Automated Symbolic Law Discovery: A Computer Vision Approach

Hengrui Xing, Ansaf Salleb-Aouissi, Nakul Verma
Department of Computer Science, Columbia University, New York, USA
h.xing@columbia.edu, ansafsalleb@columbia.edu, verma@cs.columbia.edu

Abstract

One of the most exciting applications of modern artificial intelligence is to automatically discover scientific laws from experimental data. This is not a trivial problem as it involves searching for a complex mathematical relationship over a large set of explanatory variables and operators that can be combined in an infinite number of ways.

Inspired by the incredible success of deep learning in computer vision, we tackle this problem by adapting various successful network architectures into the symbolic law discovery pipeline. The novelty of our approach is in (1) encoding the input data as an image with super-resolution, (2) developing an appropriate deep network pipeline, and (3) predicting the importance of each mathematical operator from the relationship image. This allows us to prior the exponentially large search with the predicted importance of the symbolic operators, which can significantly accelerate the discovery process.

We apply our model to a variety of plausible relationships—both simulated and from physics and mathematics domains—involving different dimensions and constituents. We show that our model is able to identify the underlying operators from data, achieving a high accuracy and AUC (91% and 0.96 on average resp.) for systems with as many as ten independent variables. Our method significantly outperforms the current state of the art in terms of data fitting (R^2), discovery rate (recovering the true relationship), and succinctness (output formula complexity). The discovered equations can be seen as first drafts of scientific laws that can be helpful to the scientists for (1) hypothesis building, and (2) understanding the complex underlying structure of the studied phenomena. Our approach holds a real promise to help speed up the rate of scientific discovery.

Introduction

A persistent goal in the history of scientific research is to extract the relationship between variables from data. In most scientific fields, these relationships can be reduced to simple expressions composed from elementary mathematical operators. Physicists, for example, have formulated laws governing natural phenomena including motion, electricity, etc.

A key objective in modern artificial intelligence is to automate this process. Symbolic regression (SR) attempts to achieve this by searching in the space of mathematical expressions. Specifically, for a data pair (x, y), SR seeks to identify a mapping f that best describes the relationship $x \rightarrow y$, i.e., that minimizes a certain loss function. f is required to be a symbolic expression assembled from mathematical operators, explanatory variables, and constants.

In conventional regression, one prescribes a parametric model for f. This almost always suffers from the trade-off between expressiveness and interpretability. Simple models (such as linear regression) can severely underfit a complex system. Composite models (such as neural network), on the other hand, are not easily understood by humans (Fan, Xiong, and Wang 2020). SR, in contrast, can fit to an arbitrarily complex target function and provide readily interpretable results. Even if it does not precisely uncover the ground truth functional form, the result still offers useful insight to the development of scientific theory.

SR is classically implemented via genetic programming (Koza 1994). The mapping f is expressed as a symbolic tree. Each internal node of the tree is an operator and each leaf node is either an explanatory variable or a constant. The algorithm randomly initializes a population of symbolic trees. These instances then fit the dataset through a series of evolutionary steps, or generations. In each generation, new trees are created by genetic operations, typically including point mutation and crossover (subtree exchange). The best-performing trees are then selected to proceed to the next generation, based on a fitness measurement on the training set (e.g., root mean square error). Empirically, the performance increases with each generation and we can obtain a plausible symbolic relationship after sufficient rounds.

There is a serious bottleneck of this genetic approach which is seldom discussed in other SR literature – the operator set. Before running the algorithm, one needs to define the search space by providing a set of available operators (`add`, `log`, `sin`, etc.). The model will fail to uncover the true relationship $x \rightarrow y$ if the set lacks some constituent operator. On the other hand, the speed and performance of the algorithm will be severely hindered if the set is too large. This is because: 1) We need a larger population and/or more generations for an exponentially large search space. 2) There are more false optima of the loss function due to extraneous operators. We will quantitatively evaluate the impact of operator set initialization in the subsequent sections.

Ideally, we should prior the algorithm with and only with the operators that constitute the actual symbolic relationship. However, this information is most often not provided with the raw data when scientists study a new system. In this paper, we focus on this symbolic operator identification problem. Formally, we are given data pair (x, y), such that each i^{th} observation satisfies $y^i = f(x^i) + \epsilon$. Here f is a symbolic relationship and ϵ is random noise. Our model determines whether an operator is contained within the expression of f.

We present **De**ep **S**ymbolic **Tr**ee **O**perator **I**dentifier (DeSTrOI) as the first practical solution. DeSTrOI first extracts the relationship between variables and encode it into an image. We employ a super-resolution residual network (ResNet) as our encoder. We then use a deep convolutional neural network (CNN) to decode from the image and predict the existence of each operator, given in an importance score. We apply a gated attention mechanism to account for high dimensional data. Our architecture is able to achieve 85%-96% accuracy and 0.90-0.99 area under curve (AUC) for our test operators (`add`, `mul`, `inv`, `sqrt`, `log`, `sin`) and up to 10 explanatory variables. We apply our results to genetic algorithms and find a $\sim$ 10% increase in R^2, $\sim$ 25% increase in discovery rate and $\sim$ 10% decrease in output formula complexity (all statistically significant). Our model also significantly improves the efficiency for other non-genetic state-of-the-art SR methods.

Our paper is organized as follows: In Methods, we describe the DeSTrOI architecture including encoder, decoder and the attention mechanism. In Results for Synthetic Data, we discuss our synthetic dataset and the training process. We also present the DeSTrOI performance and compare it to two simpler baseline models that we develop. In Results for Real Data, we apply DeSTrOI to actual problems in mathematics, physics and engineering, as real scientists would.

