Neurosymbolic Learning for Science

Anonymous Author(s)
Affiliation
Address
email

Abstract

Neurosymbolic Programming (NP) techniques have the potential to accelerate scientific discovery across fields. These models can interface with symbolic domain knowledge from scientists, such as prior knowledge and experimental context, to produce interpretable outputs. NP models combine neural and symbolic components to learn complex patterns and representations from data, using high-level concepts or known constraints. Here, we identify opportunities and challenges between current NP models and scientific workflows, with real-world examples from behavior analysis in science. We define concrete next steps to move the NP for science field forward, 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 machine learning techniques because of the need to ensure that any new theories that are formulated from data are consistent with our prior scientific knowledge, as well as to enable scientists to reason about the implications of new hypotheses and experimental designs.

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

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

We argue that NP has an important role to play in automating scientific discovery in a way that builds on existing knowledge and leads to finding new scientific insights. NP allows for a new line of attack on this challenge: represent scientific hypotheses as programs in a *Domain Specific Language* (DSL) and use neurosymbolic program synthesis to automatically discover these programs. An important benefit of this approach is that a user can incorporate complex forms of prior knowledge (for example, known features and constraints) into the design of the DSL. The NP learning algorithms can then follow principles of reasoning found in classical science. Also, models learned this way are often similar to the code that human domain experts write during manual scientific modeling. Collectively,

Submitted to NeurIPS 2021 AI for Science Workshop, Attention Track.

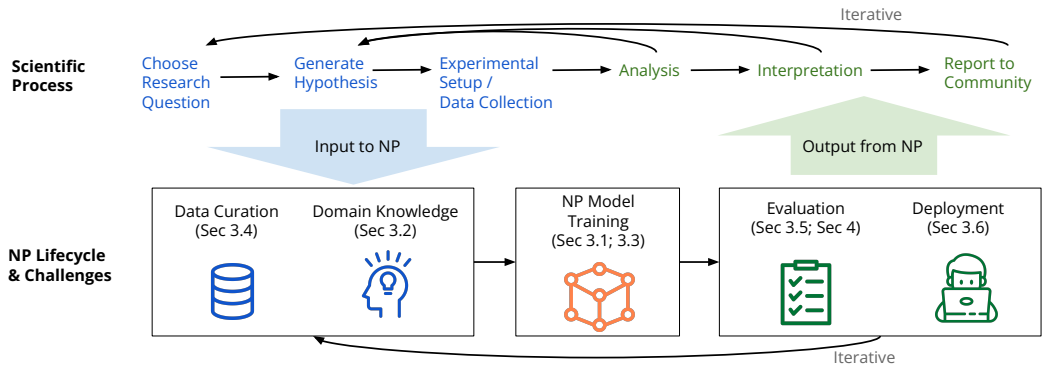

Figure 1: Synergy between the scientific and neurosymbolic programming workflow.

these characteristics enable an interactive process through which a artificial system and a human expert collaborate on evidence-based reasoning and the discovery of new scientific facts.

In this paper, we use behavior analysis as a concrete example to demonstrate the challenges and opportunities in this space. We start with an introduction to NP in Section 2. While NP has the potential to improve the scientific process end-to-end (Figure 1), there are challenges that exist at the intersection of neurosymbolic learning and science, which we outline in Section 3, with a focus on evaluation and benchmarking in Section 4.

Behavior analysis as running example. Many of the underlying challenges in behavior analysis are shared in other scientific domains, from monitoring vital signs, to modeling physical systems, to dynamics of chemical reactions. Fundamentally, behavioral data is a type of time-series data, and this data type is common across the sciences as many phenomena are spatiotemporal in nature. Behavior analysis illustrates common challenges with scientific data analysis. Scientists generally study behaviors that are rare with imbalanced data. Datasets vary across labs, organisms, and experimental setups, and are often noisy and imperfect. Finally, automatic behavior quantification is becoming increasingly crucial in a range of scientific fields, such as neuroscience, ecology, biology, and healthcare monitoring. As automatic behavior analysis and neurosymbolic learning are both developing fields, there are many exciting opportunities to explore at their intersection.

