LEARNING THE VOCABULARY OF MATHEMATICS FROM DATA

Anonymous authors
Paper under double-blind review

ABSTRACT

Symbolic regression (SR) searches for formulas built from a set of functions fixed before the search, its vocabulary. When a law needs a function outside that set, the search can only return a longer expression that approximates the missing function, yet one rarely knows in advance which functions a law will need. We show that the vocabulary of mathematics can be learned from data. We add to an SR library one learnable operator, a spline or a small neural network that the search can insert anywhere a built-in function can go; all its occurrences share one parameter vector, fitted together with the numeric constants. On three laws that contain Bessel's J_0, with only elementary functions in the library, the learned operator matched J_0 in 13 of 20 runs. In most of these runs the three selected formulas had 12 nodes in total, as many as a search whose library contains J_0, while elementary search, even with four times as much time, returned formulas of about 50 nodes with higher error. The runs whose operator matched J_0 were exactly those with the lowest training loss. We prove conditions under which several laws sharing one operator determine it, up to the scale and offset of its output, even when no single law does. When we delete familiar functions from the library, the operator recovers the square root in about two thirds of runs, and addition or subtraction in 21 of 50 runs when it is a neural network. On recordings of a real pendulum, three of ten runs select the energy law with a cosine-shaped operator. Functions such as $\sqrt{\cdot}$ and $\sin$ became standard because each describes many laws compactly, and a search that fits new functions from data could find useful functions that do not yet have names.

1 INTRODUCTION

> *"We cannot improve the language of any science without at the same time improving the science itself; neither can we, on the other hand, improve a science, without improving the language or nomenclature which belongs to it."* Lavoisier (1790), *Elements of Chemistry*

Symbolic regression (SR) searches for closed-form formulas built from a list of functions chosen before the search begins, which we call its vocabulary (Koza, 1992; Cranmer, 2023). How parsimonious a formula can be depends on this vocabulary, yet the right vocabulary is rarely known ahead of time. When a law needs a function the list lacks, the search often fails, or instead returns a longer expression that approximates the missing function with the functions it has. For example, Schmidt & Lipson (2009) removed sine and cosine from the building blocks of their search, and SR wrote the conserved energy of a pendulum as $\omega^2 + k_1\theta^2 - k_2\theta^4$, whose angle terms are the Taylor polynomial of the cosine term. On the recorded angles this polynomial fits as well as the cosine does, so the data gives the search no reason to prefer the true law. In this paper, we show that this vocabulary can be learned during the search.

Many standard functions were used in calculations long before they were named. Hipparchus tabulated chords for astronomical calculations around 140 BC, and Aryabhata's half-chord tables from 499 AD compute the sine, whose Sanskrit name *jya* passed through Arabic into the Latin *sinus* (O'Connor & Robertson, 1996). Napier published logarithms in 1614 so that multiplying, dividing and extracting roots of large numbers could be done largely by addition and subtraction (O'Connor & Robertson, 1998). Jacob and Daniel Bernoulli, Euler and Lagrange had met special cases of Bessel's functions before Bessel developed them more fully in 1824, in a study of planetary pertur-

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

bations (O'Connor & Robertson, 1997). In each case the same function appeared in many calculations and was then given a name and tables. A search with a fixed list of functions cannot add a function in this way.

Our method adds to an SR library one learnable operator φ_θ (Fig. 1, Section 2). The search can insert φ_θ wherever it can insert $\sin$ or $\sqrt{\cdot}$. Within a candidate, every occurrence of φ_θ, in every equation fitted together, uses one parameter vector θ, and θ is fit by gradient-based optimization together with the constants. We gave this search three laws that contain Bessel's J_0 and a library of elementary functions only. The learned operator matched J_0 in 13 of 20 runs. In nine of them the search selected exactly the three target laws with φ_θ in place of J_0, $\varphi_\theta(x_1)+x_2$, $\varphi_\theta(x_1+x_2)$ and $x_1\varphi_\theta(x_2)$. These laws have no constants, so the search found both the structure and the function, in the same 12 nodes as a search whose library contains J_0 (Section 3, Fig. 16). Elementary search given four times as much time approximated parts of J_0 with sines and square roots, but using about 50 nodes and with about a hundred times the error. The runs whose operator matched J_0 were exactly the runs with the lowest training loss, so on this task keeping the lowest-loss run of a few restarts returns J_0 without knowing the answer.

Why should a flexible function become J_0 and not some other function that fits? In some problems a list of required properties is satisfied by only one function. Shannon's entropy is the only measure of uncertainty, up to a constant factor, with three natural properties, and so information is measured with the logarithm (Shannon, 1948). Here the required properties are imposed by the equations that contain φ_θ. For a single law in one variable, such as $\sqrt{x}+x$, the shortest exact solution uses the whole law as φ_θ, so nothing forces φ_θ to be the missing function. A law with constants can also compensate some changes of φ_θ by changing its constants. We prove that several laws sharing φ_θ can determine it, up to the scale and offset of its output, even when no single law does. It is enough that every change of φ_θ is detected by at least one law after its constants are refitted, and that no different set of formulas of the same or smaller size can reproduce how the laws change with their parameters (Section 4).

To see whether the operator can supply a function the library is missing, we delete a known function, the square root, addition and subtraction, or sine and cosine, and test whether φ_θ recovers it (Section 5). Unlike the single J_0 task, this covers many laws at once, including recordings of real physical systems. Each run is scored by its fit, by whether φ_θ matches the deleted function on held-

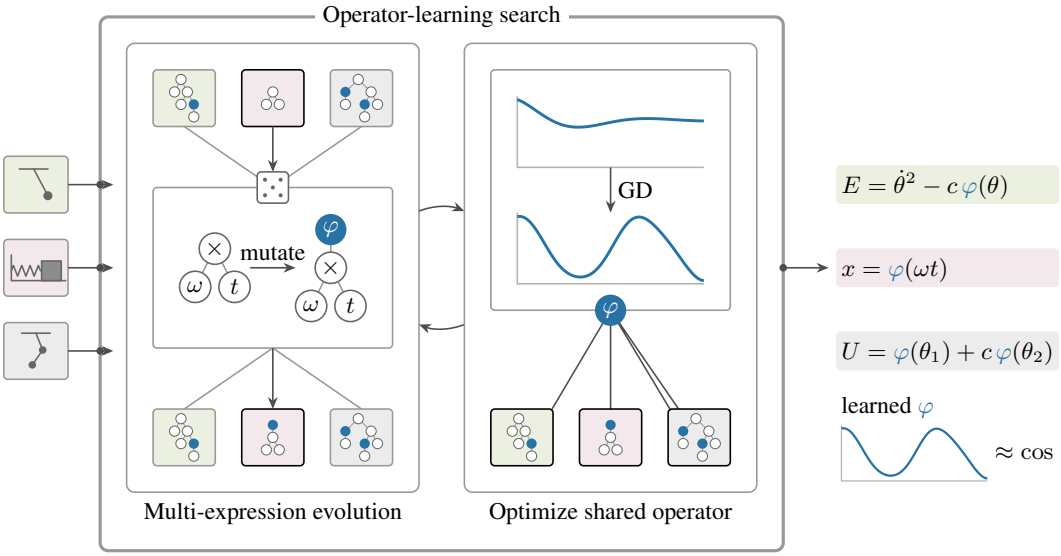

Figure 1: Overview of the method (schematic). Data from several systems enter one joint search. The search evolves one expression per system by mutation (left) and fits the parameters of a single learnable operator φ_θ, shared by every expression, by gradient-based optimization together with the constants (right). Each returned law uses the same learned φ_θ, which here resembles $\cos$. The systems and laws shown are illustrative; Sections 3 and 5 report the experiments.

out data, and by whether the selected equations are the target laws. Across ten equation sets the operator matches the square root in about two thirds of runs and addition or subtraction in 21 of 50 runs with the neural-network operator, and the library without the deleted function never returns a target law. Runs whose operator does not match still fit well in most cases, and on addition some fit better than the full library. On recordings of a real pendulum, three of ten runs select the energy law with a cosine-shaped operator.

2 A Learnable Operator in the Library

SR searches over expressions generated by a grammar with one nonterminal $\langle \mathrm{expr} \rangle$ and productions $\langle \mathrm{expr} \rangle \to u \langle \mathrm{expr} \rangle$ for unary operators $u \in \mathcal{U}$ (for example $\sin$ and $\sqrt{\cdot}$), $\langle \mathrm{expr} \rangle \to b \langle \mathrm{expr} \rangle \langle \mathrm{expr} \rangle$ for binary operators $b \in \mathcal{B}$ (for example $+$ and $\times$), and $\langle \mathrm{expr} \rangle \to t$ for inputs and constants $t \in \mathcal{T}$. It seeks an expression $f : \mathbb{R}^d \to \mathbb{R}$ with low loss on the data and small size.

We add a set $\mathcal{L}$ of learnable operator symbols. Each $\ell \in \mathcal{L}$ has an arity $a(\ell) \in \{1, 2\}$ and a parameter vector θ_ℓ, and contributes the production

$$\langle \mathrm{expr} \rangle \to \ell_{\theta_\ell} \langle \mathrm{expr} \rangle^{a(\ell)}.$$

All occurrences of ℓ share one parameter vector θ_ℓ. Every occurrence of ℓ evaluates the same θ_ℓ, and when several equations $f_1, \ldots, f_M$ are fitted jointly as one candidate (its *inner expressions*) they all share it. In all our experiments $\mathcal{L}$ holds one symbol, written φ_θ. Because every occurrence uses the same parameters, each candidate has one learned function, and each run reports the one in its selected candidate, which we can compare with $\sqrt{\cdot}$ (Fig. 4 in Appendix A illustrates the method).

2.1 Two operator families

Any differentiable function approximator can serve as φ_θ. We use two in our experiments.

B-spline operator (BOP). For arity a the operator is a tensor-product B-spline of degree d,

$$\mathrm{BOP}_a(u_1, \ldots, u_a) \;=\; \sum_{i_1, \ldots, i_a} C_{i_1 \cdots i_a} \prod_{j=1}^{a} B_{i_j, d}(u_j),$$

whose parameters θ are the control points C. The $k + d$ basis functions $B_{i,d}$ per argument use fixed, uniformly spaced knots over $[0, k]$, and outside that interval the spline continues as the polynomial of its first or last knot span. With $d = 3$ and $k = 30$ a unary operator has 33 control points and a binary one $33^2 = 1089$. All control points start at 1, so the initial operator is the constant 1.

MLP operator (NOP). For arity a the operator is a one-hidden-layer perceptron with H rectified units, $\mathrm{NOP}_a(u) = W_2 \,\mathrm{relu}(W_1 u + b_1) + b_2$ for $u \in \mathbb{R}^a$, with $\theta = (W_1, b_1, W_2, b_2)$. At $H = 16$ it has 49 parameters for a unary and 65 for a binary operator, initialized from one fixed random draw shared by all runs.

2.2 Joint optimization of structure, constants and operators

Evolutionary SR alternates discrete moves on the trees (mutation, crossover, and selection that keeps a Pareto front of loss against size) with a continuous step that fits the constants of each candidate and, in our method, the parameters of the learnable operator. The scalar constants of all inner expressions and the parameters θ are stacked into one vector and optimized together by L-BFGS with a backtracking line search, for a fixed number of iterations per call (20 on the synthetic sets and 3 on the physical tasks), and the result is kept if it lowers the loss. Derivatives with respect to the constants and θ are computed through the trees and φ_θ by reverse-mode automatic differentiation. When a candidate is mutated or crossed over, its θ is copied with it, so later optimization continues from the inherited parameters instead of reinitializing them.

The expression size counts a learnable node as one node, although it has 33 to 1089 parameters. To check that this does not favor φ_θ, Appendix D.6 also ranks expressions by the Bayesian information criterion (BIC), charging every constant and every operator parameter. Implementation details are in Appendix A.

3 LEARNING A FUNCTION THE LIBRARY NEVER HAD

Bessel's J_0 cannot be written with finitely many elementary operations, so a search without it can only approximate it. We generate data from three laws that use J_0 in different places, $y_1 = J_0(x_1) + x_2$, $y_2 = J_0(x_1 + x_2)$ and $y_3 = x_1 J_0(x_2)$, with x_1, x_2 uniform on $[0, 6]$ and 1000 training points. The three laws are fitted jointly, so a shared φ_θ has to represent J_0 on $[0, 12]$. The library holds unary $\sqrt{\cdot}$, exp, log, sin, cos and binary $+$, $-$, $\times$, $/$.

We compare four versions of the search, which we call arms, that differ only in their library, and run each with 20 seeds. ORACLE adds J_0 to the library, ABLATED uses the elementary library alone, and +BOP and +NOP add a unary spline or MLP φ_θ. All four use the settings of the synthetic sets, 100 iterations and at most 30 nodes over the three laws. Because the learnable arms take longer, we also check whether more search would have helped the elementary library with an extended ABLATED run that gets twice the size limit and about four times the spline's wall time. We say that a run "learns" J_0 when the affine fit of φ_θ to J_0 (the fit after the best scale and offset of the output), on a grid over the arguments φ_θ received, reaches $R^2 \geq 0.999$ and is at least ten times closer than the best of six elementary look-alikes, such as $\sin(x)/x$.

The spline learns J_0 in 13 of 20 runs (Table 18). Twelve of these runs apply φ_θ to arguments covering $[0, 11.6]$; seed 1007 applies it only to arguments up to 6, so it matches J_0 only there. In nine of the thirteen the search selects $\varphi_\theta(x_1) + x_2$, $\varphi_\theta(x_1 + x_2)$ and $x_1 \varphi_\theta(x_2)$, which are exactly the target laws with φ_θ in place of J_0. These laws contain no constants, so fitting them exactly requires both the right structure and $\varphi_\theta = J_0$ itself on $[0, 12]$, not only up to scale and shift. Of the other four runs that learned J_0, three select the same laws written with numerically negligible extra terms, such as $x_2 \cdot 0.99999887$ in place of x_2, or with $\varphi_\theta = -J_0$ and the signs changed to match. The ABLATED elementary search does not find a short description. Even the extended run has a median answer size of 50 nodes, and its median error on fresh data is about a hundred times the spline's. The MLP operator fails to learn J_0 in all runs.

Training loss alone picks out these runs. The 13 runs that learned J_0 are exactly the 13 spline runs with the lowest training loss, and each has a lower training loss than all 40 elementary runs. So a user who did not know that the data needed J_0, and kept the lowest-loss run of several restarts, would get the expected 12-node answer, with a normalized mean squared error of 4.8×10^{-6} on 10,000 fresh points. Keeping the lowest-loss of the 20 extended ABLATED runs instead returns a 56-node expression that is undefined on part of the domain. We also tested whether the learned function can be reused. We froze the parameters of the φ_θ from the lowest-loss +BOP run, added it to the library of a new search, and trained that search on $J_0(x_1)J_0(x_2)$ sampled at 32 points on $[0, 6]^2$. On inputs from $[6, 9]^2$, which lie inside the $[0, 12]$ range the operator was trained on but outside the new search's training range, the returned expression has a normalized error of about 10^{-5}, far below the median error of 1.9 returned by elementary search.

To check that the results were not a fluke, we also ran a similar set of experiments but with Dawson's integral (another special function) in place of J_0. All three of the target laws are matched for five of the spline runs and the operators learned also achieve an R^2 of about 0.998 against Dawson's integral, just shy of our typical 0.999 goal. The lowest-loss run learns Dawson's integral with its argument curiously shifted by 0.37. Additional information for all the experiments discussed in this section is in Appendix E.

4 WHEN IS THE LEARNED OPERATOR DETERMINED?

Because a flexible φ_θ could fit the data exactly without being a generalizable missing function, we ask when the shortest exact description of a set of laws forces φ_θ to be that function, up to the scale and offset of its output. We study the question in an idealized setting where the laws are known exactly on their domains, without noise, and the search finds a description of minimum total size.

A single law often leaves φ_θ undetermined, in two ways. First, φ_θ can absorb the whole law. For example, the shortest exact description of $f(x) = \sqrt{x} + x$ is the two-node tree $\varphi_\theta(x)$ with $\varphi_\theta(x) = \sqrt{x} + x$, not $\varphi_\theta(x) + x$ with $\varphi_\theta = \sqrt{\cdot}$. Second, a law's constants can compensate a change in φ_θ. Consider the law $f(x) = \sqrt{x + 2}$, which is written equally well as $\varphi_\theta(x + c)$ with $\varphi_\theta(u) = \sqrt{u + 2 - c}$ for any c, so this law cannot tell $\sqrt{\cdot}$ from a shifted square root. However, a

second law without that constant, such as $x_2\sqrt{x_1}$, fixes the shift. Laws that share φ_θ can therefore, in a sense, cover each other's blind spots. We formalize the problem in Appendix F. There, Theorems 1 and 2 and Corollary 3 give conditions under which several laws determine a shared operator. The paragraphs below summarize what each result says informally.

Local identification (Theorem 1). The operator φ_θ is a function determined by its parameters θ, such as the 33 control points of the spline. Suppose two laws are $J_0(x+2)$ and $J_0(3x)$, and the search writes them as $\varphi_\theta(x+c)$ and $\varphi_\theta(ax)$. With $\varphi_\theta = J_0$, $c = 2$ and $a = 3$, both formulas fit exactly. Theorem 1 keeps these formulas as they are and asks whether an operator slightly different from J_0 could also fit both laws exactly after c and a are readjusted. Some changes to φ_θ can be compensated by a law's constants. Shifting φ_θ sideways is compensated by changing c in $\varphi_\theta(x+c)$, and stretching it sideways is compensated by changing a in $\varphi_\theta(ax)$, so each law alone cannot distinguish these altered operators from J_0. The theorem says that if every change to the shape of φ_θ can be compensated in some of the laws but never in all of them, then no operator close to J_0 fits every law, other than J_0 itself up to the scale and offset of its output. In the example, the second law rules out the shift and the first rules out the stretch, so together they determine φ_θ. Repeating a law adds nothing, since the copy compensates the same changes.

Generic global recovery (Theorem 2). Theorem 1 keeps the formulas fixed, but the search can also write the same laws differently. The two laws above can be written as $\varphi_\theta(x)$ and $\varphi_\theta(ax+d)$ with $\varphi_\theta(u) = J_0(u+2)$, $a = 3$ and $d = -2$. This description fits both laws exactly and has the same total size, eight nodes, but its operator is a shifted J_0. Theorem 2 concerns such rivals: descriptions with a different structure that fit every law exactly and are either shorter or equally short with a different operator. It says that the shortest description uses the true operator, up to the scale and offset of its output, provided that no rival can follow every change the laws undergo when their true constants and operator change. The rival above can: whatever values c and a take, it matches them by readjusting a and d, so these two laws alone do not determine φ_θ. The theorem holds for almost every value of the true parameters. A rival may happen to fit at special values, as the shorter formula $\varphi_\theta(x)$ fits $J_0(x+c)$ exactly when $c = 0$, but it cannot do so over a whole range of values.

Several laws can be necessary (Corollary 3). This corollary combines the two theorems into a case that shows why several laws can be needed to pin down φ_θ. Suppose each law, taken alone, can be fit exactly by more than one operator, because its constants compensate some change of φ_θ, like the shift in $\varphi_\theta(x+c)$ or the stretch in $\varphi_\theta(ax)$. Suppose the laws together still catch every change of shape, as in Theorem 1, and that they meet the condition of Theorem 2, so no rival for all the laws together can follow every change they undergo. The rival above, which moved the shift constant from one law to the other, is a case where that condition fails. Suppose finally that each true formula is the shortest description of its own law, so the ambiguity of a single law is not merely that a shorter formula exists. Then, for almost every value of the true parameters, each law alone has shortest descriptions with different operators, while every shortest description of all the laws together uses the true operator, up to the scale and offset of its output. No single law pins down φ_θ, but the laws together do.

The theory accounts for several features of our results. An operator used by a single law can absorb that law, and square-root recovery is indeed lower in runs whose selected equations use φ_θ only once (Appendix Table 11). A shift that every law's constants can absorb is a blind spot shared by all laws, and in the Dawson runs, where each law carried its own shift constant, the lowest-loss run learned Dawson's integral with a shifted argument. The J_0 laws contain no constants, so any change of φ_θ on $[0, 12]$ changes one of them and the laws have no shared blind spot by construction. What remains is whether the search reaches the minimum, and the lowest-loss runs do (Section 3). The results assume a smooth operator family and a search that finds the minimum, and Appendix F discusses both.

5 Deleting a Function and Learning It Back

In practice, given a dataset, one typically does not know in advance which functions its laws will need. We simulate this by deleting a known function from the library and testing whether φ_θ recovers it. Every experiment uses the four arms of Section 3, with shared data, seeds and search iterations.

ORACLE is the full library, ABLATED is the full library without the deleted function, and +BOP and +NOP add a spline or MLP φ_θ with the deleted function's number of arguments.

5.1 TASKS AND SCORES

Synthetic equation sets. To know which function each law needs, we wrote the laws ourselves. The ten sets hold two to four equations in two inputs, and each equation uses the square root in a different role, alone, inside a sum or a product, or next to polynomial, trigonometric, exponential or logarithmic terms (Table 4). The library is $\{\sqrt{\cdot}, +, -, \times\}$, plus $\cos$, $\log$ or $\exp$ where a set uses them. With $\sqrt{\cdot}$ deleted, φ_θ has one argument. In a second block we delete $+$ and $-$ together, and φ_θ has two arguments. Inputs are drawn uniformly on each set's range without noise, with 1000 training and 1000 test points, and the loss is the mean squared error averaged over a set's equations.

Physical tasks. We use the pendulum, double-pendulum, air-track and double-air-track recordings of Schmidt & Lipson (2009), each in a real and a simulated version, and the tracked pixel coordinates of the pendulum bob, which trace a circle. From these we build fifteen tasks in five groups of three (Table 5). Each group is one joint search with one shared φ_θ, with the functions its laws depend on deleted. These are $\sin$ and $\cos$ for the two pendulum groups, where φ_θ has one argument, and $+$ and $-$ for the two air-track groups and $\times$ for a geometry group of circle, sphere and tracked-bob constraints, where φ_θ has two. *Invariant* tasks ask for a quantity H conserved along each trajectory, scored by the mean squared cosine between its gradient and the direction of motion, which is 0 when H is exactly conserved and at most 1. *Acceleration* tasks predict the measured acceleration (Appendix B). Where a recording has several trials the last one is the test set, and otherwise every fifth row is.

Three measures. We score each run by three measures, fit, function and law, all computed on the expression the engine selects from the run's final Pareto front. Each condition runs five search seeds for 100 iterations of the evolutionary loop, 250 learnable-arm runs in all. The *fit*, which we also call the relative loss, is the test loss divided by the median ABLATED test loss on the same set or task, so 1 means no better than the reduced library and 0 is a perfect fit. The *function* score is the affine R^2 of the least-squares fit $\varphi_\theta(u) \approx \alpha\, p(u) + \beta$ against the deleted function p, over the inputs u that φ_θ receives inside the selected expression on the held-out rows, pooled over its occurrences and the group's tasks. A function is recovered at $R^2 \geq 0.99$, and we also report 0.95. On the pendulum groups, p is whichever of $\sin$ and $\cos$ matches better. The *law check* asks whether the selected expressions are the target laws with φ_θ in place of the deleted function, up to the values of their constants. On the synthetic sets, where the target laws have fixed constants, we put the deleted function back in place of φ_θ and refit the constants. The refit must give relative RMSE at most 10^{-6} on 1000 points in the training range and on 1000 over three times its width, and no term that the refit sets to zero may change the prediction when restored. On the physical groups, where the expected laws have unknown coefficients, the selected expression must have the terms of the expected law with φ_θ in its place, such as $c_0 + c_1\omega^2 + c_2\varphi_\theta(\theta)$ for the pendulum energy (Appendix C).

