Physics 361 - Machine Learning in Physics

Lecture 25 – Interpretability, Symbolic Regression, Emulators

April 24th 2024

AI
∩
Universe
$\mathcal{L} = -\frac{1}{4} F_{\mu\nu} F^{\mu\nu}$
$+ i\bar{\psi} \not{D} \psi + h.c.$
$+ \bar{\psi}_i y_{ij} \psi_j \phi + h.c.$
$+ |D_\mu \phi|^2 - V(\phi)$

Moritz Münchmeyer

Interpretability/ Explainability

Why do we want interpretable machine learning?

- Machine learning has significantly improved the way scientists model and interpret large datasets across a broad range of the physical sciences
- However, its **"black box"** nature often limits our ability to trust and understand its results.
- **Explainable AI (XAI)** refers to methods and techniques in artificial intelligence that make the behavior and decision-making process of AI systems understandable to humans.
- We would we want explainable AI?
  - **Trust the result**
  - **Find bugs or problems**
  - **Get scientific insight**

How interpretable are ML techniques

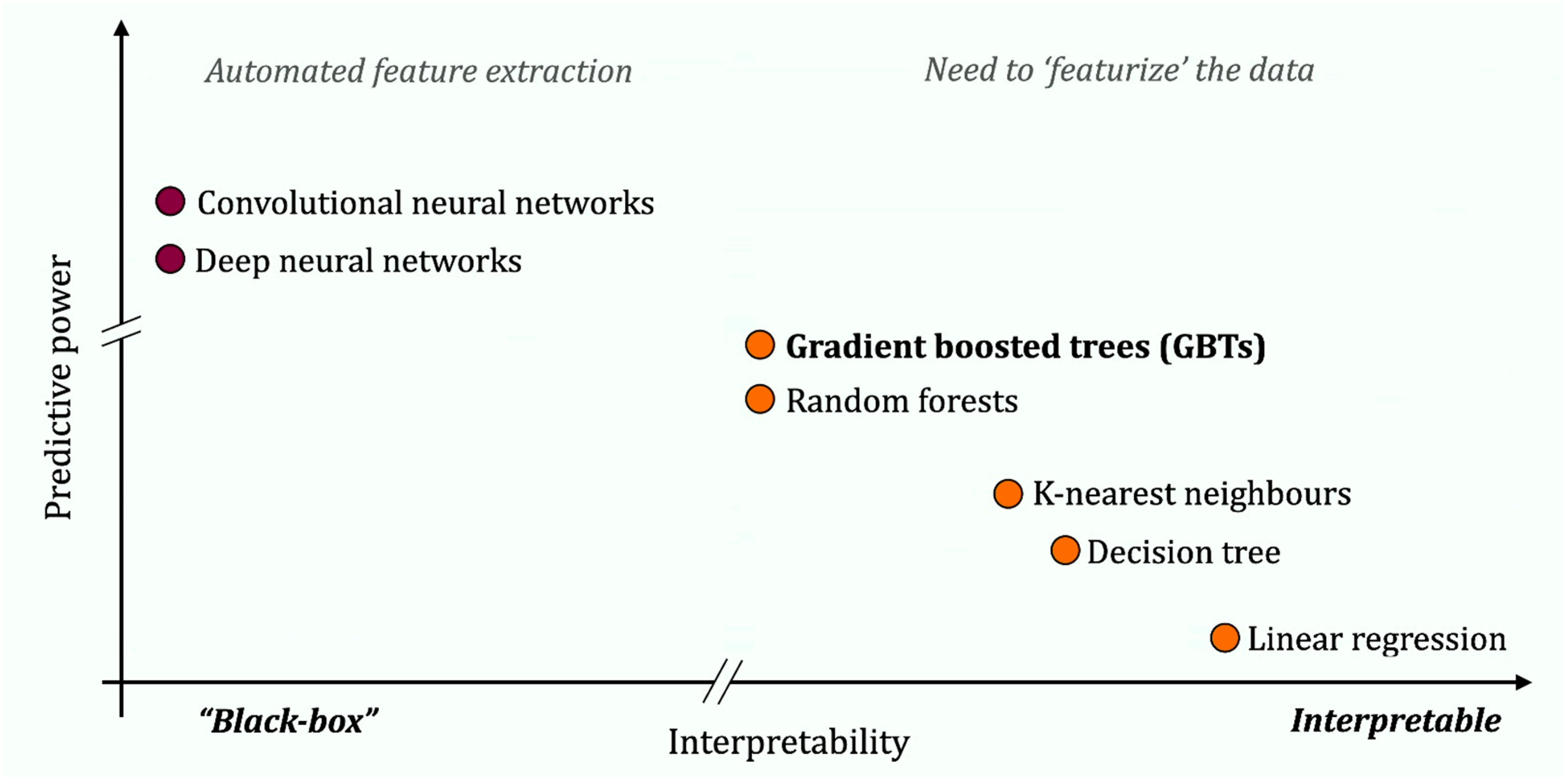


Figure credit: Luisa Lucie-Smith, Hamburg University

Feature Importance

- We already encountered one such method in the case of **Tree based methods (e.g. Gradient Boosted Trees).**
    - **Gini Importance / Mean Decrease in Impurity (MDI)**: Measures how much each feature decreases the impurity across all trees.
- For MLPs and other deep neural networks, **weight magnitudes are not directly interpretable due to non-linearity**.
- One can also **remove features during training and evaluation** to gauge their importance, but the result may be somewhat stochastic in different training runs.

Saliency Maps

- **Saliency maps** are a type of **visual explanation** technique primarily used in deep learning—especially in computer vision—to **highlight which parts of the input most influenced the model’s output**.

- A saliency map is a **heatmap over the input (e.g., an image)** where the intensity of each pixel indicates how important that pixel was for the model's prediction.

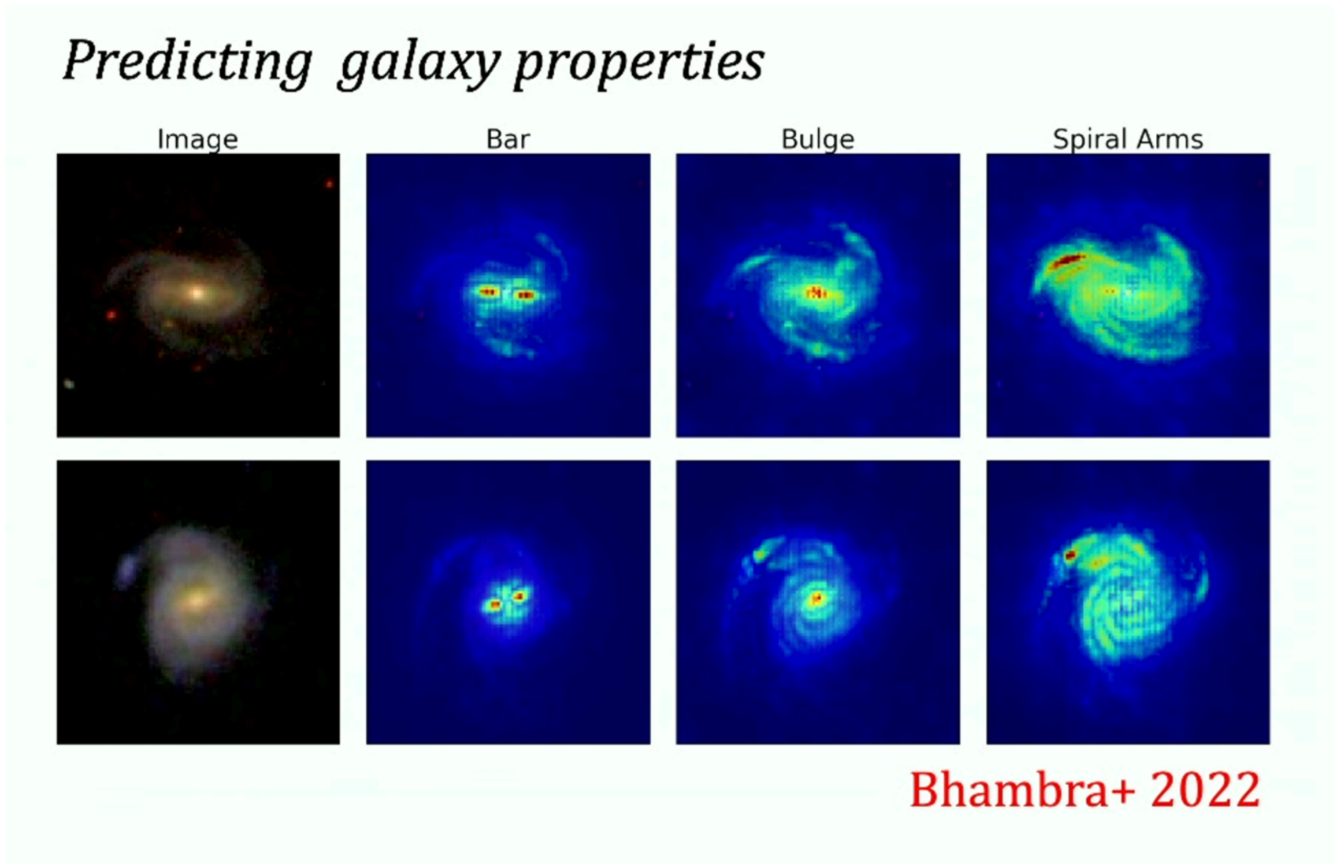
They are often computed using **gradients**:

$$\text{Saliency}(x_i) = \left| \frac{\partial f(x)}{\partial x_i} \right|$$

where $f(x)$ is the model output (e.g., the probability of a class) and x_i is the i-th input pixel.

- E.g. In an image classification task, a saliency map might show that a **cat’s face** contributed most to the model labeling the image as a **"cat"**, while the background had little effect.

Saliency Maps



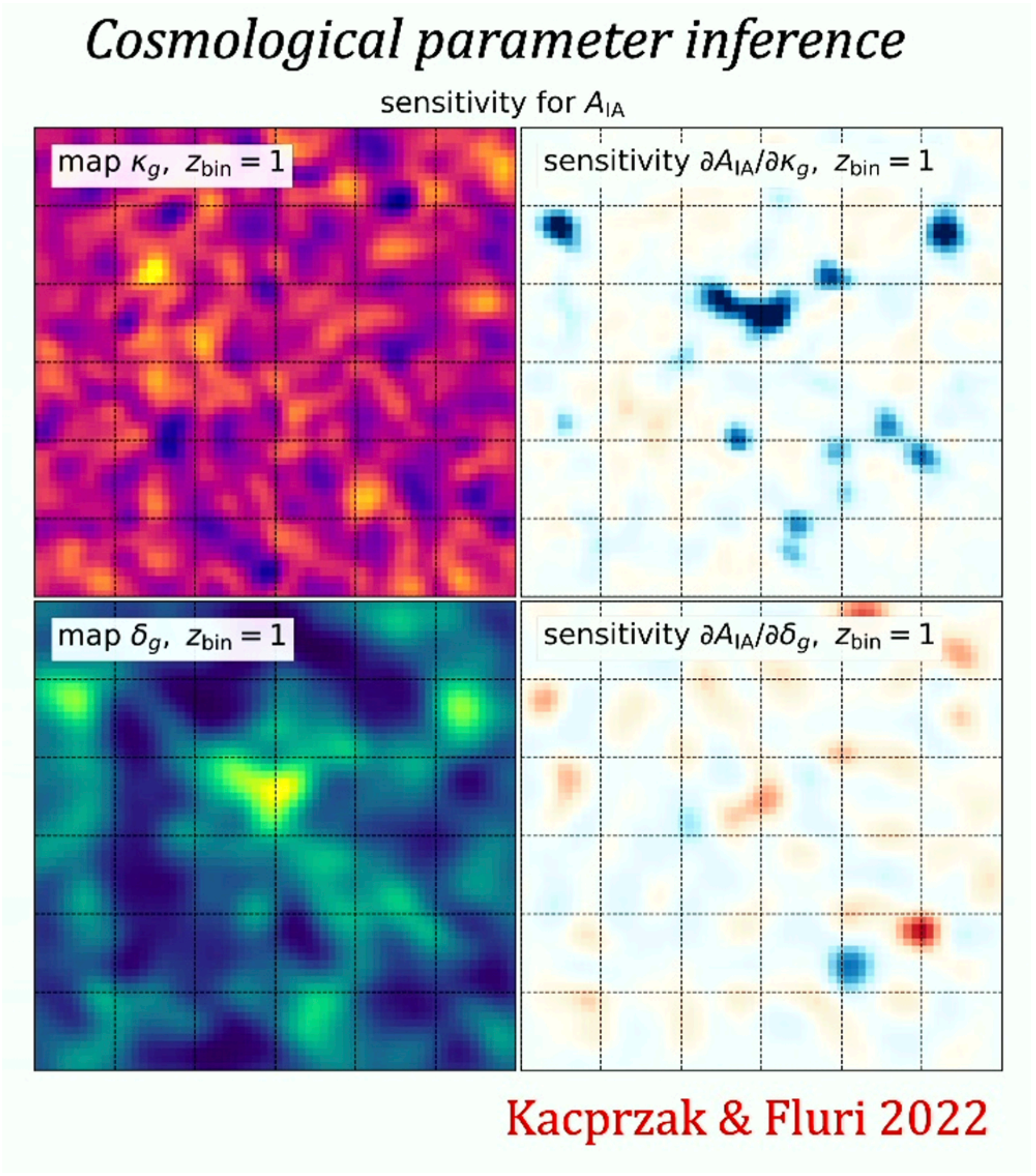
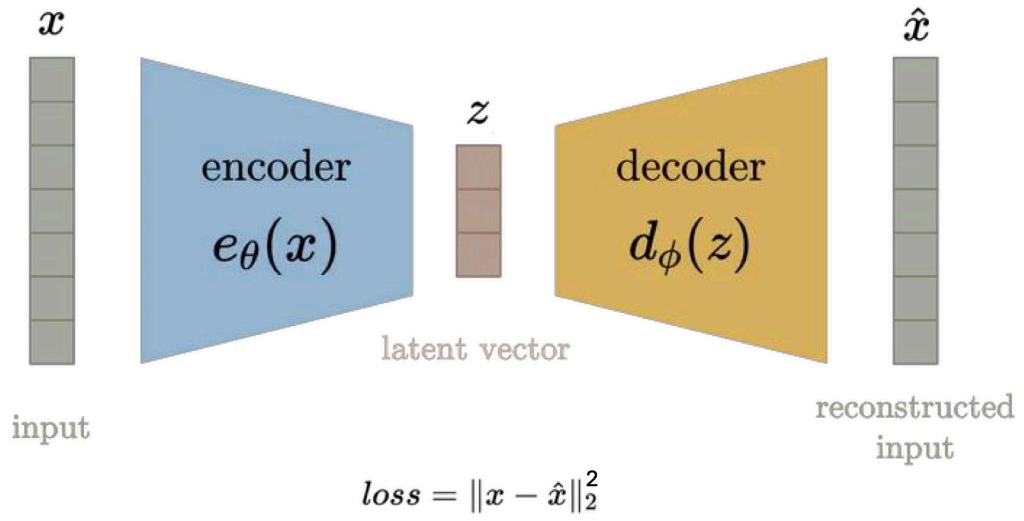