DeSTrOI will be useful even outside of the context of SR. By identifying the mathematical operators from data, it can guide scientific researchers to derive the underlying formulas. For example, our algorithm can report, with high confidence, whether or not there should be a sinusoidal component in the mapping $x \rightarrow y$. This effectively provides insight for scientists to understand the nature of this relationship.

Related Works

Symbolic Regression (SR)

Symbolic equation generation from data was first studied in computational physics (Crutchfield and Mcnamara 1987; Crutchfield and Young 1989). Later, Koza (1994) formulated SR as an artificial intelligence problem and designed the popular genetic algorithm solution. This method is modified and applied in the seminal work by Schmidt and Lipson (2009) to discover equations of motion in nonlinear dynamical systems. The authors later developed Eureqa, the first commercialized SR software.

Evolutionary algorithms have since become the predominant method for SR, and several optimizations have been developed to improve its accuracy and efficiency (Nguyen et al. 2011; Amir Haeri, Ebadzadeh, and Folino 2017). This body of works mainly focus on the evolutionary process. They attempt to achieve better performance through loss function selection, pruning, and hyper-parameter adjustments. Our work innovatively highlights the operator set as an important prior for the algorithm. We demonstrate that the operator set initialization significantly impacts the genetic algorithm performance. Some authors have considered methods for operator set optimization (Lu, Ren, and Wang 2016). However, they all rely on external information such as expert knowledge to prior the algorithm. Our DeSTrOI approach, on the other hand, seeks to identify the operators from the raw data itself.

Deep Learning for SR

Developments in machine learning (esp. deep learning) have prompted researchers to apply these methods in SR. Udrescu and Tegmark (2020) recently developed AI Feynman, a framework that constructs the underlying symbolic equation through a series of feature extraction steps. In particular, they use a multi-layer perceptron (MLP) to fit the data and identify symmetries such as translational invariance. This is, in fact, a very crude version of our architecture. We provide super-resolution ResNet as a more accurate encoder than MLP. Moreover, while they try to identify hand-crafted symmetry features, we use deep CNN to automatically learn from the images. As we show later, DeSTrOI with genetic algorithm significantly outperforms AI Feynman.

Sahoo, Lampert, and Martius (2018) developed an Equation Learner Network. It replaces the activation function of each internal node in an MLP with a mathematical operator. This model can indirectly select operators by imposing an L1 regularization. However, this architecture strictly limits the depth and width of the symbolic tree. In addition, because gradient has to propagate through each internal node, divergent operators such as `log` and `sqrt` cannot be included, severely biasing the search space.

Deep generative models are also explored in recent studies as an alternative to genetic algorithms. Kusner, Paige, and Hernández-Lobato (2017) use variational autoencoder to train a latent space representation for the symbolic expressions. Language models with recursive neural networks are applied to "translate" the dataset to a string representation of the symbolic tree (e.g., a preorder traversal). To deal with the non-differentiable loss, Anjum et al. (2019) use an evolutionary algorithm and Petersen (2019) uses deep reinforcement learning to optimize the parameters. These methods do not, in general, outperform state-of-the-art genetic algorithm solutions. They additionally suffer from the problem that their output string might not reconstruct to a valid symbolic tree (incorrect grammar). Therefore, we constrain ourselves to DeSTrOI's application to genetic algorithm, which is the mainstream solution for SR.

We note that all major works on SR assume a predefined set of operators. Therefore, DeSTrOI can be applied as an extra layer to all the existing methods. As we demonstrate in the Results for Real Data, our model significantly reduces the run time for AI Feynman. For generative models, we can use our results to constrain the model vocabulary, which has shown significant improvement in performance and efficiency for machine translation (Post and Vilar 2018).

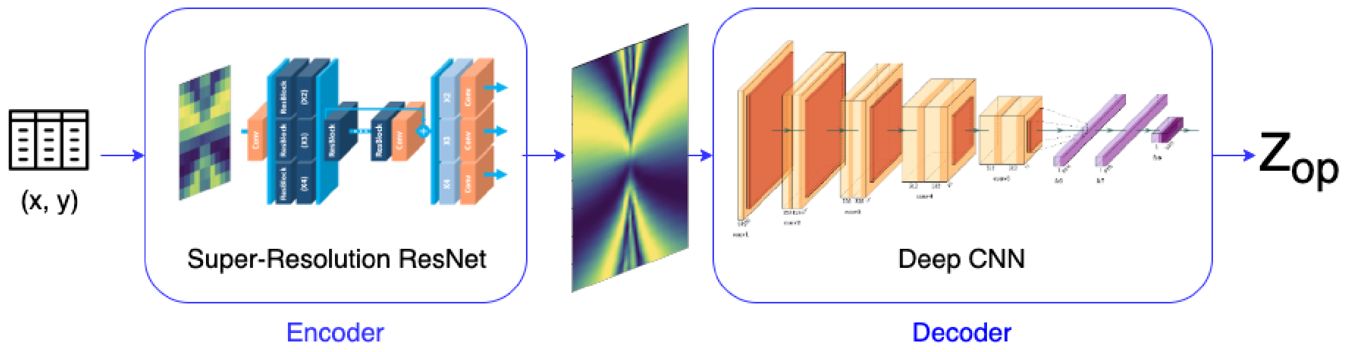


Figure 1: Diagram illustrating the general DeSTrOI architecture. In summary, (x, y) will be encoded via the super-resolution ResNet as an image that plots the underlying relationship. This encoding is then fed into a deep CNN decoder that predicts z_{op}.

