influence of human choice on the automated detection of Drosophila behavior by a supervised machine learning algorithm. *PLoS ONE*, 15(12):e0241696, December 2020. ISSN 1932-6203. doi: 10.1371/journal.pone.0241696.

Alexander Mathis, Pranav Mamidanna, Kevin M. Cury, Taiga Abe, Venkatesh N. Murthy, Mackenzie W. Mathis, and Matthias Bethge. Deeplabcut: markerless pose estimation of user-defined body parts with deep learning. *Nature Neuroscience*, 2018.

Thomas McGrath, Andrei Kapishnikov, Nenad Tomašev, Adam Pearce, Demis Hassabis, Been Kim, Ulrich Paquet, and Vladimir Kramnik. Acquisition of chess knowledge in alphazero. *arXiv preprint arXiv:2111.09259*, 2021.

Kevin P. Murphy. *Probabilistic Machine Learning: Advanced Topics*. MIT Press, 2023.

Talmo D Pereira, Joshua W Shaevitz, and Mala Murthy. Quantifying behavior to understand the brain. *Nature neuroscience*, 23(12):1537–1549, 2020.

Talmo D Pereira, Nathaniel Tabris, Arie Matsliah, David M Turner, Junyu Li, Shruthi Ravindranath, Eleni S Papadoyannis, Edna Normand, David S Deutsch, Z. Yan Wang, Grace C McKenzie-Smith, Catalin C Mitelut, Marielisa Diez Castro, John D'Uva, Mikhail Kislin, Dan H Sanes, Sarah D Kocher, Samuel S-H, Annegret L Falkner, Joshua W Shaevitz, and Mala Murthy. Sleap: A deep learning system for multi-animal pose tracking. *Nature Methods*, 19(4), 2022.

Cynthia Rudin. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature Machine Intelligence*, 1(5):206–215, 2019.

Nicolae Sapoval, Amirali Aghazadeh, Michael G Nute, Dinler A Antunes, Advait Balaji, Richard Baraniuk, CJ Barberan, Ruth Dannenfelser, Chen Dun, Mohammadamin Edrisi, et al. Current progress and open challenges for applying deep learning across the biosciences. *Nature Communications*, 13(1):1–12, 2022.

Cristina Segalin, Jalani Williams, Tomomi Karigo, May Hui, Moriel Zelikowsky, Jennifer J Sun, Pietro Perona, David J Anderson, and Ann Kennedy. The mouse action recognition system (mars) software pipeline for automated analysis of social behaviors in mice. *Elife*, 10:e63720, 2021.

Ameesh Shah, Eric Zhan, Jennifer Sun, Abhinav Verma, Yisong Yue, and Swarat Chaudhuri. Learning Differentiable Programs with Admissible Neural Heuristics. In *Advances in Neural Information Processing Systems*, volume 33, pages 4940–4952. Curran Associates, Inc., 2020.

Jennifer J Sun, Ann Kennedy, Eric Zhan, David J Anderson, Yisong Yue, and Pietro Perona. Task programming: Learning data efficient behavior representations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2876–2885, 2021.

Megan Tjandrasuwita, Jennifer J Sun, Ann Kennedy, Swarat Chaudhuri, and Yisong Yue. Interpreting expert annotation differences in animal behavior. *CV4Animals Workshop at CVPR*, 2021.

Albert Tseng, Jennifer J Sun, and Yisong Yue. Automatic synthesis of diverse weak supervision sources for behavior analysis. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022.

Berk Ustun and Cynthia Rudin. Optimized risk scores. In *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1125–1134, 2017.

Lazar Valkov, Dipak Chaudhari, Akash Srivastava, Charles Sutton, and Swarat Chaudhuri. Houdini: Lifelong learning as program synthesis. *Advances in Neural Information Processing Systems*, 31, 2018.

Abhinav Verma, Vijayaraghavan Murali, Rishabh Singh, Pushmeet Kohli, and Swarat Chaudhuri. Programmatically interpretable reinforcement learning. In *International Conference on Machine Learning*, pages 5045–5054. PMLR, 2018.

Catherine Wong, Kevin M Ellis, Joshua Tenenbaum, and Jacob Andreas. Leveraging language to learn program abstractions and search heuristics. In *International Conference on Machine Learning*, pages 11193–11204. PMLR, 2021.

Eric Zhan, Jennifer J Sun, Ann Kennedy, Yisong Yue, and Swarat Chaudhuri. Unsupervised learning of neurosymbolic encoders. *arXiv preprint arXiv:2107.13132*, 2021.