Figure credit: Luisa Lucie-Smith, Hamburg University

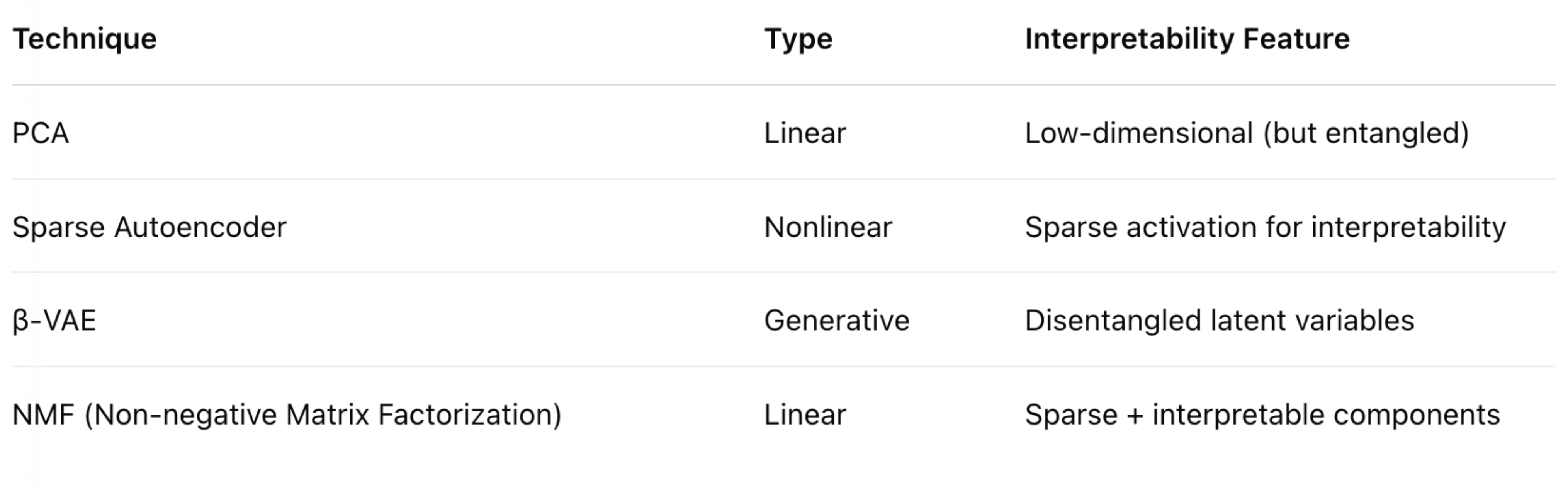
Interpretability in latent space

- If we can represent the data in a low-dimensional latent space, we may be able to interpret this space.
- For interpretability, we seek a **latent space with the following properties**:
  - **Low-dimensional**: Only a few variables capture most of the information.
  - **Sparse**: Most dimensions are zero or near-zero for any given example.
  - **Disentangled**: Each latent dimension controls an independent, semantically meaningful factor of variation in the data.



Interpretability in latent space

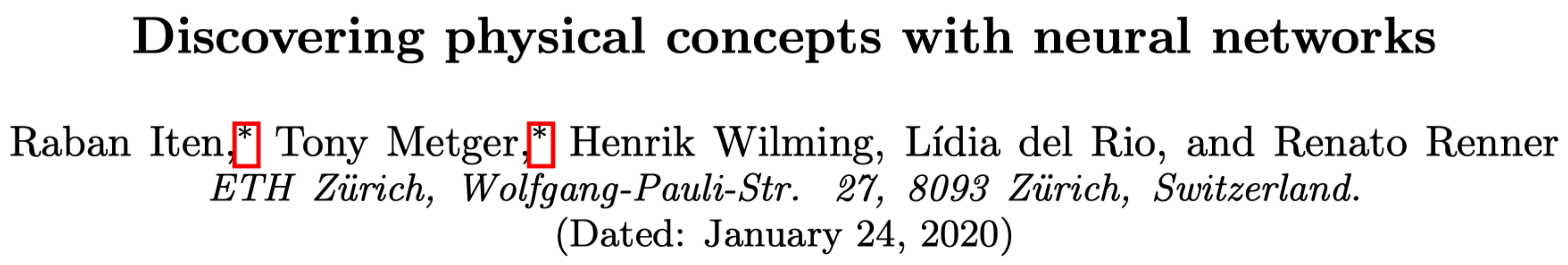
- Here are some **techniques that can be used to construct such a latent space**:



Technique	Type	Interpretability Feature
PCA	Linear	Low-dimensional (but entangled)
Sparse Autoencoder	Nonlinear	Sparse activation for interpretability
β-VAE	Generative	Disentangled latent variables
NMF (Non-negative Matrix Factorization)	Linear	Sparse + interpretable components

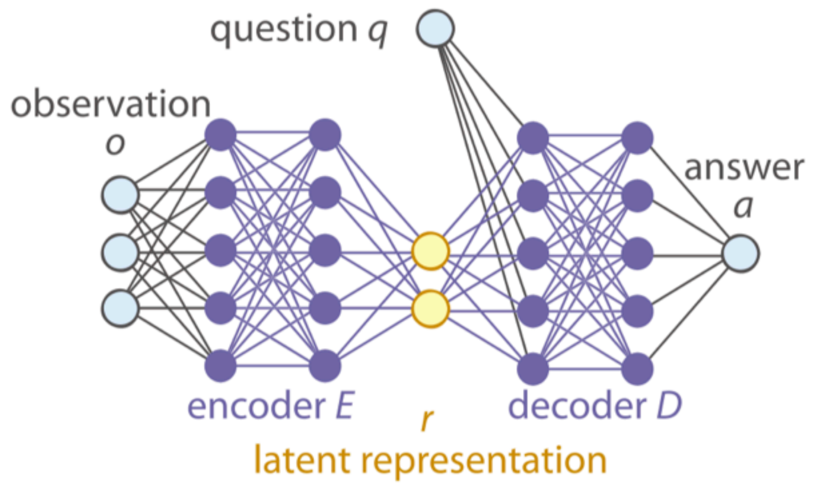
- **Once the latent space is found, one can for example**
  - **Visualize and interpret the “degrees of freedom”** of the data.
  - **Perform inference** on these latent variables.
  - Use latent variables for new tasks.

Example: SciNet model



Discovering physical concepts with neural networks

Raban Iten,[*] Tony Metger,[*] Henrik Wilming, Lídia del Rio, and Renato Renner
ETH Zürich, Wolfgang-Pauli-Str. 27, 8093 Zürich, Switzerland.
(Dated: January 24, 2020)



- If you train a network to **answer physical questions about a system, the bottleneck layer will contain the relevant physical variables** — like energy, angular momentum, or charge — without being told what those variables are.

- The b**ottleneck forces the encoder to represent only the minimal sufficient information necessary** to answer questions.

- The latent variables **often disentangle into interpretable physical quantities, like mass or velocity,** without supervision.

Example: SciNet Model

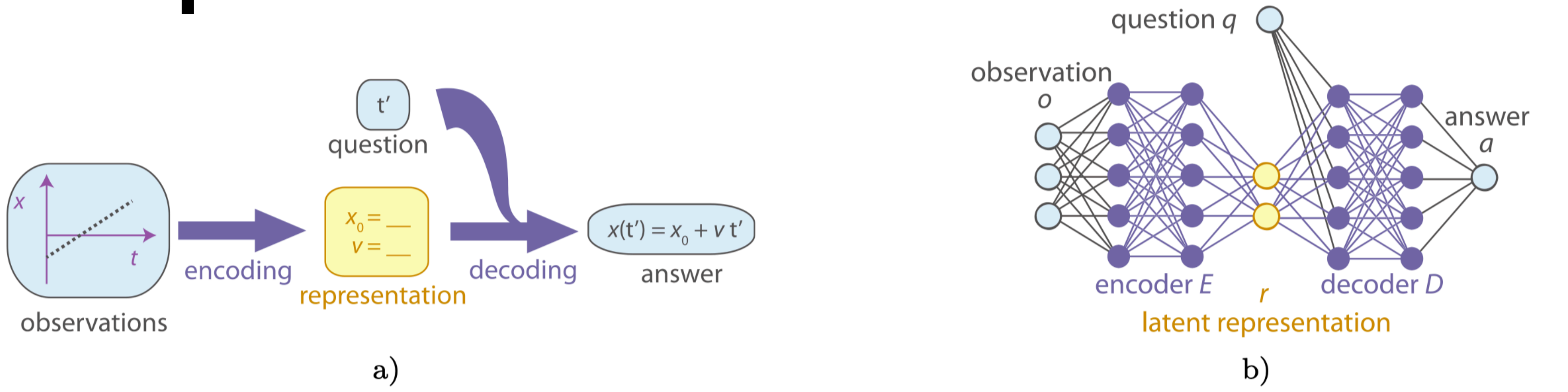


Figure 1. **Learning physical representations. (a) Human learning.** A physicist compresses experimental observations into a simple representation (*encoding*). When later asked any question about the physical setting, the physicist should be able to produce a correct answer using only the representation and not the original data. We call the process of producing the answer from the representation *decoding*. For example, the observations may be the first few seconds of the trajectory of a particle moving with constant speed; the representation could be the parameters "speed v" and "initial position x_0" and the question could be "where will the particle be at a later time t'?" **(b) Neural network structure for *SciNet*.** Observations are encoded as real parameters fed to an encoder (a *feed-forward neural network*, see Appendix D), which compresses the data into a representation (*latent representation*). The question is also encoded in a number of real parameters, which, together with the representation, are fed to the decoder network to produce an answer. (The number of neurons depicted is not representative.)