Background of 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. In this setting, a common analysis method 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]. We need methods that facilitate scientists' extraction of biological insights from pose trajectories or directly from recorded videos of animals.

Existing challenges in behavior analysis. Similar to other fields of science, data collection and annotation is expensive for behavioral experiments, and raw data is often imperfect and noisy. The process of data analysis is time-consuming and expensive, since specialized domain expertise is required for identifying behaviors-of-interest and extracting knowledge. Models need to efficiently interface 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, 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. Furthermore, when black-box models are used for studying behavior, it is difficult to diagnose errors and interpret the model output. NP models have the potential to produce symbolic descriptions of behavior (Figure 3), which enables experts to connect model interpretations with other parts of the behavior analysis workflow. For example, for describing behavioral differences across different strains of mice. Finally, to enable the use of NP models

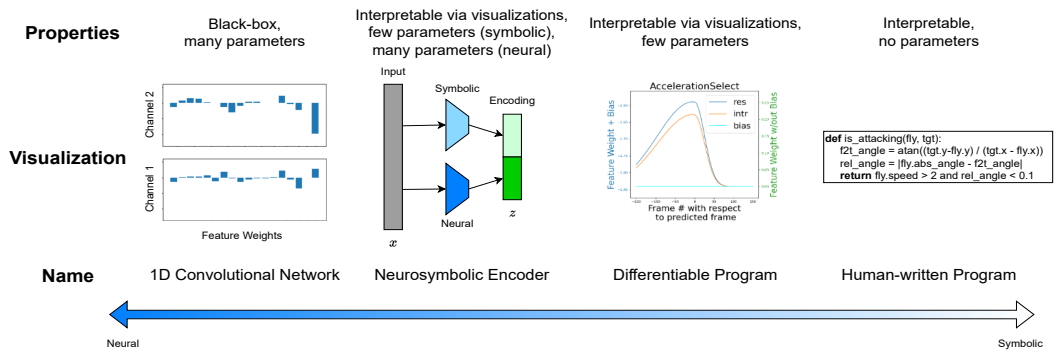

Figure 2: Space of neurosymbolic programming models in behavior analysis.

in real-world science pipelines, these models must be scalable and produce robustly reproducible interpretations, which is a shared challenge across domains in science.

2 Neurosymbolic Programming techniques

In the NP framework, the objective is to learn *neurosymbolic programs*, which incorporate latent representations computed by neural networks and symbols that explicitly capture pre-existing human knowledge, and connect these elements using rich architectures that mirror classical 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.

The range of NP methods vary in the degree to which they use neural vs. symbolic reasoning (Figure 2). The two ends of the spectrum correspond to purely neural—a 1D convolutional network—and purely symbolic techniques—a human written-program used for weak supervision [Tseng et al., 2022] —respectively, whereas the techniques close to the center are both neurosymbolic. Specifically, 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].

For an example of the strengths and weaknesses of each model, assume that we have a scientific hypothesis to test on a dataset. On one 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 and 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 other 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.

Differentiable programs. While deep learning models are powerful choices when given sufficient high-quality data, they are generally prone to overfitting and difficult to interpret. To address some of these weaknesses, NP methods have been applied in a variety of machine learning paradigms to leverage the advantages of neural networks and programs. For example, [Shah et al., 2020] introduce a Neural Admissible Relaxation (NEAR) technique in a supervised behavior classification setting. Their goal is to synthesize differentiable programs, a powerful way of engineering systems with both discrete architectures and continuous parameters. On a high level, a differentiable program α_θ for a programming language L is defined as a composition of functions $\sigma \in L$ such that the parameters of the function $\sigma_\theta \in \mathbb{R}$ 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 from Shah *et al.*'s framework to be fully differentiable. [Shah et al., 2020] claim that programs synthesized by NEAR compete with the performance of a black-box recurrent neural network while being much more human-interpretable. Figure 3 is an example program discovered by NEAR for behavior analysis.

