Symbolic Regression

Giulio CARPI LAPI **Abdou WADE**

Supervised by
Emmanuel FRANCK and **Victor MICHEL-DANSAC**

January 30, 2025

UFR de **mathématique** et d'**informatique**
Université de Strasbourg

IRMA

Inria

Motivation

- **PDEs** (e.g., Advection-Diffusion) are often **high-dimensional** and expensive to solve for many parameter variations.
- Traditional **Reduced-Order Models** (ROMs) (e.g., POD) can reduce computational cost, but rely on **linear** bases.
- **Neural networks** (autoencoders) provide **nonlinear dimension reduction**, yet can be "black-box."

Objective

Combine **Convolutional Autoencoder** (for compression) and **Symbolic Regression** (for interpretability):
to build a more **efficient** & **interpretable** ROM for the 1D Advection-Diffusion equation.

Project Overview

- **Numerical PDE Simulation:**
  - 1D **Advection-Diffusion** with **Gaussian** initial conditions.
  - Generate **PDE snapshots** over varying μ_0, σ_0.
- **Autoencoder:**
  - **Compress** PDE solutions into a **2D latent space** (Conv1d).
- **Symbolic Decoder (SINDy):**
  - Learn a **sparse symbolic expression** mapping the 2D latent space back to $u(x)$

Advection-Diffusion Equation

$$\partial_t u + a\partial_x u - D\partial_x^2 u = 0 \tag{1}$$

- a: **advection** speed.
- D: **diffusion** coefficient.

Context: fluid mechanics, **heat transfer**, **pollutant transport**, etc.

Gaussian Initial Condition

$$u(x, 0) = \frac{1}{\sigma_0\sqrt{2\pi}} \exp\left(-\frac{(x - \mu_0)^2}{2\sigma_0^2}\right) \tag{2}$$

- Analytical solution remains **Gaussian**: $\mu(t), \sigma(t)$ evolve over time.
- We **solve numerically** and **store snapshots** to form our **training dataset**.

Data Generation

- **Finite-Difference Scheme: Upwind** for advection, **centered** for diffusion, **explicit Euler** in time.
- Vary (μ_0, σ_0) over chosen sets.
- Collect **PDE solutions** at discrete time intervals $\rightarrow$ forms a **dictionary of simulations**.

Data generation: Plot

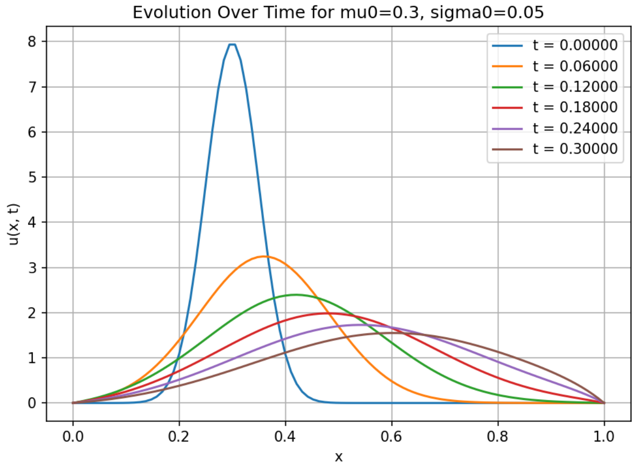

Figure: Time evolution of the advection-diffusion solution for $\mu_0 = 0.3$ and $\sigma_0 = 0.05$ at selected time points. [1]

Convolutional Autoencoder

- **Encoder:**
  - **Conv1d** layers reduce $[1, N_x] \to [1, 2]$.
- **Decoder:**
  - **ConvTranspose1d** layers expand $[1, 2] \to [1, N_x]$.
- **Training:**
  - **Loss** $= \text{MSE}(u_{\text{recon}}, u_{\text{true}})$.
  - Typically 100 epochs, Adam optimizer, LR$=10^{-3}$, batch size$=16$.