5.2 THE SQUARE ROOT

Deleting the square root and adding a learnable operator usually gives the square root back. The operator matches $\sqrt{\cdot}$ up to an affine map in about two thirds of runs, 33 of 50 with the spline and 32 of 50 with the MLP, and in nine of the ten equation sets at least one run of each arm does so (Table 1, Fig. 2). These counts barely change under a second way of scoring the match (Appendix D.2). The match holds on the inputs each operator received, not very far beyond them. On the fixed range $[0, 20]$, which extends past most runs' data, both operators match less often at $R^2 \geq 0.99$, the spline in 9 of 50 runs and the MLP in 20. However, the two differ in how far they miss. The spline continues as a cubic outside its data and often departs from $\sqrt{\cdot}$, while the MLP continues linearly and stays close, matching at $R^2 \geq 0.95$ in 49 of 50 runs (Appendix D.2). When the operator does match, the search often (in 12 spline and 14 MLP runs) writes the target laws themselves, with φ_θ exactly in place of $\sqrt{\cdot}$.

Table 1: Recovery of deleted functions. *Fit*: median over runs of the test loss divided by the median ABLATED test loss on the same set or task, so 1 is no better than the reduced library and 0 is a perfect fit. *Function*: runs whose learned operator matches the deleted function at held-out affine $R^2 \geq 0.99$ (at 0.95 in parentheses). *Law*: runs that pass the law check. On the physical groups, fit and law use the energy task (the circle for the geometry group), and the function score pools all of the group's tasks.

		+BOP (spline)			+NOP (MLP)		
Deleted	Setting	Fit	Function	Law	Fit	Function	Law
$\sqrt{\cdot}$	10 synthetic sets	0.098	33/50 (39)	12/50	0.378	32/50 (46)	14/50
$+, -$	10 synthetic sets	0.018	3/50 (14)	2/50	0.009	21/50 (32)	14/50
$\sin, \cos$	real pendulum	0.011	0/5 (0)	1/5	0.026	0/5 (3)	2/5
$\sin, \cos$	simulated pendulum	0.009	1/5 (1)	1/5	0.029	1/5 (1)	1/5
$+, -$	real air track	0.371	0/5 (0)	0/5	0.022	0/5 (0)	0/5
$+, -$	simulated air track	0.085	0/5 (0)	0/5	0.059	0/5 (0)	0/5
$\times$	geometry	0.047	0/5 (0)	0/5	0.057	0/5 (0)	0/5

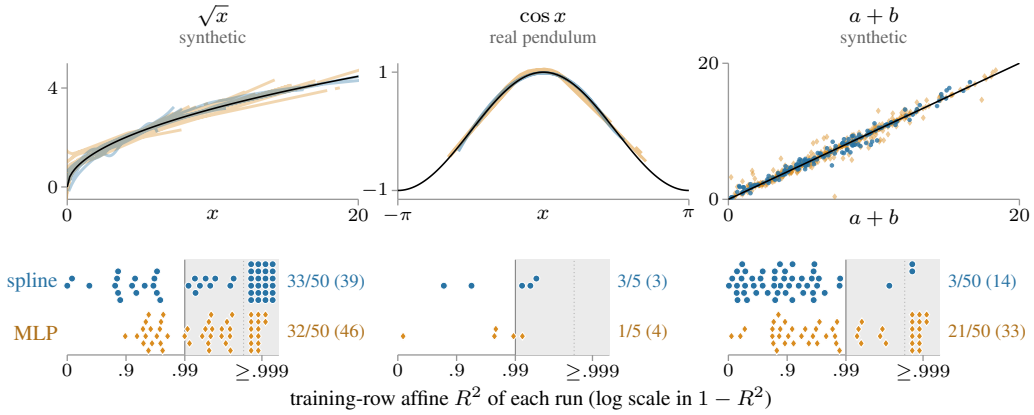

Figure 2: The learned operator against the deleted function. Curves show φ_θ on its training inputs after the training-row affine fit, for runs with training-row $R^2 \geq 0.95$, with the deleted function in black. Dots give each run's training-row affine R^2, and the counts give the runs at $R^2 \geq 0.99$ (0.95), for $\sqrt{x}$ and $a + b$ on ten synthetic sets and for $\cos x$ on the real pendulum.

5.3 ADDITION

With $+$ and $-$ deleted, φ_θ is a function of two arguments. The MLP learns addition or subtraction at held-out $R^2 \geq 0.99$ in 21 of 50 runs, and in 14 of them the selected equations are the target laws. The spline, a grid of 1089 control points, learns it in 3 runs, with the target laws in 2 (Table 1). The library without $+$ and $-$ never produces the target laws. Both arms fit well. The median run reaches a relative loss of 0.018 with the spline and 0.009 with the MLP, so each removes about 98 percent of the error that deleting $+$ and $-$ caused, and the median is below 0.1 on nine of the ten sets for the spline and on all ten for the MLP.

5.4 THE COSINE FROM A REAL PENDULUM

On the pendulum a polynomial in the angle can approximate the cosine term, as in the example of the introduction, so we exclude such polynomials in all four arms, ORACLE included. Angles are wrapped to $[-\pi, \pi)$, and no product may have a raw angle (one that has not passed through a unary operator) on one side and a variable-dependent term on the other. With the physical library $\{+, -, \times, \sin, \cos\}$, an angle can then enter nonlinearly only through a unary operator, which is $\cos$ in ORACLE, absent in ABLATED, and φ_θ in the learnable arms.

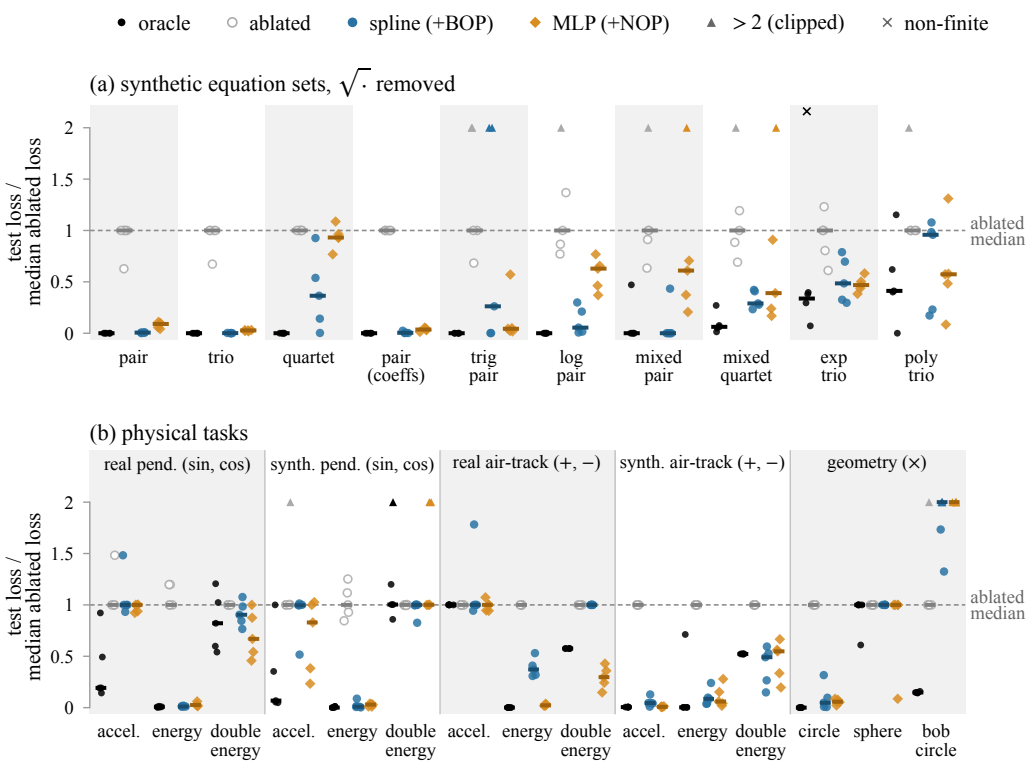

Figure 3: Adding one learnable operator removes most of the increase in test loss caused by deleting a primitive on most synthetic sets and on the single-system energy tasks, and has mixed effects on the other physical tasks. Each point is one seed's test loss divided by the median ABLATED loss on the same set or task, so 1 (dashed) is no better than the reduced library and 0 is a perfect fit. Ticks mark medians, triangles mark seeds above 2, and the cross marks the one run with a non-finite test loss. (a) Synthetic sets with $\sqrt{\cdot}$ removed, ordered by the relative loss of ORACLE. (b) Physical tasks by group, with the removed primitives in the group header; invariant tasks use the invariance score and acceleration tasks the variance-normalized MSE.

On the real pendulum with sin and cos deleted, 3 of 10 runs, one spline and two MLP runs, select an energy law of the form $c_0 + c_1\omega^2 + c_2\varphi_\theta(\theta)$ (Appendix Table 15). By the function score, pooled over the held-out rows of all three pendulum tasks, 3 of 10 runs reach 0.95 and none reaches 0.99. On the energy task alone, scoring φ_θ only on the angles it receives in the energy expression, the three energy-law runs reach 0.974, 0.993 and 0.992 (Appendix Table 16). The recorded angles span about $[-1.40, 1.47]$, and on that range an affine fit at $R^2 \geq 0.99$ cannot distinguish cos from its quartic Taylor polynomial $1 - x^2/2 + x^4/24$. The two become distinguishable only on ranges that reach $|x| \geq 2.12$, which is also why the quartic of Schmidt & Lipson (2009) fits. The selected operator is cosine-shaped on the recorded angles and appears in the energy-law form, and on a grid over the full period $[-\pi, \pi]$ no run reaches 0.99 (Appendix D.2). All arms fit the double pendulum poorly. Even the full library, cosine included, reaches a median relative loss of only 0.82 on the recording and 1 on the simulation, while 3 of 5 MLP runs fit the recording better than the median ORACLE run. The likely cause is our search settings, since the three pendulum tasks were learned as one group and the number of optimizer iterations was not tuned for them.

5.5 FAILURE MODES

Across these experiments, a run whose operator does not match the deleted function, which we call a miss, usually still fits the data well (Fig. 3). On the addition block the 45 spline misses with a finite loss have a median relative loss of 0.022 and the 29 MLP misses 0.014, and 38 and 27 of them, respectively, are below 0.1. On the three sets where the full library does not fit exactly, 10 spline

misses and 11 MLP misses fit the test data better than the median ORACLE run. On the square-root block the misses remove about half of the error the deletion caused, with median relative losses of 0.49 (spline, 17 misses) and 0.54 (MLP, 18 misses). On the physical groups no run recovers a deleted binary operation, but the learned operator still gives a good fit on some tasks. The MLP brings the real air track's energy task to a median relative loss of 0.022, every MLP run fits the real double air track better than the median ORACLE run, and both arms fit the geometry group's circle task to about 0.05. Neither arm fits the tracked bob of the geometry group, where the median relative loss is 5.3 (spline) and 3.1 (MLP).

6 Related Work

Fixed vocabularies. Genetic programming searches over expression trees built from a fixed function set (Koza, 1992), and Schmidt & Lipson (2009) used it to find conservation laws in recorded trajectories. We build on SymbolicRegression.jl and PySR (Cranmer, 2023). Other searches change how expressions are found (Udrescu & Tegmark, 2020; Biggio et al., 2021; Kamienny et al., 2022; Petersen et al., 2021; Brunton et al., 2016). In all of them, and in SRBench (La Cava et al., 2021), the user chooses the function set.

Learned functions from a fixed family. EQL uses a fixed menu of primitives as network activations (Martius & Lampert, 2016; Sahoo et al., 2018), NOMTO adds pretrained neural operators held fixed (Garmaev & Fink, 2025), symbolic metamodels fit Meijer G-functions (Alaa & van der Schaar, 2019; Crabbé et al., 2020), Abroshan et al. (2023) evolve tree structures while training parametric primitives from a few analytic families by gradient descent, Pietropolli et al. (2023) add gradient-trained weights to GP trees, shared per operator type in one variant, and Symbolic-KAN and SINDy-KAN select or combine terms from an analytic library (Faroughi et al., 2026; Howard et al., 2026). These methods tune a few coefficients of functions written in advance, while the tens to a thousand parameters of our spline or network set the shape of the learned function.

Free functions with a fixed skeleton or at every edge. Universal differential equations fit a network for an unknown term of a differential equation of fixed form (Rackauckas et al., 2020), which Podina et al. (2023) convert to a formula with SR. KANs put an independent spline on every edge and may replace each with the closest library function (Liu et al., 2025), and KAN-SR uses them to guide a symbolic search (Bühler & Guillén-Gosálbez, 2025). In our method the search chooses where to place the learned function, and every occurrence uses the same parameters.

Library learning. Automatically defined functions (Koza, 1994) and DreamCoder (Ellis et al., 2021) add reusable compositions of existing primitives, and LaSR induces textual concepts that guide PySR (Grayeli et al., 2024).

7 Conclusion

The list of functions a symbolic search may use, its vocabulary, can be learned from data during the search. With one shared learnable operator, a search given only elementary functions fitted an operator that matches Bessel's J_0 in 13 of 20 runs and selected three laws with it that are as short as the answer of a search whose library contains J_0, while elementary search returned formulas about four times longer. We proved conditions under which several laws that share one operator determine it up to an affine map, even when no single law does. After a deletion the operator matches the square root, addition or cosine in some runs, and most other runs still have low error.

Our results have limitations. We learn one operator at a time, supply its number of arguments, and in many cases learn from synthetic equations. For J_0, the training loss neatly separates the runs that learn the missing function from those that do not, but this is messier for the square-root sets. Finally, a learnable operator generally lengthens the search (Table 6), though this can be worth it.

Functions such as $\sqrt{\cdot}$, $\sin$ and J_0 became standard because many laws use them. We chose J_0 because the target function is known, and the next test is a set of laws that require a function no one has named.

AI USE STATEMENT

In this work, we used generative AI tools (Claude Code and Codex) to generate synthetic datasets, to assist in the writing of proofs, to design or provide feedback on our research methodology and experiments, to implement methods, to clean and reformat datasets, and to support qualitative and thematic data analysis. Additionally, we used generative AI tools to create and edit software code and to draft parts of the paper. We have reviewed all AI-assisted work. All AI-generated code was reviewed and tested for correctness by the authors, dataset cleaning outputs were spot-checked against the raw data, all proofs were checked by the authors, synthetic datasets were inspected for quality and validity before use, and all AI-drafted text was reviewed and edited by the authors. We take responsibility for the final content of this work, including text, claims, or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

Section 5.1 and Appendix B describe every search and optimizer setting, the synthetic equations and sampling ranges, the physical datasets of Schmidt & Lipson (2009), the preprocessing, the train/test splits and the invariance objective. Appendix C defines every metric and the analysis protocol.

REFERENCES

Ralph Abraham, Jerrold E. Marsden, and Tudor Ratiu. *Manifolds, Tensor Analysis, and Applications*, volume 75 of *Applied Mathematical Sciences*. Springer, New York, second edition, 1988. doi: 10.1007/978-1-4612-1029-0.

Mahed Abroshan, Saumitra Mishra, and Mohammad Mahdi Khalili. Symbolic metamodels for interpreting black-boxes using primitive functions. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pp. 6649–6657, 2023. doi: 10.1609/aaai.v37i6.25816.

Ahmed M. Alaa and Mihaela van der Schaar. Demystifying black-box models with symbolic metamodels. In *Advances in Neural Information Processing Systems 32 (NeurIPS)*, 2019.

Luca Biggio, Tommaso Bendinelli, Alexander Neitz, Aurelien Lucchi, and Giambattista Parascandolo. Neural symbolic regression that scales. In *Proceedings of the 38th International Conference on Machine Learning (ICML)*, 2021.

Steven L. Brunton, Joshua L. Proctor, and J. Nathan Kutz. Discovering governing equations from data by sparse identification of nonlinear dynamical systems. *Proceedings of the National Academy of Sciences*, 113(15):3932–3937, 2016. doi: 10.1073/pnas.1517384113.

Marco Andrea Bühler and Gonzalo Guillén-Gosálbez. KAN-SR: A Kolmogorov-Arnold network guided symbolic regression framework, 2025.

Jonathan Crabbé, Yao Zhang, William Zame, and Mihaela van der Schaar. Learning outside the black-box: The pursuit of interpretable models. In *Advances in Neural Information Processing Systems 33 (NeurIPS)*, 2020.

Miles Cranmer. Interpretable machine learning for science with PySR and SymbolicRegression.jl. *arXiv preprint arXiv:2305.01582*, 2023.

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)*, pp. 835–850. ACM, 2021. doi: 10.1145/3453483.3454080.

Salah A. Faroughi, Farinaz Mostajeran, Amirhossein Arzani, and Shirko Faroughi. Symbolic-KAN: Kolmogorov-Arnold networks with discrete symbolic structure for interpretable learning, 2026.

Sergei Garmaev and Olga Fink. A neural operator-based approach to symbolic discovery of PDEs, 2025.

Arya Grayeli, Atharva Sehgal, Omar Costilla-Reyes, Miles Cranmer, and Swarat Chaudhuri. Symbolic regression with a learned concept library. In *Advances in Neural Information Processing Systems*, volume 37, 2024. doi: 10.52202/079017-1419.

Amanda A. Howard, Nicholas Zolman, Bruno Jacob, Steven L. Brunton, and Panos Stinis. SINDy-KANs: Sparse identification of non-linear dynamics through Kolmogorov-Arnold networks, 2026.

Pierre-Alexandre Kamienny, Stéphane d'Ascoli, Guillaume Lample, and François Charton. End-to-end symbolic regression with transformers. In *Advances in Neural Information Processing Systems 35 (NeurIPS)*, 2022.

John R. Koza. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press, Cambridge, MA, 1992. ISBN 0-262-11170-5.

John R. Koza. *Genetic Programming II: Automatic Discovery of Reusable Programs*. MIT Press, Cambridge, MA, 1994.

William La Cava, Patryk Orzechowski, Bogdan Burlacu, Fabrício Olivetti de França, Marco Virgolin, Ying Jin, Michael Kommenda, and Jason H. Moore. Contemporary symbolic regression methods and their relative performance. In *Advances in Neural Information Processing Systems 34, Datasets and Benchmarks Track*, 2021.

Antoine Laurent Lavoisier. *Elements of Chemistry, in a New Systematic Order, Containing All the Modern Discoveries*. William Creech, Edinburgh, 1790. Translated by Robert Kerr from *Traité élémentaire de chimie* (Paris, 1789). Quotation from the Preface of the Author.

Ziming Liu, Yixuan Wang, Sachin Vaidya, Fabian Ruehle, James Halverson, Marin Soljačić, Thomas Y. Hou, and Max Tegmark. KAN: Kolmogorov-Arnold networks. In *International Conference on Learning Representations (ICLR)*, 2025.

Georg Martius and Christoph H. Lampert. Extrapolation and learning equations. *arXiv preprint arXiv:1610.02995*, 2016.

John J. O'Connor and Edmund F. Robertson. The trigonometric functions. MacTutor History of Mathematics, University of St Andrews, 1996. URL `https://mathshistory.st-andrews.ac.uk/HistTopics/Trigonometric_functions/`.

John J. O'Connor and Edmund F. Robertson. Friedrich wilhelm Bessel. MacTutor History of Mathematics, University of St Andrews, 1997. URL `https://mathshistory.st-andrews.ac.uk/Biographies/Bessel/`.

John J. O'Connor and Edmund F. Robertson. John Napier. MacTutor History of Mathematics, University of St Andrews, 1998. URL `https://mathshistory.st-andrews.ac.uk/Biographies/Napier/`.

Brenden K. Petersen, Mikel Landajuela, T. Nathan Mundhenk, Claudio P. Santiago, Soo K. Kim, and Joanne T. Kim. Deep symbolic regression: Recovering mathematical expressions from data via risk-seeking policy gradients. In *International Conference on Learning Representations (ICLR)*, 2021.

Gloria Pietropolli, Federico Julian Camerota Verdù, Luca Manzoni, and Mauro Castelli. Parametrizing GP trees for better symbolic regression performance through gradient descent. In *Proceedings of the Companion Conference on Genetic and Evolutionary Computation (GECCO '23 Companion)*, pp. 619–622, 2023. doi: 10.1145/3583133.3590574.

Lena Podina, Brydon Eastman, and Mohammad Kohandel. Universal physics-informed neural networks: Symbolic differential operator discovery with sparse data. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 27948–27956, 2023. URL `https://proceedings.mlr.press/v202/podina23a.html`.

Christopher Rackauckas, Yingbo Ma, Julius Martensen, Collin Warner, Kirill Zubov, Rohit Supekar, Dominic Skinner, Ali Ramadhan, and Alan Edelman. Universal differential equations for scientific machine learning, 2020.

Subham Sahoo, Christoph Lampert, and Georg Martius. Learning equations for extrapolation and control. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 4442–4450. PMLR, 2018.

Arthur Sard. The measure of the critical values of differentiable maps. *Bulletin of the American Mathematical Society*, 48(12):883–890, 1942. doi: 10.1090/S0002-9904-1942-07811-6.

Michael Schmidt and Hod Lipson. Distilling free-form natural laws from experimental data. *Science*, 324(5923):81–85, 2009. doi: 10.1126/science.1165893.

Claude E. Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27 (3):379–423, 1948.

Silviu-Marian Udrescu and Max Tegmark. AI Feynman: A physics-inspired method for symbolic regression. *Science Advances*, 6(16):eaay2631, 2020. doi: 10.1126/sciadv.aay2631.

A Method Details

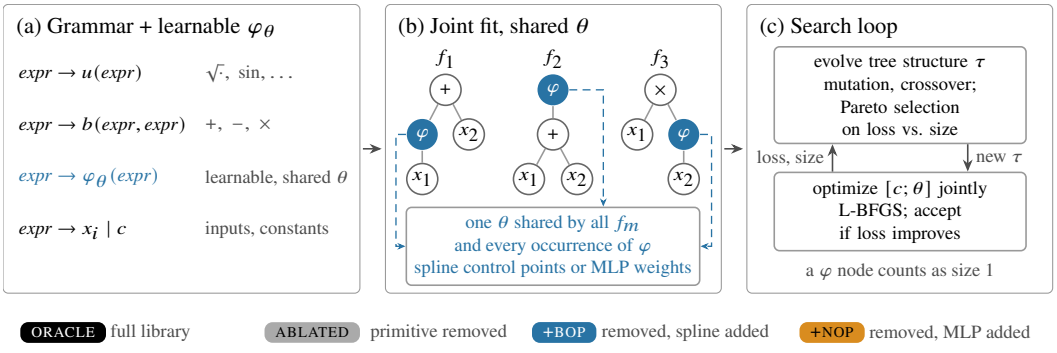

Figure 4: The method adds one learnable operator, shared by every equation of a joint fit, to an ordinary symbolic search. (a) The expression grammar gains one production φ_θ whose parameters θ (spline control points or MLP weights) are fitted along with the constants. (b) A single θ serves every occurrence of φ_θ in every equation $f_1, \ldots, f_M$ of the joint fit. (c) Evolution over tree structure alternates with joint L-BFGS optimization of constants and operator parameters; a learnable node counts as one unit of size. The legend lists the four arms run in every experiment.

B-spline operator. For each input the knot vector is $\kappa = (0^{(d)}, 0, 1, \ldots, k, k^{(d)})$: integer knots on $[0, k]$ with the end knots repeated so that the total multiplicity at each end is $d + 1$; it spans $k + d$ B-spline basis functions, so the control tensor has $(k + d)^a$ entries for arity a. An input outside $[0, k]$ is assigned to the first or last knot span and the polynomial piece of that span is evaluated, so the operator is a degree-d polynomial extrapolation beyond its support rather than clamped or undefined. The implementation also carries, per input, an affine map (a scale and an offset) and the knot vector itself as further trainable components, and the MLP's weights and biases can be frozen separately; Table 2 lists what is trained in every run reported here (spline control points, all MLP weights and biases), with the affine maps and knots left at their defaults (identity map, uniform knots). Because B-spline bases sum to one on the support, initializing every control point to 1 makes the initial operator the constant 1. All arithmetic is single precision.

MLP operator. The MLP is a chain of two dense layers, $a \to H \to 1$, with ReLU on the hidden layer and identity output, built with Lux.jl. Parameters are initialized by the library defaults (Kaiming-uniform weights, uniform biases) from one fixed pseudo-random seed at construction time, so the initial θ is identical across seeds, runs and population members. Evaluation is batched over a whole data column; the spline is evaluated element-wise.

Table 2: Only the spline control points and the MLP weights and biases are trained in this paper; every other component of the learnable operators (`LearnableConfig`) stays at its default.