Symbolic Operator Identification

To the best of our knowledge, only Zhong et al. (2018) have studied the symbolic operator identification problem. They proposed DL-GEP, a method that attempts to identify operators from a line plot of the data. Some important limitations of their work that we have addressed in our model are:

- DL-GEP only works for univariate data, whereas DeSTrOI is applicable for data with arbitrary number of explanatory variables. We have verified that it achieves sound test performance for up to 10 explanatory variables. Note that most real scientific problems are multivariate.
- The core DL-GEP mechanism crucially relies on synthetically generated line plots. It is unclear how these plots get produced from discrete data (e.g., no description of interpolation method). We, instead, provide a well-defined encoder to generate an image for each relationship.
- DL-GEP fails to (significantly) outperform baseline genetic algorithm. (It has better performance for some instances and worse for others. The authors have not included a statistical evaluation.) DeSTrOI gives significantly better results on a variety of metrics.

Deep Learning for Computer Vision

Our model leverages tools and techniques from learning-based computer vision. Super-resolution seeks to recover a high resolution image from a low resolution sample. The state-of-the-art models provide solutions using ResNet (Lim et al. 2017; Yu et al. 2018) and generative-adversarial networks (GAN) (Ledig et al. 2017). To our knowledge, super-resolution has not appeared in SR literature. We choose the ResNet architecture as our encoder because of its lower reliance on sample size. Furthermore, We utilizes a deep CNN for our decoder. Our model is modified from the VGG architecture developed by Simonyan and Zisserman (2015).

Multiple Instance Learning (MIL)

Our model employs MIL to address higher dimensional data. This technique, first explored by Dietterich, Lathrop, and Lozano-Pérez (1997), pools an unordered bag of instances and predicts a label for the bag. Our model design is inspired by MIL's success in medical imaging. Ilse, Tomczak, and Welling (2018) have implemented deep attention-based MIL to detect a cancer site from multiple medical scans. We innovatively leverage this mechanism to aggregate different projections of a multi-dimensional dataset.

Methods

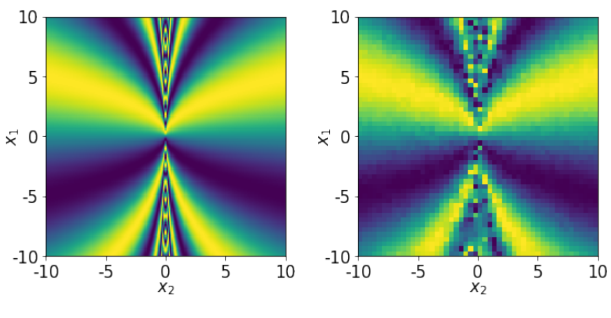
Our **De**ep **S**ymbolic **Tr**ee **O**perator **I**dentifier (DeSTrOI) predicts whether some mathematical operator (`add`, `log`, `sin`, etc.) exists in a symbolic relationship f given a pair of data (x, y) that satisfy $y^i = f(x^i) + \epsilon$ (ϵ is a random noise term). Each input $x^i \in \mathbb{R}^k$ is an observation vector for the k explanatory variables, and the output $y^i \in \mathbb{R}$ corresponds to the response variable. For every operator `op`, we assign a label z_{op} to indicate whether it exists within f. For example, for the relationship $y = \sin(x_1\sqrt{1/|x_2|})$, we would have $z_{\texttt{add}} = 0$, $z_{\texttt{sin}} = 1$, $z_{\texttt{sqrt}} = 1$, etc. For $k = 1$, operator selection does not have significant impact for SR (Zhong et al. 2018). Therefore, we will focus on the harder task of $k \geq 2$.

We use the encoder-decoder paradigm to develop our model. A general diagram of our model architecture is shown in Figure 1. For each relationship f, the encoder outputs an image (a plot of f). Specifically, we construct a grid of points in the explanatory variable space $\mathbb{R}^k$ for a particular cubic region. With (x, y) as input, our encoder predicts the value of y at each point on the grid. The result will be a single channel k-d image. As an example, the (ground truth) plot of $y = \sin(x_1\sqrt{1/|x_2|})$ is shown in Figure 2a. Our encoder model will attempt to reconstruct this plot using (x, y). We then use this visual representation of the relationship to predict z_{op} via the decoder model.

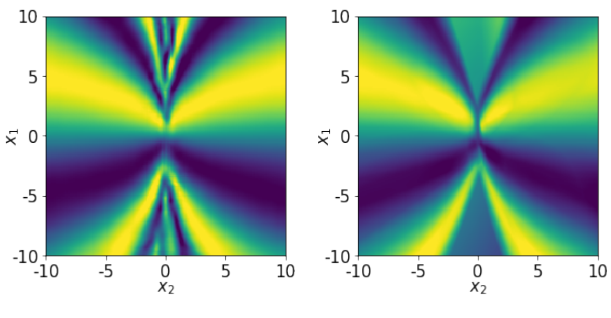
We use a super-resolution ResNet as encoder and a deep CNN as decoder. We further apply a gated attention mechanism for higher dimensional data (explained in the subsequent sections). Ultimately, we apply our results as a prior for SR using genetic algorithm. We probabilistically include each operator based on its predicted importance. This significantly accelerates the discovery of the underlying formula.

Super-Resolution Encoder

For a data pair (x, y), we first generate a naive image encoding: We divide the cubic region in $\mathbb{R}^k$ into a grid of cells. We assign a value for each cell as the average y value of



(a) Ground truth (b) Naive encoding (0.10)



(c) Super-Resolution (0.06) (d) MLP (0.09)

Figure 2: Encoding images for $y = \sin(x_1\sqrt{1/|x_2|})$. Parenthesized values are reconstruction MAE.

data points within this region. For cells that does not contain a data point, we interpolate its value by averaging the values of its immediate neighbors. To account for adjacent cells without enclosing data points, we sweep through the grid a number of times and iteratively recompute the averages.

For our example function $y = \sin(x_1\sqrt{1/|x_2|})$, this naive encoding is shown in Figure 2b. In comparison to the ground truth, it is noisy and discontinuous, especially around $x_2 = 0$. These will be very problematic for our CNN decoder.