Training data: (x, q, a)

Where:

- x = **data** (observations of the system)
- q = **question** (e.g., "what is the position at time t?")
- a = **answer** (the ground truth to that question)

Example: Harmonic Oscillator

x: a few positions $x(t)$ at early times

q: a future time t_{future}

a: the position $x(t_{\text{future}})$

Example from cosmology

Discovering the building blocks of dark matter halo density profiles with neural networks

Luisa Lucie-Smith,[1,*] Hiranya V. Peiris,[2,3] Andrew Pontzen,[2] Brian Nord,[4,5,6] Jeyan Thiyagalingam,[7] and Davide Piras[2]

IVE: interpretable variational encoder

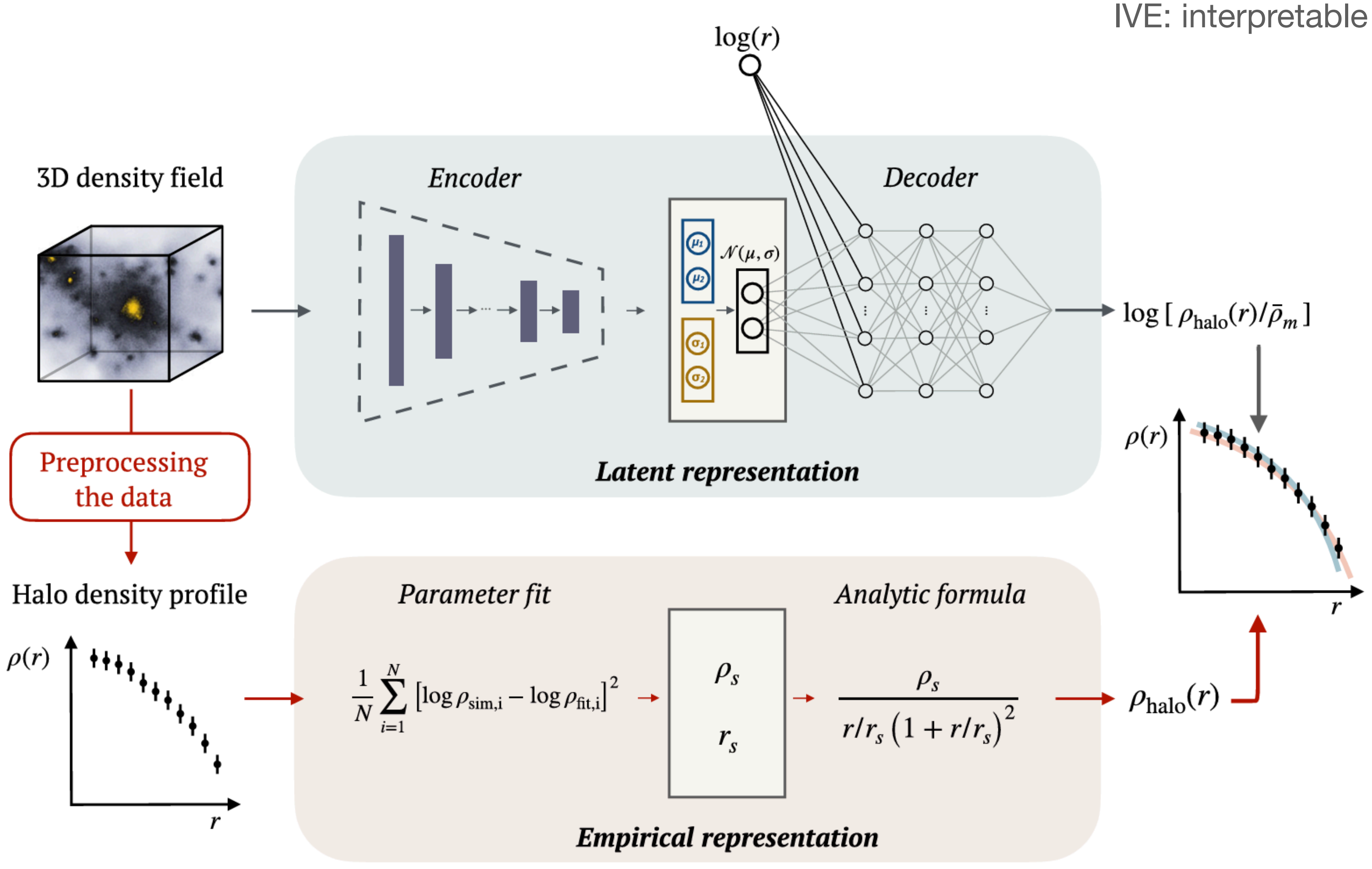


FIG. 1. The IVE consists of an encoder compressing the 3D density field containing each halo into a low-dimensional latent representation, followed by a decoder mapping the latent representation and a given value of r to the spherically averaged density $\rho(r)$. In this illustration, the latent space is two dimensional; however, the dimensionality of the latent space can be increased to any arbitrary value. The latent representation only retains the information required by the model to predict the halo density profiles, allowing us to interpret the representation as independent factors of variability in the density profiles. The decoder plays a role similar to an analytic fitting formula, which takes as input a set of halo-specific parameters and returns $\rho(r)$ for any given r. The encoder is equivalent to the parameter fitting procedure, in that they return those halo-specific parameters used by the analytic formula (decoder). However, the inputs to the encoder and the fitting procedure are fundamentally different: the former extracts information directly from the 3D density field containing the halo, whereas the latter uses processed information of that field, i.e., the spherically averaged density profiles themselves. The latter data processing is a step motivated by human intuition and physical frameworks such as the secondary infall model [34].

Analyze mutual information in latent space

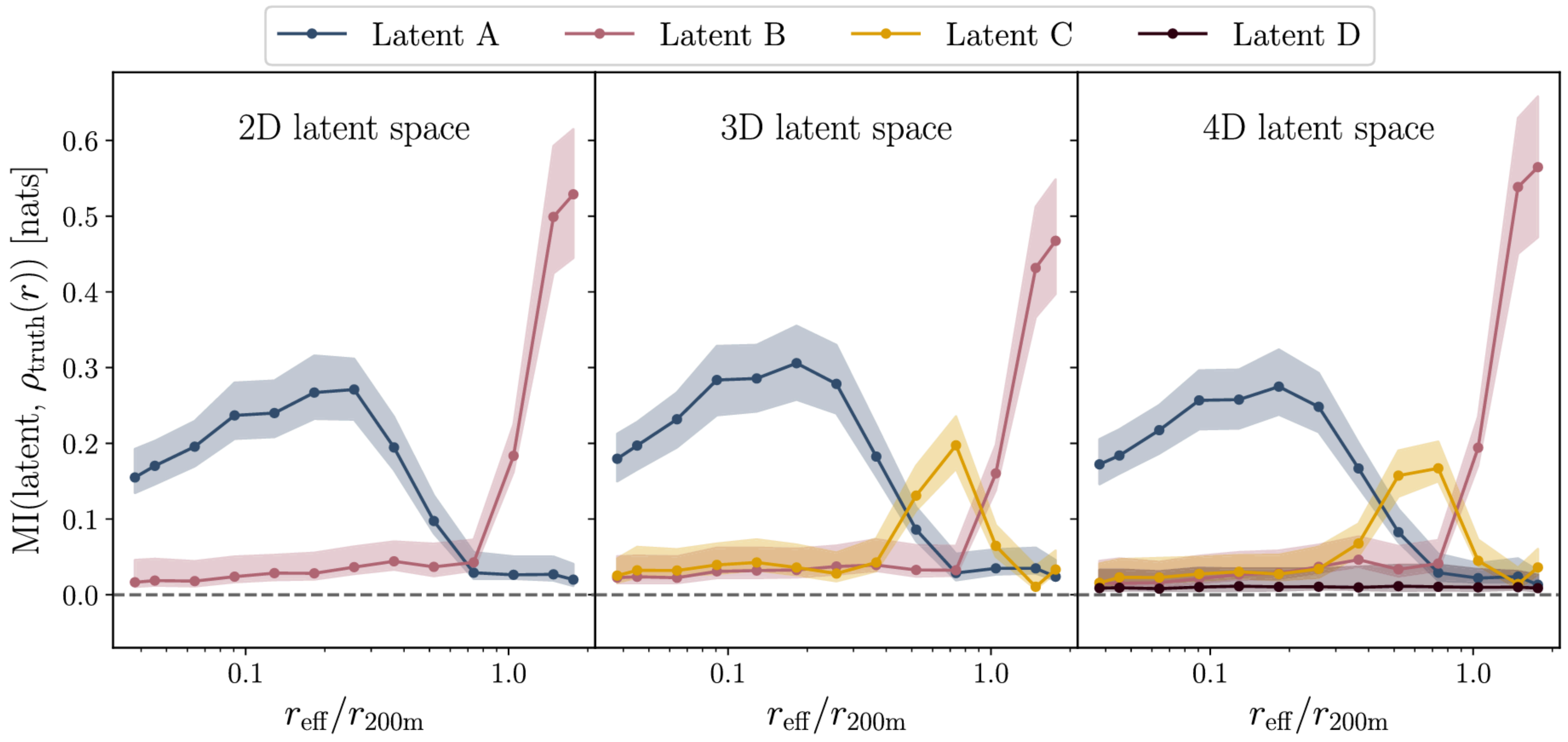


FIG. 4. Mutual information between each latent variable and the ground-truth $\log[\rho_i(r)/\bar{\rho}_{\rm m}]$ in every ith radial bin. The three panels show the results for the IVE$_{\rm infall}$ model with latent dimensionality 2, 3, and 4. The solid lines show the mutual information when adopting a bandwidth of 0.2; the bands show the scatter in the mutual information estimate when adopting bandwidths of 0.1 (upper band limit) and 0.3 (lower band limit). These values of bandwidths cover sufficient range to undersmooth and oversmooth the distributions, thus demonstrating that our results are insensitive to the specific bandwidth choice.

Mutual information

To estimate the **mutual information (MI)** between a feature X and a target Y, you're quantifying **how much knowing X reduces uncertainty about Y** — and vice versa.

- How to estimate the mutual information depends on whether X and/or Y are continuous or discrete.
- Discrete case:

$$I(X;Y) = \sum_{x \in \mathcal{X}} \sum_{y \in \mathcal{Y}} p(x,y) \log\left(\frac{p(x,y)}{p(x)p(y)}\right)$$

  - $p(x,y)$: Joint probability of $X = x$ and $Y = y$
  - $p(x)p(y)$: What the joint probability would be if X and Y were **independent**
  - $\log\left(\frac{p(x,y)}{p(x)p(y)}\right)$: Measures how "surprised" we are that x and y occur together more or less often than if they were independent.

  So, mutual information measures the **average surprise** (in bits if using log base 2) due to the dependence between X and Y.

- Scikit-Learn has various functions to help calculating this from samples.

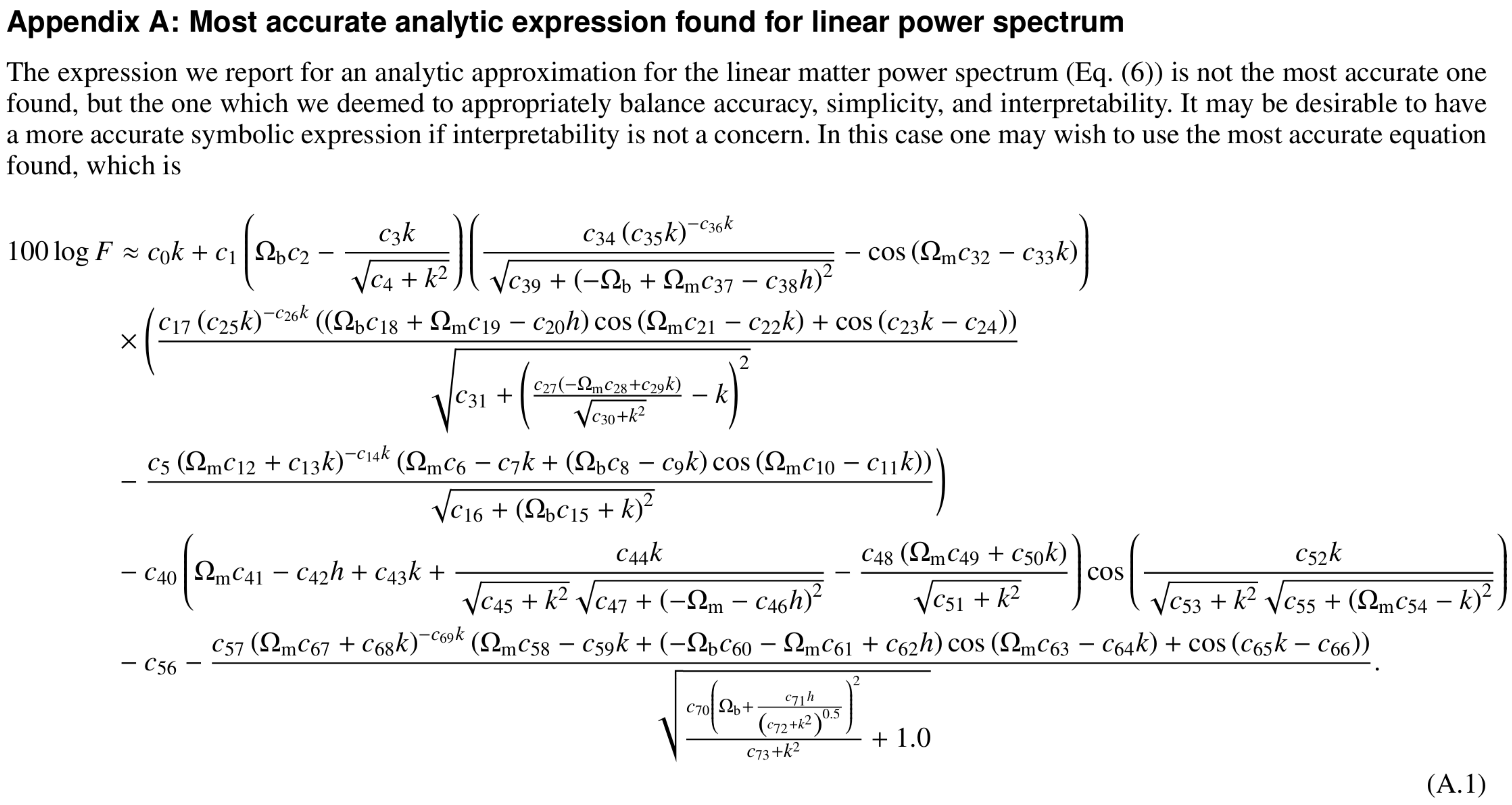
Symbolic Regression of Functions

Discovering symbolic laws from data

- **The goal of physics is to find the mathematical laws that describe nature.**
- **We could hope that neural networks "look at the data" and automatically find the underlying laws.**
    - In simple setups this is now possible. E.g. discover newtons laws from looking at how particles mode.
- **Discovering Symbolic Laws = Symbolic Regression**
- **Symbolic Regression is an old topic that pre-dates machine learning, but Machine Learning may enhance it.**
- Analytic expressions **CAN be more interpretable than arbitrary neural networks, if they are sufficiently simple** and/or relatable to the physics of the domain.