Operator	Component	Trained	Count (unary / binary)
BOP	control points C	yes	33 / 1089
BOP	knot vectors κ_d	no	37 per input
BOP	input scales w_d	no	1 / 2
BOP	input offsets b_d	no	1 / 2
NOP	weights W_1, W_2	yes	32 / 48
NOP	biases b_1, b_2	yes	17 / 17
tree	scalar constants	yes	varies

Where the parameters live. A candidate in a joint fit is a tuple of inner expressions $(f_1, \ldots, f_M)$ together with named parameter vectors held alongside them, one per trainable component of each learnable symbol. Before an inner expression is evaluated, the operator table is rebuilt from those vectors and attached to every inner expression, so all f_m see the same operator. Copies of a candidate, made for mutation, crossover, or the hall of fame, copy the vectors, so no two population members share θ. A learnable symbol that does not occur anywhere in the current tuple contributes no optimizable parameters, so the dimension of the continuous problem varies from candidate to candidate. The SR engine exposes the flat parameter vector through a small interface (get, set, gradient), and our implementation lays it out as the scalar constants of f_1, then of f_2, and so on, followed by the active learnable vectors.

Optimizer. The engine's constant optimizer is Optim.jl's L-BFGS with a backtracking line search, run for a fixed number of iterations (Table 3) with a cap on function evaluations; a single free parameter is handled by Newton's method. The optimized vector replaces the original only if the loss decreased. When restarts are enabled the starting vector is perturbed multiplicatively by a factor of one plus half a standard normal draw per coordinate, and the best of the restarts is kept; the synthetic runs use no restarts, the physical runs use the engine default. Gradients are reverse-mode derivatives computed by Mooncake.jl through the expression evaluator and the bound operators.

Size and selection. The size of a joint candidate is the sum over inner expressions of their node counts; no per-operator cost is set, so every node, including a learnable one, has size one, and the maximum size applies to the sum. The expression we report from a run is the one the engine selects from its final Pareto front by its standard rule: among front members whose loss is within a factor 1.5 of the lowest, the one with the largest decrease in log-loss per unit of size relative to the next smaller member; we also record the front member of lowest training loss.

Grammar. For completeness, with $\mathcal{V} = \{\langle \mathrm{expr} \rangle\}$ and $\Sigma = \mathcal{U} \cup \mathcal{B} \cup \mathcal{L} \cup \mathcal{T}$ the productions are $\langle \mathrm{expr} \rangle \to u\,\langle \mathrm{expr} \rangle$ for $u \in \mathcal{U}$, $\langle \mathrm{expr} \rangle \to b\,\langle \mathrm{expr} \rangle\,\langle \mathrm{expr} \rangle$ for $b \in \mathcal{B}$, $\langle \mathrm{expr} \rangle \to \ell_{\theta_\ell}\,\langle \mathrm{expr} \rangle^{a(\ell)}$ for $\ell \in \mathcal{L}$, and $\langle \mathrm{expr} \rangle \to t$ for $t \in \mathcal{T}$.

Further limitations of the implementation. The spline operator starts as the constant function 1 and the MLP starts from one fixed random draw shared by every seed, run and population member, so diversity in θ comes only from optimization along different search lineages. The spline's support is $[0, 30]$ with no input rescaling, and inputs to a φ_θ node routinely leave this range (Appendix D reports the fraction); behavior there is polynomial extrapolation. Neither the smoothness of the spline nor the norm of the MLP weights is penalized. Each synthetic set uses one data draw, so seeds vary the search only. For single-trial physical recordings the test rows are interleaved in time with the training rows and are not independent of them. Gradients of the invariance objective are central finite differences, and the SR engine reuses a parent's cached loss after algebraic simplification, which under a finite-difference loss can occasionally leave a front member with a slightly stale loss (next paragraph).

Stale losses after simplification. The engine simplified expressions during the search and reused the parent's cached loss for the simplified child. When a simplification folded a learnable operator using its default instead of its fitted parameters, the cached loss could be wrong. We therefore

rescored all 900 final candidates, the selected and the minimum-loss candidate of each of the 450 searches, from their stored parameters. Nine disagreed, in six searches. Three are selected candidates, all in the geometry spline arm (seeds 1001, 1002 and 1005), whose true training loss is 53 to 86% higher than the recorded one. Six are minimum-loss candidates, from the same three geometry runs, one `sqrt_poly_trio` MLP run and two runs of the ablated real-pendulum arm. Every test loss, relative loss, operator score and structure count in the paper is computed from the rescored candidates, and the geometry spline exemplar of Table 14 is the seed with the median rescored training loss. The recorded values enter only the loss trajectories and Pareto fronts of Appendix D and the BIC reselection of Appendix D.6, which ranks the recorded losses of front members. Rescoring the final candidates cannot reverse any effect that an incorrect cached loss had on selection during those six searches, so this caveat applies to the geometry spline results.

B Experimental Details

Arms and settings. Table 3 lists every search, data, optimizer and operator setting for both tracks. Three asymmetries between tracks are deliberate: the constant optimizer runs 20 iterations per call on the synthetic sets and 3 on the physical tasks; the physical runs use the engine's default number of optimizer restarts and the synthetic runs none; and the population layout differs. Within a track all four arms share every setting. Search seeds are 1001 to 1005 in both tracks with a single data seed. Each search ran as one Slurm job with 4 CPU cores and 12 GB of memory (32 GB for the add/sub block) on AMD Genoa or Intel Sapphire Rapids nodes with Julia 1.12; the median job took about 0.3 h on the synthetic sets and 0.6 h on the physical tasks, the longest 3.6 h, for roughly 250 node-hours (about 1000 CPU-hours) across the 450 searches.

Synthetic equation sets. Table 4 lists the ten sets. Inputs are drawn independently and uniformly on the range given in the table with a fixed pseudo-random seed, $n = 1000$ training points in single precision, and targets are computed exactly. An independent test set of 1000 points is drawn from a second fixed seed on the same range. Because the data seed does not vary across search seeds, all seeds of all arms of a set see the same data. The M equations of a set are fit as one joint candidate whose inner expressions $f_1, \ldots, f_M$ each see both inputs; the loss is $\frac{1}{n}\sum_j \frac{1}{M}\sum_m (y_{m,j} - f_m(x_j))^2$. Holding out $\sqrt{\cdot}$ removes it from every inner expression's library and, in the learnable arms, adds one unary φ_θ shared by all f_m; holding out $+$ and $-$ leaves $\{\times\}$ and adds one binary φ_θ. Sets that use $\cos$, $\log$ or $\exp$ keep those operators in every arm.

Table 3: Within a track, all four arms share every search, optimizer and operator setting.

Setting	**Synthetic sets**	**Physical tasks**
Search		
Search seeds per condition	5 (seeds 1001–1005)	5 (seeds 1001–1005)
Outer iterations (`niterations`)	100	100
Populations × size	10×30	4×50
Tournament size	SR default	15
Max. size (sum over equations)	30	150 pendulum groups; 30 otherwise
Nesting constraints	none	pendulum groups: no trig in trig, no φ_θ in φ_θ
Raw-angle product constraint	–	pendulum groups only
Parallelism	multithreading	multithreading
Data		
Samples	1000 per set (train), 1000 (test)	256 rows per task
Data seed	1 (test: 1+10000)	1
Test split	independent draw	last trial, else every 5th row
Noise	none	measured / simulated data
Precision	Float32	Float32
Constant and operator optimization		
Algorithm	L-BFGS, backtracking	L-BFGS, backtracking
Iterations per call	20	3
Members optimized per iteration	all	all
Restarts	0	2 (engine default)
Gradients	Mooncake reverse-mode	Mooncake reverse-mode
Candidate-gradient for tangency	–	central FD, step $10^{-3} s_j$
Tangent estimator	–	recorded, else local quadratic
Gradient-coverage threshold	–	0.9
Learnable operators		
B-spline degree d, resolution k	3, 30	3, 30
B-spline trainable parameters	control points only	control points only
MLP hidden width H	16	16
MLP trainable parameters	all weights and biases	all weights and biases
Initialization	BOP $\equiv$ 1; NOP fixed seed	BOP $\equiv$ 1; NOP fixed seed
Recording		
Candidate snapshots	every iteration	every iteration
Checkpoints	every 5 iterations	every 5 iterations
Operator sample rows per snapshot	1024	1024

Table 4: Every equation of a synthetic set uses $\sqrt{\cdot}$, in a different position, so one shared learnable operator must approximate it in every equation of the set. The equations of a set are fitted jointly. The library for a set is its unary operators plus $\{+, -, \times\}$. The main block removes $\sqrt{\cdot}$ (unary φ_θ); the second block removes $+$ and $-$ (binary slot).

Set	**Equations**	**Inputs**	**Unary ops**
`sqrt_pair`	$y_1 = \sqrt{x_1} + x_2;\ y_2 = \sqrt{x_1 + x_2}$	$[0, 10]^2$	$\sqrt{\cdot}$
`sqrt_pair_coeffs`	$y_1 = 2.31\sqrt{x_1} + x_2;\ y_2 = \sqrt{x_1 + 0.44\,x_2}$	$[0, 10]^2$	$\sqrt{\cdot}$
`sqrt_mixed_pair`	$y_1 = \sqrt{x_1}\,(x_2 + 1);\ y_2 = (x_1 + \sqrt{x_2})(1 + x_1 x_2)$	$[0.5, 10]^2$	$\sqrt{\cdot}$
`sqrt_trig_pair`	$y_1 = \sqrt{x_1}\cos x_2 + \cos x_1;\ y_2 = \sqrt{x_1 + x_2} + \cos(x_1 x_2)$	$[0, 6.28]^2$	$\sqrt{\cdot}$, cos
`sqrt_log_pair`	$y_1 = \sqrt{x_1} + \log(x_2 + 1);\ y_2 = \sqrt{x_1}\log(x_2 + 2)$	$[0.1, 10]^2$	$\sqrt{\cdot}$, log
`sqrt_trio`	$y_1 = \sqrt{x_1} + x_2;\ y_2 = \sqrt{x_1 + x_2};\ y_3 = x_1 + \sqrt{x_2}$	$[0, 10]^2$	$\sqrt{\cdot}$
`sqrt_poly_trio`	$y_1 = \sqrt{x_1} + x_2^2;\ y_2 = x_1\sqrt{x_2 + 1};\ y_3 = \sqrt{x_1 x_2} + 0.5\,x_1^2$	$[0, 8]^2$	$\sqrt{\cdot}$
`sqrt_exp_trio`	$y_1 = e^{0.3x_1} + \sqrt{x_2};\ y_2 = \sqrt{x_1 + x_2}\,e^{0.1x_2};\ y_3 = \sqrt{x_1 x_2} + e^{-x_1}$	$[0, 5]^2$	$\sqrt{\cdot}$, exp
`sqrt_quartet`	$y_1 = \sqrt{x_1} + x_2;\ y_2 = \sqrt{x_1 + x_2};\ y_3 = x_1 + \sqrt{x_2};\ y_4 = x_1 + x_2 + \sqrt{x_1 x_2}$	$[0, 10]^2$	$\sqrt{\cdot}$
`sqrt_mixed_quartet`	$y_1 = \sqrt{x_1 x_2} + 0.3\,x_1;\ y_2 = \sqrt{x_1 + x_2^2};\ y_3 = x_1\sqrt{x_2};\ y_4 = \sqrt{x_1 + 1}\,\sqrt{x_2 + 2}$	$[0, 8]^2$	$\sqrt{\cdot}$

Physical data. All physical data are the supporting datasets of Schmidt & Lipson (2009). Each recording holds a trial index, time, the state variables and, for some systems, their time derivatives. Values are used in single precision without smoothing, unit conversion or standardization; scale enters only through the per-variable standard deviations inside the invariance score. Table 5 lists the tasks by group. Each group is one joint search in which the three tasks share the learnable operator. The pendulum groups search over $\{\sin, \cos\}$ and $\{+, -, \times\}$ in the full arm and over $\{+, -, \times\}$ when ablated, with a unary learnable operator; the air-track and geometry tasks declare no unary operators, so those groups search over $\{+, -, \times\}$, reduced to $\{\times\}$ or $\{+, -\}$ when ablated, with a binary learnable operator. The geometry group is mixed: two simulated constraints and the real tracked bob. The acceleration task for the simulated pendulum is derived from the energy recording by taking its angular-velocity derivative column as the target.

Table 5: The fifteen physical tasks form five groups, each one joint search with a shared learnable operator and with the primitives its laws depend on removed (group header). All tasks are built from the recordings of Schmidt & Lipson (2009). *energy* and *geom.*: invariant tasks, where SR learns H with $H(\mathbf{x}(t)) = C$ along trajectories (an energy-like conservation law or a geometric constraint), scored by the invariance score of Appendix B; *accel.*: supervised tasks, where SR predicts the acceleration a from the state, scored by variance-normalized MSE. Variables marked $^\circ$ are angles, wrapped to $[-\pi, \pi)$ before the search. The expected law restates the result of Schmidt & Lipson (2009) for reference and is not used by the search. ‡`pendulum_a` is derived from the `pendulum_h` recording, with $a = \dot{\omega}$ taken from its derivative column. Fitted coefficients are denoted c_i.

Task	Type	Inputs	Expected law
Group `synthetic_pendulum_trig` (simulated; held out: $\sin, \cos$)			
`pendulum_a`‡	accel.	$\theta^\circ,\ \omega\ \rightarrow\ a$	$a = -9.8 \sin\theta$
`pendulum_h`	energy	$\theta^\circ,\ \omega$	$c_1\omega^2 + c_2 \cos\theta = C$
`double_pend_h`	energy	$\theta_1^\circ,\ \theta_2^\circ,\ \omega_1,\ \omega_2$	$c_1\omega_1^2 + c_2\omega_2^2 + c_3\omega_1\omega_2\cos(\theta_1 - \theta_2) + c_4\cos\theta_1 + c_5\cos\theta_2 = C$
Group `real_pendulum_trig` (real; held out: $\sin, \cos$)			
`real_pend_a`	accel.	$\theta^\circ,\ \omega\ \rightarrow\ a$	$a = c_0 + c_1\sin\theta + c_2\omega$
`real_pend_h`	energy	$\theta^\circ,\ \omega$	$c_1\omega^2 + c_2\cos\theta = C$
`real_double_pend_h`	energy	$\theta_1^\circ,\ \theta_2^\circ,\ \omega_1,\ \omega_2$	$c_1\omega_1^2 + c_2\omega_2^2 + c_3\omega_1\omega_2\cos(\theta_1 - \theta_2) + c_4\cos\theta_1 + c_5\cos\theta_2 = C$
Group `synthetic_airtrack_add_sub` (simulated; held out: $+, -$)			
`linear_a`	accel.	$x,\ v\ \rightarrow\ a$	$a = c_0 + c_1 x$
`linear_h`	energy	$x,\ v$	$c_1 x^2 + c_2 v^2 = C$
`double_linear_h`	energy	$x_1,\ x_2,\ v_1,\ v_2$	$c_1 v_1^2 + c_2 v_2^2 + c_3 x_1^2 + c_4(x_2 - x_1)^2 + c_5 x_2^2 = C$
Group `real_airtrack_add_sub` (real; held out: $+, -$)			
`real_linear_a`	accel.	$x,\ v\ \rightarrow\ a$	$a = c_0 + c_1 x + c_2 v$
`real_linear_h`	energy	$x,\ v$	$c_1 x^2 + c_2 v^2 = C$
`real_double_linear_h`	energy	$x_1,\ x_2,\ v_1,\ v_2$	$c_1 v_1^2 + c_2 v_2^2 + c_3 x_1^2 + c_4(x_2 - x_1)^2 + c_5 x_2^2 = C$
Group `geometry_mul` (mixed; held out: $\times$)			
`circle`	geom.	$x,\ y$	$x^2 + y^2 = C$
`sphere`	geom.	$x,\ y,\ z$	$x^2 + y^2 + z^2 = C$
`real_pend_circle`	geom.	$x,\ y$	$(x - x_0)^2 + (y - y_0)^2 = C$

Tangents. Invariant tasks need the trajectory tangent $\dot{x}_i$ at every sample. Where the recording provides derivative columns they are used. Otherwise, within each trial, a quadratic in time is fit by least squares to a window of five consecutive samples of each variable and its derivative at the sample's own time is taken as the tangent; the window is centered on the sample where possible and kept inside the trial near its ends. Tangents are computed on the full recording before subsampling or angle wrapping.

Subsampling and splits. Each task uses 256 rows sampled uniformly at random (fixed seed) from the recording and then sorted in time. If the recording contains more than one trial, whole trials are held out: the last trial is the test set (for the double-pendulum recordings, trial 0 trains and trial 1 tests). If it contains a single trial, every fifth row in time order is a test row. Trial identifier and time are never inputs to the search.

Angle wrapping and the raw-angle constraint. Angular variables are wrapped to $[-\pi, \pi)$ after tangents are estimated, which removes accumulated revolutions from the inputs without introducing spurious derivatives at the wrap. On the two pendulum groups we additionally forbid, in all four arms, any product whose one side contains a raw angle (an angle that has not passed through a unary operator) and whose other side depends on any variable. Products of an angle with a constant remain allowed, as do all uses of angular velocities, and passing an angle through a unary operator ($\sin$, $\cos$, or φ_θ) lifts the restriction from that subexpression. The rule is checked on the tree structure. The reason is empirical: without it the ablated arm replaces $\sin\theta$ by a low-order polynomial in θ that fits the trajectories almost as well over the observed angular range, so removing the primitive barely changes the loss. With the rule and the physical library $\{+, -, \times, \sin, \cos\}$, a raw angle can enter an expression only linearly, through sums and constant multiples, so in the ablated arm the nonlinear dependence cannot be expressed and in the learnable arms it must go through φ_θ. Two further constraints apply to the pendulum groups in all arms: a trigonometric operator may not be nested inside another, and a learnable operator may not be nested inside another.

Invariance score. For an invariant task let $s \in \mathbb{R}^d$ be the per-variable standard deviations of the training inputs, and for a candidate H write $\tilde{g}_i = s \odot \nabla H(x_i)$ and $\tilde{t}_i = \dot{x}_i \oslash s$ for the gradient and tangent rescaled to unit-free form ($\odot$, $\oslash$: elementwise product and division). Samples whose rescaled tangent is negligible are excluded; call the rest S. A sample is *active* if its rescaled gradient is not negligible; call the active set A. Then

$$\mathcal{L}_{\mathrm{inv}}(H) = \begin{cases} \dfrac{1}{|A|} \displaystyle\sum_{i \in A} \dfrac{\langle \tilde{g}_i, \tilde{t}_i \rangle^2}{\|\tilde{g}_i\|^2 \, \|\tilde{t}_i\|^2} & \text{if } |A|/|S| \geq 0.9, \\ +\infty & \text{otherwise,} \end{cases}$$

the mean squared cosine between the gradient of H and the direction of motion. It is adapted from the criterion of Schmidt & Lipson (2009), which compares ratios of partial derivatives of a candidate with ratios of time derivatives estimated from the data for pairs of variables; the squared cosine uses all variables at once. It is 0 exactly when the gradient of H is orthogonal to the estimated direction of motion at every sampled point of the active set, which holds when H is conserved along the trajectory; with finitely many samples the converse can fail. It is 1 when the gradient is parallel to the motion; it is unchanged by rescaling H or changing the units of any variable. The coverage rule rejects the constant function and any candidate that is flat on more than a tenth of the data. Gradients of candidates are central finite differences with step $10^{-3} s_j$ in variable j. Acceleration tasks are scored by mean squared error divided by the variance of the target. The loss that drives a joint search is the unweighted mean over the group's tasks of their training scores; any task with a non-finite score makes the candidate's loss infinite. Train and test scores are recorded per task.

C Analysis Protocol

What is recorded. Every run saves its configuration and seeds, its final Pareto front with each candidate stored so that it can be re-evaluated (expression, operator definitions and all learned parameters), the loss trajectory over iterations, and, for the physical tasks, per-task train and test scores. For each occurrence of φ_θ in a front candidate the run also stores the inputs that occurrence received on the training and test rows and its outputs. Runs that fail are counted and excluded from medians but listed in Appendix D.

Candidate choice. Every figure uses the expression the engine itself selects from the final front by its standard trade-off between loss and size. We also record the front member of lowest training loss.

Fit. For the physical tasks the per-task test score of the selected candidate is read directly. For the synthetic sets no test score is recorded during the search; the selected candidate is re-evaluated on the held-out draw, giving the test mean squared error averaged over outputs. Fig. 3 plots each seed's value divided by the median ABLATED value on the same set or task; values above 2 are drawn at the top of the axis and failed runs are marked separately.

Operator recovery. The recorded inputs of each φ_θ occurrence are passed to every primitive of the same arity, and each is flagged as available or held out in that run. Per occurrence, and pooled over all occurrences of φ_θ, we compute the coefficient of determination of the least-squares fit learned $\approx \alpha$ primitive $+ \beta$ (the affine R^2, equal to the squared correlation) with the fitted α, β, and the root mean squared error normalized by the primitive's range. The held-out score is this pooled affine R^2 against the held-out primitive on the inputs the operator received on the *test* rows, with the affine map fitted on those rows. On the physical tasks whose test set is a whole held-out trial (the double pendulum, the double air track, `sphere`) these inputs lie largely outside the range the operator was fitted on, so the score also measures extrapolation. We therefore also report a training-row score: the same R^2 on the *training* rows, pooled over occurrences, whose fitted map is then also applied unchanged to the test rows. Table 1 gives the held-out counts and Table 10 the training-row ones, and every figure or table that uses the training-row score says so. When $\sin$ and $\cos$ are deleted we take the better of $\sin$ and $\cos$ and say which. For binary operators we also regress the learned outputs on $\{1, a, b\}$ (addition and subtraction) or $\{1, a, b, ab\}$ (multiplication) as a diagnostic only: a learned $\lambda a + \mu b + c$ with $\lambda \neq \mu$ is linear in its inputs but is addition only after its inputs are rescaled, so it does not count as a recovery. All recovery counts use the affine fit of the output against $a + b$, $a - b$ or ab. Fig. 6 shows the median-loss seed; the strips show all seeds.

Structure. The law check asks whether the selected expressions are the target laws with φ_θ in place of the deleted function, up to the values of their constants. On the synthetic sets the target laws have fixed constants, and the check is numerical. The held-out primitive replaces φ_θ, the constants are refitted by Levenberg–Marquardt, and the check passes when the relative RMSE is at most 10^{-6} on 1000 points in the training range and on 1000 points over three times its width (with at least 99% of predictions finite, so a law undefined on fewer than 1% of the points can pass), and when no term that the refit sets to zero would change the in-range prediction by more than 10^{-3} in relative RMSE if restored. The last condition rejects expressions that keep a term with a coefficient near zero, such as $-0.0017\, x_1 x_2 (x_1 + x_2)$ inside φ_θ. No ABLATED run passes, and every run that passes for the square root also matches $\sqrt{\cdot}$ at held-out $R^2 \geq 0.99$. Table 6 gives the per-set counts. On the physical tasks the expected laws of Table 5 have unknown coefficients and φ_θ is not exactly the deleted function, so the check compares forms. The selected expression and the expected law, with φ_θ in place of the deleted function, are expanded into sums of terms. A term is dropped when removing it changes the expression on the training rows by less than 10^{-3} in relative RMSE, with φ_θ evaluated as the primitive it matches best for this purpose only, and the check passes when the remaining terms are those of the expected law, with any coefficients and constant. The pendulum energy, for example, passes when it has the form $c_0 + c_1\omega^2 + c_2\varphi_\theta(\theta)$. For the physical tasks we also report how many inner expressions use φ_θ and list the selected expressions.

Cost and aggregation. Wall-clock time is compared with the ORACLE median for the same set or group. With five seeds per condition we plot every seed and report medians; arms are not paired, so no paired tests are used. An operator is *recovered* if its pooled test-row affine $R^2 \geq 0.99$. For every +BOP run we also report the fraction of in-context inputs outside the spline support $[0, 30]$.

D Additional Results

Arms are named as in Section 3, seeds are 1001–1005, and all losses are test losses of the selected expression unless stated.