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

Figure 3: A program generated by a neurosymbolic programming framework [Shah et al., 2020].

The program in Figure 3 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.

Though NEAR and follow-up work [Cui and Zhu, 2021] demonstrate great promise in terms of accuracy on behavior classification tasks, their programs are still written in a fairly generic DSL, but the results may be difficult to interpret by domain experts. [Tjandrasuwita et al., 2021] directly addresses this challenge in the context of explaining the difference between different sets of behavior expert annotations. Their primary contribution is to 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

The techniques outlined above demonstrate some benefits of the NP framework. However, there are still major gaps between current NP approaches and practical use cases for scientists. There exist challenges across the life-cycle of NP algorithm development, deployment, and its synergy with the scientific workflow, from dealing with noisy input data to scalable model training, to evaluation and tool building and deployment (see 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 science.

3.1 Scalability challenge

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-level similar to performing machine learning (ML) with programs as models, a key difference is that ML approaches hinge 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, and similar to other scientific data, contains noisy labels. 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. The differentiable program synthesis library (dPads) [Cui and Zhu, 2021] addresses the computational efficiency challenge of having to synthesize discrete program architectures and perform gradient optimization at runtime. To do so, dPads trades off computation with the cost of storing many more parameters in memory, with heuristics to mitigate memory usage. However, training fully neural models is often more efficient than NP models, which requires searching through an exponentially large space of symbolic architectures. 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 the 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 symbolic regression [Cranmer et al., 2020], where the goal is to learn exact mathematical relationships between variables by searching a space of mathematical expressions. In this case, the regression scales poorly with the number of interactions between variables. [Cranmer et al., 2020] posit a solution that separately models the variable to variable interaction and the variable transformation through a graph neural network. Distilling the neural networks to symbolic functions is then tractable via symbolic regression.

3.2 Encoding and Learning Domain Knowledge

The success of NP techniques often depends on how a DSL is defined. However, it may be not only time-consuming but also unclear about how to handcraft domain-specific components that work best in a scientific context. As such, works in *library learning* propose algorithms that consolidate common patterns in successful programs generated in program synthesis and 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 not only synthesizing interpretable programs but also inducing high-level DSL's, given low-level, generic primitives. These models have been applied to tasks such as physics equation discovery [Ellis et al., 2021]. Another example that demonstrates the potential of library learning is generative modeling in molecular chemistry [Guo et al., 2022]. The proposed library learning method is capable of handling highly data-limited settings, often found in science due to the cost of obtaining high-quality labels from experts. Library learning methods also have the advantage of enabling greater possibilities for including domain-specific knowledge, as all components in the library are written in terms of human-interpretable primitives.

Challenges of library learning for science. While DreamCoder and molecular generation have shown promise in their respective domains, 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 frames of videos with noisy data. In addition, it is highly expensive to collect behavior annotations across many 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 situations found in behavior analysis [Leng et al., 2020, Segalin et al., 2021].

Another challenge is that domain experts still need to interpret library components learned by the NP 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. In addition to the challenge of scaling library learning to real behavior analysis settings, building a smooth interface between expertise in science, program synthesis and library learning methods, found in NP, would likely require significant engineering and research efforts (Section 3.6).

3.3 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 discontinuous, 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 modeled 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, scientific applications require 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.

Continuous relaxations. We approach the continuous program optimization problem 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 shifted sigmoid function with a high temperature. This smoothening is not restricted to a one dimension 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, that are essential for the NP framework. For example, in Houdini [Valkov et al., 2018], continuous relaxations enabled constructing 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.

However, in both of these methods, Smooth Interpretation allowed the authors to posit 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.

3.4 Dealing with raw, noisy, and imperfect data

Scientific data provides an opportunity to study NP models with imperfect data in real-world conditions, such as with missing data, experiment noise, and distribution shifts.