To fix this, we apply a modified enhanced deep super-resolution network (EDSR) (Lim et al. 2017) that removes the noise and increases the resolution of the naive encoding. EDSR is composed from a series of 8 residual blocks. Each block contains 2 convolutional layers with 64 filters. We train this model using the mean absolute error (MAE) loss. The output encoding for our example function is illustrated in Figure 2c. The noisy regions in the naive encoding is properly smoothed and MAE is almost halved.

An alternative encoding method is to train a multi-layer perceptron (MLP) network to predict the value of each cell. This method is included in the Appendix for reference. For completeness, we have included the MLP encoding for our example function in Figure 2d. In general, super-resolution has much better performance and efficiency.

CNN Decoder

Once we have a high quality visual relationship between x and y, we feed it to our decoder, which predicts the existence of each symbolic operator. For $k = 2$, we apply a deep CNN model modified from VGG-16 (Simonyan and Zisserman

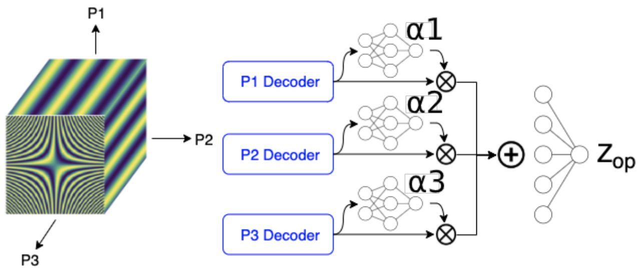


Figure 3: Diagram illustrating the gated attention MIL mechanism. Each 2D projection of the data will be separately encoded. Before prediction, their activation will be pooled together according to their attention weights α's.

2015). The architecture contains 5 convolution blocks, with 64, 128, 256, 512, 512 filters, respectively. Each block has 2 convolutional layers and 1 max-pooling layer. The output of the last block then propagates through 3 fully connected layers, with 4096, 4096, 1000 neurons, respectively. We use an Exponential Linear Unit (ELU) activation for each kernel since the encoding contains both positive and negative values. We regularize the model with an L2 weight decay and 50% dropouts for the fully connected layers.

We train a different model for each `op`. These models do not share weights since we do not presume a predefined set of operators. Our model can then be easily extended to incorporate new operators on demand.

Attention-Based MIL

For a scientific system with more explanatory variables (i.e., $k > 2$), we construct a new (x', y) for each pair of explanatory variables by hiding the other variable values. In effect, we project the original k-d dataset onto each of the orthogonal 2-d planes. For a relationship f, we now have $\binom{k}{2}$ sets of data points (x', y). We can then leverage multiple instance learning (MIL) to perform the decoding.

As illustrated in Figure 3, our encoder generates a bag of $\binom{k}{2}$ images. We input them into the original decoder and pool their activation in the last layer before predicting. Suppose the activation vectors are $\{\boldsymbol{h}_1, \ldots, \boldsymbol{h}_L\}$. They are aggregated through a gated attention mechanism (Ilse, Tomczak, and Welling 2018):

$$\boldsymbol{a} = \sum_{l=1}^{L} \alpha_l \boldsymbol{h}_l \tag{1}$$

$\boldsymbol{a}$ is then used to predict the label for the bag. The α_l are attention weights calculated from:

$$\alpha_l = \frac{\exp[\boldsymbol{w}^T(\tanh(\boldsymbol{V}\boldsymbol{h}_l) \odot \sigma(\boldsymbol{U}\boldsymbol{h}_l))]}{\sum_{l=1}^{L}\exp[\boldsymbol{w}^T(\tanh(\boldsymbol{V}\boldsymbol{h}_l) \odot \sigma(\boldsymbol{U}\boldsymbol{h}_l))]} \tag{2}$$

Here $\boldsymbol{w} \in \mathbb{R}^M$, $\boldsymbol{V} \in \mathbb{R}^{M\times 1000}$ and $\boldsymbol{U} \in \mathbb{R}^{M\times 1000}$ are trainable parameters. (In practice, we choose $M = 250$.) $\odot$ is element wise multiplication and $\sigma(\cdot)$ is the sigmoid function. This mechanism learns what makes a projection relevant for the operator prediction task. This is more powerful than simple aggregates (e.g., mean, max).

Results for Synthetic Data

DeSTrOI is first trained and tested on our synthetic dataset with randomly generated symbolic trees. Our dataset consists of multiple (x, y) pairs, each sampled from a different relationship. In the next Results for Real Data section, we will demonstrate the model's applicability in real problems of science and engineering.

Synthetic Dataset

We generate symbolic trees with minimum depth 2 (e.g., $y = \sin(\log(x))$) and maximum depth 4 (e.g., $y = \sin(\log(\sin(\log(x))))$). This corresponds to the complexity of typical symbolic formulas in physical sciences. We select each internal node randomly from the set of six operators: {`add`, `muliply`, `inverse`, `sqrt`, `log`, `sin`}. As discussed before, we can easily extend our model to new operators on demand. The leaf nodes are either randomly selected explanatory variables or random constants in the range $[-1, 1]$. We also ensure that the symbolic relationships all have reasonable value ranges. In practice, we exclude instances that give y values exceeding ± 100 for explanatory variables within $[-10, 10]$. Moreover, we add random Gaussian noise $\epsilon \sim \mathcal{N}(0, 0.01)$ independently to each y to emulate the inherent imprecision in real data. A more detailed description of data generation is included in the Appendix. As shown in Table 1, our simulation scheme gives relatively balanced class labels for each of the operators. We randomly split the dataset into 80% for training, 10% for early-stopping, and 10% for testing.

Baseline Models

Since we are working on the novel operator identification task, there are no established baselines. We present two simple and intuitive baseline models to benchmark the performance of DeSTrOI.