Are symbolic expressions always “interpretable”?

Example: https://arxiv.org/abs/2311.15865 A precise symbolic emulator of the linear matter power spectrum



Appendix A: Most accurate analytic expression found for linear power spectrum

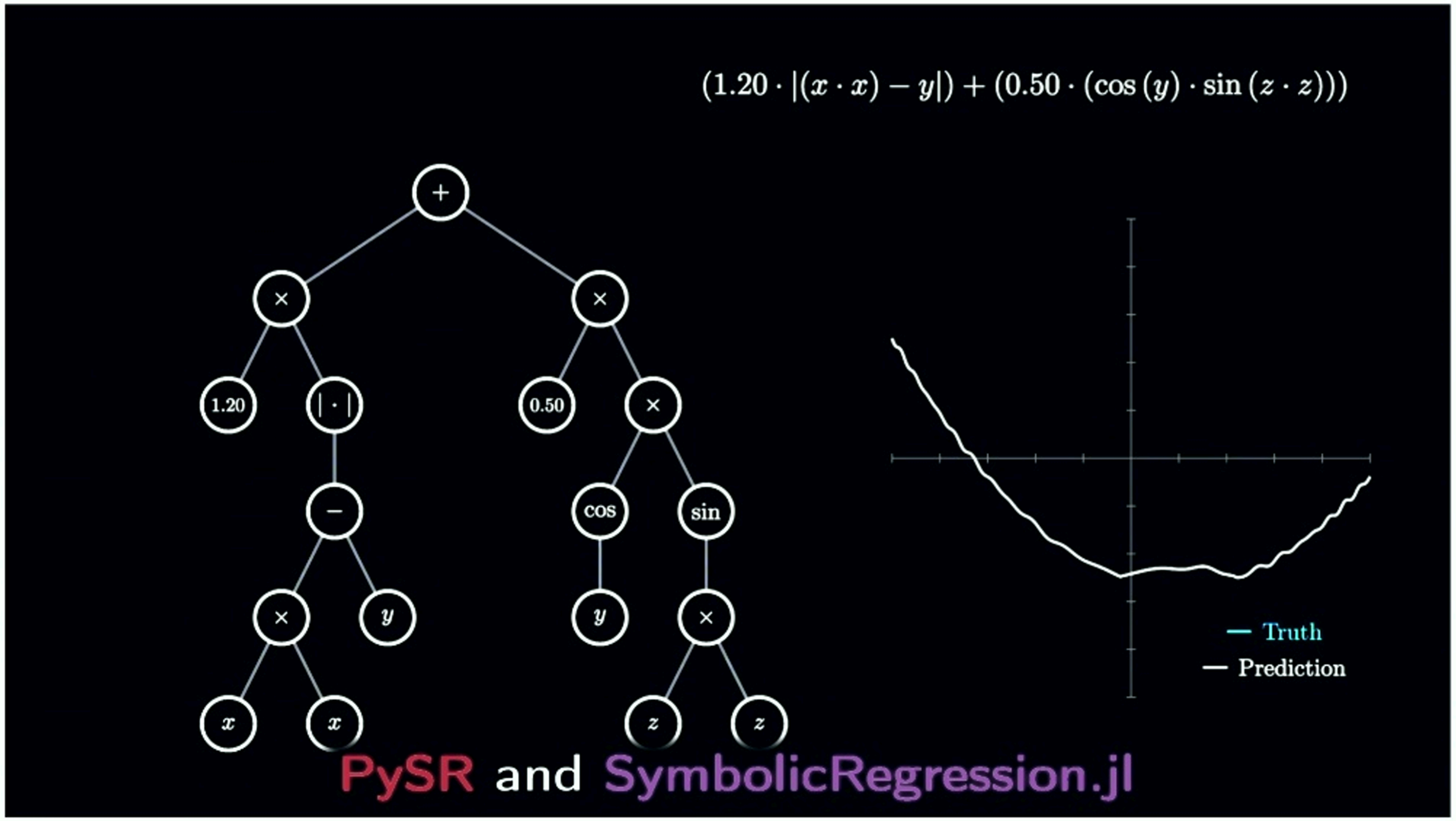
The expression we report for an analytic approximation for the linear matter power spectrum (Eq. (6)) is not the most accurate one found, but the one which we deemed to appropriately balance accuracy, simplicity, and interpretability. It may be desirable to have a more accurate symbolic expression if interpretability is not a concern. In this case one may wish to use the most accurate equation found, which is

$$\begin{aligned}
100\log F \approx\; & c_0 k + c_1\left(\Omega_{\mathrm{b}} c_2 - \frac{c_3 k}{\sqrt{c_4+k^2}}\right)\left(\frac{c_{34}\left(c_{35}k\right)^{-c_{36}k}}{\sqrt{c_{39}+\left(-\Omega_{\mathrm{b}}+\Omega_{\mathrm{m}} c_{37}-c_{38}h\right)^2}} - \cos\left(\Omega_{\mathrm{m}} c_{32}-c_{33}k\right)\right) \\
& \times\left(\frac{c_{17}\left(c_{25}k\right)^{-c_{26}k}\left(\left(\Omega_{\mathrm{b}} c_{18}+\Omega_{\mathrm{m}} c_{19}-c_{20}h\right)\cos\left(\Omega_{\mathrm{m}} c_{21}-c_{22}k\right)+\cos\left(c_{23}k-c_{24}\right)\right)}{\sqrt{c_{31}+\left(\frac{c_{27}\left(-\Omega_{\mathrm{m}} c_{28}+c_{29}k\right)}{\sqrt{c_{30}+k^2}}-k\right)^2}}\right. \\
& \left. - \frac{c_5\left(\Omega_{\mathrm{m}} c_{12}+c_{13}k\right)^{-c_{14}k}\left(\Omega_{\mathrm{m}} c_6 - c_7 k + \left(\Omega_{\mathrm{b}} c_8 - c_9 k\right)\cos\left(\Omega_{\mathrm{m}} c_{10}-c_{11}k\right)\right)}{\sqrt{c_{16}+\left(\Omega_{\mathrm{b}} c_{15}+k\right)^2}}\right) \\
& - c_{40}\left(\Omega_{\mathrm{m}} c_{41} - c_{42}h + c_{43}k + \frac{c_{44}k}{\sqrt{c_{45}+k^2}\sqrt{c_{47}+\left(-\Omega_{\mathrm{m}}-c_{46}h\right)^2}} - \frac{c_{48}\left(\Omega_{\mathrm{m}} c_{49}+c_{50}k\right)}{\sqrt{c_{51}+k^2}}\right)\cos\left(\frac{c_{52}k}{\sqrt{c_{53}+k^2}\sqrt{c_{55}+\left(\Omega_{\mathrm{m}} c_{54}-k\right)^2}}\right) \\
& - c_{56} - \frac{c_{57}\left(\Omega_{\mathrm{m}} c_{67}+c_{68}k\right)^{-c_{69}k}\left(\Omega_{\mathrm{m}} c_{58}-c_{59}k+\left(-\Omega_{\mathrm{b}} c_{60}-\Omega_{\mathrm{m}} c_{61}+c_{62}h\right)\cos\left(\Omega_{\mathrm{m}} c_{63}-c_{64}k\right)+\cos\left(c_{65}k-c_{66}\right)\right)}{\sqrt{\frac{c_{70}\left(\Omega_{\mathrm{b}}+\frac{c_{71}h}{\left(c_{72}+k^2\right)^{0.5}}\right)^2}{c_{73}+k^2}+1.0}}.
\end{aligned} \tag{A.1}$$

SR with Genetic algorithms, PySR code

- Symbolic Regression is usually done with Genetic algorithms, building a population of possible expressions that evolve (i.e. are modified).
- See the PySR paper https://arxiv.org/abs/2305.01582 Interpretable Machine Learning for Science with PySR and SymbolicRegression.jl

Credit: Miles Cranmer


$(1.20 \cdot |(x \cdot x) - y|) + (0.50 \cdot (\cos(y) \cdot \sin(z \cdot z)))$
+
×
×
1.20
|·|
0.50
×
−
cos
sin
×
y
y
×
x
x
z
z
Truth
Prediction
PySR and SymbolicRegression.jl

PySR algorithm

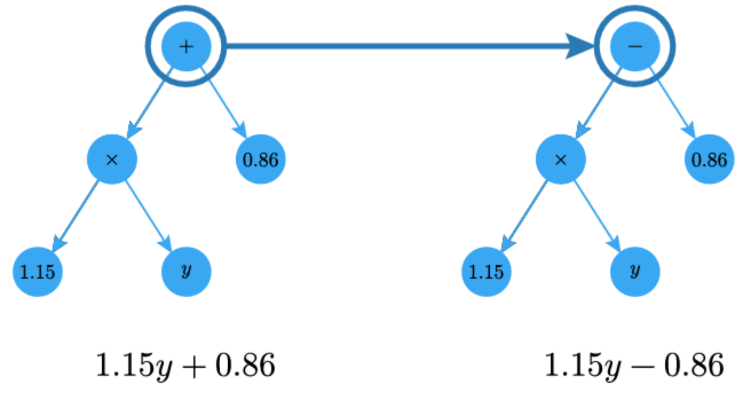


Figure 1: A mutation operation applied to an expression tree.