Current NP models are not typically developed or evaluated in these settings, but by incorporating prior knowledge and known constraints, 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 learned models to generate weak supervision for the full dataset as in [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 label 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 frames. 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 performance and scalability before widespread deployment.

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]. Recent work [Zhan et al., 2021] has tackled this using NP for behavior analysis and demonstrated that integrating domain knowledge results in more meaningful discovered categories compared to fully neural methods. However, these methods have not been widely developed. For broader impact across science, in addition to the algorithmic challenges discussed in previous sections, future research work needs to be robust to variations in experimental noise and produce reproducible interpretations of discovered structures in the data that are useful for scientists.

Distribution Shifts. Distribution shifts are common in real-world scientific 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

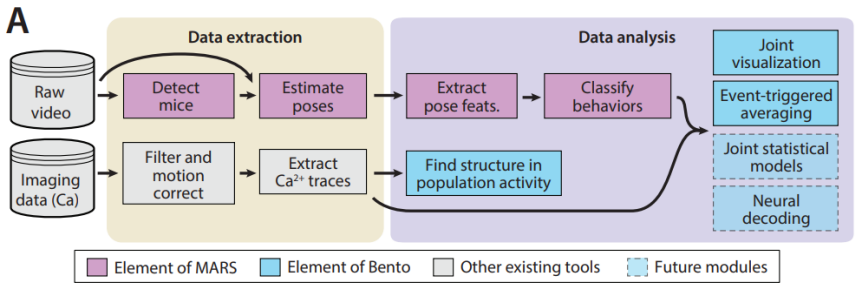

Figure 4: Functionalities of MARS and Bento [Segalin et al., 2021] in the behavior analysis pipeline.

tackle this challenge. For example, in behavior analysis, when the physical behavioral area changes in size or shape, the relative size of mice also changes. This causes errors in behavior classifiers trained in a previous known area, but NP programs can be adjusted to the new task accordingly.

Future work to improve model interpretability and distributional robustness of NP approaches is important for facilitating the integration of these models in real-world science workflows.

3.5 Evaluation

Similar to other ML models, evaluation in NP is crucial for measuring progress in model development and understanding gaps in model performance. This is also important before real-world deployment in scientific discovery pipelines. As an important next step in reducing the gap between NP models and science, we discuss evaluation and benchmarking in more detail in Section 4, and focus on interpretability in this subsection.

The main goal of model interpretability is to obtain model insights that are understandable and actionable to humans, to assist scientists in their analysis workflow. The following are commonly described properties of explanations that can be found in the machine learning framwork [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 include 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 that is able to quantify these different dimensions in a range of scientific contexts.

3.6 Cross-Domain Analysis Science Tools

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 (as reviewed 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 not only provide scientists with a user-friendly way of generating differentiable programs, but also greater means of understanding the programs that NP pipelines generate. 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. Though we foresee various benefits of integrating NP algorithms into scientific pipelines, following through with this vision will take major engineering efforts to design a suitable interface between scientists and the algorithmic backends. Domain expertise vary 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, we need user studies that substantiate the above intuitions with either quantitative or qualitative evidence. Taking the first steps to realize and evaluate the effectiveness of NP algorithms via human-computer interaction 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 Benchmarking

Much of the innovations in ML algorithms for scientific applications have focused on single domains, for example behavioral neuroscience [Sun et al., 2021], chemistry [Guo et al., 2022], or physics [Cranmer et al., 2020]. While many individual fields of science have seen successes through NP, consolidating underlying insights that are generalizable across domains in science remains a significant open challenge. To encourage the development of new methods that work across a diverse set of scientific problems and enable comparisons between these methods, we propose to build a benchmark featuring low-dimensional spatiotemporal data, a setting where NP methods have been demonstrated to be successful [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) model improvements across a variety of metrics, 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. Constructing a benchmark presents unique challenges. First, 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 to interpret these structures. This imposes a physical and manual 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.