- **Genetic Algorithm (GA)**. A natural predictor to use is to directly perform genetic algorithm. We use the entire set of 6 operators for each symbolic relationship and predict based on whether the best-performing instance contains a given operator. We use the GPLearn software package that provides a sophisticated genetic algorithm for SR [1]. In addition to the low accuracy, GA has a few drawbacks as a predictor: 1) It does not provide a confidence level for the predictions. 2) GA is much more computationally expensive than parametric methods, especially in the testing phase.

- **Multi-Layer Perceptron (MLP)**. We can use an MLP network to predict operator existence. We flatten the set of (x^i, y^i) pairs for each symbolic relationship into a vector of dimension $m \times (k+1)$ and use it as the input. (m is the number of data pairs.) In our experiment, we use 4 hidden layers each with 2048 neurons. In addition, the model prediction should be invariant under permutation of the input (x^i, y^i) pairs. To account for this, we augment the data by randomly shuffling the pairs 20 times. At testing time, we average the predicted operator importance scores over all the permutations. Note that this method is much more computation and memory intensive than DeSTrOI.

k	2	3	5	10
`add`	0.43	0.42	0.42	0.42
`mul`	0.42	0.41	0.41	0.41
`inv`	0.43	0.39	0.39	0.39
`sqrt`	0.59	0.59	0.58	0.58
`log`	0.55	0.56	0.56	0.57
`sin`	0.56	0.56	0.56	0.58

Table 1: Class imbalance in our synthetic data set for different k. Values are given in proportion of positive instances.

[1]Stephens, T. 2018. GPLearn: Genetic Programming in Python. URL: https://github.com/trevorstephens/gplearn

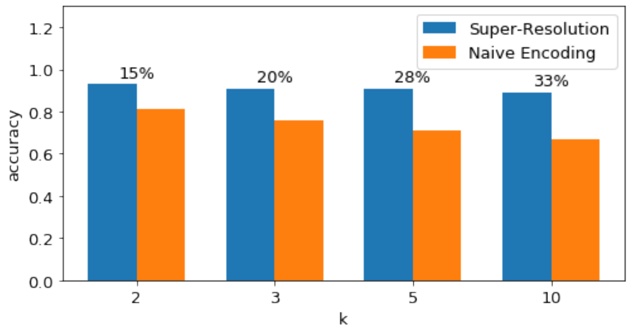


Figure 4: DeSTrOI accuracy with and without the super-resolution encoder. We also show percentage improvement.

The baseline performance for $k = 2, 3$ is shown in Table 2a. For higher k, these models are infeasible: GA takes too long to converge ($>$ 100 hours) and MLP requires too much memory ($>$ 100GB). Observe that both baselines yield insufficient accuracies to be useful for real life applications.

DeSTrOI

The accuracy of our model on a held-out testing set is shown in Table 2a. The area under curve (AUC) scores are shown in Table 2b. DeSTrOI significantly outperforms the baseline methods for all the operator types and number of independent variables. On average, we obtain 91% accuracy and 0.96 AUC. Such a high performance suggests that DeSTrOI is reliable for real scientific problems. Comparatively, `sin` and `log` predictors are the most accurate while `inv` is the hardest to predict. The experiment also verifies that our architecture is robust against image orientation and rescaling (More details in Appendix).

We further verify our model by removing the encoder component. In Figure 4, we compare the DeSTrOI accuracy when using the super-resolution encoding vs. the naive encoding. Our super-resolution encoder gives a significant contribution to the performance of our model, especially for higher k (where the naive encoding is more prone to noise).

k	2			3			5	10
Model	GA (baseline)	MLP (baseline)	**DeSTrOI**	GA (baseline)	MLP (baseline)	**DeSTrOI**	**DeSTrOI**	**DeSTrOI**
`add`	0.72	0.65	**0.94**	0.71	0.61	**0.91**	**0.91**	**0.90**
`mul`	0.73	0.66	**0.90**	0.71	0.57	**0.89**	**0.88**	**0.86**
`inv`	0.73	0.60	**0.88**	0.70	0.55	**0.87**	**0.86**	**0.85**
`sqrt`	0.78	0.77	**0.91**	0.79	0.76	**0.89**	**0.91**	**0.86**
`log`	0.80	0.65	**0.96**	0.77	0.60	**0.95**	**0.95**	**0.93**
`sin`	0.73	0.73	**0.96**	0.70	0.70	**0.96**	**0.96**	**0.92**
Average	0.75	0.68	**0.93**	0.73	0.63	**0.91**	**0.91**	**0.89**

(a) Accuracy

k	2		3		5	10
Model	MLP (baseline)	**DeSTrOI**	MLP (baseline)	**DeSTrOI**	**DeSTrOI**	**DeSTrOI**
`add`	0.71	**0.97**	0.58	**0.96**	**0.96**	**0.96**
`mul`	0.70	**0.95**	0.57	**0.94**	**0.94**	**0.94**
`inv`	0.61	**0.92**	0.65	**0.92**	**0.89**	**0.91**
`sqrt`	0.77	**0.96**	0.77	**0.96**	**0.96**	**0.94**
`log`	0.73	**0.99**	0.72	**0.99**	**0.98**	**0.98**
`sin`	0.77	**0.99**	0.68	**0.98**	**0.99**	**0.97**
Average	0.71	**0.96**	0.66	**0.96**	**0.95**	**0.95**

(b) Area Under Curve (AUC)

Table 2: Performance of our models for different operator and k. DeSTrOI significantly outperforms the baseline models. For larger k (i.e. $k = 5$ and 10), the baseline models are computationally infeasible. In addition, GA baseline does not give a probabilistic score so it does not yield an AUC.