D.1 Per-seed results and structural recovery

Table 6 gives, per equation set or task group and learnable arm, how many seeds recovered the target skeleton, how many equations used the learned operator, its shape score and its cost relative to the oracle. Tables 7, 8 and 9 list every seed's test loss for the physical groups, the synthetic sets and the individual physical tasks.

Table 6: Structure recovery varies widely across sets, while φ_θ appears in most inner expressions of every set and group. Structure: runs whose selected equations pass the law check of Section 5.1 once the held-out primitive replaces φ_θ. Inners w/ φ_θ: median number of inner expressions that use the learnable operator / total. Affine R^2: training-row affine R^2 of the learnable operator against the held-out primitive, pooled over all its occurrences in the selected candidate, median [min, max] over seeds. Time: median wall-clock relative to the oracle arm.

Experiment	Held out	Arm	ok	Structure	Inners w/ φ_θ	Affine R^2 (train)	Time $\times$
sqrt_pair	sqrt	+BOP	5/5	1/5	2/2	1.00 [0.58, 1.00]	4.04
sqrt_pair	sqrt	+NOP	5/5	3/5	2/2	1.00 [0.98, 1.00]	7.89
sqrt_pair	+/-	+BOP	5/5	2/5	2/2	1.00 [0.91, 1.00]	5.68
sqrt_pair	+/-	+NOP	5/5	5/5	2/2	1.00 [1.00, 1.00]	5.55
sqrt_trio	sqrt	+BOP	5/5	4/5	3/3	1.00 [1.00, 1.00]	3.73
sqrt_trio	sqrt	+NOP	5/5	2/5	3/3	1.00 [1.00, 1.00]	8.87
sqrt_trio	+/-	+BOP	4/5	0/4	3/3	0.85 [0.46, 0.97]	7.75
sqrt_trio	+/-	+NOP	5/5	5/5	3/3	1.00 [1.00, 1.00]	5.17
sqrt_quartet	sqrt	+BOP	5/5	1/5	4/4	0.99 [0.97, 1.00]	2.65
sqrt_quartet	sqrt	+NOP	5/5	0/5	3/4	1.00 [0.98, 1.00]	6.50
sqrt_quartet	+/-	+BOP	4/5	0/4	4/4	0.96 [0.96, 0.99]	6.10
sqrt_quartet	+/-	+NOP	5/5	0/5	4/4	0.99 [0.96, 0.99]	4.71
sqrt_pair_coeffs	sqrt	+BOP	5/5	0/5	2/2	1.00 [0.00, 1.00]	2.44
sqrt_pair_coeffs	sqrt	+NOP	5/5	4/5	2/2	1.00 [0.99, 1.00]	6.58
sqrt_pair_coeffs	+/-	+BOP	5/5	0/5	2/2	0.94 [0.79, 0.96]	5.38
sqrt_pair_coeffs	+/-	+NOP	5/5	2/5	2/2	0.90 [0.86, 1.00]	3.65
sqrt_trig_pair	sqrt	+BOP	5/5	2/5	2/2	1.00 [0.17, 1.00]	2.09
sqrt_trig_pair	sqrt	+NOP	5/5	3/5	2/2	1.00 [1.00, 1.00]	5.30
sqrt_trig_pair	+/-	+BOP	5/5	0/5	2/2	0.30 [0.02, 0.42]	6.05
sqrt_trig_pair	+/-	+NOP	5/5	2/5	2/2	0.86 [0.83, 1.00]	4.01
sqrt_poly_trio	sqrt	+BOP	5/5	0/5	3/3	0.95 [0.85, 1.00]	1.83
sqrt_poly_trio	sqrt	+NOP	5/5	1/5	3/3	0.95 [0.94, 1.00]	5.68
sqrt_poly_trio	+/-	+BOP	5/5	0/5	3/3	0.48 [0.28, 0.72]	3.22
sqrt_poly_trio	+/-	+NOP	5/5	0/5	3/3	0.92 [0.83, 0.99]	2.59
sqrt_log_pair	sqrt	+BOP	5/5	0/5	1/2	0.97 [0.86, 1.00]	2.37
sqrt_log_pair	sqrt	+NOP	5/5	0/5	2/2	1.00 [0.97, 1.00]	8.94
sqrt_log_pair	+/-	+BOP	5/5	0/5	2/2	0.84 [0.48, 0.99]	5.22
sqrt_log_pair	+/-	+NOP	5/5	0/5	2/2	0.97 [0.96, 0.98]	2.88
sqrt_exp_trio	sqrt	+BOP	5/5	0/5	3/3	0.93 [0.86, 0.97]	3.05
sqrt_exp_trio	sqrt	+NOP	5/5	0/5	3/3	0.96 [0.90, 0.97]	6.55
sqrt_exp_trio	+/-	+BOP	5/5	0/5	3/3	0.89 [0.14, 0.97]	4.38
sqrt_exp_trio	+/-	+NOP	5/5	0/5	3/3	0.95 [0.82, 0.99]	3.23
sqrt_mixed_pair	sqrt	+BOP	5/5	4/5	2/2	1.00 [0.92, 1.00]	2.80
sqrt_mixed_pair	sqrt	+NOP	5/5	1/5	2/2	0.99 [0.95, 1.00]	5.55
sqrt_mixed_pair	+/-	+BOP	5/5	0/5	2/2	0.63 [0.52, 0.74]	3.57
sqrt_mixed_pair	+/-	+NOP	5/5	0/5	2/2	1.00 [0.92, 1.00]	3.92
sqrt_mixed_quartet	sqrt	+BOP	5/5	0/5	2/4	1.00 [1.00, 1.00]	2.17
sqrt_mixed_quartet	sqrt	+NOP	5/5	0/5	3/4	0.96 [0.95, 1.00]	6.39
sqrt_mixed_quartet	+/-	+BOP	5/5	0/5	3/4	0.94 [0.87, 0.97]	4.64
sqrt_mixed_quartet	+/-	+NOP	5/5	0/5	3/4	0.47 [0.11, 1.00]	2.58
real_pendulum_trig	sin/cos	+BOP	5/5	n/a	2/3	0.99 [0.84, 1.00]	1.48
real_pendulum_trig	sin/cos	+NOP	5/5	n/a	2/3	0.98 [0.19, 0.99]	2.27
synthetic_pendulum_trig	sin/cos	+BOP	5/5	n/a	1/3	0.66 [0.20, 1.00]	1.06
synthetic_pendulum_trig	sin/cos	+NOP	5/5	n/a	3/3	0.74 [0.59, 0.99]	1.86
real_airtrack_add_sub	+/-	+BOP	5/5	n/a	1/3	0.27 [0.09, 0.35]	2.63
real_airtrack_add_sub	+/-	+NOP	5/5	n/a	2/3	0.54 [0.45, 0.76]	2.73
synthetic_airtrack_add_sub	+/-	+BOP	5/5	n/a	3/3	0.23 [0.08, 0.56]	2.36
synthetic_airtrack_add_sub	+/-	+NOP	5/5	n/a	3/3	0.74 [0.27, 0.92]	2.21
geometry_mul	*	+BOP	5/5	n/a	2/3	0.82 [0.34, 0.83]	2.86
geometry_mul	*	+NOP	5/5	n/a	2/3	0.77 [0.74, 0.87]	2.89

Table 7: Physical task groups: mean test score per seed.

Group	Arm	Median	s1001	s1002	s1003	s1004	s1005
real_pendulum_trig	oracle	0.053	0.039	0.053	0.037	0.077	0.066
real_pendulum_trig	ablated	0.15	0.15	0.14	0.16	0.14	0.16
real_pendulum_trig	+BOP	0.061	0.071	0.053	0.061	0.056	0.065
real_pendulum_trig	+NOP	0.046	0.059	0.039	0.033	0.070	0.046
synthetic_pendulum_trig	oracle	0.056	0.067	0.056	0.056	0.15	0.050
synthetic_pendulum_trig	ablated	0.14	0.12	0.14	0.15	0.14	0.14
synthetic_pendulum_trig	+BOP	0.061	0.061	0.067	0.049	0.061	0.061
synthetic_pendulum_trig	+NOP	0.063	0.063	0.18	0.063	0.15	0.058
real_airtrack_add_sub	oracle	0.031	0.031	0.031	0.031	0.031	0.031
real_airtrack_add_sub	ablated	0.22	0.22	0.22	0.22	0.22	0.22
real_airtrack_add_sub	+BOP	0.11	0.14	0.12	0.10	0.11	0.10
real_airtrack_add_sub	+NOP	0.019	0.016	0.019	0.011	0.024	0.029
synthetic_airtrack_add_sub	oracle	0.036	0.036	0.036	0.036	0.14	0.036
synthetic_airtrack_add_sub	ablated	0.47	0.47	0.47	0.47	0.47	0.47
synthetic_airtrack_add_sub	+BOP	0.054	0.058	0.078	0.035	0.054	0.051
synthetic_airtrack_add_sub	+NOP	0.047	0.054	0.047	0.057	0.041	0.045
geometry_mul	oracle	0.11	0.11	0.066	0.11	0.11	0.11
geometry_mul	ablated	0.28	0.31	0.28	0.28	0.28	0.28
geometry_mul	+BOP	0.22	0.22	0.26	0.15	0.15	0.22
geometry_mul	+NOP	0.18	0.079	0.18	0.18	0.18	0.18

Table 8: Synthetic equation sets: test MSE of the selected candidate per seed.

Set	Held out	Arm	Median	s1001	s1002	s1003	s1004	s1005
sqrt_pair	none	oracle	0	0	0	0	0	0
sqrt_pair	sqrt	ablated	3.9×10^{-3}	3.9×10^{-3}	3.9×10^{-3}	2.5×10^{-3}	3.9×10^{-3}	3.9×10^{-3}
sqrt_pair	sqrt	+BOP	2.3×10^{-5}	2.3×10^{-5}	2.1×10^{-5}	4.7×10^{-6}	3.0×10^{-5}	2.6×10^{-5}
sqrt_pair	sqrt	+NOP	3.5×10^{-4}	3.9×10^{-4}	2.3×10^{-4}	1.6×10^{-4}	4.3×10^{-4}	3.5×10^{-4}
sqrt_pair	+/-	ablated	0.23	0.23	0.23	0.23	0.23	0.23
sqrt_pair	+/-	+BOP	8.6×10^{-4}	2.8×10^{-3}	7.2×10^{-4}	8.6×10^{-4}	2.3×10^{-4}	3.2×10^{-3}
sqrt_pair	+/-	+NOP	1.4×10^{-7}	4.4×10^{-8}	1.4×10^{-7}	1.3×10^{-7}	1.5×10^{-7}	1.3×10^{-6}
sqrt_trio	none	oracle	0	0	0	0	0	0
sqrt_trio	sqrt	ablated	0.017	0.017	0.012	0.017	0.017	0.017
sqrt_trio	sqrt	+BOP	2.0×10^{-5}	9.8×10^{-6}	4.6×10^{-5}	3.0×10^{-5}	6.8×10^{-6}	2.0×10^{-5}
sqrt_trio	sqrt	+NOP	4.7×10^{-4}	5.4×10^{-4}	5.1×10^{-4}	3.3×10^{-4}	4.7×10^{-4}	3.8×10^{-4}
sqrt_trio	+/-	ablated	0.26	0.26	0.26	0.32	0.26	0.26
sqrt_trio	+/-	+BOP	8.6×10^{-4}	4.8×10^{-4}	7.2×10^{-4}	*nonfinite*	1.0×10^{-3}	3.5×10^{-3}
sqrt_trio	+/-	+NOP	7.1×10^{-7}	3.0×10^{-7}	1.8×10^{-6}	1.0×10^{-6}	5.2×10^{-7}	7.1×10^{-7}
sqrt_quartet	none	oracle	0	0	0	0	0	0
sqrt_quartet	sqrt	ablated	0.18	0.18	0.18	0.18	0.18	0.18
sqrt_quartet	sqrt	+BOP	0.065	0.065	0.025	0.16	4.5×10^{-4}	0.095
sqrt_quartet	sqrt	+NOP	0.16	0.14	0.16	0.17	0.16	0.19
sqrt_quartet	+/-	ablated	1.38	1.38	1.38	1.38	1.38	1.15
sqrt_quartet	+/-	+BOP	0.034	0.039	0.023	0.030	0.068	*nonfinite*
sqrt_quartet	+/-	+NOP	0.016	0.010	0.013	0.021	0.016	0.018
sqrt_pair_coeffs	none	oracle	0	0	0	0	0	0
sqrt_pair_coeffs	sqrt	ablated	0.014	0.014	0.014	0.014	0.014	0.014
sqrt_pair_coeffs	sqrt	+BOP	5.8×10^{-5}	3.3×10^{-4}	5.8×10^{-5}	2.4×10^{-5}	1.7×10^{-5}	5.9×10^{-5}
sqrt_pair_coeffs	sqrt	+NOP	5.0×10^{-4}	1.8×10^{-4}	7.5×10^{-4}	4.1×10^{-4}	6.6×10^{-4}	5.0×10^{-4}
sqrt_pair_coeffs	+/-	ablated	0.50	0.50	0.50	0.50	0.50	0.50
sqrt_pair_coeffs	+/-	+BOP	1.9×10^{-3}	1.1×10^{-3}	1.9×10^{-3}	5.3×10^{-5}	5.4×10^{-3}	2.6×10^{-3}
sqrt_pair_coeffs	+/-	+NOP	8.0×10^{-7}	2.3×10^{-6}	1.6×10^{-7}	8.3×10^{-7}	2.4×10^{-7}	8.0×10^{-7}
sqrt_trig_pair	none	oracle	0	0	0	0	0	0
sqrt_trig_pair	sqrt	ablated	6.3×10^{-3}	0.11	6.3×10^{-3}	0.014	4.3×10^{-3}	6.3×10^{-3}
sqrt_trig_pair	sqrt	+BOP	1.7×10^{-3}	1.7×10^{-3}	8.8×10^{-6}	0.073	0.090	8.0×10^{-6}
sqrt_trig_pair	sqrt	+NOP	2.7×10^{-4}	3.6×10^{-3}	2.7×10^{-4}	3.1×10^{-4}	1.5×10^{-4}	1.2×10^{-4}
sqrt_trig_pair	+/-	ablated	0.61	0.61	0.61	0.43	0.43	0.61
sqrt_trig_pair	+/-	+BOP	0.012	0.012	0.010	5.5×10^{-3}	0.14	0.024
sqrt_trig_pair	+/-	+NOP	4.4×10^{-3}	7.1×10^{-7}	7.3×10^{-3}	4.4×10^{-3}	2.1×10^{-6}	9.4×10^{-3}
sqrt_poly_trio	none	oracle	0.031	0.030	0.086	0.031	0.046	0
sqrt_poly_trio	sqrt	ablated	0.075	0.94	0.075	0.075	0.075	0.075
sqrt_poly_trio	sqrt	+BOP	0.072	0.074	0.072	0.081	0.013	0.017
sqrt_poly_trio	sqrt	+NOP	0.043	0.043	0.098	0.036	0.043	6.3×10^{-3}
sqrt_poly_trio	+/-	ablated	1.18	1.18	1.26	1.83	1.18	1.18
sqrt_poly_trio	+/-	+BOP	0.25	0.10	0.21	0.25	0.45	0.29
sqrt_poly_trio	+/-	+NOP	0.028	0.028	0.11	0.011	0.031	0.013
sqrt_log_pair	none	oracle	0	0	0	0	0	9.5×10^{-15}
sqrt_log_pair	sqrt	ablated	5.2×10^{-3}	0.020	5.2×10^{-3}	4.0×10^{-3}	7.2×10^{-3}	4.5×10^{-3}
sqrt_log_pair	sqrt	+BOP	2.9×10^{-4}	3.5×10^{-5}	1.6×10^{-3}	2.9×10^{-4}	8.1×10^{-5}	1.1×10^{-3}
sqrt_log_pair	sqrt	+NOP	3.3×10^{-3}	4.0×10^{-3}	2.4×10^{-3}	3.3×10^{-3}	3.4×10^{-3}	1.9×10^{-3}
sqrt_log_pair	+/-	ablated	0.013	0.013	0.013	0.013	0.013	0.013
sqrt_log_pair	+/-	+BOP	3.8×10^{-5}	1.2×10^{-4}	3.8×10^{-5}	1.2×10^{-5}	1.1×10^{-4}	2.2×10^{-5}
sqrt_log_pair	+/-	+NOP	1.8×10^{-4}	1.8×10^{-4}	3.6×10^{-4}	1.4×10^{-4}	4.8×10^{-5}	2.6×10^{-4}
sqrt_exp_trio	none	oracle	0.016	0.014	*nonfinite*	3.3×10^{-3}	0.018	0.019
sqrt_exp_trio	sqrt	ablated	0.047	0.038	0.047	0.058	0.029	0.047
sqrt_exp_trio	sqrt	+BOP	0.023	0.014	0.033	0.037	0.023	0.015
sqrt_exp_trio	sqrt	+NOP	0.022	0.020	0.022	0.018	0.023	0.027
sqrt_exp_trio	+/-	ablated	0.067	0.093	0.065	0.11	0.047	0.067
sqrt_exp_trio	+/-	+BOP	3.6×10^{-3}	4.7×10^{-3}	3.6×10^{-3}	3.1×10^{-3}	5.2×10^{-3}	2.4×10^{-3}
sqrt_exp_trio	+/-	+NOP	4.4×10^{-3}	3.6×10^{-3}	7.7×10^{-3}	4.4×10^{-3}	3.4×10^{-3}	4.9×10^{-3}
sqrt_mixed_pair	none	oracle	0	0	0	0	0	0.27
sqrt_mixed_pair	sqrt	ablated	0.58	0.58	0.36	0.53	7.82	0.58
sqrt_mixed_pair	sqrt	+BOP	3.1×10^{-5}	5.9×10^{-5}	3.1×10^{-5}	1.6×10^{-5}	7.9×10^{-6}	0.25
sqrt_mixed_pair	sqrt	+NOP	0.35	0.12	3.52	0.41	0.35	0.21
sqrt_mixed_pair	+/-	ablated	29.88	29.88	29.88	29.88	37.42	29.88
sqrt_mixed_pair	+/-	+BOP	1.53	1.53	6.16	0.11	0.076	4.00
sqrt_mixed_pair	+/-	+NOP	0.18	0.18	0.45	0.27	0.042	0.12
sqrt_mixed_quartet	none	oracle	0.022	0.095	0.027	0.018	4.6×10^{-3}	0.022
sqrt_mixed_quartet	sqrt	ablated	0.35	0.42	1.15	0.35	0.24	0.31
sqrt_mixed_quartet	sqrt	+BOP	0.10	0.098	0.14	0.15	0.10	0.082
sqrt_mixed_quartet	sqrt	+NOP	0.14	0.14	0.084	0.32	0.060	0.72
sqrt_mixed_quartet	+/-	ablated	0.18	0.18	0.18	0.18	0.18	0.18
sqrt_mixed_quartet	+/-	+BOP	0.012	0.018	7.4×10^{-3}	0.012	0.012	0.015
sqrt_mixed_quartet	+/-	+NOP	7.8×10^{-3}	0.015	0.019	3.1×10^{-3}	7.8×10^{-3}	6.6×10^{-3}

Table 9: Physical tasks: test score per seed.

Task	Arm	Median	s1001	s1002	s1003	s1004	s1005
real_pend_a	oracle	1.3×10^{-3}	1.3×10^{-3}	1.3×10^{-3}	6.4×10^{-3}	3.4×10^{-3}	9.9×10^{-4}
real_pend_a	ablated	7.0×10^{-3}	0.010	7.0×10^{-3}	7.0×10^{-3}	7.0×10^{-3}	7.0×10^{-3}
real_pend_a	+BOP	7.0×10^{-3}	7.0×10^{-3}	0.010	7.0×10^{-3}	6.5×10^{-3}	6.9×10^{-3}
real_pend_a	+NOP	7.0×10^{-3}	7.0×10^{-3}	6.5×10^{-3}	6.4×10^{-3}	7.0×10^{-3}	7.0×10^{-3}
real_pend_h	oracle	1.9×10^{-3}	1.1×10^{-3}	2.0×10^{-3}	1.9×10^{-3}	4.4×10^{-4}	3.1×10^{-3}
real_pend_h	ablated	0.24	0.24	0.24	0.28	0.24	0.28
real_pend_h	+BOP	2.5×10^{-3}	2.2×10^{-3}	3.7×10^{-3}	4.4×10^{-3}	2.4×10^{-3}	2.5×10^{-3}
real_pend_h	+NOP	6.1×10^{-3}	4.9×10^{-3}	6.8×10^{-3}	6.1×10^{-3}	0.014	3.3×10^{-3}
real_double_pend_h	oracle	0.16	0.11	0.16	0.10	0.23	0.19
real_double_pend_h	ablated	0.19	0.19	0.19	0.19	0.19	0.19
real_double_pend_h	+BOP	0.17	0.20	0.14	0.17	0.16	0.19
real_double_pend_h	+NOP	0.13	0.17	0.10	0.087	0.19	0.13
pendulum_a	oracle	1.0×10^{-3}	8.3×10^{-4}	1.0×10^{-3}	7.3×10^{-4}	0.015	5.4×10^{-3}
pendulum_a	ablated	0.015	0.015	0.015	0.015	0.033	0.015
pendulum_a	+BOP	0.015	0.015	0.015	7.9×10^{-3}	0.015	0.015
pendulum_a	+NOP	0.013	0.015	3.5×10^{-3}	0.016	0.013	5.8×10^{-3}
pendulum_h	oracle	5.6×10^{-5}	5.6×10^{-5}	6.5×10^{-6}	1.9×10^{-5}	2.7×10^{-3}	2.8×10^{-3}
pendulum_h	ablated	0.23	0.19	0.23	0.28	0.21	0.25
pendulum_h	+BOP	2.0×10^{-3}	2.0×10^{-3}	0.020	1.4×10^{-3}	3.1×10^{-3}	5.7×10^{-4}
pendulum_h	+NOP	6.6×10^{-3}	7.7×10^{-3}	6.6×10^{-3}	8.7×10^{-3}	2.3×10^{-3}	2.6×10^{-3}
double_pend_h	oracle	0.17	0.20	0.17	0.17	0.42	0.14
double_pend_h	ablated	0.17	0.17	0.17	0.17	0.16	0.16
double_pend_h	+BOP	0.17	0.16	0.17	0.14	0.17	0.17
double_pend_h	+NOP	0.17	0.17	0.52	0.17	0.44	0.17
real_linear_a	oracle	8.0×10^{-4}	8.0×10^{-4}	8.0×10^{-4}	8.0×10^{-4}	8.0×10^{-4}	8.0×10^{-4}
real_linear_a	ablated	8.0×10^{-4}	8.0×10^{-4}	8.0×10^{-4}	8.0×10^{-4}	8.0×10^{-4}	8.0×10^{-4}
real_linear_a	+BOP	8.0×10^{-4}	7.5×10^{-4}	1.4×10^{-3}	8.0×10^{-4}	8.0×10^{-4}	8.0×10^{-4}
real_linear_a	+NOP	8.0×10^{-4}	7.5×10^{-4}	8.0×10^{-4}	8.0×10^{-4}	8.5×10^{-4}	7.5×10^{-4}
real_linear_h	oracle	3.4×10^{-4}	3.4×10^{-4}	3.4×10^{-4}	3.4×10^{-4}	3.4×10^{-4}	3.4×10^{-4}
real_linear_h	ablated	0.49	0.49	0.49	0.49	0.49	0.49
real_linear_h	+BOP	0.18	0.26	0.20	0.16	0.18	0.15
real_linear_h	+NOP	0.011	9.2×10^{-3}	0.011	0.010	0.014	0.018
real_double_linear_h	oracle	0.091	0.091	0.091	0.091	0.091	0.091
real_double_linear_h	ablated	0.16	0.16	0.16	0.16	0.16	0.16
real_double_linear_h	+BOP	0.16	0.16	0.16	0.16	0.16	0.16
real_double_linear_h	+NOP	0.047	0.038	0.047	0.023	0.057	0.068
linear_a	oracle	4.1×10^{-3}	4.1×10^{-3}	4.1×10^{-3}	4.1×10^{-3}	4.1×10^{-3}	4.1×10^{-3}
linear_a	ablated	0.78	0.78	0.78	0.78	0.78	0.78
linear_a	+BOP	0.035	0.035	0.10	0.035	7.5×10^{-3}	0.032
linear_a	+NOP	6.1×10^{-3}	4.1×10^{-3}	7.2×10^{-3}	10.0×10^{-3}	6.1×10^{-3}	6.1×10^{-3}
linear_h	oracle	1.2×10^{-3}	1.2×10^{-3}	1.2×10^{-3}	1.2×10^{-3}	0.31	1.2×10^{-3}
linear_h	ablated	0.44	0.44	0.44	0.44	0.44	0.44
linear_h	+BOP	0.038	0.043	0.11	0.018	0.038	0.016
linear_h	+NOP	0.026	0.026	0.067	0.12	8.2×10^{-3}	0.018
double_linear_h	oracle	0.10	0.10	0.10	0.10	0.10	0.10
double_linear_h	ablated	0.20	0.20	0.20	0.20	0.20	0.20
double_linear_h	+BOP	0.097	0.097	0.029	0.052	0.12	0.10
double_linear_h	+NOP	0.11	0.13	0.066	0.039	0.11	0.11
circle	oracle	3.5×10^{-10}	3.5×10^{-10}	3.5×10^{-10}	3.5×10^{-10}	3.5×10^{-10}	3.5×10^{-10}
circle	ablated	0.46	0.46	0.46	0.46	0.46	0.46
circle	+BOP	0.022	3.7×10^{-3}	0.15	0.044	0.022	2.7×10^{-3}
circle	+NOP	0.026	0.010	0.026	0.035	0.025	0.040
sphere	oracle	0.31	0.31	0.19	0.31	0.31	0.31
sphere	ablated	0.31	0.31	0.31	0.31	0.31	0.31
sphere	+BOP	0.31	0.31	0.31	0.31	0.31	0.31
sphere	+NOP	0.31	0.026	0.31	0.31	0.31	0.31
real_pend_circle	oracle	9.4×10^{-3}	9.4×10^{-3}	0.010	9.0×10^{-3}	9.4×10^{-3}	9.4×10^{-3}
real_pend_circle	ablated	0.064	0.15	0.064	0.064	0.064	0.064
real_pend_circle	+BOP	0.34	0.35	0.34	0.084	0.11	0.35
real_pend_circle	+NOP	0.20	0.20	0.19	0.20	0.20	0.19

D.2 FIXED-GRID SHAPE SCORES AND OPERAND COVERAGE

We re-evaluated every stored learned operator from its saved parameters. The re-evaluation reproduces the stored operator outputs to within 0.1% of their range in 245 of the 250 learnable-arm runs. The five that do not are the spline runs on the real air-track group, whose binary splines received 85 to 97% of their inputs far outside the support $[0, 30]$; there the re-evaluation differs by 0.3 to 26% of the output range. Every score reported elsewhere in this paper, for these runs as for all others, uses the operator outputs stored by the search itself, so these five runs are excluded from this subsection only.

