AI surrogate models, interpretability, and uncertainty quantification

Presenter: Alessandro Fanfarillo

Oct 15th 2025

AMD @ CASTIEL

AMD

together we advance_

Agenda

1. Inverse problem solution based on AI
   - Heat source-term estimation based on Neural Network
2. Introduction to Bayesian Neural Networks
   - Uncertainty Quantification for source-term estimation problem
3. Interpretability of AI models
   - Knowledge distillation: Decision Trees, Symbolic Regression

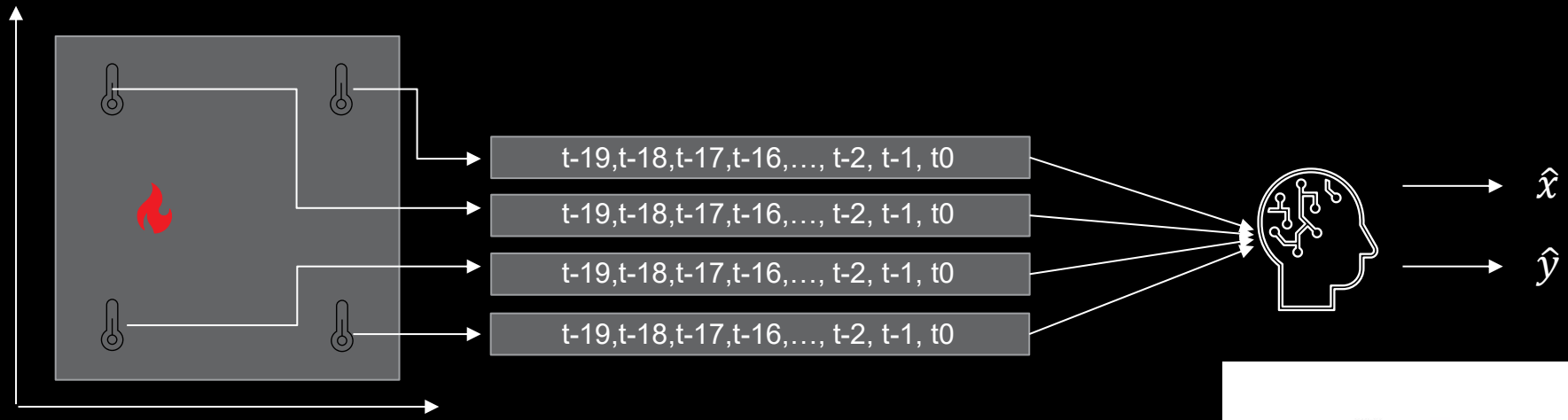
Introduction

- Inverse problem of heat source-term estimation used as main example
- Surrogate AI models can be used in HPC applications, but deeper understanding is required
- How is the model going to behave in rare/extreme events?
- How to quantify uncertainty in the predictions coming out of an AI surrogate?
- How to interpret what the model learned?
- What features are the most meaningful for the model?
- How big/complex a model should be?
- How much data is enough?