Symbolic Regression (SR)

We now apply DeSTrOI to the overarching SR task. We employ the genetic algorithm for SR provided by the GPLearn software. We run the evolution 1) with DeSTrOI (i.e., with each operator probabilistically included based on its predicted importance) and 2) with all 6 operators as the baseline. We use the same hyper parameter settings and measure the SR predictive ability (R^2), discovery rate (percentage discovering the true formula) and result complexity (# of internal nodes).

The result after 20 generations is shown in Figure 5. For all k, DeSTrOI shows (statistically) significantly better performance for all three metrics. That is, our method fits the data more accurately and do so with a more succinct symbolic expression. It also has a much better capability of discovering the true formula. In addition, as shown in detail in the Appendix, DeSTrOI consistently beats the baseline for all number of generations.

Results for Real Data

We now seek to apply DeSTrOI to real problems in science and engineering. Our experiments are summarized in Table 4. The formulas are selected to have different k and combinations of constituent operators. For all experiments, we include the predicted operator importance value in Table 3. We are able to give correct predictions for the majority of cases, except for the two cells marked with asterisks.

We subsequently apply genetic algorithm as described in the Methods section. Our DeSTrOI-priored algorithm is compared to the baseline where we input all 6 operators. Both algorithms are run for 20 generations with the same hyper-parameters (where both are able to converge). We perform 100 independent trials for each formula. The comparative performance is illustrated in Figure 7. (Detailed statistics are included in the Appendix.) DeSTrOI outperforms the vanilla genetic algorithm for all the metrics. In general, DeSTrOI is able to find a more succinct formula that better describes the data.

Additionally, we apply DeSTrOI to AI Feynman (Udrescu and Tegmark 2020), a non-genetic state-of-the-art method. Our model pre-selects the operators and prunes the search tree for the AI Feynman algorithm. Besides Table 4, we also use AF1, the example formula provided in the AI Feynman publication ($\sqrt{(x_2 - x_1)^2 + (x_4 - x_3)^2}$). As shown in Figure 6, we significantly reduce its run time for all 5 formulas. (Note that AI Feynman is a brute-force method, so we cannot improve its accuracy. The accuracy scores are lower than baseline genetic algorithm and included in the Appendix.)

Simple Harmonic Oscillator (SHO). SHO is a well-studied phenomenon in classical (and quantum) mechanics. It describes the motion of a massive particle in a quadratic potential (e.g., a ball attached to a spring). We view the time t and spring strength k as explanatory variables. We assign the oscillation amplitude constant $A = 2$. The visual encoding of this relationship is shown in Figure 8a. Here we use a trick and allow both k and t to be negative since De-

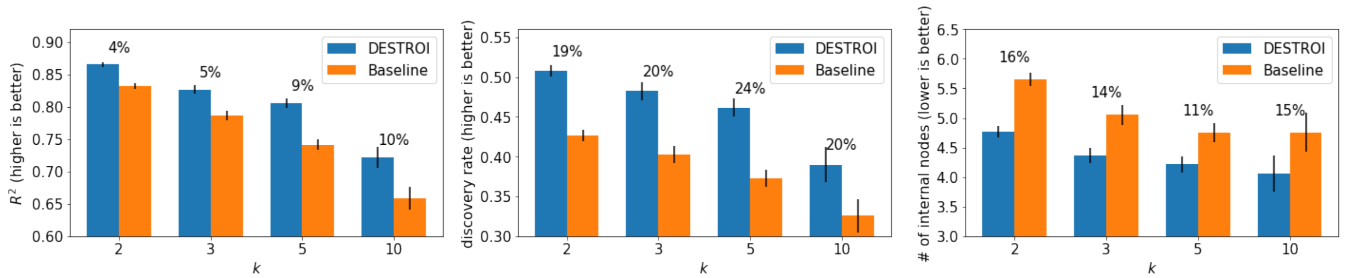


Figure 5: Performance of DeSTrOI as applied to SR vs. baseline. The fittest tree instance at the 20th generation is selected. The result is shown for R^2, discovery rate, and complexity (number of internal nodes). The percentages show the improvement from baseline. Error bars show ± 1 standard error.

Experiment	SHO	IIG	FSA	TRE
add	0.00	0.14	1.00	0.78
multiply	1.00	1.00	1.00	1.00
inverse	0.00	0.63*	1.00*	0.99
square root	1.00	0.38	1.00	0.0
log	0.00	0.94	0.00	1.00
sin	1.00	0.30	0.08	0.01

Table 3: Predicted importance of the operators for our experiments. The incorrect predictions are marked with asterisks.

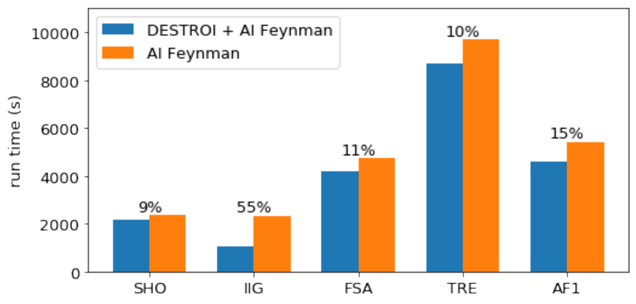


Figure 6: Run time of AI Feynman with and without DeSTrOI . We also show the percentage improvement.

STrOI requires a region symmetric around the axes. Negative time can be achieved by a translation (and is inherent for quantum SHO). For k, we replicate the data from the positive half-plane. In practice, we can always do the same for non-negative explanatory variables. Remarkably, our method discovers the correct formula 73% of the time whereas the baseline only does so 15% of the time.