Fixed-grid shape. A unary operator is scored on 201 evenly spaced points over the run's own training input range and over a common range, $[0, 20]$ for the square root and $[-\pi, \pi]$ for the angle; a binary operator on a 41×41 grid over the box spanned by its training operands. On each grid a new affine map is fitted and its R^2 reported. The spline is evaluated with its polynomial continuation outside $[0, 30]$; restricting the grids to $[0, 30]$ changes no square-root count and any other count by at most one. Table 10 gives the counts. On their own input range the operators match about as often as under the pooled score. On the common square-root range the spline falls from 31 to 9 of 50: its training inputs end at a median of 10.0 (the MLP's at 14.3), so in the median spline run half of $[0, 20]$ lies beyond its data, where it continues as a cubic. The MLP, whose rectified units continue linearly, keeps 49 of 50 at $R^2 \geq 0.95$. On the real pendulum the pooled and own-range counts agree at 4 of 10 but differ by one run in each direction (spline seed 1004: pooled 0.992, grid 0.972; MLP seed 1003: pooled 0.989, grid 0.990). On $[-\pi, \pi]$ no operator reaches 0.99. Table 16 gives the grid scores of every run.

Table 10: On a grid over each run's own input range the learned operators match the deleted primitive about as often as under the pooled score; on a common range that extends beyond the data, far fewer do. Runs matching at affine $R^2 \geq 0.99$, with the count at 0.95 in parentheses. *Pooled*: the training-row score (Appendix C). *Own range*: 201 points over the run's training input range, or a 41×41 grid over its operand box. *Common range*: $[0, 20]$ for $\sqrt{\cdot}$, $[-\pi, \pi]$ for the angle. The physical binary row excludes the five real air-track spline runs that could not be re-evaluated.

Deleted	Setting	Arm	Pooled	Own range	Common range
$\sqrt{\cdot}$	10 synthetic sets	+BOP	33/50 (39)	31/50 (38)	9/50 (15)
		+NOP	32/50 (46)	32/50 (49)	20/50 (49)
$\sin, \cos$	real pendulum	+BOP	3/5 (3)	2/5 (3)	0/5 (0)
		+NOP	1/5 (4)	2/5 (4)	0/5 (5)
$\sin, \cos$	simulated pendulum	+BOP	1/5 (2)	1/5 (2)	0/5 (0)
		+NOP	1/5 (2)	1/5 (2)	0/5 (2)
$+, -$	10 synthetic sets	+BOP	3/50 (14)	1/50 (9)	n/a
		+NOP	21/50 (33)	20/50 (33)	n/a
$+, -; \times$	physical groups	both	0/25 (0)	0/25 (0)	n/a

Operand coverage. For each binary run we bin its training operands on a 20×20 grid over their box and count the occupied bins, and we flag an operand as constant (at most one distinct value, or no spread) and the pair as lying on a curve (ratio of the second to the first singular value of the centered operands below 0.01, or $|\text{Spearman}| > 0.99$). None of the 125 binary runs is flagged; the largest $|\text{Spearman}|$ is 0.84. The median occupied fraction is 0.91 (spline) and 0.78 (MLP) on the synthetic sets, 0.95 for the 21 MLP runs that match addition or subtraction, 0.19 and 0.24 on the simulated air track, 0.43 for the MLP on the real air track, and 0.11 on the geometry group, so the physical binary operators saw a much smaller part of their operand plane. One caveat applies to the grid scores: in four MLP runs on `sqrt_mixed_pair` one side of the operand box is 11 to 133 times longer than the other, and in three of them the grid score against $a - b$ also exceeds 0.99, so there the grid does not separate addition from the longer operand alone. The other 16 MLP grid matches have box sides within a factor of 4.

Reuse and recovery. Table 11 counts recoveries by the number of equations of the selected set that use φ_θ. This is observational: the number of equations is chosen by the search and differs between sets, so the table cannot separate the effect of reuse from that of the set. Recovery does not

rise with the number of occurrences of φ_θ either (Spearman correlation with the pooled score 0.02 for the spline and -0.43 for the MLP on the square-root sets).

Table 11: Square-root recovery is lower in runs whose selected set uses φ_θ in one equation than in runs that use it in two or more, and addition recovery does not rise with the number of equations; the cells are small and confounded with the set. Runs matching the deleted primitive at pooled affine $R^2 \geq 0.99$, by the number of equations that use φ_θ in the selected set.

Deleted	Arm	1 equation	2	3	4
$\sqrt{\cdot}$	+BOP	3/9	17/23	11/15	2/3
$\sqrt{\cdot}$	+NOP	2/5	19/25	11/20	n/a
$+, -$	+BOP	n/a	3/25	0/20	0/5
$+, -$	+NOP	n/a	12/25	8/20	1/5

D.3 Recovered expressions

Tables 12–14 show the selected expression of one seed per set, group and learnable arm, with the learned operator written as φ_θ. Read against the target equations of Table 4, they show what "structure recovered" means in practice and what the failures look like. In these tables the seed is the one with the median training loss, while the figures use the median test loss, so the two can show different seeds. For the geometry spline the median is over the rescored training losses (Appendix A). Table 15 lists the energy expression of every real-pendulum run, and Table 16 gives every run's shape scores.

D.4 The secondary synthetic block: addition and subtraction removed

Fig. 5 is the counterpart of Fig. 3a for the block in which $+$ and $-$ are removed and a binary learnable operator is added. The MLP's median relative loss is below 0.1 on all ten sets and below 10^{-5} on three; the spline is below 0.1 on nine. Shape recovery is asymmetric: the MLP learns addition up to an affine map in 21 of 50 runs (32 at $R^2 \geq 0.95$), the spline in 3 (14).

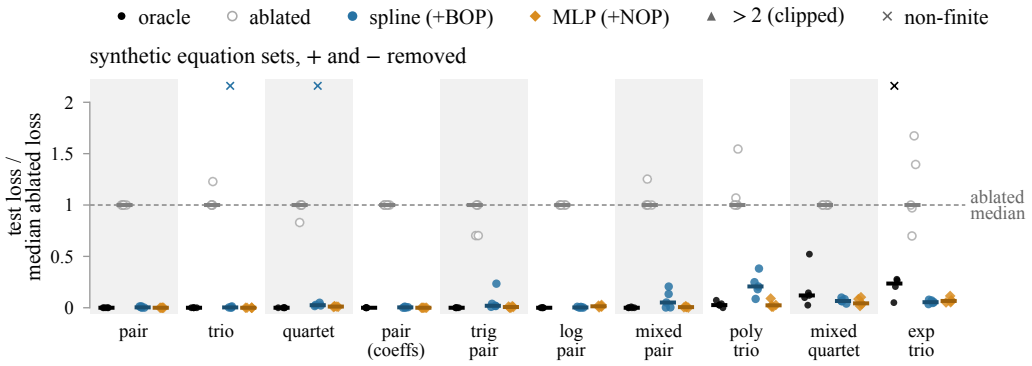

Figure 5: With $+$ and $-$ removed, the binary MLP operator brings the median loss below 0.1 of the ABLATED level on every synthetic set, and the spline on nine of ten. Encoding as in Fig. 3a.

D.5 Learned operators for every seed

Fig. 7 shows, for every set, group and seed, the learned operator on the inputs it received during the search against the held-out primitive rescaled by the training-row affine fit (dashed gray), with the training-row R^2 per arm in each cell. Unary operators come first (the ten square-root sets and the two pendulum groups), then binary operators drawn as learned output against the affine-aligned reference, on which perfect recovery is the diagonal. Cells marked off-scale have one arm's outputs orders of magnitude beyond the other's; their R^2 is still given.

Table 12: Exemplar recovered expressions: synthetic sets, held-out $\sqrt{\cdot}$

Experiment	Held out	Arm	Exemplar (selected candidate, median-loss seed)
sqrt_pair	sqrt	+BOP	f_1: $x_2 + \varphi_\theta(x_1 + \varphi_\theta(\varphi_\theta(x_1 + \varphi_\theta(\varphi_\theta(\varphi_\theta(\varphi_\theta(\varphi_\theta(\varphi_\theta(\varphi_\theta(2\ x_1)))))))) - 0.553)$ f_2: $\varphi_\theta(x_1 + x_2) + 0.164$
sqrt_pair	sqrt	+NOP	f_1: $x_2 + \varphi_\theta(x_1)$ f_2: $\varphi_\theta(x_1 + x_2)$
sqrt_trio	sqrt	+BOP	f_1: $x_2 + \varphi_\theta(x_1 + 0.248)$ f_2: $\varphi_\theta(x_1 + x_2 + 0.249)$ f_3: $x_1 + \varphi_\theta(x_2 + 0.251)$
sqrt_trio	sqrt	+NOP	f_1: $x_2 + \varphi_\theta(x_1)$ f_2: $\varphi_\theta(-0.00172\, x_1\, x_2\, (x_1 + x_2) + x_1 + x_2)$ f_3: $x_1 + \varphi_\theta(x_2)$
sqrt_quartet	sqrt	+BOP	f_1: $x_2 + 1.3\, \varphi_\theta(x_1)$ f_2: $0.0272\, x_1\, x_2 + 2.4$ f_3: $x_1 + 1.29\, \varphi_\theta(x_2)$ f_4: $(1.53\, x_1 + x_2 + 1.44)\, \varphi_\theta(\varphi_\theta(x_2))$
sqrt_quartet	sqrt	+NOP	f_1: $x_2 + \varphi_\theta(x_1)$ f_2: $\varphi_\theta(x_1 + x_2)$ f_3: $x_1 + \varphi_\theta(x_2)$ f_4: $1.46\, x_1 + 1.46\, x_2$
sqrt_pair_coeffs	sqrt	+BOP	f_1: $x_2 + \varphi_\theta(x_1)$ f_2: $0.528\, \varphi_\theta(0.673\, x_1 + 0.296\, x_2)$
sqrt_pair_coeffs	sqrt	+NOP	f_1: $x_2 + \varphi_\theta(0.654\, x_1) + 1.23$ f_2: $\varphi_\theta(0.122\, x_1 + 0.054\, x_2) + 1.23$
sqrt_trig_pair	sqrt	+BOP	f_1: $(1.35 - \cos(0.399\, x_1 + 0.751))\, \cos(x_2) + \cos(x_1)$ f_2: $\varphi_\theta(0.195\, x_1 + 0.195\, x_2) + \cos(x_1\, x_2)$
sqrt_trig_pair	sqrt	+NOP	f_1: $\varphi_\theta(x_1)\ \cos(x_2) + \cos(x_1)$ f_2: $\varphi_\theta(x_1 + x_2) + \cos(x_1\, x_2)$
sqrt_poly_trio	sqrt	+BOP	f_1: $0.282\, x_1 + x_2^2 + 0.759$ f_2: $x_1\, (0.235\, x_2 + 1.23)$ f_3: $0.0637\, x_2\, (x_1 + 4.5) + \varphi_\theta(x_1)$
sqrt_poly_trio	sqrt	+NOP	f_1: $x_2^2 + \varphi_\theta(x_1)$ f_2: $0.238\, x_1\, (x_2 + 5.12)$ f_3: $0.5\, x_1^2 + \varphi_\theta(x_1\, x_2)$
sqrt_log_pair	sqrt	+BOP	f_1: $0.0642\, x_1 + \log((x_1 + 1.29)\, (x_2 + 0.998)) + 0.0855$ f_2: $\varphi_\theta(\log(x_1\ \log(7.27\, x_2^2 + 9.99)))$
sqrt_log_pair	sqrt	+NOP	f_1: $1.22\, \varphi_\theta(x_1) + \varphi_\theta(x_2)$ f_2: $\varphi_\theta(x_1) - \varphi_\theta(-0.0185\, x_1\, x_2)$
sqrt_exp_trio	sqrt	+BOP	f_1: $\varphi_\theta(x_1 + \varphi_\theta(x_2) + 1.49)$ f_2: $\varphi_\theta(x_2 + \varphi_\theta(x_1))$ f_3: $0.676\, \varphi_\theta(x_1 + x_2 + \varphi_\theta(\varphi_\theta(-0.271\, x_1 + \varphi_\theta(x_2)))) - 0.691)$
sqrt_exp_trio	sqrt	+NOP	f_1: $\varphi_\theta(2\, x_1 + x_2 + 2.21)$ f_2: $\varphi_\theta(x_1 + 1.77\, x_2)$ f_3: $x_1\, \varphi_\theta(0.0466\, x_2) + \varphi_\theta(x_2)$
sqrt_mixed_pair	sqrt	+BOP	f_1: $x_2\, \varphi_\theta(x_1) + \varphi_\theta(x_1)$ f_2: $x_1\, x_2\, (x_1 + \varphi_\theta(x_2)) + x_1 + \varphi_\theta(x_2)$
sqrt_mixed_pair	sqrt	+NOP	f_1: $1.29\, \varphi_\theta(x_2\, (x_1 + \varphi_\theta(x_1 - 1.65) + 3.08))$ f_2: $x_2\, (x_1 + 0.0859)\, (x_1 + \varphi_\theta(x_2)) + 1.06$
sqrt_mixed_quartet	sqrt	+BOP	f_1: $(x_1 + 1.82)\, \varphi_\theta(0.171\, x_2)$ f_2: $0.844\, x_2 + 1.26$ f_3: $x_1\, \varphi_\theta(x_2)$ f_4: $0.518\, x_1 + 0.518\, x_2 + 1.07$
sqrt_mixed_quartet	sqrt	+NOP	f_1: $\varphi_\theta(\varphi_\theta(x_1\, x_2 + x_1) - 2.51)$ f_2: $0.879\, x_2 + 1.08$ f_3: $\varphi_\theta(x_1\, \varphi_\theta(2\, x_2)) - 2.24$ f_4: $\varphi_\theta(\varphi_\theta(x_1\, x_2))$

Table 13: Exemplar recovered expressions: synthetic sets, held-out $+$, $-$

Experiment	Held out	Arm	Exemplar (selected candidate, median-loss seed)
sqrt_pair	+/-	+BOP	f_1: $\varphi_\theta(\sqrt{x_1}, 0.97\,x_2)\;\varphi_\theta(\sqrt{x_1}, x_2)$ f_2: $\varphi_\theta(0.968\,x_1, x_2)$
sqrt_pair	+/-	+NOP	f_1: $\varphi_\theta(x_2, \sqrt{x_1})$ f_2: $\sqrt{\varphi_\theta(x_1, x_2)}$
sqrt_trio	+/-	+BOP	f_1: $9.5\;\varphi_\theta(x_2, 1.46\;\varphi_\theta(0.784, x_1))$ f_2: $2.98\;\sqrt{\varphi_\theta(x_1, 0.756\,x_2)}$ f_3: $9.49\;\varphi_\theta(x_1, 1.45\;\varphi_\theta(0.843, x_2))$
sqrt_trio	+/-	+NOP	f_1: $\varphi_\theta(\sqrt{x_1}, x_2)$ f_2: $\sqrt{\varphi_\theta(x_1, x_2)}$ f_3: $\varphi_\theta(\sqrt{x_2}, x_1)$
sqrt_quartet	+/-	+BOP	f_1: $\varphi_\theta(\varphi_\theta(\sqrt{x_1}, 0.0719), x_2)$ f_2: $\sqrt{\varphi_\theta(x_2, x_1)}$ f_3: $\varphi_\theta(\sqrt{\varphi_\theta(x_2, 0.309)}, x_1)$ f_4: $1.4\;\varphi_\theta(x_1, x_2)$
sqrt_quartet	+/-	+NOP	f_1: $\varphi_\theta(\sqrt{\varphi_\theta(x_1, 0.473)}, x_2)$ f_2: $\sqrt{\varphi_\theta(x_2, x_1)}$ f_3: $\varphi_\theta(\sqrt{\varphi_\theta(x_2, 0.457)}, x_1)$ f_4: $1.42\;\varphi_\theta(x_1, x_2)$
sqrt_pair_coeffs	+/-	+BOP	f_1: $3.86\;\varphi_\theta(x_2, \sqrt{x_1})$ f_2: $\varphi_\theta(4.12, \varphi_\theta(x_1, \varphi_\theta(x_2, \varphi_\theta(x_1, \varphi_\theta(1.06\;\sqrt{\varphi_\theta(4.04, x_1)}$, $\varphi_\theta(\varphi_\theta(x_2, \varphi_\theta(x_1, x_1)), -0.352))))))$
sqrt_pair_coeffs	+/-	+NOP	f_1: $\varphi_\theta(x_2, \sqrt{x_1})$ f_2: $\sqrt{\varphi_\theta(x_1, 0.19\,x_2)}$
sqrt_trig_pair	+/-	+BOP	f_1: $\varphi_\theta(x_1, x_2)$ f_2: $\varphi_\theta(\varphi_\theta(x_1, -0.502), \varphi_\theta(\varphi_\theta(-0.0471, x_2), \varphi_\theta(\varphi_\theta(0.118, \cos(x_1\,x_2)), -0.863)))$
sqrt_trig_pair	+/-	+NOP	f_1: $\varphi_\theta(-0.596\,x_1, -0.531\;\cos(x_2))$ f_2: $\varphi_\theta(\sqrt{\varphi_\theta(x_1, x_2)}, \cos(x_1\,x_2))$
sqrt_poly_trio	+/-	+BOP	f_1: $x_2\;\varphi_\theta(x_2, -0.408)$ f_2: $x_1\;\varphi_\theta(x_2, -0.0301)$ f_3: $x_1^2\;\varphi_\theta(x_2, x_1)$
sqrt_poly_trio	+/-	+NOP	f_1: $\varphi_\theta(\sqrt[4]{x_1}, x_2^2)$ f_2: $x_1\;\sqrt{\varphi_\theta(0.877, x_2)}$ f_3: $\varphi_\theta(x_1\;\varphi_\theta(\sqrt{x_1}, -0.567), \sqrt{x_2})$
sqrt_log_pair	+/-	+BOP	f_1: $\varphi_\theta(\sqrt{\varphi_\theta(\varphi_\theta(0.805, \sqrt{x_2}), \sqrt{x_2})}, x_1)$ f_2: $\sqrt{x_1}\;\sqrt{\varphi_\theta(0.502, x_2)}$
sqrt_log_pair	+/-	+NOP	f_1: $\sqrt{\varphi_\theta(x_1, x_2)}$ f_2: $0.806\;\sqrt[4]{x_1}\;\sqrt[4]{x_1\;\varphi_\theta(x_2, 1.05)}$
sqrt_exp_trio	+/-	+BOP	f_1: $\varphi_\theta(\varphi_\theta(x_1, -0.0334), x_2)$ f_2: $0.767\;\varphi_\theta(x_2, x_1)$ f_3: $\sqrt{x_2\;\varphi_\theta(x_1, e^{-0.977\;\varphi_\theta(-0.384, x_2)})}$
sqrt_exp_trio	+/-	+NOP	f_1: $\varphi_\theta(\sqrt{x_2}, \varphi_\theta(\varphi_\theta(x_1, 0.923), 1.28))$ f_2: $\varphi_\theta(\sqrt{\varphi_\theta(x_1, -0.711)}, x_2)$ f_3: $\sqrt{\varphi_\theta(0.848\,x_1\,x_2, x_2)}$
sqrt_mixed_pair	+/-	+BOP	f_1: $\sqrt{x_1}\;\varphi_\theta(-2.01, x_2)$ f_2: $31.4\,x_2\;\varphi_\theta(x_1, \sqrt{x_2})$
sqrt_mixed_pair	+/-	+NOP	f_1: $\sqrt{x_1\;\varphi_\theta(x_2\;\varphi_\theta(x_2, -1.58), -1.58)}$ f_2: $x_2\;\varphi_\theta(x_1, -1.58)\;\varphi_\theta(x_1, \varphi_\theta(0.193, x_2))$
sqrt_mixed_quartet	+/-	+BOP	f_1: $\varphi_\theta(x_1, x_2)$ f_2: $1.25\;\varphi_\theta(\varphi_\theta(0.315\,x_2, \varphi_\theta(1.69, x_1)), x_2)$ f_3: $x_1\;\sqrt{x_2}$ f_4: $1.01\;\varphi_\theta(\varphi_\theta(\varphi_\theta(1.59, x_2), x_2), x_1)$
sqrt_mixed_quartet	+/-	+NOP	f_1: $\varphi_\theta(\sqrt{x_1\,x_2}, \sqrt{x_1})$ f_2: $\sqrt{\varphi_\theta(x_2^2, x_1)}$ f_3: $x_1\;\sqrt{x_2}$ f_4: $\varphi_\theta(1.91, \sqrt{\varphi_\theta(\varphi_\theta(-0.917, x_1), x_1\,x_2)})$