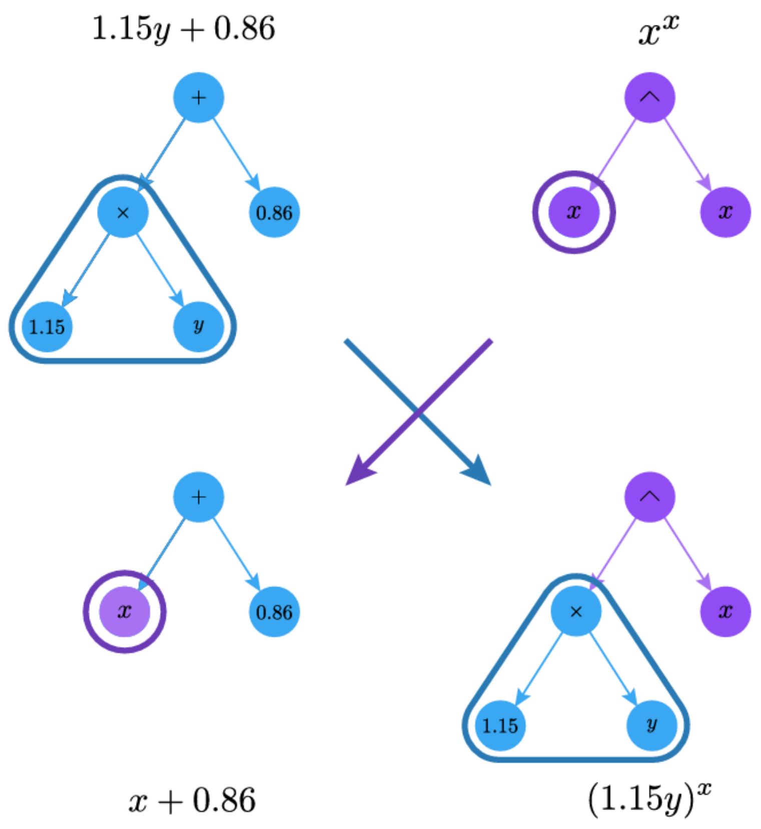
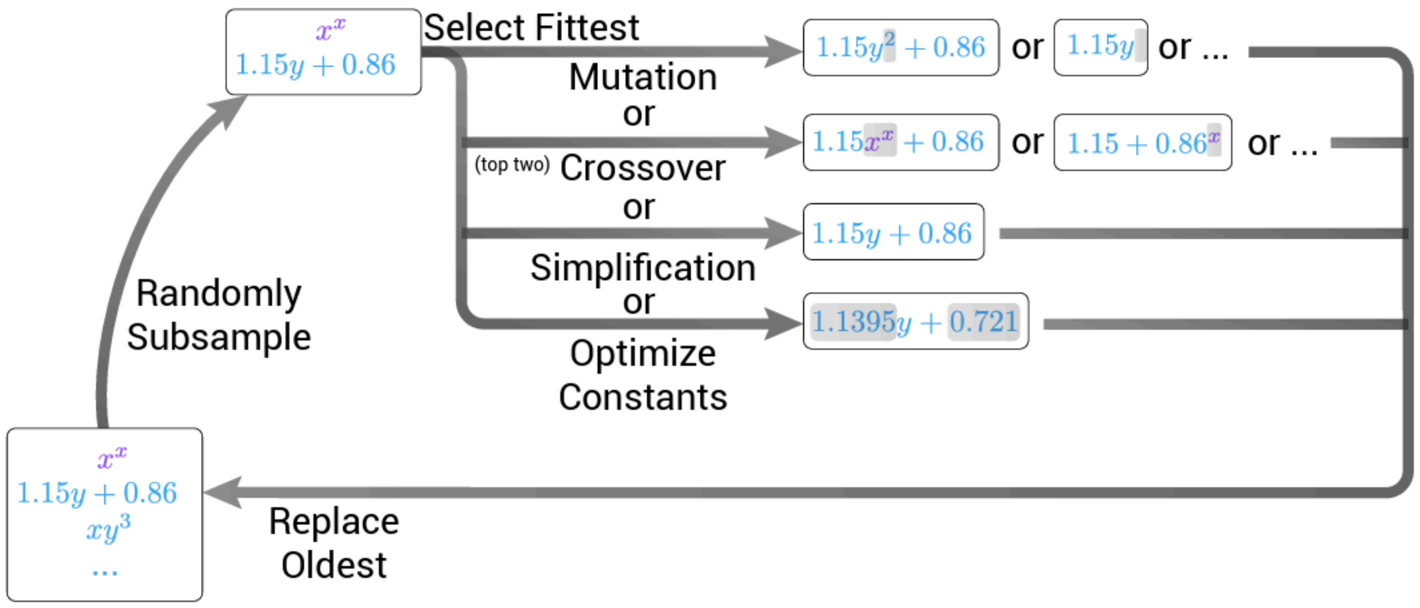
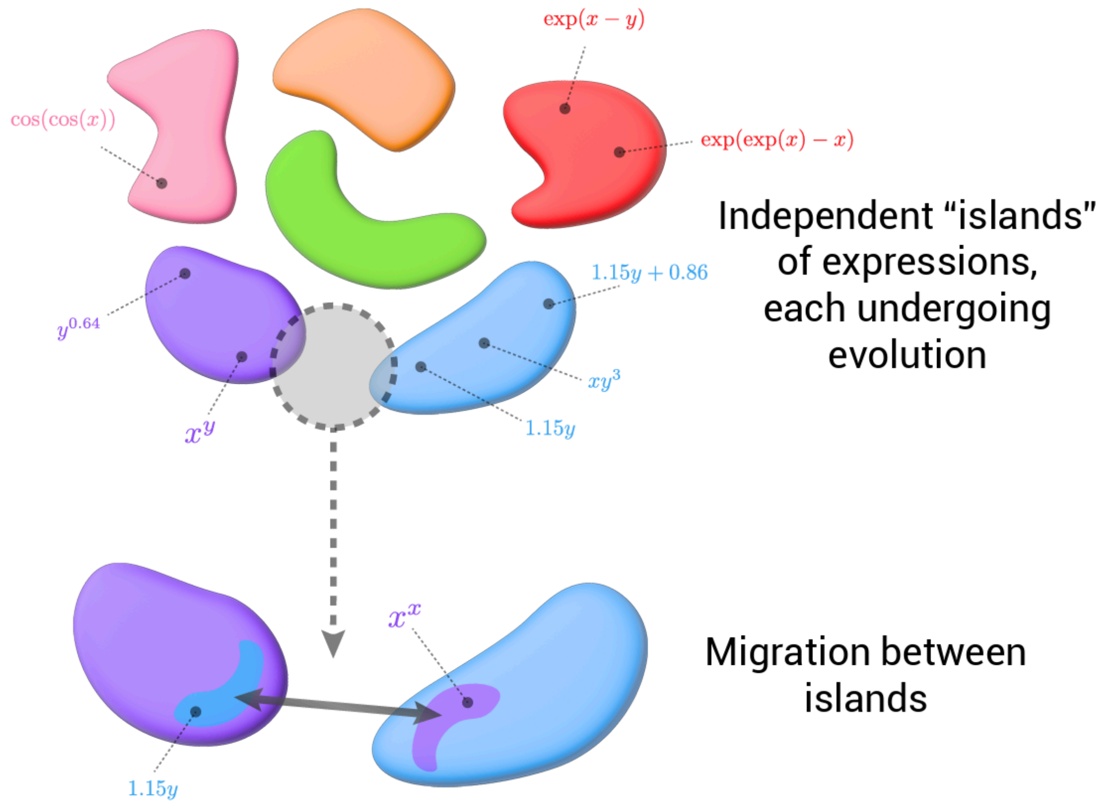


Figure 2: A crossover operation between two expression trees.



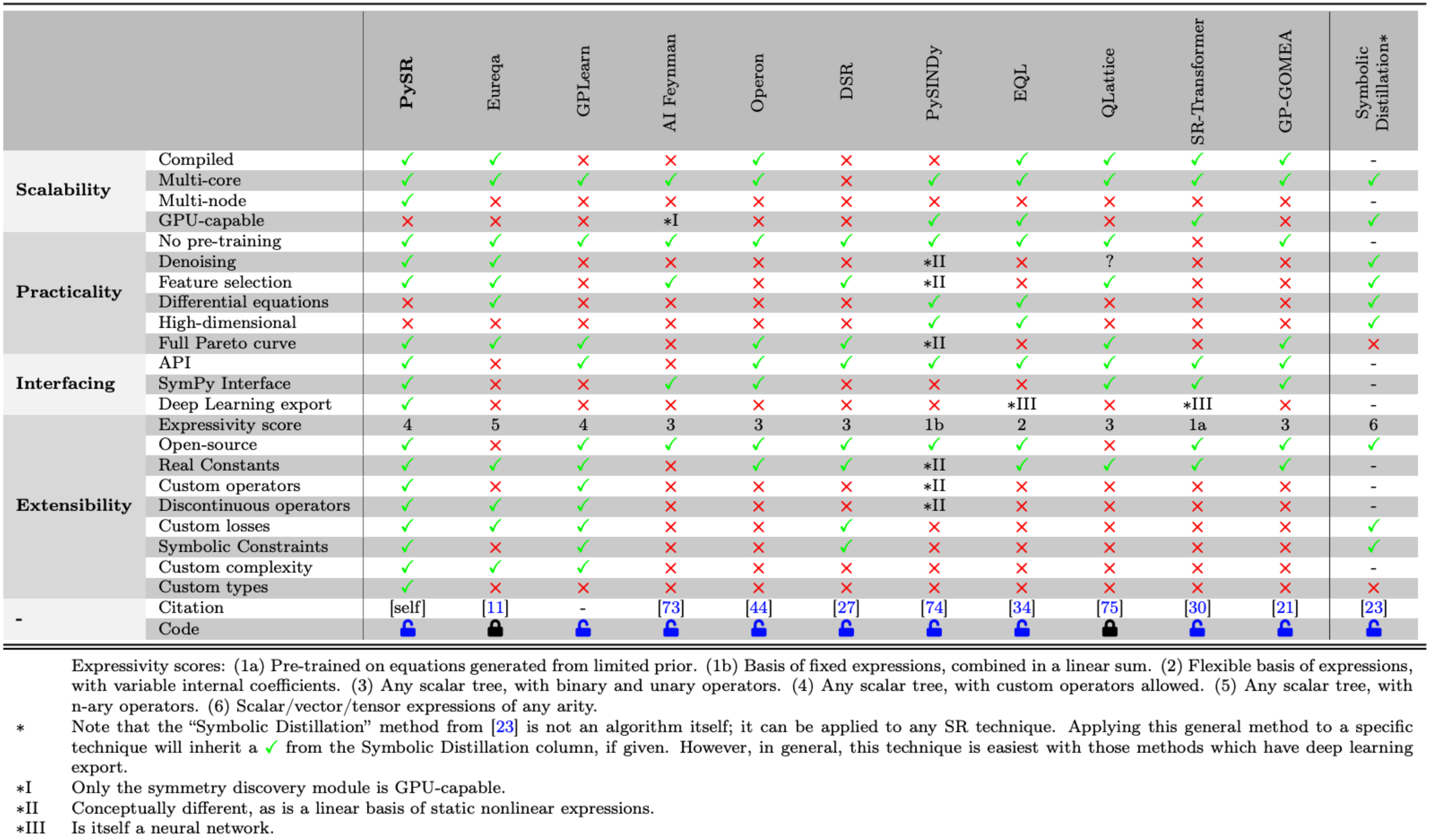
The inner loop of PySR. A population of expressions is randomly subsampled. Among this subsample, a tournament is performed, and the winner is selected for breeding: either by mutation, crossover, simplification, or explicit optimization. Examples of mutation and crossover operations are visualized in figs. 1 and 2.



The outer loop of PySR. Several populations evolve independently according to the algorithm described in fig. 3. At the end of a specified number of rounds of evolution, migration between islands is performed.

Symbolic Regression methods

Comparison from https://arxiv.org/abs/2305.01582



		PySR	Eureqa	GPLearn	AI Feynman	Operon	DSR	PySINDy	EQL	QLattice	SR-Transformer	GP-GOMEA	Symbolic Distillation*
Scalability	Compiled	✓	✓	✗	✗	✓	✗	✗	✓	✓	✓	✓	-
	Multi-core	✓	✓	✓	✓	✓	✗	✓	✓	✓	✓	✓	✓
	Multi-node	✓	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	-
	GPU-capable	✗	✗	✗	∗I	✗	✗	✓	✓	✗	✓	✗	✓
Practicality	No pre-training	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	-
	Denoising	✓	✓	✗	✗	✗	✗	∗II	✗	?	✗	✗	✓
	Feature selection	✓	✓	✗	✓	✗	✓	∗II	✗	✓	✗	✗	✓
	Differential equations	✗	✓	✗	✗	✗	✗	✓	✓	✗	✗	✗	✓
	High-dimensional	✗	✗	✗	✗	✗	✗	✓	✓	✗	✗	✗	✓
	Full Pareto curve	✓	✓	✓	✗	✓	✓	∗II	✗	✓	✗	✓	✗
Interfacing	API	✓	✗	✓	✗	✓	✓	✓	✓	✓	✓	✓	-
	SymPy Interface	✓	✗	✗	✓	✓	✗	✗	✗	✓	✓	✓	-
	Deep Learning export	✓	✗	✗	✗	✗	✗	✗	∗III	✗	∗III	✗	-
Extensibility	Expressivity score	4	5	4	3	3	3	1b	2	3	1a	3	6
	Open-source	✓	✗	✓	✓	✓	✓	✓	✓	✗	✓	✓	✓
	Real Constants	✓	✓	✓	✗	✓	✓	∗II	✓	✓	✓	✓	-
	Custom operators	✓	✗	✓	✗	✗	✗	∗II	✗	✗	✗	✗	-
	Discontinuous operators	✓	✓	✓	✗	✗	✗	∗II	✗	✗	✗	✗	-
	Custom losses	✓	✓	✓	✗	✗	✓	✗	✗	✗	✗	✗	✓
	Symbolic Constraints	✓	✗	✓	✗	✗	✓	✗	✗	✗	✗	✗	✓
	Custom complexity	✓	✓	✓	✗	✗	✗	✗	✗	✗	✗	✗	-
	Custom types	✓	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
-	Citation	[self]	[11]	-	[73]	[44]	[27]	[74]	[34]	[75]	[30]	[21]	[23]
	Code	🔓	🔒	🔓	🔓	🔓	🔓	🔓	🔓	🔒	🔓	🔓	🔓

Expressivity scores: (1a) Pre-trained on equations generated from limited prior. (1b) Basis of fixed expressions, combined in a linear sum. (2) Flexible basis of expressions, with variable internal coefficients. (3) Any scalar tree, with binary and unary operators. (4) Any scalar tree, with custom operators allowed. (5) Any scalar tree, with n-ary operators. (6) Scalar/vector/tensor expressions of any arity.

∗ Note that the "Symbolic Distillation" method from [23] is not an algorithm itself; it can be applied to any SR technique. Applying this general method to a specific technique will inherit a ✓ from the Symbolic Distillation column, if given. However, in general, this technique is easiest with those methods which have deep learning export.