Isothermal Ideal Gas (IIG). Ideal gas is a model for homogeneous gases. When the gas expands, it transfers heat to the container. Physicists are especially interested in this process under constant temperature (isothermal expansion). The energy transfer depends on temperature T and volume V. We assign $k = 1$ and $V_0 = 3$. The visual encoding is shown in Figure 8b and we use the same trick to allow negative T and V. Our method discovers the correct formula 24% of the time whereas the baseline only does so 2% of the time.

Frustum Surface Area (FSA). (Conical) frustum is a geometric object obtained by removing the top of a circular cone. Its lateral surface is a fraction of a ring and its area is prescribed by top radius r, bottom radius R, and height h. We project the data points onto the 3 orthogonal surfaces and the encoding images are included in the appendix.

Tsiolkovsky Rocket Equation (TRE). TRE is a fundamental result in modern aerospace engineering. It predicts the final velocity v_f of an ideal rocket after it exhausts its propellant. v_i is the initial velocity; m_i and m_f are the initial and final masses; g is the gravitational acceleration impeding the rocket and t is the travel time. We project the data points onto the $\binom{5}{2} = 10$ orthogonal surfaces and the encoding images are included in the appendix. Two of the most representative projections and their attention values are plotted in Figure 9. We can verify that our attention mechanism works properly. DeSTrOI assigns a large weight for `inv` to the image on the left which involves m_f. The image on the right is less informative and therefore little attention is assigned.

Conclusion and Future Work

For decades, researchers have tried to automate the discovery of symbolic laws. In this paper, we identify the symbolic operator identification problem as a core bottleneck for classical SR. We present DeSTrOI, a computer-vision-based deep learning architecture that encodes the data into images and accurately identifies the underlying operators. As a powerful prior, DeSTrOI significantly improves the performance of SR for both synthetic data and real scientific problems. From raw observation data, we are able to discover symbolic laws that are more accurate and succinct. We also converge on the true formula much more frequently. Note that DeSTrOI can also serve as a prior for human. Our predicted operator importance may inspire physicists and mathematicians on how to proceed with searching for a symbolic law.

Recently, there are ongoing projects for SR using deep generative models (Petersen 2019). Post and Vilar (2018) have shown that a constrained vocabulary can give better performance and beam search efficiency for natural language generation. As a potential next step, we can use DeSTrOI to constrain the vocabulary for the generative models for SR. Furthermore, we will apply DeSTrOI to open prob-

Experiment	Formula	k	Operators	# of Constants	Domain
SHO	$x = A\sin(\sqrt{k}t)$	2	{`mul`, `sqrt`, `sin`}	1	(Quantum) Mechanics
IIG	$\delta E = kT\ln(V/V_0)$	2	{`mul`, `inv`,`log`}	2	Thermodynamics
FSA	$A = \pi(r+R)\sqrt{(r-R)^2+h^2}$	3	{`add`, `mul`, `sqrt`}	1	Geometry
TRE	$v_f = v_i\ln(m_i/m_f) - gt$	5	{`add`, `mul`,`inv`, `log`}	0	Aerospace Engineering

Table 4: Summary of our experiment formulas. We select a variety of equations with different k and combinations of operators.

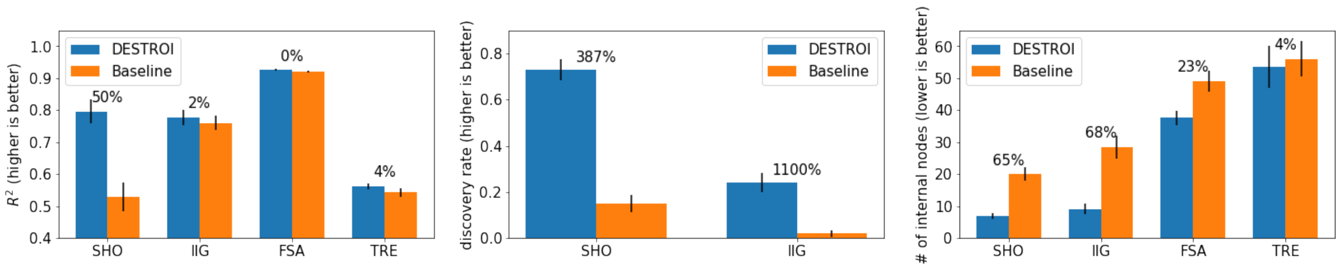
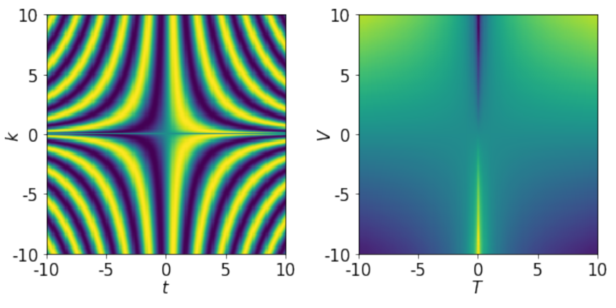


Figure 7: Performance of DeSTrOI as applied to the experiments vs. baseline. The fittest tree instance at the 20th generation is selected. The result is shown for R^2, discovery rate, and complexity (number of internal nodes). The percentages show the improvement from baseline. Error bars show ±1 standard error. (Discovery rate for the more complex FSA and TRE formulas are all 0 and thus omitted.)