Table 14: Exemplar recovered expressions: physical tasks

Experiment	Held out	Arm	Exemplar (selected candidate, median-loss seed)
real_pendulum_trig	sin/cos	+BOP	real_double_pend_h: $\varphi_\theta(-0.0959\,x_4) + 1.43\,\varphi_\theta(x_1 - x_2)$ real_pend_a: $-1.07\,x_1$ real_pend_h: $-\varphi_\theta(x_1) - \varphi_\theta(x_2)$
real_pendulum_trig	sin/cos	+NOP	real_double_pend_h: $0.0095\,x_1 - 0.000167\,x_4 - 0.0094$ $(\varphi_\theta(x_1) - \varphi_\theta(0.558\,x_2))\,\varphi_\theta(x_1) + 0.0344\,(0.00719\,x_2 - 3.15$ $\varphi_\theta(1.43) - 0.00719\,\varphi_\theta(x_4) + 0.0217)\,\varphi_\theta(1.12) - 0.0344$ $\varphi_\theta(-0.583) + 0.0094\,\varphi_\theta(1.43) + 0.0094\,\varphi_\theta(x_1) - 0.0192$ $\varphi_\theta(x_2) - 0.0094\,\varphi_\theta(-x_1 - 1.33) - 0.000467$ $\varphi_\theta(-x_1 + x_3 - 1.92) - 0.494$ real_pend_a: $-1.07\,x_1$ real_pend_h: $2\,x_2^2 + 1.05\,\varphi_\theta(x_1) + 1.05\,\varphi_\theta(0.0259 - x_1) + 0.0215$
synthetic_pendulum_trig	sin/cos	+BOP	double_pend_h: $-2\,x_1 + x_2$ pendulum_a: $-7.02\,x_1$ pendulum_h: $x_2^2 + \varphi_\theta(-0.925\,x_1) + \varphi_\theta(x_1)$
synthetic_pendulum_trig	sin/cos	+NOP	double_pend_h: $x_1 + \varphi_\theta(0.411) - \varphi_\theta(0.452) + \varphi_\theta(-x_1 + x_2 + 0.0792)$ pendulum_a: $2\,\varphi_\theta(-1.99)\,\varphi_\theta(x_1) - \varphi_\theta(x_1)$ pendulum_h: $x_2^2 + \varphi_\theta(-0.688)\,\varphi_\theta^2(2\,x_1) + \varphi_\theta(1.64) + 0.198$
real_airtrack_add_sub	+/-	+BOP	real_double_linear_h: x_2 real_linear_a: $-6.02\,x_1$ real_linear_h: $x_2\,\varphi_\theta(\varphi_\theta(-0.27, x_1), 0.601)$
real_airtrack_add_sub	+/-	+NOP	real_double_linear_h: $\varphi_\theta(x_3, \varphi_\theta(\varphi_\theta(x_1, x_2), x_4))$ real_linear_a: $-6.02\,x_1$ real_linear_h: $\varphi_\theta(x_2, \varphi_\theta(x_1, x_1))$
synthetic_airtrack_add_sub	+/-	+BOP	double_linear_h: $\varphi_\theta(x_1\,\varphi_\theta(x_2, -0.154\,x_3\,x_4), x_2)$ linear_a: $\varphi_\theta(-0.441, x_1)$ linear_h: $\varphi_\theta(\varphi_\theta(x_1, 1.12), -0.0259\,x_2^2)$
synthetic_airtrack_add_sub	+/-	+NOP	double_linear_h: $\varphi_\theta(\varphi_\theta(6.87, x_3), x_4)$ linear_a: $-2.33\,\varphi_\theta(x_1, -1.16)$ linear_h: $\varphi_\theta(x_1\,\varphi_\theta(x_1, -2.39), x_2^2)$
geometry_mul	*	+BOP	circle: $\varphi_\theta(x_1, 0.509) + \varphi_\theta(x_2, 0.513) - x_1 - x_2$ real_pend_circle: $\varphi_\theta(x_1 - 0.970, x_1 - x_2 - 2.22)$ sphere: x_3
geometry_mul	*	+NOP	circle: $\varphi_\theta(x_1, x_2) + \varphi_\theta(x_2, x_1)$ real_pend_circle: $-x_2 + \varphi_\theta(x_2, x_1)$ sphere: x_3

Table 15: Three real-pendulum runs select an energy-law expression containing the learned operator. Selected expression for the energy task in every run of the real pendulum group, simplified by collecting constants (c is an additive constant); θ is the wrapped angle and ω the angular velocity. *Operator*: affine R^2 of φ_θ against cos (against sin where marked, the better match) over all its occurrences in the group, on the held-out rows and on the training rows it was fitted on. *Loss*: the energy task's test loss relative to the ABLATED median (ORACLE median 0.0081). In two spline runs φ_θ also takes ω, and the cosine-shaped operator replaces the kinetic term. Per-task and fixed-grid scores are in Appendix Table 16.

				Operator		
Arm	Seed	Selected H	Form	held-out	fitted	Loss
+BOP	1001	$\varphi_\theta(\omega) + \varphi_\theta(\theta)$	$\varphi_\theta(\omega)$	0.881	0.994	0.0093
+BOP	1002	$0.182\,\omega^2 - 1.084\,\varphi_\theta(\theta) + c$	**energy law**	0.868	0.944	0.0155
+BOP	1003	$-(1.811 + 0.250\,\omega^2)\,\varphi_\theta(1.281 - \theta)$	product, shift	0.006 (sin)	0.835 (sin)	0.0187
+BOP	1004	$2.970\,\omega^2 + 2.970\,\varphi_\theta(\theta + 0.253) - 1.970\,\theta + c$	shift, linear θ	0.466	0.992	0.0102
+BOP	1005	$-\varphi_\theta(\omega) - \varphi_\theta(\theta)$	$\varphi_\theta(\omega)$	0.176	0.996	0.0105
+NOP	1001	$2\,\omega^2 + 1.052\,[\varphi_\theta(\theta) + \varphi_\theta(0.026 - \theta)] + c$	shift	0.041	0.187	0.0207
+NOP	1002	$0.351\,\omega^2 - 0.304\,\varphi_\theta(\theta) + c$	**energy law**	0.971	0.992	0.0287
+NOP	1003	$\varphi_\theta(\theta) - 0.698\,\omega^2$	**energy law**	0.961	0.989	0.0257
+NOP	1004	$\varphi_\theta(-0.210\,\theta) - 0.332\,\omega^2$	rescaled	0.965	0.977	0.0607
+NOP	1005	$0.523\,[\varphi_\theta(\theta) + \varphi_\theta(-0.896\,\theta)] - 0.873\,\omega^2$	rescaled	0.602	0.979	0.0138
Runs matching at $R^2 \geq 0.99$ (≥ 0.95), of 10				0 (3)	4 (7)	

Table 16: The three energy-law runs match cos at $R^2 \geq 0.95$ on the energy task's own inputs, while the held-out score, pooled over the whole group's test rows, passes two of them at $R^2 \geq 0.95$ and none at 0.99. Affine R^2 of φ_θ against the better-matching of cos and sin (Ref.) for every real-pendulum run. *Law*: the energy expression has the form $c_0 + c_1\omega^2 + c_2\varphi_\theta(\theta)$ (Table 15). *Group*: all occurrences of φ_θ in the group's selected candidate; test is the held-out score, with its affine map fitted on the test rows, which include the unseen double-pendulum trial; train is the training-row score. *Energy task*: only the occurrences in the `real_pend_h` expression; the map fitted on its training rows is applied unchanged to its interleaved test rows. *Grid*: 201 points over the run's own training input range and over $[-\pi, \pi]$, each with its own affine fit (Appendix D.2). Last row: runs at $R^2 \geq 0.99$, and at 0.95 in parentheses.

				Group		Energy task		Grid	
Arm	Seed	Law	Ref.	test	train	test	train	own range	$[-\pi, \pi]$
+BOP	1001		cos	0.881	0.994	0.997	0.997	0.994	0.827
+BOP	1002	✓	cos	0.868	0.944	0.974	0.968	0.920	0.880
+BOP	1003		sin	0.006	0.835	0.634	0.573	0.607	0.601
+BOP	1004		cos	0.466	0.992	0.997	0.997	0.972	0.460
+BOP	1005		cos	0.176	0.996	0.998	0.998	0.993	0.510
+NOP	1001		cos	0.041	0.187	0.961	0.959	0.003	0.957
+NOP	1002	✓	cos	0.971	0.992	0.993	0.993	0.992	0.985
+NOP	1003	✓	cos	0.961	0.989	0.992	0.994	0.990	0.984
+NOP	1004		cos	0.965	0.977	0.964	0.977	0.976	0.962
+NOP	1005		cos	0.602	0.979	0.924	0.923	0.988	0.983
Runs matching of 10				0 (3)	4 (7)	5 (8)	5 (8)	4 (7)	0 (5)

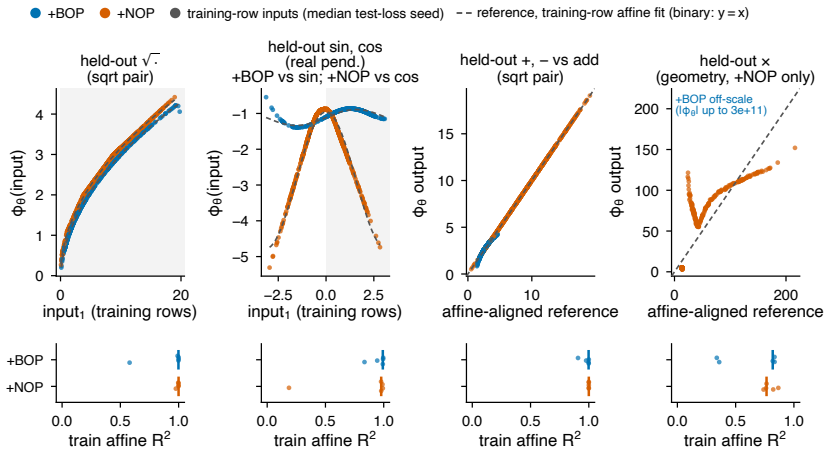

Figure 6: The learned operator of the median-loss seed matches the shape of the deleted primitive in both synthetic panels and, for the MLP, on the real pendulum; on the geometry group it does not match multiplication. Unary panels: φ_θ against its input, the affine-fitted primitive dashed, the spline support $[0, 30]$ shaded; the spline's median pendulum seed is the one run that matches sin better than cos ($R^2 = 0.84$, below both thresholds), the MLP's matches cos ($R^2 = 0.98$). Binary panels: output of φ_θ against the affine-aligned primitive, where recovery is the diagonal; the spline's geometry outputs are off-scale. Strips: training-row affine R^2 of every seed (all seeds in Appendix D).

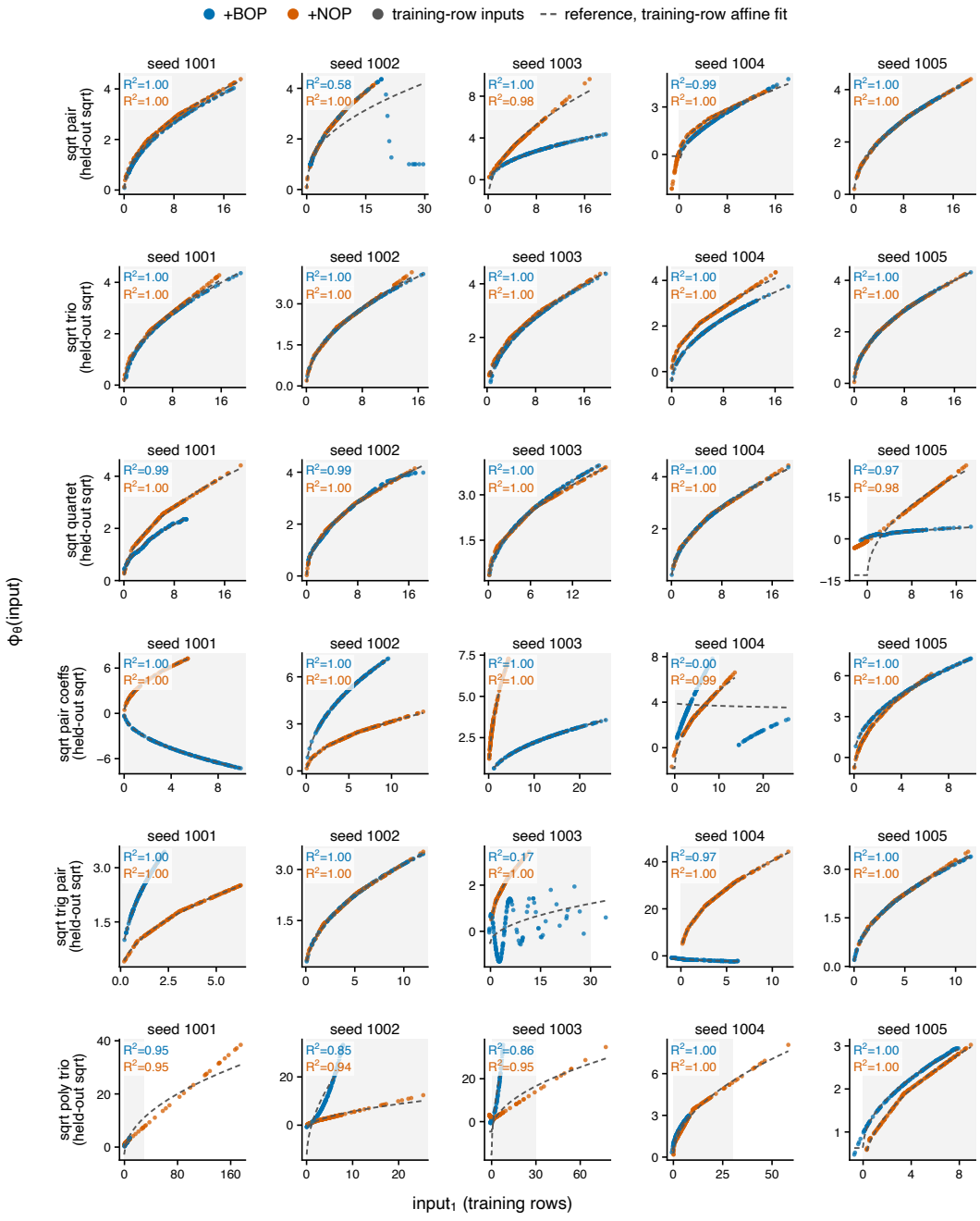

Figure 7: Learned operators of every seed against the removed primitive (page 1 of 5: square-root sets).

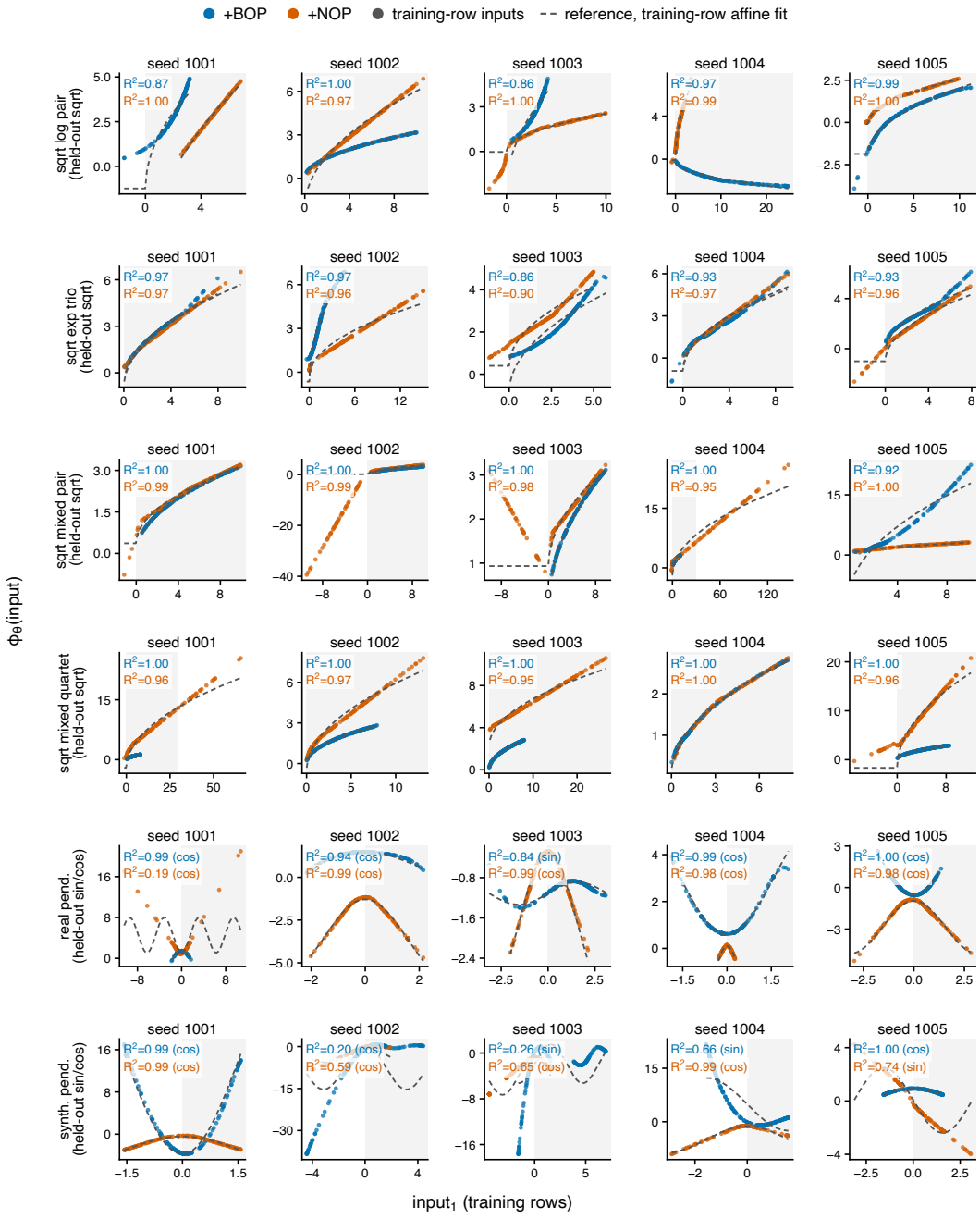

Figure 8: Learned operators of every seed, continued (square-root sets and pendulum groups).

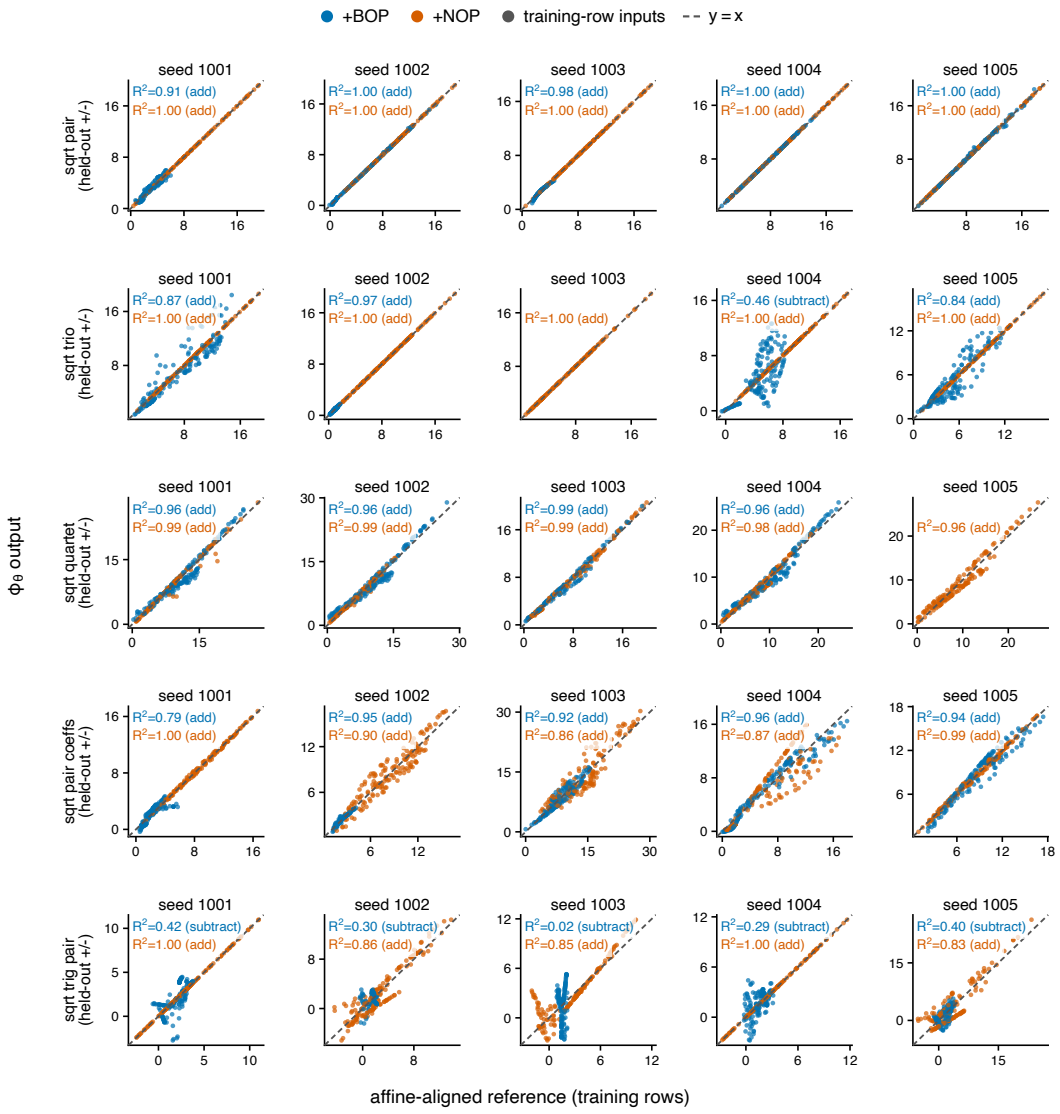

Figure 9: Learned binary operators of every seed: learned output against the affine-aligned reference, addition and subtraction removed.

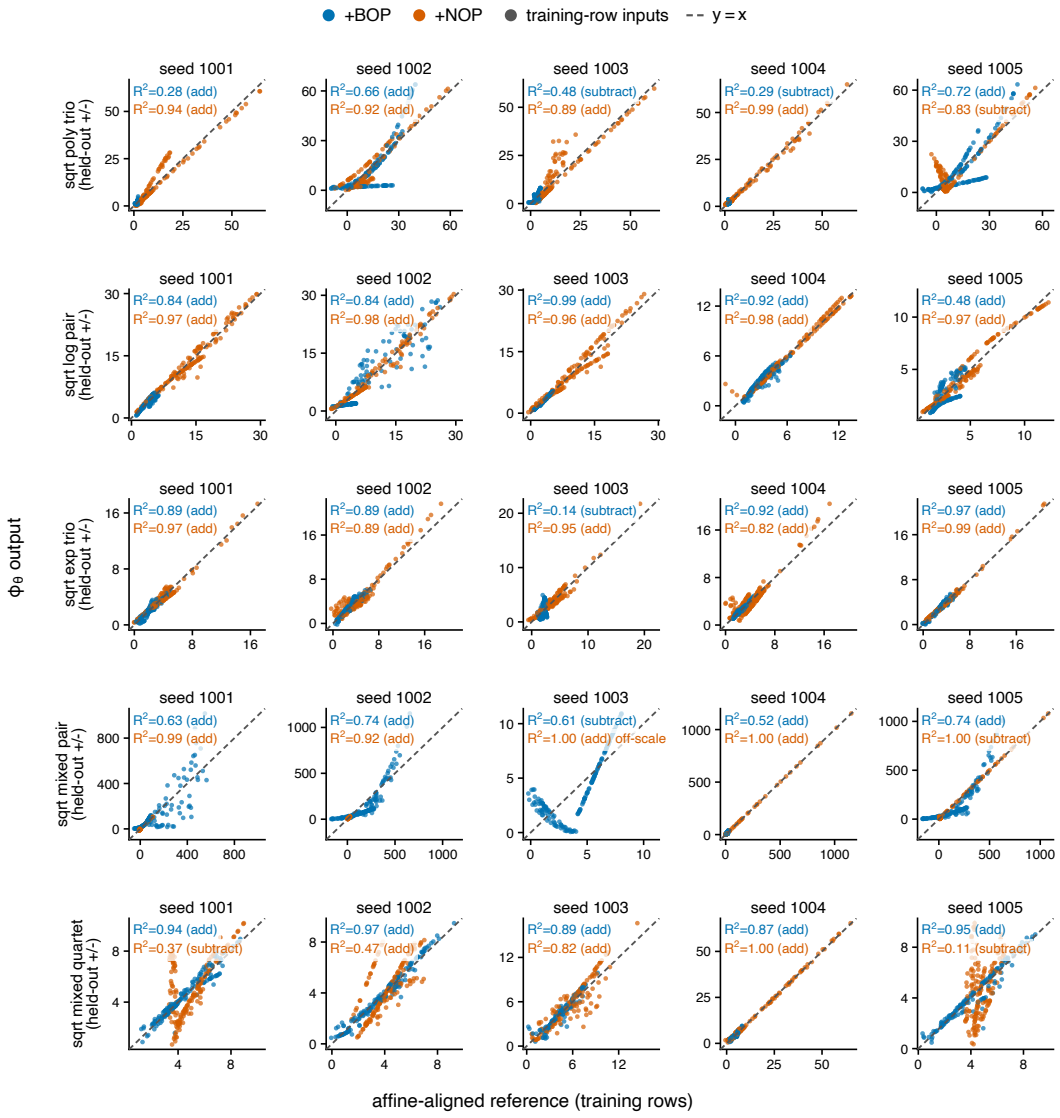

Figure 10: Learned binary operators of every seed, continued.