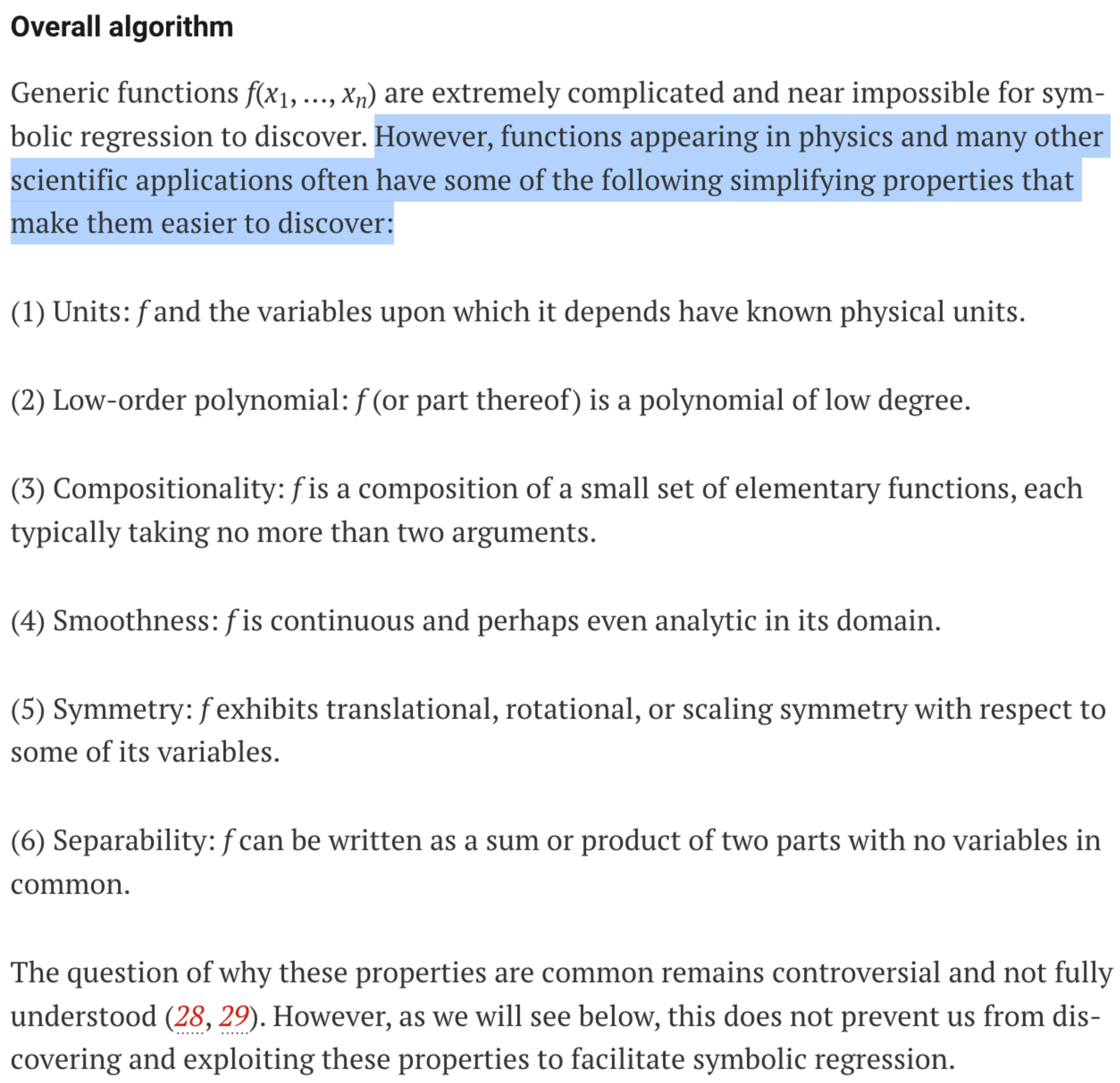
∗I Only the symmetry discovery module is GPU-capable.

∗II Conceptually different, as is a linear basis of static nonlinear expressions.

∗III Is itself a neural network.

Example: AI Feynman

- https://www.science.org/doi/10.1126/sciadv.aay2631 AI Feynman: A physics-inspired method for symbolic regression
  - We develop a recursive multidimensional symbolic regression algorithm that combines neural network fitting with a suite of physics-inspired techniques. **We apply it to 100 equations from the Feynman Lectures on Physics, and it discovers all of them.**



Overall algorithm

Generic functions $f(x_1, \ldots, x_n)$ are extremely complicated and near impossible for symbolic regression to discover. However, functions appearing in physics and many other scientific applications often have some of the following simplifying properties that make them easier to discover:

(1) Units: f and the variables upon which it depends have known physical units.

(2) Low-order polynomial: f (or part thereof) is a polynomial of low degree.

(3) Compositionality: f is a composition of a small set of elementary functions, each typically taking no more than two arguments.

(4) Smoothness: f is continuous and perhaps even analytic in its domain.

(5) Symmetry: f exhibits translational, rotational, or scaling symmetry with respect to some of its variables.

(6) Separability: f can be written as a sum or product of two parts with no variables in common.

The question of why these properties are common remains controversial and not fully understood (28, 29). However, as we will see below, this does not prevent us from discovering and exploiting these properties to facilitate symbolic regression.

Combine low-dimensional latent representations with SR

- https://arxiv.org/abs/2006.11287 **Discovering Symbolic Models from Deep Learning with Inductive Biases**
    - We develop a general approach to distill symbolic representations of a learned deep model by introducing strong inductive biases. We focus on Graph Neural Networks (GNNs).
    - The technique works as follows: **we first encourage sparse latent representations when we train a GNN in a supervised setting, then we apply symbolic regression to components of the learned model to extract explicit physical relations.** We find the correct known equations, including force laws and Hamiltonians, can be extracted from the neural network

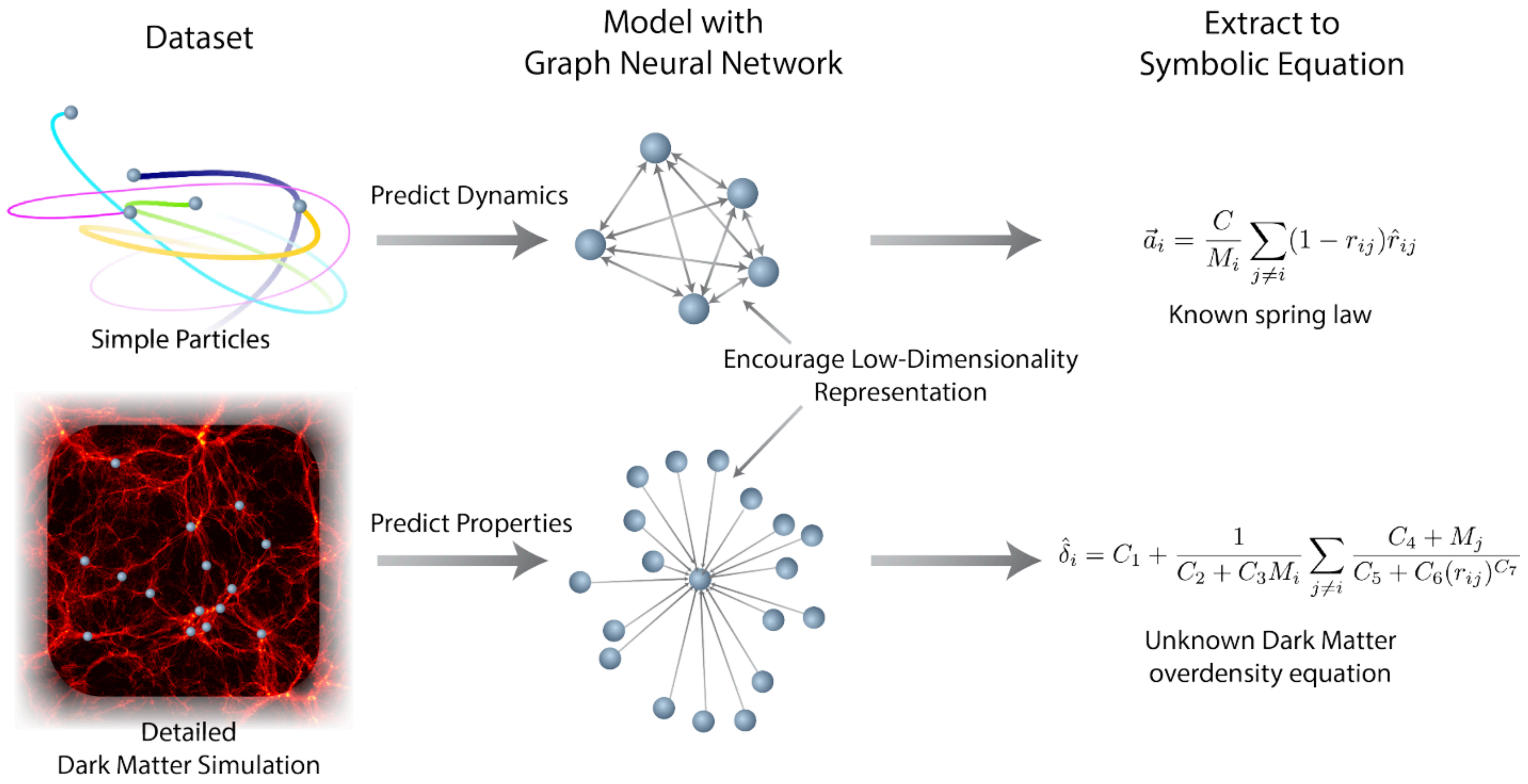

Dataset
Model with Graph Neural Network
Extract to Symbolic Equation
Simple Particles
Predict Dynamics
$\vec{a}_i = \frac{C}{M_i} \sum_{j \neq i} (1 - r_{ij}) \hat{r}_{ij}$
Known spring law
Encourage Low-Dimensionality Representation
Detailed Dark Matter Simulation
Predict Properties
$\hat{\delta}_i = C_1 + \frac{1}{C_2 + C_3 M_i} \sum_{j \neq i} \frac{C_4 + M_j}{C_5 + C_6 (r_{ij})^{C_7}}$
Unknown Dark Matter overdensity equation

Figure 1: A cartoon depicting how we extract physical equations from a dataset.

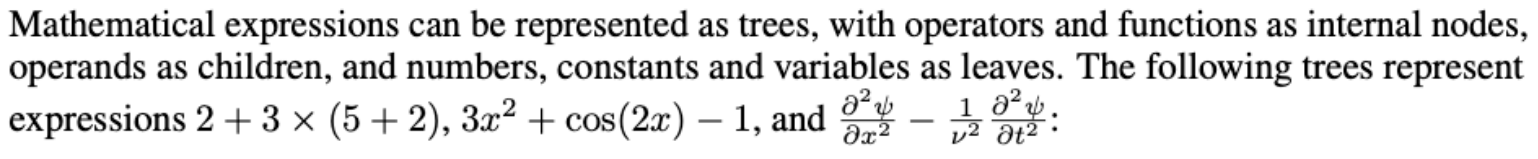
Aside: Symbolic Regression with Transformers

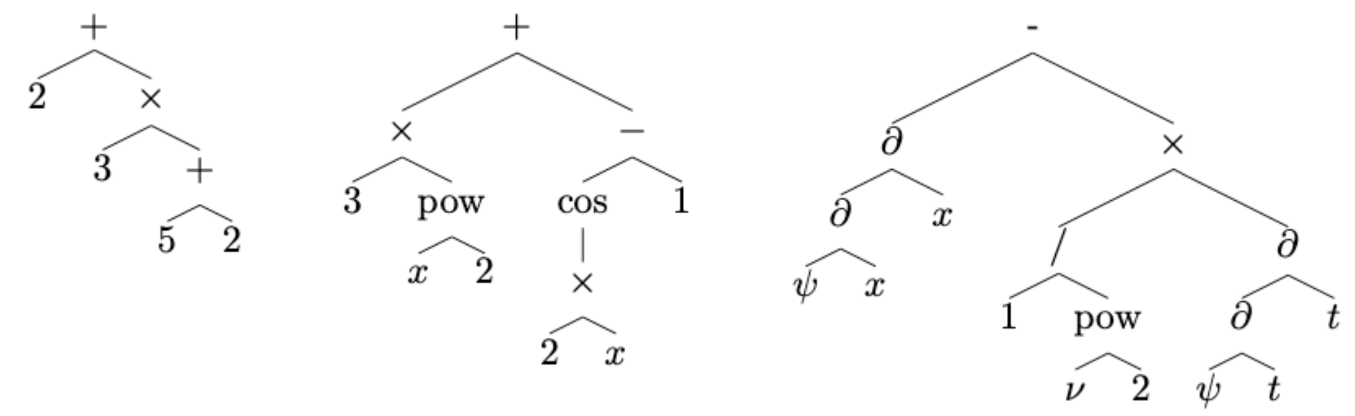
Foundation Model combining symbolic laws and data?

- One can **train a multimodal transformer that has seen lots of pairs of data and corresponding equations.**
- The transformer might then be **able to “guess” the underlying equation of novel** data, e.g. **a function, ODE or even PDE system**.
- The hope would be that the model learns some kind of “organic understanding of equations”.
- This is not currently a competitive approach, but there is some experimentation in this direction.