Convolutional Autoencoder

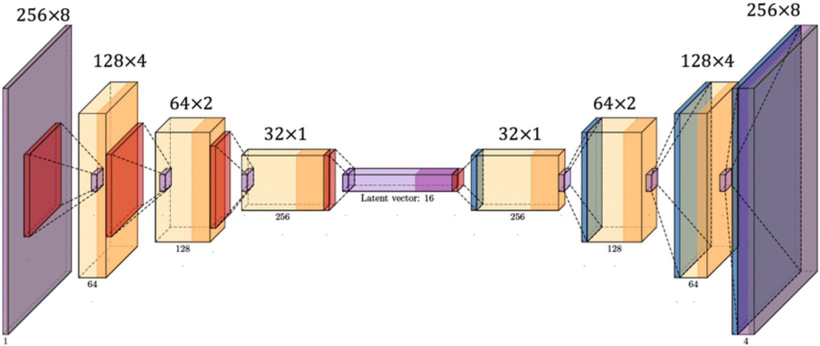

Figure: Autoencoder convolutional neural network (CNN) structure. [2]

Autoencoder Results: Loss Evolution

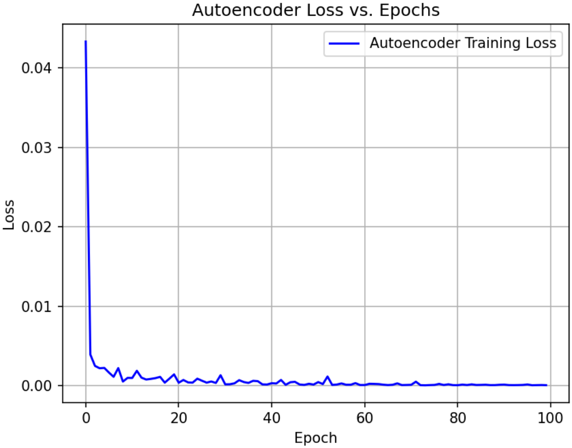

Figure: Training loss evolution for the autoencoder. [3]

Autoencoder Results: Reconstruction Plots

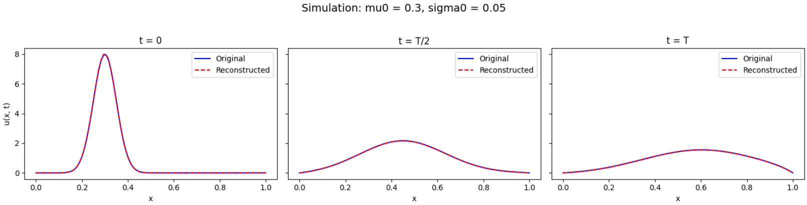

Figure: Comparison: Ground truth PDE snapshot vs. autoencoder reconstruction. [4]

Symbolic Decoder (SINDy)

- **Motivation:** standard NN decoders are black-box. We want a **sparse, interpretable** formula.
- **Approach**:
  1. Map PDE snapshots $\to (z_1, z_2)$ with the trained encoder.
  2. Build a library $\Theta(z_1, z_2)$ (polynomials, sines, cosines, exponentials).
  3. **L1-regularized regression**: $\min_\alpha \|\hat{u} - \Theta\alpha\|^2 + \lambda\|\alpha\|_1$.
  4. Prune small $\alpha \to 0 \to$ yields a **sparse symbolic expression**.

Symbolic Decoder Architecture

- **Encode** PDE snapshots $\rightarrow$ Autoencoder $\rightarrow$ 2D latent space.
- **Symbolic Decoder:** $\hat{u}(x_i) = \Theta(z_1, z_2) \cdot \alpha$.
- **Loss:** $MSE(u_{\text{true}}, \hat{u}) + \lambda \|\alpha\|_1$.
- **Threshold:** Zero out small coefficients.