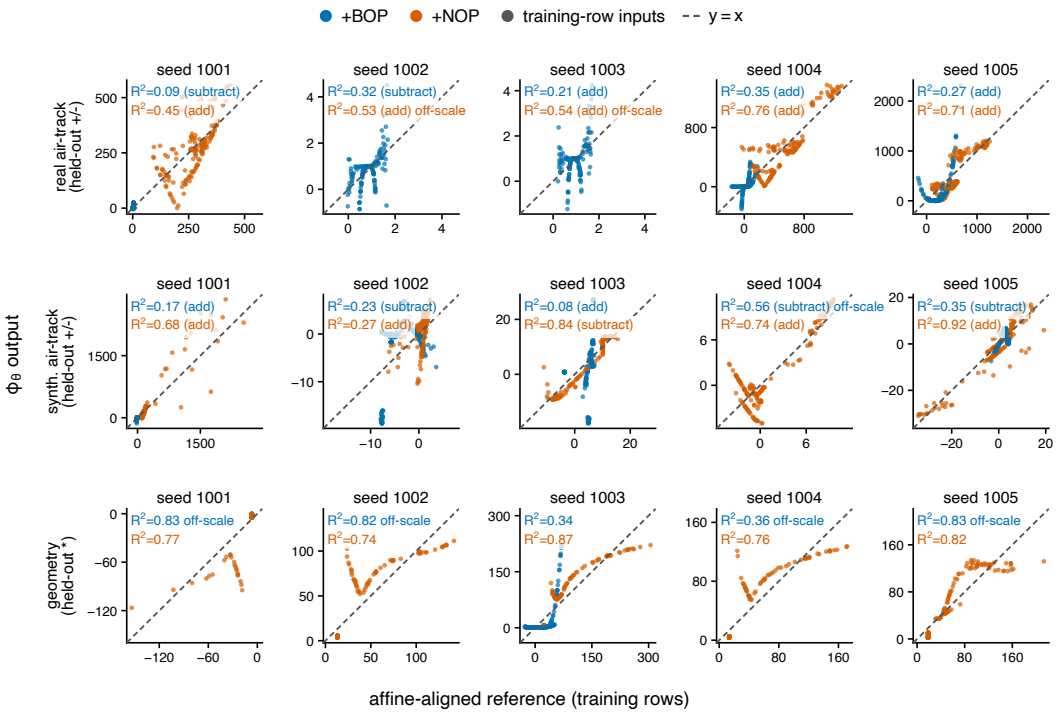

Figure 11: Learned binary operators of every seed, continued (air-track and geometry groups).

D.6 Charging the learnable operator's parameters

A learnable node has size one in the search (Section 2.2). We tested whether adding its parameter count to the expression cost would have changed the selected expressions. There is no new search: we reselect from each run's stored final Pareto front, so front members that a different cost would have created during the search were never stored and cannot be chosen. The selection rule of Appendix A reproduces all 450 stored selections. The optimized parameter counts p are 33 (unary spline), 49 (unary MLP), 65 (binary MLP) and 1089 (binary spline). We apply three costs to every candidate that uses φ_θ: (a) size plus p and (b) size plus $\log_2 p$, both followed by the standard selection rule, and (c) a BIC-style score $n \ln L + k \ln n$ minimized over the whole front, where L is the stored training loss, n the number of training rows times outputs (2000 to 4000 on the synthetic sets, 544 to 612 on the physical groups) and k the number of constants plus p. On the physical groups L is an invariance or normalized loss, so (c) is a heuristic analog there.

Costs (a) and (b) change no selection. The rule chooses among front members within a factor 1.5 of the lowest loss, and in no learnable run does a φ_θ-free member fall inside that window: the best φ_θ-free member has 2.2 to 5.9×10^9 times the lowest training loss (median 935). Every eligible member uses φ_θ and receives the same added cost, so an added size cost leaves their order unchanged under this rule. Cost (c) changes selections. It changes 206 of 250 selections and chooses a φ_θ-free expression in 17, of which 15 are binary-spline runs on the physical groups, where no run recovers the operation in any case, and 2 are addition-spline runs on `sqrt_trig_pair`. Table 17 gives the recovery counts. Against the same-seed ABLATED run, the learnable arm's BIC is lower in 5 of 5 seeds for every physical MLP arm and both pendulum spline arms, in 0 of 5 for the physical binary splines, and in at least 3 of 5 for every synthetic set and arm except the addition spline on `sqrt_poly_trio` (0) and `sqrt_trig_pair` (2).

Table 17: Recovery counts after reselecting each run's stored final front under a parameter charge. *Operator*: runs matching the deleted primitive by the held-out score at $R^2 \geq 0.99$ (≥ 0.95), of 50 per arm on the synthetic sets and of 10 runs (both arms) on the real pendulum. *Law*: for the square root, runs whose selected equations equal the target laws after refitting their constants on 200 points with relative RMSE at most 10^{-4}, a looser test than the law check of Section 5.1, whose counts for the standard selection are in brackets; for the pendulum, runs whose energy expression has the form $c_0 + c_1\omega^2 + c_2\varphi_\theta(\theta)$. Size charges (a) and (b) leave every selection, and so every count, as in the paper.

		$\sqrt{\cdot}$ operator		$\sqrt{\cdot}$ law		$+, -$ operator		Pendulum	
Selection	Changed	spline	MLP	spline	MLP	spline	MLP	operator	law
Standard, (a), (b)	$0/250$	33 (39)	32 (46)	13 [12]	18 [14]	3 (14)	21 (32)	0 (3)	3
BIC-style (c)	$206/250$	31 (37)	28 (42)	14	16	3 (12)	24 (33)	0 (1)	2

D.7 Search dynamics, Pareto fronts and spline support

Fig. 12 plots the lowest loss on the front against wall-clock time, median over seeds, for every set and group; the learnable arms reach their final loss within the budget on the synthetic sets and cost about one to nine times the oracle's time. Figs. 13 and 14 show the final loss-versus-size fronts of each arm; because a φ_θ node has size one, the learnable arms' fronts should be read with the parameter counts of Table 2 in mind. Fig. 15 gives, for every spline run, the fraction of the operator's training inputs that fell outside its support $[0, 30]$, where the spline extrapolates polynomially; on the physical groups this is about half of all inputs because angles and velocities are negative half the time.

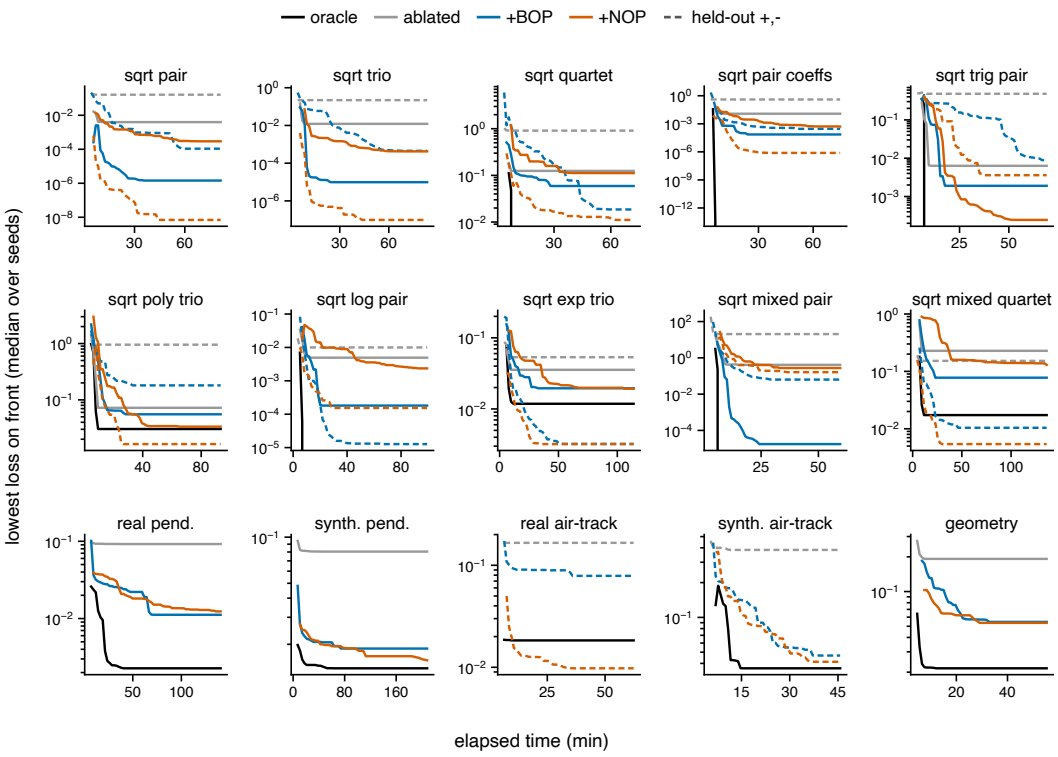

Figure 12: The learnable arms reach their final loss within the budget on the synthetic sets, at a higher wall-clock cost than ORACLE. Lowest loss on the front against wall-clock time, median over seeds, for every set and group.

E BESSEL'S J_0 AND DAWSON'S INTEGRAL

Equations, library and search. The J_0 study uses $y_1 = J_0(x_1) + x_2$, $y_2 = J_0(x_1 + x_2)$, $y_3 = x_1 J_0(x_2)$ with x_1, x_2 uniform on $[0, 6]$, so a shared operator receives arguments on about $[0, 12]$. The Dawson study uses the same forms with Dawson's integral $F(x) = e^{-x^2} \int_0^x e^{t^2}\, dt$ and inputs on $[0, 4]$. Neither function can be written with finitely many elementary operators. Both use 1000 training and 1000 test samples; NMSE is the mean squared error divided by the target variance, averaged over outputs, on a fresh draw of 10,000 points, and a failed or non-finite run counts as $+\infty$. The library is unary $\sqrt{\cdot}$, exp, log, sin, cos and binary $+$, $-$, $\times$, $/$, with the target added in ORACLE only. Search settings are those of the synthetic sets (Section 5.1) with a size cap of 30 summed over a set's equations; the spline has 33 parameters and the MLP 49. Every arm runs search seeds 1001–1020.

Arms. ORACLE, ABLATED, +BOP and +NOP as in Section 3, and one extended ABLATED arm with a size cap of 60 and 1017 (J_0) or 1072 (F) iterations, about four (J_0) or three (F) times the spline's wall time. Wall times come from a node shared by up to 32 runs, so comparisons between arms are approximate.

Identification. This is the rule Section 3 uses when it says a run learns J_0. The learned operator is evaluated on 1000 uniform points spanning the arguments it received on the training data and compared, after a least-squares affine map on the output, with the target and with fixed look-alikes (J_0: $\cos x$, $\sin(\pi x)/(\pi x)$, $\sin(x)/x$, $\cos(x)e^{-x/4}$, $\cos(x)/\sqrt{1+x}$, $1 - x^2/4$). It is identified when $R^2 \geq 0.999$ against the target and $10(1-R^2)$ is at most the smallest $1-R^2$ over the look-alikes. If a look-alike fitted to the target itself on the same grid leaves $1 - R^2 < 0.001$, the run is undetermined and counted as not identified.

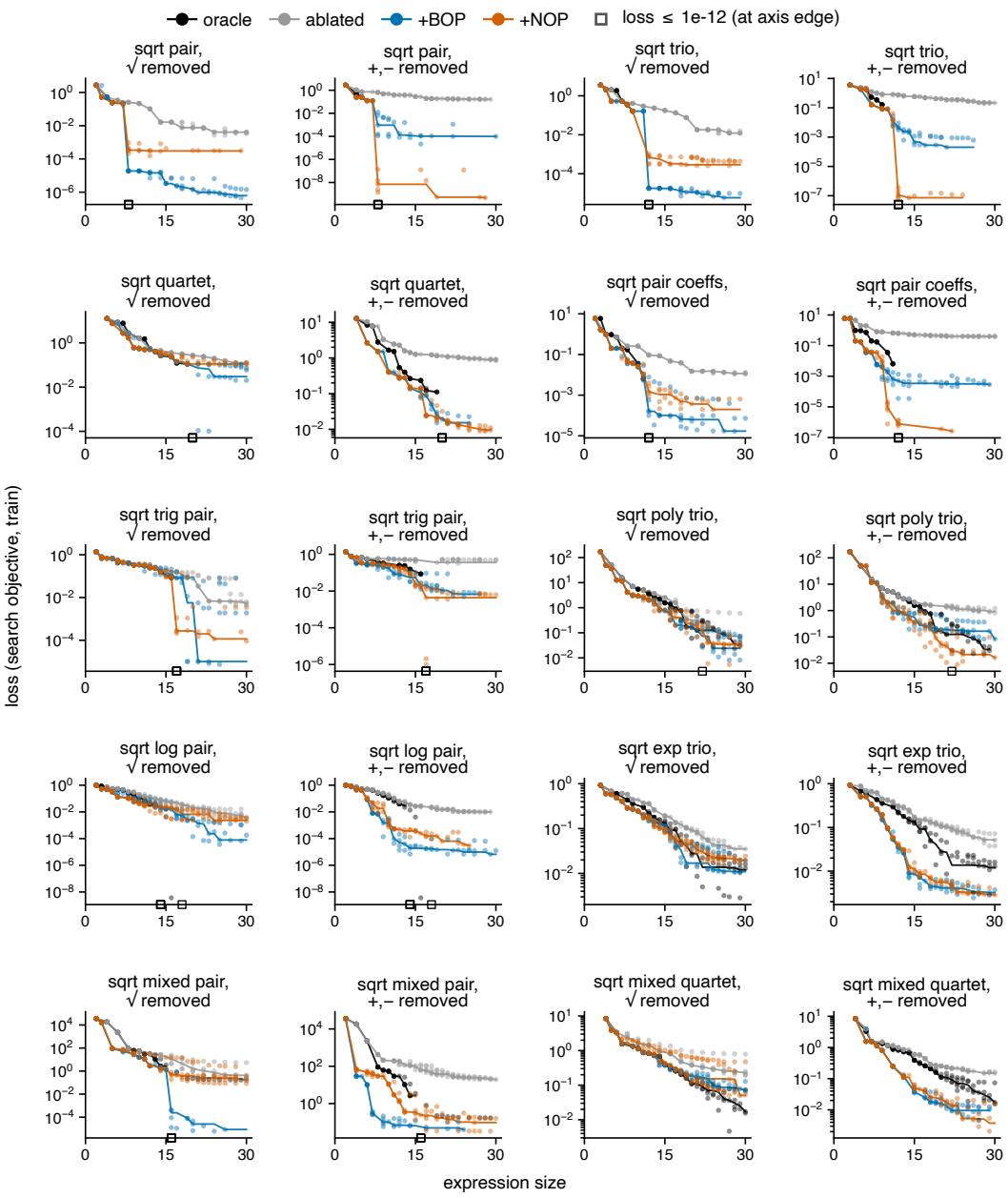

Figure 13: Final loss-versus-size fronts per arm on the synthetic sets, one panel per removed operator. Because a φ_θ node has size one, the plotted sizes of the learnable arms do not count the parameters of φ_θ; the ORACLE front is the same in both panels of a set. Lines are the median over seeds at each size, points are individual seeds' front members, and members with zero loss are marked on the axis edge.

Results for J_0. The spline identifies J_0 in 13 of 20 seeds (Clopper–Pearson 95% interval $[0.41, 0.85]$), 17 reach $R^2 \geq 0.99$, and one is undetermined. The identified operators reach R^2 from 0.9998 to 1.0000 against J_0 and from 0.81 to 0.90 against $\sin(x)/x$, their closest look-alike. Ten spline runs select the target laws in 12 nodes; nine of them identify J_0, and the tenth (seed 1017) reaches $R^2 = 0.9978$. Three further identified runs select the target laws with negligible extra terms (seeds 1009 and 1019) or with $\varphi_\theta = -J_0$ and matching signs (seed 1011), and one (seed 1007) replaces $J_0(x_1 + x_2)$ with a product of sines. Four of the seven runs that do not identify J_0 replace $J_0(x_1 + x_2)$ with a constant, and one (1001) contains $\sqrt{8.45/x_1}$ and reaches 1.7×10^9 on the fresh

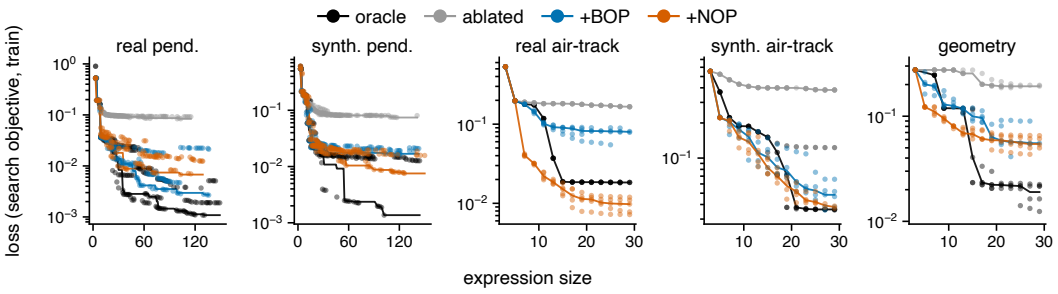

Figure 14: Final loss-versus-size fronts per arm on the physical task groups.

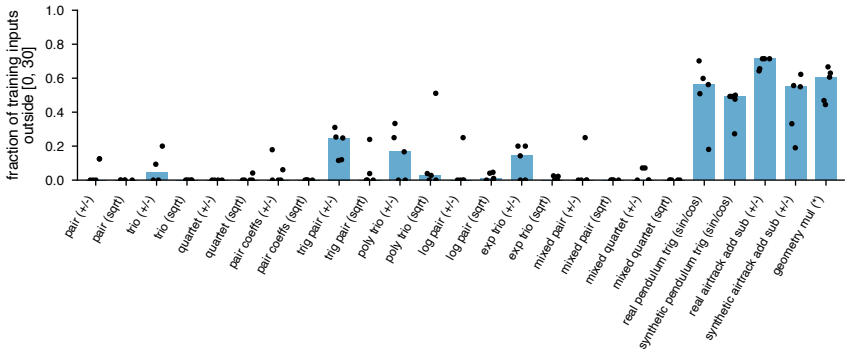

Figure 15: On the physical task groups, the median fraction of the spline operator's training inputs outside its support $[0, 30]$ is approximately one half or higher; the synthetic pendulum group's median is 0.493. Most square-root sets have a low median. Bars show medians and dots show individual runs.

draw. The spline's median NMSE, 2.1×10^{-5}, is 109 times below that of the extended ABLATED run. The MLP identifies J_0 in no seed (3 at $R^2 \geq 0.99$), with median NMSE 0.044.

Results for Dawson's integral. No spline run identifies F, and 7 reach $R^2 \geq 0.99$. Five select the target laws in 12 nodes, with R^2 from 0.9978 to 0.9987 against F and about 0.97 against its closest look-alike, $x/(1 + 2x^2)$. The four spline runs with the lowest training loss change the argument of φ_θ by a constant in every law: three shift it, as in $\varphi_\theta(x_1 + 0.37) + x_2$, $\varphi_\theta(x_1 + x_2 + 0.37)$ and $x_1\varphi_\theta(x_2 + 0.37)$ (seed 1017, the lowest-loss run), and one (seed 1019) scales it by 1.29. The identification rule allows only an output scale and shift, so it compares these operators with F on their own arguments (R^2 from 0.76 to 0.93). Compared with F after undoing the shift or scale, on the stored operator grid, they reach R^2 from 0.9977 to 0.99998; this comparison is not part of the identification rule. The spline's median NMSE is 9.4×10^{-4}, against 2.0×10^{-3} for the extended ABLATED run. The MLP identifies F in no seed.

Selection by training loss. The 13 identified spline seeds are the 13 with the lowest training MSE (highest 3.7×10^{-6}; next 2.7×10^{-5}, seed 1017) and lie below all 40 elementary runs (lowest 4.9×10^{-6}). Keeping the lowest-training-loss run of each arm selects spline seed 1003 (12 nodes, training MSE 4.2×10^{-7}, NMSE 4.8×10^{-6}) and extended ABLATED seed 1013 (56 nodes, 4.9×10^{-6}, non-finite on the fresh draw because it contains $\sqrt{x_1 + x_2 - 0.155}$). With $N = 3000$ training residuals and k counting every numeric constant and operator parameter, $N \ln \mathrm{MSE} + k \ln N$ is -43764 for the spline ($k = 33$) and -36554 for the extended ABLATED run ($k = 13$). If three seeds are drawn at random from the twenty spline seeds, the probability that at least one identifies J_0 is $1 - \binom{7}{3}/\binom{20}{3} = 0.97$.

Transfer. A second study freezes the selected operator of spline seeds 1001–1010 and gives each as a fixed library function to a new search for $J_0(x_1)J_0(x_2)$ from 32 or 1000 points on $[0, 6]^2$, tested on $[6, 9]^2$. Over the ten frozen operators, the median extrapolation NMSE at $n = 32$ is 1.9, the same

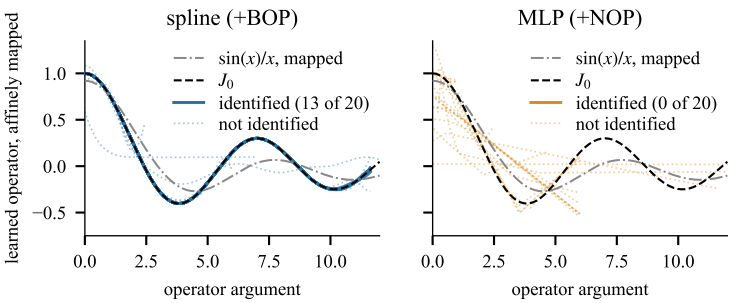

Figure 16: Bessel's J_0, absent from the library, is learned by the spline in 13 of 20 seeds and by the MLP in none. Each curve is one seed's learned operator on the arguments it received, affinely mapped onto J_0 (dashed), and solid curves pass the identification rule of Section 3. Dash-dotted: $\sin(x)/x$, the closest look-alike of the identified seeds, mapped the same way.

as elementary search (1.9). The five that identified J_0 reach a median of 2.7×10^{-5} at $n = 32$ and 2.1×10^{-5} at $n = 1000$, against 2.1 and 2.9 for the other five. The operator of seed 1003, the lowest-loss run of all twenty, reaches 1.0×10^{-5} at $n = 32$ and 8.0×10^{-6} at $n = 1000$.

Table 18: Discovery of J_0 and Dawson's integral F, selected candidate, 20 seeds per arm. NMSE on 10,000 fresh in-range points; failed or non-finite predictions count as ∞. Identified: runs passing the identification rule / runs with $R^2 \geq 0.99$ against the target. Nodes: median total size of the three selected laws. Iter.: search iterations. Min: median wall time per run in minutes. $|\theta|$: parameters of the learned operator.