Second, the space of NP models is broad. Each algorithm presents a different methodology for encoding expert knowledge into the machine learning lifecycle. This requires comparing models on multiple evaluation metrics. However, not all algorithms can be systematically evaluated on the same set of metrics. For instance, certain classes of NP models use stochastic search to discover the programmatic structure and the programs found by such models may not be exactly 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 problem.

Finally, 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 consider training and inference efficiency and performance into account.

5 Conclusion

Neurosymbolic programming offers clear promise to accelerate scientific discovery. The benefits are in its ability to incorporate prior knowledge and the symbolic nature of the solutions. This approach is compatible with current scientific workflows and thus has the potential to optimize scientific discovery end-to-end. 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 opportunities and challenges of NP in a concrete scientific application, behavior analysis. A key promise of NP 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.

References

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

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, and Yisong Yue. Neurosymbolic Programming. *Foundations and Trends in Programming Languages*, 7(3):158–243, December 2021. ISSN 2325-1107. doi: 10.1561/2500000049. URL https://doi.org/10.1561/2500000049.

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. URL https://proceedings.neurips.cc/paper/2020/hash/c9f2f917078bd2db12f23c3b413d9cba-Abstract.html.

Guofeng Cui and He Zhu. Differentiable Synthesis of Program Architectures. In *Advances in Neural Information Processing Systems*, volume 34, pages 11123–11135. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper/2021/hash/5c5a93a042235058b1ef7b0ac1e11b67-Abstract.html.

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. URL https://proceedings.mlr.press/v70/devlin17a.html. 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. doi: 10.1145/3453483.3454080. URL https://doi.org/10.1145/3453483.3454080.

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.

Sumit Gulwani. Automating String Processing in Spreadsheets using Input-Output Examples. In *PoPL'11, January 26-28, 2011, Austin, Texas, USA*, January 2011. URL https://www.microsoft.com/en-us/research/publication/automating-string-processing-spreadsheets-using-input-output-examples/.

Minghao Guo, Veronika Thost, Beichen Li, Payel Das, Jie Chen, and Wojciech Matusik. Data-Efficient Graph Grammar Learning for Molecular Generation. March 2022. URL https://openreview.net/forum?id=l4IHywGq6a.

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*, IUI '99, pages 145–152, New York, NY, USA, December 1998. Association for Computing Machinery. ISBN 978-1-58113-098-0. doi: 10.1145/291080.291104. URL https://doi.org/10.1145/291080.291104.

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. URL https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7743940/.

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. URL https://www.nature.com/articles/s41593-018-0209-y.

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. URL probml.ai.

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.

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. URL https://proceedings.neurips.cc/paper/2020/hash/342285bb2a8cadef22f667eeb6a63732-Abstract.html.

Jennifer J. Sun, Ann Kennedy, Eric Zhan, David J. Anderson, Yisong Yue, and Pietro Perona. Task Programming: Learning Data Efficient Behavior Representations. pages 2876–2885, 2021. URL https://openaccess.thecvf.com/content/CVPR2021/html/Sun_Task_Programming_Learning_Data_Efficient_Behavior_Representations_CVPR_2021_paper.html.

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 *Proceedings of the 35th International Conference on Machine Learning*, pages 5045–5054. PMLR, July 2018. URL `https://proceedings.mlr.press/v80/verma18a.html`. ISSN: 2640-3498.

Catherine Wong, Kevin M. Ellis, Joshua Tenenbaum, and Jacob Andreas. Leveraging Language to Learn Program Abstractions and Search Heuristics. In *Proceedings of the 38th International Conference on Machine Learning*, pages 11193–11204. PMLR, July 2021. URL `https://proceedings.mlr.press/v139/wong21a.html`. ISSN: 2640-3498.

Eric Zhan, Jennifer J. Sun, Ann Kennedy, Yisong Yue, and Swarat Chaudhuri. Unsupervised Learning of Neurosymbolic Encoders, July 2021. URL `http://arxiv.org/abs/2107.13132`. arXiv:2107.13132 [cs].