Symbolic Decoder Architecture

$$\hat{u}(x_i) = \Theta(z_1, z_2)\cdot\alpha, \quad \mathsf{Loss} = \mathrm{MSE}(u_{\mathsf{true}}, \hat{u}) + \lambda\|\alpha\|_1.$$

- Each grid point x_i has a **separate set** of coefficients.
- **func7, func9, etc.** denote specific library functions we appended to fill the library size.
- After training, each $u(x_i)$ is a **sparse combination** of these library terms.

Example Symbolic Expressions (SINDy)

Outputs from the final SINDy decoder:

Output 3:

$$0.0048\, z_1 + 0.0190\ \cos(0.0000\, z_2)$$

Output 7:

$$0.0034\, z_1^2 + 0.0622\ \cos(0.0000\, z_2) + 0.0010\, \mathrm{func9}(z)$$

Output 11:

$$0.0389\, z_1 + 0.0373\, z_2 + 0.1785\ \cos(0.0000\, z_2) + 0.0054\, \mathrm{func7}(z)$$

Many coefficients are zero, leaving these dominant terms.

Interpretability of Symbolic Expressions

- Each spatial output $u(x_i)$ expressed as a **sparse function** of (z_1, z_2).
- Decoder relies on **small-frequency trigonometric components**.
- Some expressions include **polynomial terms** like z_1^2, z_2^2, or cross terms func7(z) $\rightarrow$ indicates **nonlinear interactions** in latent space.

Interpretability of Symbolic Expressions

- **Interpretation:** The PDE solution manifold is captured by a small set of **basis functions** in **latent coordinates**.
- **Caution:** We get a **separate expression** per x_i, so interpretability is **partial**. Next steps might treat x explicitly in the library.

SINDy Results: Training Loss

Loss Evolution over 10,000 epochs:

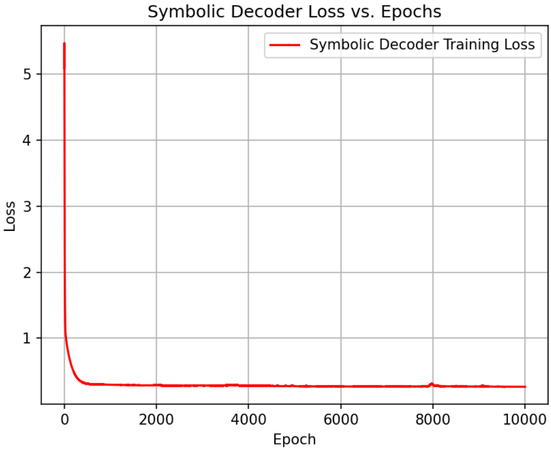

Figure: Training loss evolution for the symbolic decoder, including both MSE and L_1 penalty. [5]

SINDy Results: Reconstruction Plots

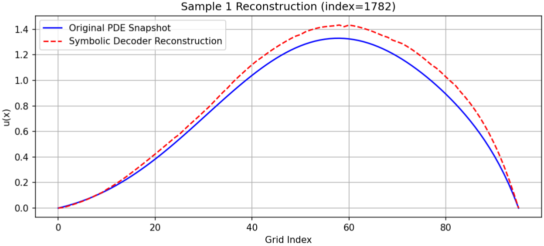

Figure: Ground truth (blue) vs. SINDy reconstruction (red). [6]

SINDy Results: Reconstruction Plots

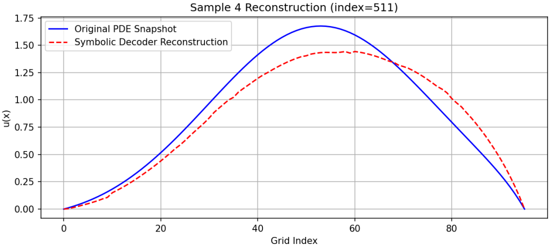

Figure: Ground truth (blue) vs. SINDy reconstruction (red). [7]