Target	Arm	Runs	Median NMSE	NMSE range	Identified	Nodes	Iter.	Min	$\lvert\theta\rvert$
J_0	ORACLE	20/20	0	0–0	–	12	100	12	0
J_0	ABLATED	20/20	3.4×10^{-1}	4.1×10^{-2}–3.8×10^{-1}	–	23.5	100	11	0
J_0	ABLATED (extended)	20/20	2.3×10^{-3}	4.8×10^{-5}–∞	–	50	1017	102	0
J_0	+BOP	20/20	2.1×10^{-5}	4.8×10^{-6}–1.7×10^{9}	13 / 17	12	100	26	33
J_0	+NOP	20/20	4.4×10^{-2}	2.2×10^{-2}–3.6×10^{-1}	0 / 3	22	100	58	49
F (Dawson)	ORACLE	20/20	0	0–0	–	12	100	10	0
F (Dawson)	ABLATED	20/20	1.3×10^{-1}	6.1×10^{-2}–3.7×10^{-1}	–	24	100	10	0
F (Dawson)	ABLATED (extended)	20/20	2.0×10^{-3}	3.5×10^{-4}–5.3×10^{-3}	–	51	1072	100	0
F (Dawson)	+BOP	20/20	9.4×10^{-4}	1.6×10^{-5}–∞	0 / 7	18	100	31	33
F (Dawson)	+NOP	20/20	1.4×10^{-2}	8.6×10^{-4}–3.4×10^{-1}	0 / 0	21	100	97	49

F IDENTIFICATION OF THE SHARED OPERATOR

This appendix states and proves the results of Section 4 in full. Two structural properties can make several laws identify a shared operator. The laws can constrain different changes in the operator even after their constants are refitted, and an inequivalent explanation with other trees can fail to reproduce how the collection of laws varies. We prove a local theorem for the first property and a generic global theorem for the second, over the entire symbolic search, and a corollary that gives conditions under which each law alone has inequivalent minimum-size operators while their joint minimum identifies one equivalence class. No particular target formula is assumed.

All proofs in this appendix have been formally verified using the Lean 4 proof assistant with the Mathlib library, taking Sard's theorem (Sard, 1942) and the constant-rank theorem (Abraham et al., 1988) as axioms, since these standard results have not yet been formalized.

F.1 PROBLEM AND MEANING OF UNIQUENESS

Fix a finite primitive library $\mathcal{P}$ with specified arities, finite variable sets for each law, and a learned symbol φ of fixed arity a. Its interpretation belongs to a finite-parameter family $\{\varphi_\theta : \theta \in \Theta\}$, with $\Theta \subseteq \mathbb{R}^p$ open. Every occurrence in every tree uses the same θ. $\mathcal{F}_i$ contains all finite arity-respecting expression trees for task i, and $\mathrm{size}(T)$ is its number of nodes. Real constant leaves,

repeated variables, and arbitrary nesting are allowed. Each constant occurrence is a separately valued parameter. Repeated subtrees are counted separately.

For complete target laws $f_i : \Omega_i \to \mathbb{R}$ on domains $\Omega_i \subseteq \mathbb{R}^{n_i}$ with their usual topology, minimize

$$\sum_{i=1}^{m} \operatorname{size}(T_i) \quad \text{over } \theta, T_1, \dots, T_m, c_1, \dots, c_m, \qquad T_i[\varphi_\theta, c_i] = f_i \text{ on } \Omega_i. \tag{1}$$

All evaluations must be defined. The learned operator, its placement, the surrounding expressions, and all ordinary constants are unknown.

Fix an operator-identification domain U. As a working equivalence, write

$$\varphi_\eta \sim \varphi_\theta \quad \Longleftrightarrow \quad \varphi_\eta(u) = \alpha\varphi_\theta(\pi u) + \beta \quad (u \in U) \tag{2}$$

for some $\alpha \neq 0$, $\beta \in \mathbb{R}$, and $\pi \in \Pi$, where Π is a fixed subgroup of argument permutations preserving U. Thus output scaling, output shifting, and these argument permutations do not count as different discoveries. The global theorem also works for any other equivalence relation fixed in advance. Identification concerns operator functions on U, not their network weights or the syntax of equivalent trees.

The results concern finite-dimensional parameter families. They do not require a polynomial operator. The stated smoothness assumptions must hold. The term *generic* means outside a Lebesgue-null set of true operator-and-constant parameters in a specified open family. It does not mean every individual physical instance.

F.2 Complementary constraints after retuning constants

For this section fix any candidate tree tuple $\mathbf{T}$. Let

$$\mathsf{L}_i(\theta, c_i) = T_i[\varphi_\theta, c_i]$$

be the entire predicted law on Ω_i, taking values in a Banach space Y_i of functions. Assume these maps are C^1 near the true parameters. For example, $Y_i = C(\Omega_i)$ can be used on a compact observation domain when the evaluation map is differentiable in the uniform norm. The output represents the complete function, rather than finitely many observed rows.

At $u_* = (\theta_*, c_{1*}, \dots, c_{m*})$, put

$$A_i = D_\theta \mathsf{L}_i, \qquad B_i = D_{c_i} \mathsf{L}_i, \qquad N_i = \{v \in \mathbb{R}^p : A_i v \in \operatorname{im} B_i\}.$$

A direction $v \in N_i$ changes the operator in a way that law i cannot detect to first order: its constants can compensate, since $A_i v + B_i w_i = 0$ for some w_i.

Lemma 1 (Eliminating ordinary constants)**.** *The operator directions invisible to the joint law tuple are exactly*

$$N_{\text{joint}} = \bigcap_{i=1}^{m} N_i. \tag{3}$$

In particular,

$$\dim N_i = p + \operatorname{rank} B_i - \operatorname{rank}[A_i\ B_i], \qquad \dim N_{\text{joint}} = p + \sum_i \operatorname{rank} B_i - \operatorname{rank} D\mathsf{L},$$

where $\mathsf{L} = (\mathsf{L}_1, \dots, \mathsf{L}_m)$ and its constants are law-specific. No full-rank assumption on any B_i is needed.

Proof. The joint linearized equations are $A_i v + B_i w_i = 0$ for all i. Their operator-coordinate projection is contained in every N_i. Conversely, if v belongs to every N_i, choose the compensator w_i independently for each law and assemble them into one joint parameter variation. This proves equation 3.

The projection $\ker[A_i\ B_i] \to N_i$ is surjective, with kernel $\{0\} \times \ker B_i$. Rank-nullity gives the individual dimension formula. Apply the same argument to $D\mathsf{L}$, whose constant-parameter block is diagonal with blocks B_i, to obtain the joint formula. □

Let $Q_* = \{\theta : \varphi_\theta \sim \varphi_{\theta_*}\}$ be the allowed equivalence class in parameter space. Assume it is an embedded submanifold near θ_*, and let $G = T_{\theta_*} Q_*$. It includes parameter redundancies that leave the operator function unchanged. The regularity of Q_* is an explicit assumption; singular parameterizations require separate treatment.

Theorem 1 (Complementary laws give local identification)**.** *Assume the following neighborhood conditions for the fixed tuple. The full derivative $D\mathsf{L}$ and each constant derivative $D_{c_i}\mathsf{L}_i$ have constant rank. The equivalence class Q_* admits a C^1 choice of compensating constants $s(\theta)$ through the true constants such that*

$$\mathsf{L}(\theta, s(\theta)) = \mathsf{L}(u_*) \qquad (\theta \in Q_* \text{ near } \theta_*).$$

If

$$\bigcap_i N_i = G, \tag{4}$$

then every sufficiently nearby exact joint fit has an operator equivalent to φ_{θ_*} on U.

If, in addition, each $D\mathsf{L}_i$ has constant rank nearby and each N_i properly contains G, then every individual law admits a curve of nearby exact fits with inequivalent operators. Thus the individual laws fail to identify the operator locally while the joint fit identifies its equivalence class locally.

Proof. By the constant-rank theorem (Abraham et al., 1988), the exact joint fiber

$$S = \mathsf{L}^{-1}(\mathsf{L}(u_*))$$

is a C^1 manifold near u_*, with tangent $\ker D\mathsf{L}$. The finite-dimensional source and finite-dimensional derivative range are complemented, so the constant-rank theorem applies to the Banach-valued map.

Restrict the operator-coordinate projection p_θ to S. The kernel of its derivative consists of constant variations $w_i \in \ker B_i$. Its dimension is constant by the hypotheses on the B_i. Therefore $p_\theta|_S$ has constant rank. Its local image A is an embedded manifold near θ_*, with tangent

$$T_{\theta_*} A = \bigcap_i N_i.$$

The compensating-constant section shows that Q_* is locally contained in A. Under equation 4, these two manifolds have the same dimension. Their inclusion is consequently locally open, so $A = Q_*$ after shrinking the neighborhood. This proves local joint identification, with both the operator parameters and ordinary constants restricted to that neighborhood.

For individual law i, choose $v \in N_i \setminus G$ and a compensator w_i such that $A_i v + B_i w_i = 0$. Constant rank of $D\mathsf{L}_i$ makes its exact fiber a manifold. The tangent vector (v, w_i) integrates to a C^1 curve in this fiber. In local coordinates normal to Q_*, that curve has nonzero velocity because $v \notin G$. It therefore leaves Q_* for every sufficiently small nonzero curve parameter. These are exact complete-law alternatives, not merely first-order matches. □

The compensating section is a property to verify from the expressions and their constants. Declaring two operators equivalent does not itself guarantee that the chosen trees can absorb that transformation at unchanged cost.

For a matrix form of the condition, choose an inner product on each finite-dimensional space $\operatorname{im} A_i + \operatorname{im} B_i$, and let P_i project onto the orthogonal complement of $\operatorname{im} B_i$ there. Define

$$I_i = A_i^* P_i A_i.$$

Then $\ker I_i = N_i$. The compensating section ensures $G \subseteq N_i$ for every law. The conditions $\bigcap_i N_i = G$ and $N_i \supsetneq G$ for every i are equivalently

$$\sum_i I_i \Big|_{G^\perp} \succ 0, \qquad I_i|_{G^\perp} \text{ singular for every } i. \tag{5}$$

Indeed, $v^*(\sum_i I_i)v = \sum_i \|P_i A_i v\|^2$. Each law misses an inequivalent operator direction, but no such direction is missed by them all. Repeating the same constraints does not shrink their intersection.

F.3 Excluding other trees and remote operator aliases

Local identification does not rule out another tree, or a remote parameter value of the same tree. We now give a sufficient structural condition that handles both globally.

Fix any true skeleton tuple $\mathbf{T}_*$, with total size B. Let

$$u = (\theta, c_1, \dots, c_m) \in W \subseteq \mathbb{R}^n$$

range over an open set on which its complete laws $\mathsf{L}_*(u)$ are defined. Its tree shapes are arbitrary. For every rival skeleton tuple $\mathbf{T}$ with size at most B, let v include all its shared operator parameters and constant values, and let $\mathsf{L}_{\mathbf{T}}(v)$ be its complete law tuple wherever defined.

Assume primitive evaluations and $(\theta, z) \mapsto \varphi_\theta(z)$ are C^∞ on their open domains. Consequently evaluations at finitely many fixed task inputs are smooth on open parameter domains. At a complete feasible parameter value, the parameter derivatives of each law are continuous functions of the task inputs. These hypotheses allow open-domain primitives such as logarithms: the proof uses only open parameter neighborhoods where the finitely many selected evaluations are defined.

Call a complete-law equality

$$\mathsf{L}_{\mathbf{T}}(v) = \mathsf{L}_*(u)$$

a *bad match* if either $\mathrm{size}(\mathbf{T}) < B$, or $\mathrm{size}(\mathbf{T}) = B$ and its operator is inequivalent to the true operator. At such a pair define the finite-dimensional spaces of law variations

$$\mathcal{V}_*(u) = \mathrm{im}\, D_u \mathsf{L}_*(u), \qquad \mathcal{V}_{\mathbf{T}}(v) = \mathrm{im}\, D_v \mathsf{L}_{\mathbf{T}}(v).$$

They are spans of continuous functions across all task domains, defined using pointwise parameter derivatives. The rival derivatives are taken in the ambient operator-and-constant parameter space. At a boundary of complete-law admissibility this may include directions that cannot preserve definedness at every input simultaneously. Separation from this potentially larger space is a stronger sufficient condition; the proof requires only open neighborhoods for each finite set of evaluations.

Theorem 2 (Generic global recovery)**.** *Suppose that at every bad match, for every rival tuple of cost at most B,*

$$\boxed{\mathcal{V}_*(u) \not\subseteq \mathcal{V}_{\mathbf{T}}(v).} \tag{6}$$

Then, outside a Lebesgue-null subset of W, the global minimum in equation 1 is B, and every global minimizer uses an operator equivalent to the true operator on U.

The condition permits actual bad matches at exceptional parameters. It says that a wrong explanation cannot reproduce every infinitesimal change of the true collection of laws by retuning its own parameters. It is a condition on entire response families, rather than a rank test solely at the intended solution.

The following lemma handles intersections that are themselves singular and rival parameterizations with redundant coordinates.

Lemma 2 (Projection of matches with incompatible tangents)**.** *Let $F : W \subseteq \mathbb{R}^n \to \mathbb{R}^M$ and $H : V \subseteq \mathbb{R}^d \to \mathbb{R}^M$ be smooth on open sets. Let E be any subset of their coincidence set $F(u) = H(v)$ such that*

$$\mathrm{rank}[DF(u)\ DH(v)] > \mathrm{rank}\, DH(v) \qquad ((u, v) \in E).$$

Then the projection of E into W has Lebesgue measure zero.

Proof. Put $R(u, v) = H(v) - F(u)$. At a point of E, let $r = \mathrm{rank}\, DR$. Choose r component rows J of R whose derivatives are linearly independent. On the open full-row-rank locus W_J of DR_J, the set

$$M_J = \{R_J = 0\} \cap W_J$$

is a smooth manifold. At the chosen point, its tangent is $\ker DR$, since the selected rows span the full derivative row space.

The derivative of projection $M_J \to W$ has image

$$\{\delta u : DF(u)\delta u \in \mathrm{im}\, DH(v)\}.$$

Its rank is therefore

$$n - \big(\operatorname{rank}[DF(u)\ DH(v)] - \operatorname{rank} DH(v)\big) < n.$$

Thus the chosen truth parameter is a critical value of this smooth projection. Sard's theorem (Sard, 1942) says that these critical values have measure zero. There are finitely many possible ranks and component-row subsets. Every point of E is a critical point of at least one of these projections, so their finite union covers its projected image. No manifold assumption on E or constant-rank assumption on the full residual R was used. □

Proof of Theorem 2. A feasible true tuple gives an upper bound B. There are finitely many rival skeletons of cost at most B: replace each arbitrary real constant leaf by one constant-slot label, leaving its value in v. Thus nesting, repeated calls, and arbitrary constant values remain in the search.

Choose a countable dense subset of every Ω_i. Evaluations on these sets separate continuous tuples of functions. At any bad match, the sum $\mathcal{V}_*(u) + \mathcal{V}_{\mathbf{T}}(v)$ is finite-dimensional. Finitely many of the chosen evaluations are injective on this sum: if the current evaluation kernel contains a nonzero function tuple, continuity gives another chosen point where it is nonzero, reducing the kernel dimension. After finitely many reductions the kernel is zero.

For the resulting finite evaluation vector L, set

$$F_L(u) = L\mathsf{L}_*(u), \qquad H_L(v) = L\mathsf{L}_{\mathbf{T}}(v).$$

Injectivity on the tangent sum preserves both ranks, so equation 6 gives the strict rank inequality in Lemma 2. The finite evaluation domain of the rival is open: it requires only finitely many calls to primitives with open domains. We therefore apply that lemma on this domain to the subset of exact bad matches whose separation is detected by L.

There are only countably many finite subsets of the countable evaluation set. They detect every bad match. The union of their projected bad-match sets is null, by the lemma. Finally take the finite union over rival skeletons. Outside this exceptional set, no cheaper tuple fits and no equally cheap tuple fits with an inequivalent operator. The true tuple is feasible, which proves the claimed global minimum and identification. □

The proof uses finite evaluations only to detect finite-dimensional tangent spaces. It does not replace equality of complete laws by finite-sample interpolation. The conclusion is global over all rival parameter values, including distant aliases of the same tree.

F.4 When multiple laws are necessary at the optimum

The two theorems can be combined without assuming that the intended single-law trees are shortest.

Corollary 3 (Individual ambiguity and joint identification)**.** *Let the true parameters be $u = (\theta, c_1, \dots, c_m)$ in an open set W. Suppose the hypotheses of Theorem 1, including $N_i \supsetneq G$ for every i and $\bigcap_i N_i = G$, hold at every truth in this set. Suppose also that Theorem 2's separation condition holds for the joint search.*

For each law separately, assume either a uniform lower bound proving that its true tree is minimum-size, or the following one-law version of tangent separation: every exact match to that law by a smaller tree has a true-law tangent space not contained in the smaller tree's tangent space.

Then, for almost every true parameter tuple in W, each individual law has inequivalent globally minimum-size learned operators, while the joint globally minimum explanation identifies the true operator's equivalence class.

Proof. For a single law, the proof of Theorem 2, retaining only cheaper rivals, excludes all such rivals outside a null subset of its coordinates (θ, c_i). Its true tree is feasible and therefore minimum-size there. Alternatively, an independently established lower bound supplies the same conclusion directly.

The coordinate projection from W onto (θ, c_i) is a submersion. Pullbacks of null sets under this projection are null by Fubini's theorem. Hence, outside a finite union of null subsets of W, every

true single-law tree is minimum-size. Theorem 1 supplies nearby exact single-law alternatives with inequivalent operators at those unchanged tree sizes. They are therefore also single-law global minima.

Exclude in addition the null set in Theorem 2. That theorem identifies the operator equivalence class across all joint global minima. This proves both assertions simultaneously. □

Complexity lower bounds can sometimes be checked directly from the laws. For example, a binary tree depending essentially on k distinct variables needs at least k leaves and hence at least $2k-1$ nodes. Such a bound restricts the optimum because of the observations; it does not restrict the search space in advance. The one-law separation alternative in the corollary is available when a simple count is not sharp.

F.5 A STRUCTURAL BOUND FROM SHARING ONE PARAMETER VECTOR

At a true/rival match, write the variations of law i as

$$\delta f_i = A_i^{\mathrm{op}}\delta\theta + A_i^{\mathrm{const}}\delta c_i, \qquad \delta f_i = B_i^{\mathrm{op}}\delta\eta + B_i^{\mathrm{const}}\delta d_i,$$

where θ and η are shared across their respective tuples, while the constants are law-specific. Let Q_i be the quotient by $\operatorname{im} B_i^{\mathrm{const}}$, and define

$$e_i = \operatorname{rank}[A_i^{\mathrm{const}}\ B_i^{\mathrm{const}}] - \operatorname{rank} B_i^{\mathrm{const}}, \qquad s = \operatorname{rank}\begin{bmatrix} Q_1 B_1^{\mathrm{op}} \\ \vdots \\ Q_m B_m^{\mathrm{op}} \end{bmatrix}.$$

Here e_i counts changes in law i produced by its true constants that its rival constants cannot reproduce. After those rival constants are eliminated, the shared rival operator supplies $s \le p$ independent changes across the entire collection.

Lemma 3 (Shared-parameter dimension bound)**.** *The tangent difference in Theorem 2 satisfies*

$$\operatorname{rank}[D_u \mathsf{L}_*\ D_v \mathsf{L}_{\mathbf{T}}] - \operatorname{rank} D_v \mathsf{L}_{\mathbf{T}} \ \ge\ \sum_i e_i - s. \tag{7}$$

Consequently, $\sum_i e_i > s$ at every bad match of a rival tuple is sufficient to exclude that tuple for almost every truth.

Proof. The direct sum of rival constant-variation spaces is contained in $\mathcal{V}_{\mathbf{T}}$. Quotienting $\mathcal{V}_* + \mathcal{V}_{\mathbf{T}}$ and $\mathcal{V}_{\mathbf{T}}$ by that direct sum leaves their dimension difference unchanged. In this quotient, the true operator directions form the stacked map $A = (Q_i A_i^{\mathrm{op}})_i$, its constant directions form the block-diagonal map $C = \operatorname{diag}(Q_i A_i^{\mathrm{const}})$, and the rival operator directions form $B = (Q_i B_i^{\mathrm{op}})_i$. The difference is therefore

$$\operatorname{rank}[A\ C\ B] - \operatorname{rank} B \ \ge\ \operatorname{rank} C - s = \sum_i e_i - s.$$

A positive difference is exactly the tangent-separation condition. The proof of Theorem 2 excludes this rival outside a null set. □

The important constraint is that the rival has one shared η, rather than a separate operator parameter vector for each law. The uncanceled private changes accumulate across laws, while the rank s is at most the dimension of that one shared vector. This bound is sufficient, not necessary: a law with $e_i = 0$ can still contribute through changes in the true shared operator.

A cruder check can avoid solving for any matching parameters. If $\operatorname{rank} A_i^{\mathrm{const}} \ge \rho_i$ throughout the truth region and rival tree i has q_i' constant slots, then

$$e_i \ge (\rho_i - q_i')_+.$$

Thus $\sum_i (\rho_i - q_i')_+ > p$ excludes that entire rival family generically. Every affordable rival still needs this or another separation argument; the bound for one rival is not a full-search proof.

F.6 How to verify the structural conditions

Complementarity. Differentiate the complete laws with respect to the shared operator parameters and every ordinary constant. Eliminate the images of the constant derivatives before comparing operator directions. Under the regularity and compensation hypotheses, the positive-definite sum in equation 5 is a concrete test: every inequivalent operator change must affect at least one law in a way that changing its constants cannot undo. A pointwise rank calculation alone does not prove constant rank in a neighborhood; both a rank lower bound and a matching structural upper bound are needed.

Tangent separation. A stronger sufficient condition for equation 6 is

$$\operatorname{rank} D_u \mathsf{L}_*(u) > \sup_v \operatorname{rank} D_v \mathsf{L}_{\mathbf{T}}(v).$$

The rival's number of parameters bounds the right side, and parameter redundancies can lower it further. When the two ranks are equal, their images can still be different. A linear functional w on the law-variation space satisfying

$$w D_v \mathsf{L}_{\mathbf{T}} = 0, \qquad w D_u \mathsf{L}_* \neq 0$$

certifies separation at a match. It selects a change in the collection of laws that the rival cannot reproduce. These conditions must address all affordable rival trees, including the true tree evaluated at inequivalent remote parameters.

Interaction identities. Tree structure can force identities that exclude rivals before a rank comparison is needed. If a tree adds functions of disjoint variable blocks, its mixed derivatives across those blocks vanish. If it multiplies such functions, it satisfies $f\,\partial_i\partial_j f - (\partial_i f)(\partial_j f) = 0$ across the blocks. A family of laws violating a forced identity cannot arise from that tree. Shared-operator constraints must likewise be checked jointly across laws. These are examples of structural checks, not additional assumptions that the observed laws have a particular formula.

Complete observations. For the local ambiguity claim, use complete function-valued law maps, or a finite signature proved to determine the complete laws in the family. A kernel computed from arbitrary finite samples can miss changes elsewhere and does not prove exact single-law ambiguity. The global exclusion theorem avoids this issue by applying its tangent condition to complete-law matches and using evaluations only within its proof.

Piecewise-smooth models. A finite-signature version follows from Lemma 2: apply it separately on smooth truth-parameter strata, require the rank condition at every bad signature match, and include every smooth parameter stratum of every rival. Countably many rival strata give a countable union of null exceptional sets on each truth stratum. Lower-dimensional rival strata cannot be discarded; a recovery claim restricted to a lower-dimensional truth stratum requires its own application. This supplies a route for spline and ReLU families when their finite signatures admit suitable smooth stratifications. It does not automatically establish the local complete-function differentiability assumptions: a fixed activation pattern at sampled points is insufficient for a claim about the entire law.

F.7 Scope of the result

The local theorem explains how different laws remove different operator ambiguities after ordinary constants are optimized. The global theorem supplies a separate structural requirement excluding competing trees and distant operator aliases for almost every truth in a smooth finite-dimensional family. The corollary combines these into a theorem in which individual minimum-size explanations remain ambiguous but the joint minimum identifies one operator equivalence class.

The hypotheses are sufficient and require verification for the chosen library and law family. A failed rank test does not prove nonidentification. The generic exceptional set can include special physical parameter values. No claim is made that all experimental instances satisfy the conditions, that optimization is computationally easy, or that evolutionary search reaches the global optimum. The operator equivalence and identification domain are fixed before applying the theorem.