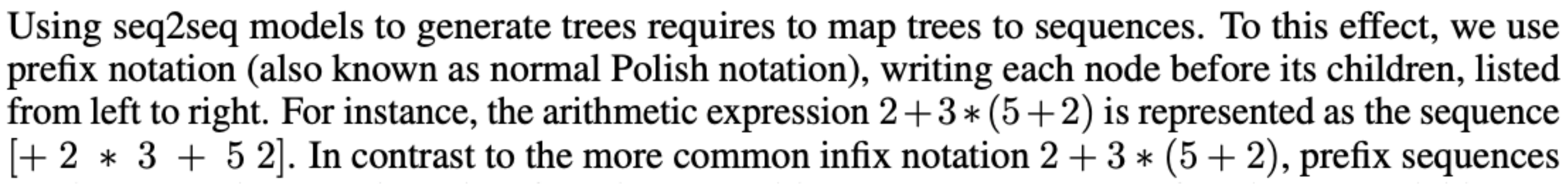
How to encode equations?

- https://arxiv.org/abs/1912.01412 Deep Learning for Symbolic Mathematics
- This paper used **sequence-to-sequence models (e.g. transformer)** on two problems of symbolic mathematics: function integration and ordinary differential equations (ODEs)
- Representing mathematical expressions as trees


Mathematical expressions can be represented as trees, with operators and functions as internal nodes, operands as children, and numbers, constants and variables as leaves. The following trees represent expressions $2 + 3 \times (5 + 2)$, $3x^2 + \cos(2x) - 1$, and $\frac{\partial^2 \psi}{\partial x^2} - \frac{1}{\nu^2} \frac{\partial^2 \psi}{\partial t^2}$:

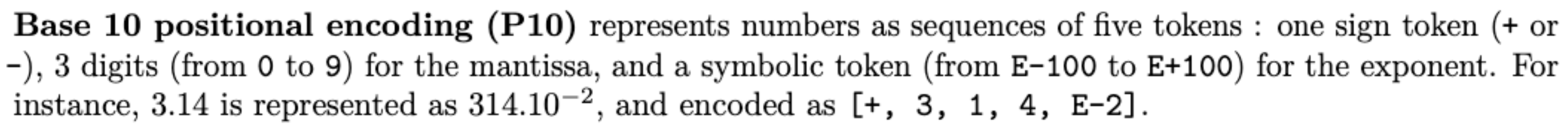


- The tree can then be represented as a sequence of tokens.


Using seq2seq models to generate trees requires to map trees to sequences. To this effect, we use prefix notation (also known as normal Polish notation), writing each node before its children, listed from left to right. For instance, the arithmetic expression $2+3*(5+2)$ is represented as the sequence $[+\ 2\ *\ 3\ +\ 5\ 2]$. In contrast to the more common infix notation $2 + 3 * (5 + 2)$, prefix sequences

How to encode equations?

- Real numbers can also be discretized as tokens, for example as follows (2112.01898):



Base 10 positional encoding (P10) represents numbers as sequences of five tokens : one sign token (`+` or `-`), 3 digits (from `0` to `9`) for the mantissa, and a symbolic token (from `E-100` to `E+100`) for the exponent. For instance, 3.14 is represented as 314.10^{-2}, and encoded as `[+, 3, 1, 4, E-2]`.

- There are other strategies to tokenize equations and numbers.

SR with transformers

End-to-end symbolic regression with transformers

Pierre-Alexandre Kamienny, Stéphane d'Ascoli, Guillaume Lample, François Charton

https://arxiv.org/pdf/2204.10532

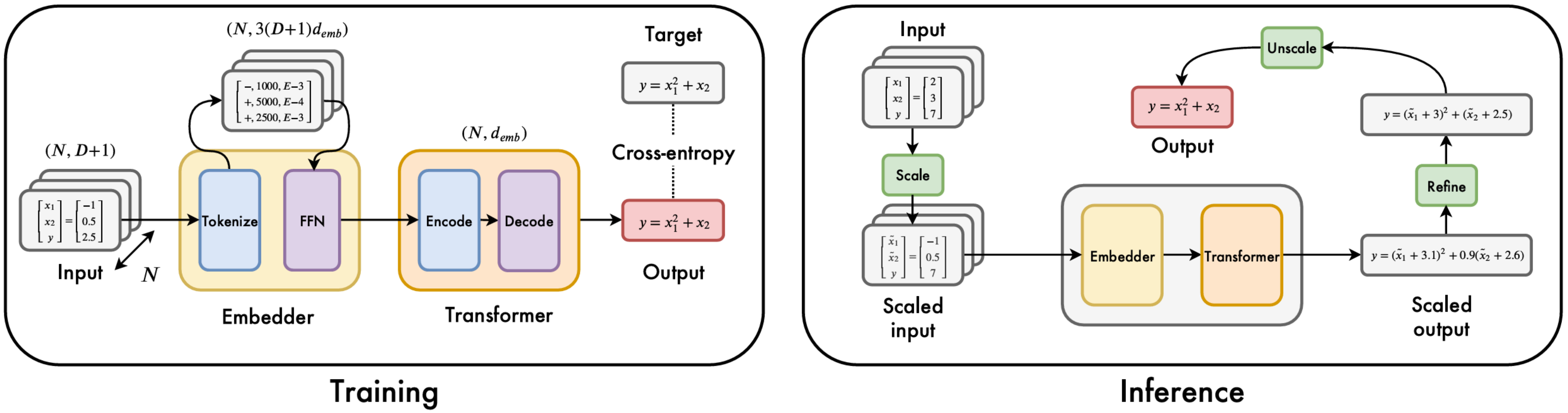
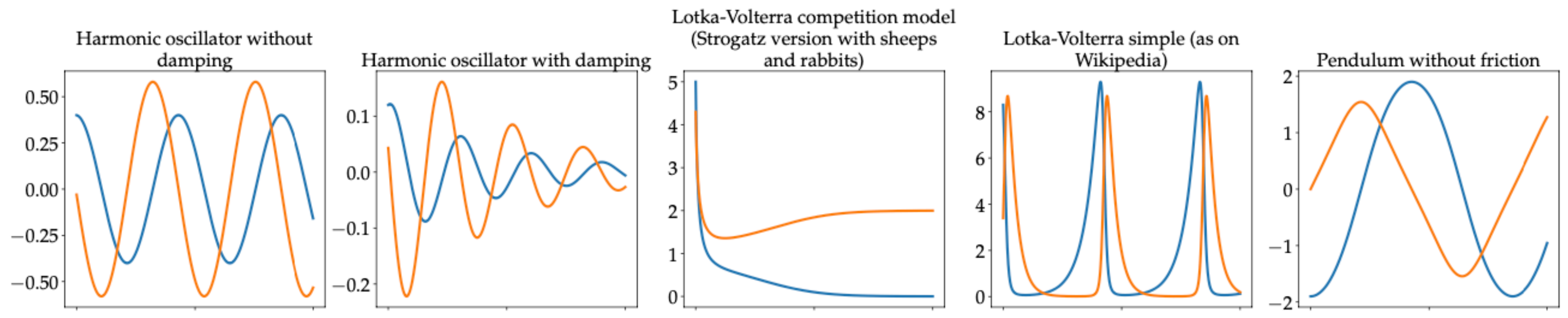


Figure 2: **Sketch of our model.** During training, the inputs are all whitened. At inference, we whiten them as a pre-processing step; the predicted function must then be unscaled to account for the whitening.

SR for ODEs: ODEFormer

- Can we give a neural network observed data (solutions to differential equations) and ask it to find the underlying equation of motion (differential equations)?
- Example:: https://arxiv.org/abs/2310.05573 **ODEFormer: Symbolic Regression of Dynamical Systems with Transformers**
- Example: 2-dimensional ODE

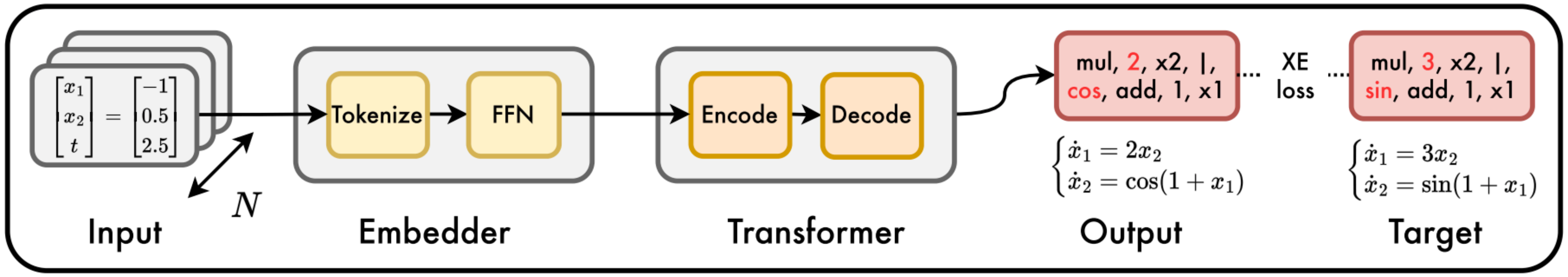
$$\dot{x} = f(x)$$



ID	System description	Equation	Parameters	Initial values
24	Harmonic oscillator without damping	$\begin{cases} x_1 \\ -c_0 x_0 \end{cases}$	2.1	[0.4, -0.03], [0.0, 0.2]
25	Harmonic oscillator with damping	$\begin{cases} x_1 \\ -c_0 x_0 - c_1 x_1 \end{cases}$	4.5, 0.43	[0.12, 0.043], [0.0, -0.3]
26	Lotka-Volterra competition model (Strogatz version with sheeps and rabbits)	$\begin{cases} x_0 (c_0 - c_1 x_1 - x_0) \\ x_1 (c_2 - x_0 - x_1) \end{cases}$	3.0, 2.0, 2.0	[5.0, 4.3], [2.3, 3.6]
27	Lotka-Volterra simple (as on Wikipedia)	$\begin{cases} x_0 (c_0 - c_1 x_1) \\ -x_1 (c_2 - c_3 x_0) \end{cases}$	1.84, 1.45, 3.0, 1.62	[8.3, 3.4], [0.4, 0.65]
28	Pendulum without friction	$\begin{cases} x_1 \\ -c_0 \sin(x_0) \end{cases}$	0.9	[-1.9, 0.0], [0.3, 0.8]

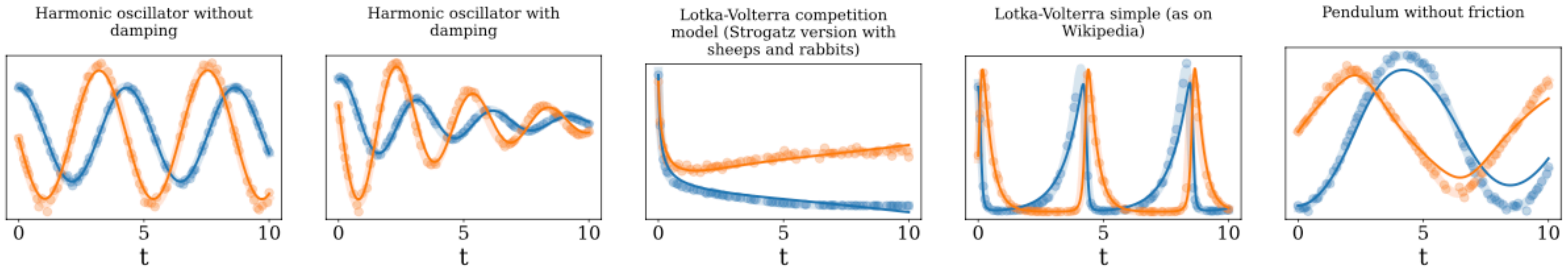
ODEFormer

- Use ideas we discussed: Tokenizing equations, transformer encoder-decoder



$\begin{bmatrix} x_1 \\ x_2 \\ t \end{bmatrix} = \begin{bmatrix} -1 \\ 0.5 \\ 2.5 \end{bmatrix}$
N
Input
Tokenize
FFN
Embedder
Encode
Decode
Transformer
mul, 2, x2, |, cos, add, 1, x1
XE loss
mul, 3, x2, |, sin, add, 1, x1
$\begin{cases} \dot{x}_1 = 2x_2 \\ \dot{x}_2 = \cos(1+x_1) \end{cases}$
$\begin{cases} \dot{x}_1 = 3x_2 \\ \dot{x}_2 = \sin(1+x_1) \end{cases}$
Output
Target

Figure 2: **Sketch of our method to train ODEFormer**. We generate random ODE systems, integrate a solution trajectory on a grid of N points $x \in \mathbb{R}^D$, and train ODEFormer to directly output the ODE system in symbolic form, supervising the predicted expression via cross-entropy loss.



- Could we discover the equations of motion of an observed system for which we don't know the answer?