SINDy Results: Reconstruction Plots

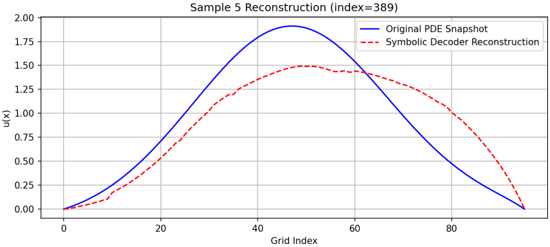

Figure: Ground truth (blue) vs. SINDy reconstruction (red). [8]

Prospects

- **Higher-Dimensional PDEs:** Convolutional autoencoders with **Conv2d**/**Conv3d**.
- **Library Engineering:** Tailor **symbolic library** to PDE physics.

Conclusion

- Combined **convolutional autoencoder** for nonlinear dimension reduction with **SINDy** for **symbolic decoding**.
- Achieved both **efficiency** and **interpretability** for 1D advection-diffusion PDE.
- Future work: extension to **multi-dimensional PDEs**, refined **library design**, **real-time** parameter sweeps.

Thank you for your attention!

Questions?

Giulio Carpi Lapi and Abdou Wade.
Time evolution of the advection-diffusion solution at selected time points.
Generated using Python and Matplotlib, 2025.

Fangcao Xu, Guido Cervone, Gabriele Franch, and Mark Salvador.
Multiple geometry atmospheric correction for image spectroscopy using deep learning.
Journal of Applied Remote Sensing, 14(02):024518, 2020.
Figure: *autoencoder-CNN-structure.png* taken from this paper.

Giulio Carpi Lapi and Abdou Wade.
Generated plot of the loss evolution of the autoencoder over epochs.
Generated using Python and Matplotlib, 2025.

Giulio Carpi Lapi and Abdou Wade.
Generated plot of the autoencoder reconstruction.
Generated using Python and Matplotlib, 2025.

Giulio Carpi Lapi and Abdou Wade.
Generated plot of the loss evolution of the sindy decoder over epochs.
Generated using Python and Matplotlib, 2025.

Giulio Carpi Lapi and Abdou Wade.
Generated plot of the symbolic decoder's reconstruction.
Generated using Python and Matplotlib, 2025.

Giulio Carpi Lapi and Abdou Wade.
Generated plot of the symbolic decoder's reconstruction.
Generated using Python and Matplotlib, 2025.

Giulio Carpi Lapi and Abdou Wade.
Generated plot of the symbolic decoder's reconstruction.
Generated using Python and Matplotlib, 2025.

Pieter Wesseling.
Principles of Computational Fluid Dynamics, volume 29 of *Springer Series in Computational Mathematics*.
Springer Berlin Heidelberg, Berlin, Heidelberg, 2001.

David Skinner.
Green's functions for pdes.
Lecture Notes, Mathematical Methods, University of Cambridge, 2024.
Available at `http://www.damtp.cam.ac.uk/user/dbs26/1BMethods/All.pdf`.

E. Franck.
Scientific machine learning (sciml) master course, 2024.

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.

Norbert Stoop.
Advection-diffusion, nonlinear transport.
https://math.mit.edu/~stoopn/18.086/Lecture7.pdf, 2016.
Lecture 7 from "18.086 - Computational Science and Engineering II (Spring 2016)" at MIT Department of Mathematics.

Inductiveload.
Normalized Gaussian curves with expected value μ and variance σ^2. The corresponding parameters are $b = \mu$ and $c = \sigma$.
https://en.wikipedia.org/wiki/Gaussian_function#/media/File:Normal_Distribution_PDF.svg, 2016.

Giulio Carpi Lapi and Abdou Wade.

Generated plot of gaussian curves with different values for μ_0 and σ_0^2 as initial conditions.
Generated using Python and Matplotlib, 2024.

Giulio Carpi Lapi and Abdou Wade.
Generated plot of the autoencoder reconstruction.
Generated using Python and Matplotlib, 2025.

Giulio Carpi Lapi and Abdou Wade.
Generated plot of the autoencoder reconstruction.
Generated using Python and Matplotlib, 2025.