(a) SHO (b) IIG

Figure 8: Encodings for SHO and IIG. We take all explanatory variables from -10 to 10. ($A = 2, k = 1, V_0 = 3$)

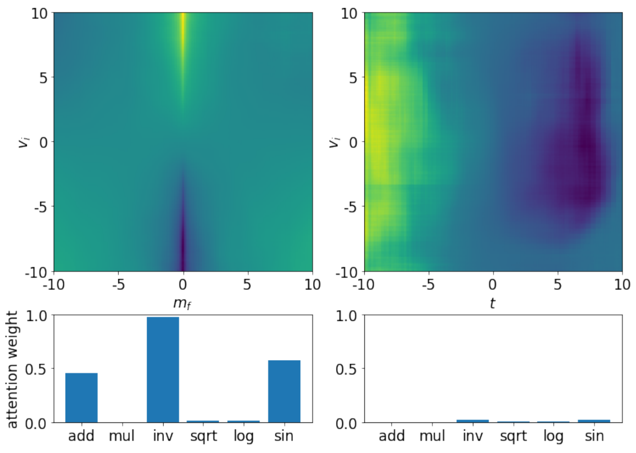


Figure 9: Two of the (projected) encoding images of the TRE function. We take all variables from -10 to 10. The bar plot shows the predicted attention weights for each image.

lems in science and engineering and potentially provide interesting results for the scientific community.

Acknowledgments

This work is partially supported by Columbia University's Alliance Program Grant.

References

Amir Haeri, M.; Ebadzadeh, M. M.; and Folino, G. 2017. Statistical genetic programming for symbolic regression. *Applied Soft Computing* 60: 447 – 469. ISSN 1568-4946. doi:https://doi.org/10.1016/j.asoc.2017.06.050. URL http://www.sciencedirect.com/science/article/pii/S1568494617303939.

Anjum, A.; Sun, F.; Wang, L.; and Orchard, J. 2019. A Novel Continuous Representation of Genetic Programmings using Recurrent Neural Networks for Symbolic Regression. *CoRR* abs/1904.03368.

Crutchfield, J. P.; and Mcnamara, B. S. 1987. Equations of motion from a data series. *Complex Systems* 452.

Crutchfield, J. P.; and Young, K. 1989. Inferring statistical complexity. *Phys. Rev. Lett.* 63: 105–108. doi:10.1103/PhysRevLett.63.105. URL https://link.aps.org/doi/10.1103/PhysRevLett.63.105.

Dietterich, T. G.; Lathrop, R. H.; and Lozano-Pérez, T. 1997. Solving the multiple instance problem with axis-parallel rectangles. *Artificial Intelligence* 89(1): 31–71. ISSN 0004-3702. doi:https://doi.org/10.1016/S0004-3702(96)00034-3.

Fan, F.; Xiong, J.; and Wang, G. 2020. On Interpretability of Artificial Neural Networks. *CoRR* abs/2001.02522.

Ilse, M.; Tomczak, J. M.; and Welling, M. 2018. Attention-based Deep Multiple Instance Learning. *CoRR* abs/1802.04712.

Koza, J. R. 1994. Genetic programming as a means for programming computers by natural selection. *Statistics and Computing* 4(2): 87–112. doi:10.1007/BF00175355. URL https://doi.org/10.1007/BF00175355.

Kusner, M. J.; Paige, B.; and Hernández-Lobato, J. M. 2017. Grammar Variational Autoencoder. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, 1945–1954. JMLR.org.

Ledig, C.; Theis, L.; Huszár, F.; Caballero, J.; Cunningham, A.; Acosta, A.; Aitken, A.; Tejani, A.; Totz, J.; Wang, Z.; and Shi, W. 2017. Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 105–114.

Lim, B.; Son, S.; Kim, H.; Nah, S.; and Lee, K. 2017. Enhanced Deep Residual Networks for Single Image Super-Resolution. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, 1132–1140. doi:10.1109/CVPRW.2017.151.

Lu, Q.; Ren, J.; and Wang, Z. 2016. Using Genetic Programming with Prior Formula Knowledge to Solve Symbolic Regression Problem. *Computational intelligence and neuroscience* 2016: 1021378–1021378. doi:10.1155/2016/1021378. URL https://pubmed.ncbi.nlm.nih.gov/26819577.

Nguyen, Q. U.; Hoai, N.; O'Neill, M.; McKay, R.; and Galván-López, E. 2011. Semantically-based crossover in genetic programming: Application to real-valued symbolic regression. *Genetic Programming and Evolvable Machines* 12: 91–119. doi:10.1007/s10710-010-9121-2.

Petersen, B. K. 2019. Deep symbolic regression: Recovering mathematical expressions from data via policy gradients. *CoRR* abs/1912.04871.

Post, M.; and Vilar, D. 2018. Fast Lexically Constrained Decoding with Dynamic Beam Allocation for Neural Machine Translation. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, 1314–1324. Association for Computational Linguistics. doi:10.18653/v1/N18-1119.

Sahoo, S.; Lampert, C.; and Martius, G. 2018. Learning Equations for Extrapolation and Control. In Dy, J.; and Krause, A., eds., *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, 4442–4450. Stockholmsmässan, Stockholm Sweden: PMLR.

Schmidt, M.; and Lipson, H. 2009. Distilling free-form natural laws from experimental data. *Science (New York, N.Y.)* 324(5923): 81–85. doi:10.1126/science.1165893. URL http://dx.doi.org/10.1126/science.1165893.

Simonyan, K.; and Zisserman, A. 2015. Very Deep Convolutional Networks for Large-Scale Image Recognition. *CoRR* abs/1409.1556.

Udrescu, S.-M.; and Tegmark, M. 2020. AI Feynman: A physics-inspired method for symbolic regression. *Science Advances* 6(16). doi:10.1126/sciadv.aay2631. URL https://advances.sciencemag.org/content/6/16/eaay2631.

Yu, J.; Fan, Y.; Yang, J.; Xu, N.; Wang, Z.; Wang, X.; and Huang, T. S. 2018. Wide Activation for Efficient and Accurate Image Super-Resolution. *CoRR* abs/1808.08718.

Zhong, J.; Lin, Y.; Lu, C.; and Huang, Z. 2018. A Deep Learning Assisted Gene Expression Programming Framework for Symbolic Regression Problems. In *ICONIP*.