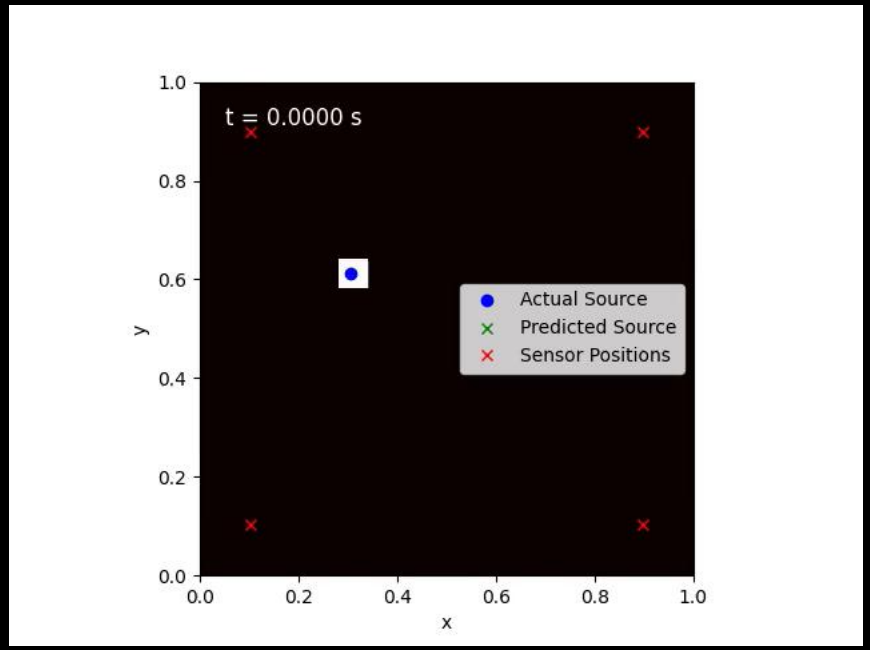
Heat source-term estimation problem

- A **source-term estimation problem** is about finding the location and strength of something that is being released or emitted into an environment. Source could be chemical, gas, pollutant, heat, sound, etc.
- Source-term estimation typically involves using measurements (e.g., sensors detecting the substance) and a mathematical model of how the substance spreads (e.g., how gas flows in the air)
- This is an inverse problem, we start from the effects and try to find out the cause, ill-posed (non-uniqueness, instability, non-existence)
- Inverse problems can be solved by running multiple simulations of where the source might be and assess whether the effects are consistent with the observations
- **Problem**: we have a 2D plate with 4 thermometers and a single heat source. We want to find the position of the source based on the readings
- AI is very effective in “solving” these kind of problems: *Fanfarillo A. - Quantifying Uncertainty in Source Term Estimation with TensorFlow Probability* https://ieeexplore.ieee.org/abstract/document/8944939

Traditional Neural Network solution



- Neural network (NN) based on 4 LSTM layers and 1 linear layer as output
- A window of 20 timesteps is expected by the LSTM layer
- Input to NN is 4 x 20:
  - 4 thermometers reporting data at each timestep
  - 20 timesteps expected by NN, most recent timestep kicks out oldest
- Output is the x and y coordinates on the 2D plate on where the heat source is supposed to be
- The 2D plate is 50 x 50 units
- The (perfect) thermometers are located at: (5,5), (44, 5), (5, 44), (44,44)
- Total simulation is 200 timesteps
- A fixed number of simulations with randomly located sources is used for training



What to do next?

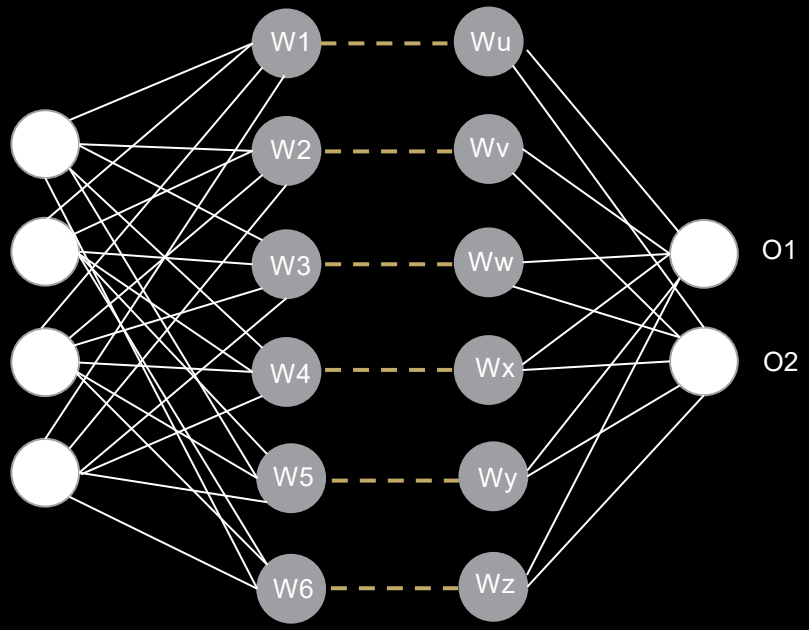
- The model seems to behave most of the times
- Was the training dataset representative “enough”?
- Was the model complex “enough”?
- Was the problem formulation good “enough”?
- Next steps:
  - Uncertainty quantification: allows you to understand how “sure” the model was about a prediction (systemic uncertainty) and/or how accurate a particular prediction can be based on the problem itself (aleatoric uncertainty). Bayesian Neural Networks, Gaussian Processes, Polynomial Chaos Expansion (PCE)
  - Interpret what the model has learned and how: knowledge distillation, feature importance, sensitivity analysis

Traditional vs. Bayesian Neural Networks

- Traditional Neural Network:
  - Deterministic weights: each connection in the network has a fixed weight value learned during training
  - Point estimates: the model provides single-point predictions without quantifying uncertainty
  - Overfitting risk: may overfit the training data, especially with limited data, leading to poor generalization on new data
- Bayesian Neural Network:
  - Probabilistic weights: weights are treated as probability distributions, capturing a range of possible values
  - Uncertainty quantification: BNNs provide a measure of confidence in their predictions by considering the uncertainty in the weights
  - Regularization: the Bayesian framework helps prevent overfitting by integrating prior knowledge and updating beliefs with new data