Briefly: Emulators for expensive function evaluations

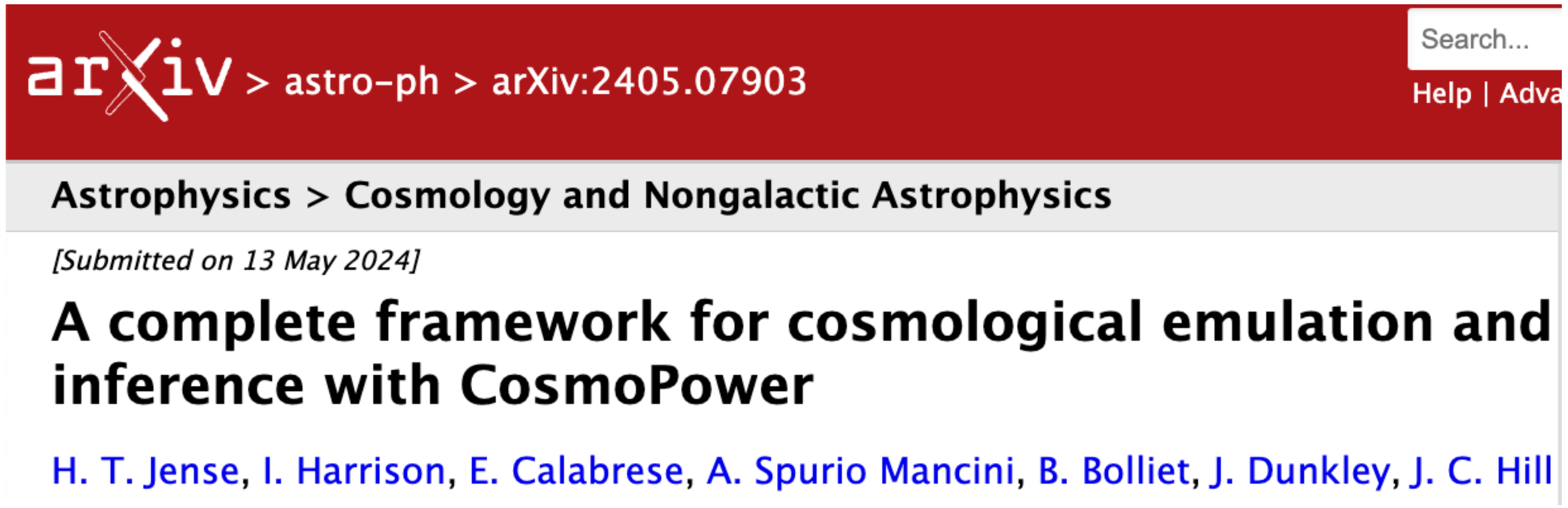
What are emulators?

- Emulators are **machine learning models that replace more precise and expensive analytic methods or simulations** with an approximate but **much faster** machine-learned model.

$$f(\mathbf{x}) \approx \hat{f}_{\mathrm{ML}}(\mathbf{x})$$

- This is sometimes called an **emulator or a surrogate model**. We have discussed such cases before but not in this language.
- E.g. We have seen how to use **diffusion models to emulate real simulations**. But the idea also works for **general functions, not just simulations**.
- Emulators can have other advantages over the exact method:
    - They are **auto-differentiable**. This can make high-dimensional Bayesian analysis tractable.
- Common methods to learn emulators of functions are **Gaussian Processes** and **Neural Networks**.
- Analyses where emulators can be switched with the full slower method are in some sense **interpretable (since ML is purely used for speedup)**.

Emulators to speed up exact codes



arXiv > astro-ph > arXiv:2405.07903

Search...

Help | Adva

Astrophysics > Cosmology and Nongalactic Astrophysics

[Submitted on 13 May 2024]

A complete framework for cosmological emulation and inference with CosmoPower

H. T. Jense, I. Harrison, E. Calabrese, A. Spurio Mancini, B. Bolliet, J. Dunkley, J. C. Hill

- A complicated and slow code (“Einstein-Boltzmann solver”) calculates predictions for observables in cosmology for given cosmological parameters.

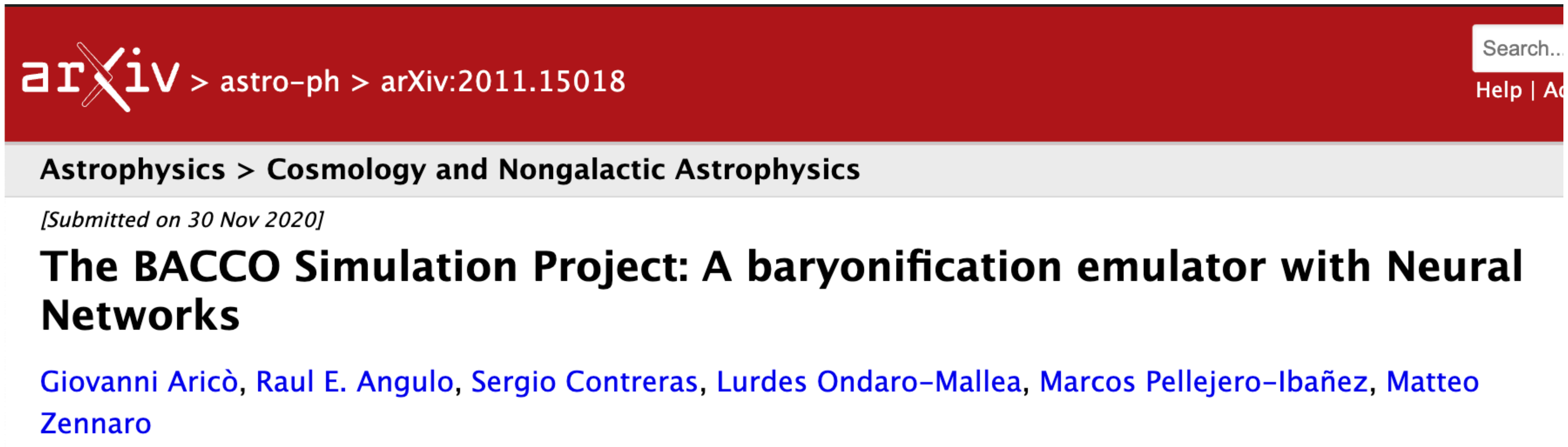
- This code needs to be called thousands of times in cosmological parameter inference.

- Instead, one can train a neural network to approximate and interpolate this computation.

- Can make analyses that would take weeks run in seconds on a laptop.

Emulators for interpolating simulation results

- Emulators are also used **to make continuous functions from discrete simulation data**.
- If you have "summary statistic" T($\mathbf{x}_i$) that extracts information from a model (see SBI unit), you may be able to simulate the summary statistic at some discrete parameters ($\mathbf{x}_i$). If you train a neural network to learn T($\mathbf{x}$) from this data, you can then use a continuous function during inference.
- Example from cosmology:



arXiv > astro-ph > arXiv:2011.15018

Search..
Help | Ad

Astrophysics > Cosmology and Nongalactic Astrophysics

[Submitted on 30 Nov 2020]

The BACCO Simulation Project: A baryonification emulator with Neural Networks

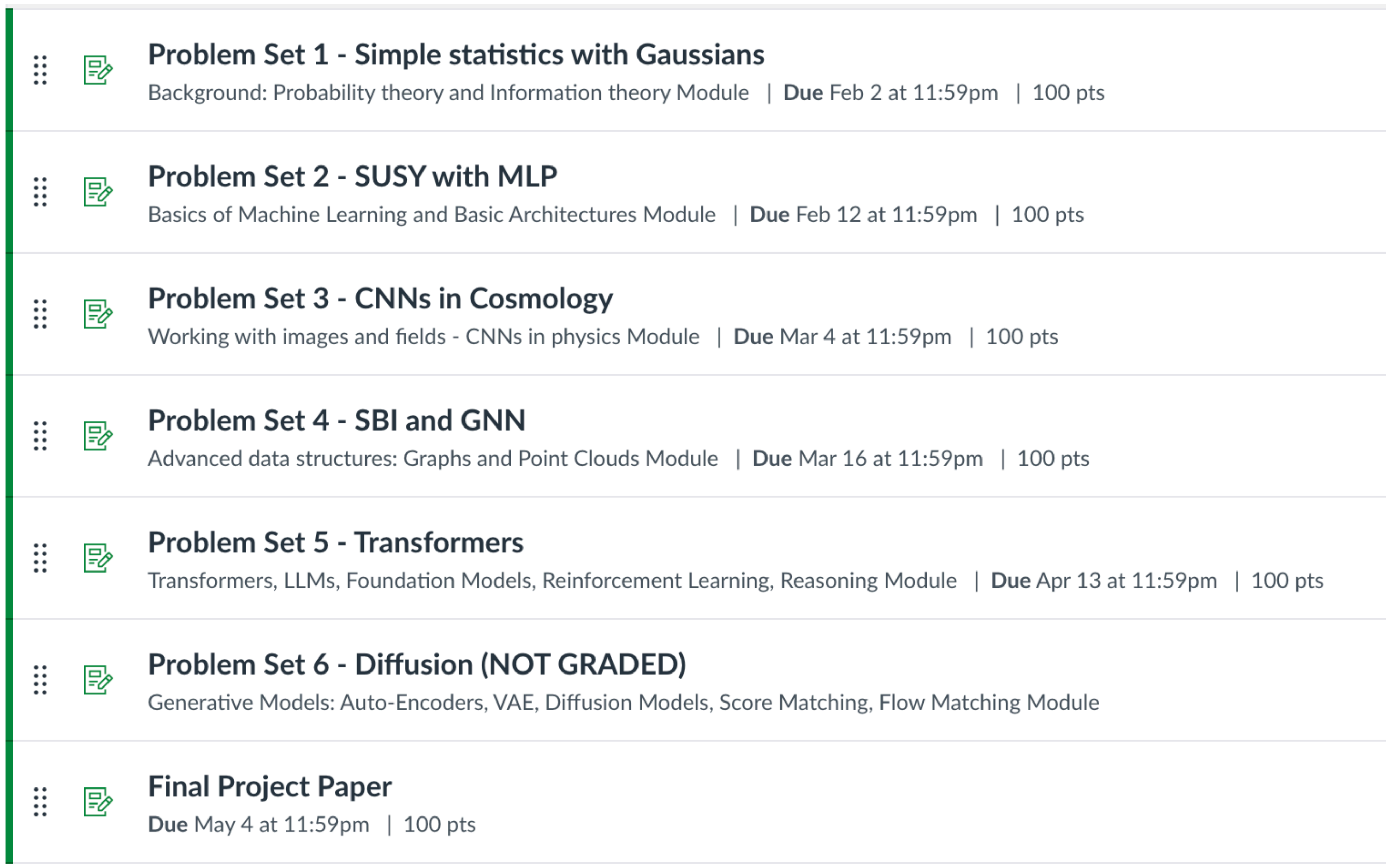
Giovanni Aricò, Raul E. Angulo, Sergio Contreras, Lurdes Ondaro-Mallea, Marcos Pellejero-Ibañez, Matteo Zennaro

Summary of the Semester

What we covered

- Background: Probability theory and Information theory
- Basics of Machine Learning and Basic Architectures
- Working with images and fields - CNNs in physics
- Simulation-based inference & Uncertainty Quantification
- Advanced data structures: Graphs and Point Clouds
- Transformers, LLMs, Foundation Models, Reinforcement Learning, Reasoning
- Generative Models: Auto-Encoders, VAE, Diffusion Models, Score Matching, Flow Matching
- Brief outline of other advanced topics:
    - PDE solving, Inverse problems, Anomaly detection
    - Interpretability, Symbolic Regression, Emulators

Assignments



Problem Set 1 - Simple statistics with Gaussians
Background: Probability theory and Information theory Module | **Due** Feb 2 at 11:59pm | 100 pts

Problem Set 2 - SUSY with MLP
Basics of Machine Learning and Basic Architectures Module | **Due** Feb 12 at 11:59pm | 100 pts

Problem Set 3 - CNNs in Cosmology
Working with images and fields - CNNs in physics Module | **Due** Mar 4 at 11:59pm | 100 pts

Problem Set 4 - SBI and GNN
Advanced data structures: Graphs and Point Clouds Module | **Due** Mar 16 at 11:59pm | 100 pts

Problem Set 5 - Transformers
Transformers, LLMs, Foundation Models, Reinforcement Learning, Reasoning Module | **Due** Apr 13 at 11:59pm | 100 pts

Problem Set 6 - Diffusion (NOT GRADED)
Generative Models: Auto-Encoders, VAE, Diffusion Models, Score Matching, Flow Matching Module

Final Project Paper
Due May 4 at 11:59pm | 100 pts

What’s next?

- We covered a lot of material from **basic stuff to state-of-the-art**, which should give you a good high-level **overview of what’s out there in AI for Science**.

- Almost all the things we covered will be useful in physics research or other sciences, and are not likely to be outdated soon.

- I hope that this course will allow you to get more deeply into a topic you are interested in, and understand publications in the field.

- **There is a lot of exciting work to be done in AI for Physics or AI for Science**, with potentially big consequences for both physics and humanity.

Thanks for participating in this course!