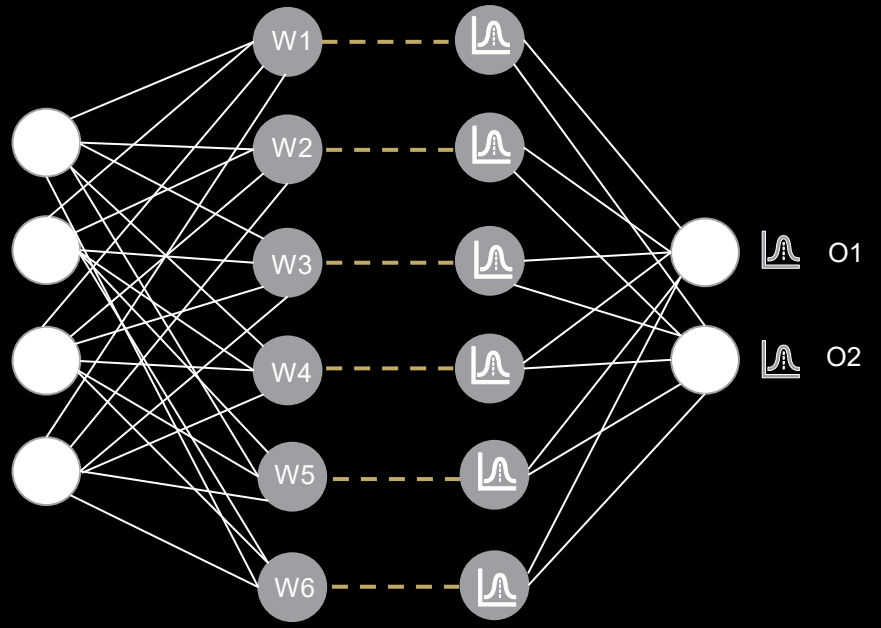
Traditional vs. Bayesian Neural Networks (BNN)

Traditional Neural Network



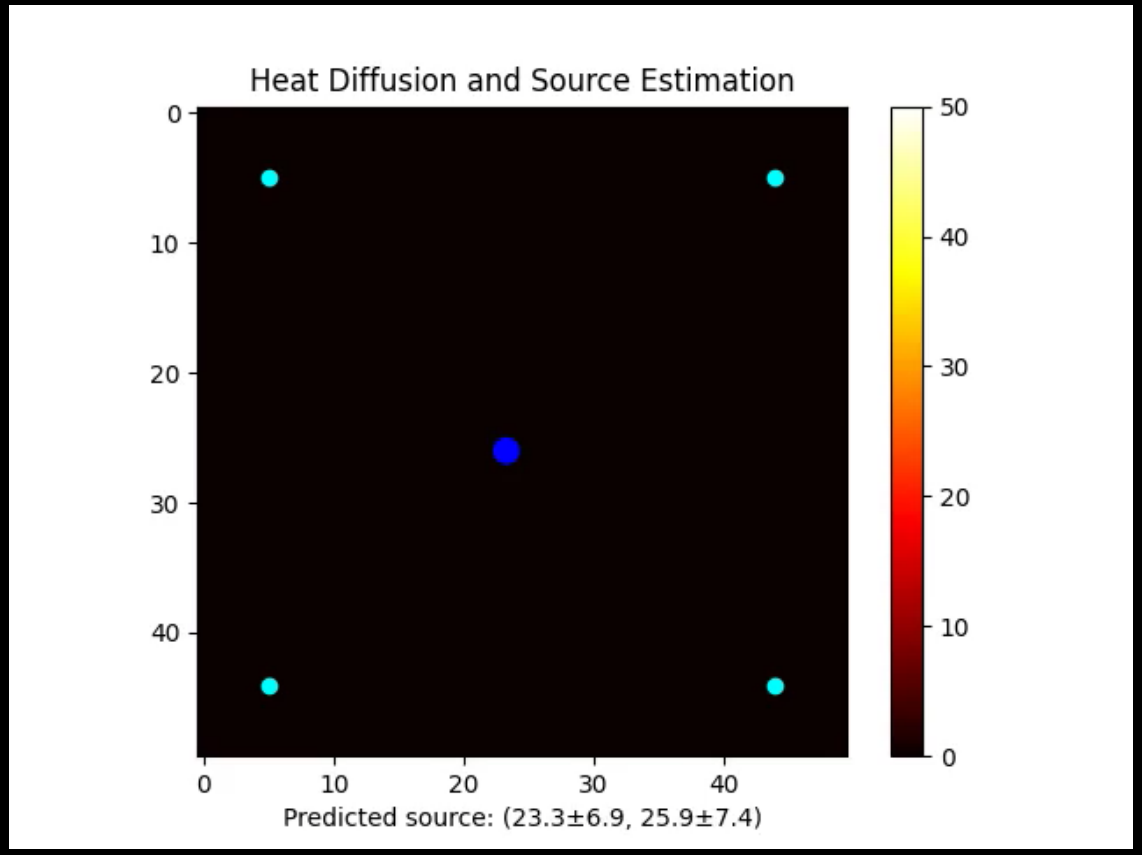
Weights and output variables are scalars

Bayesian Neural Network



Weights and output variables are random variables

Bayesian Neural Networks (video)



BNN Nuts and Bolts

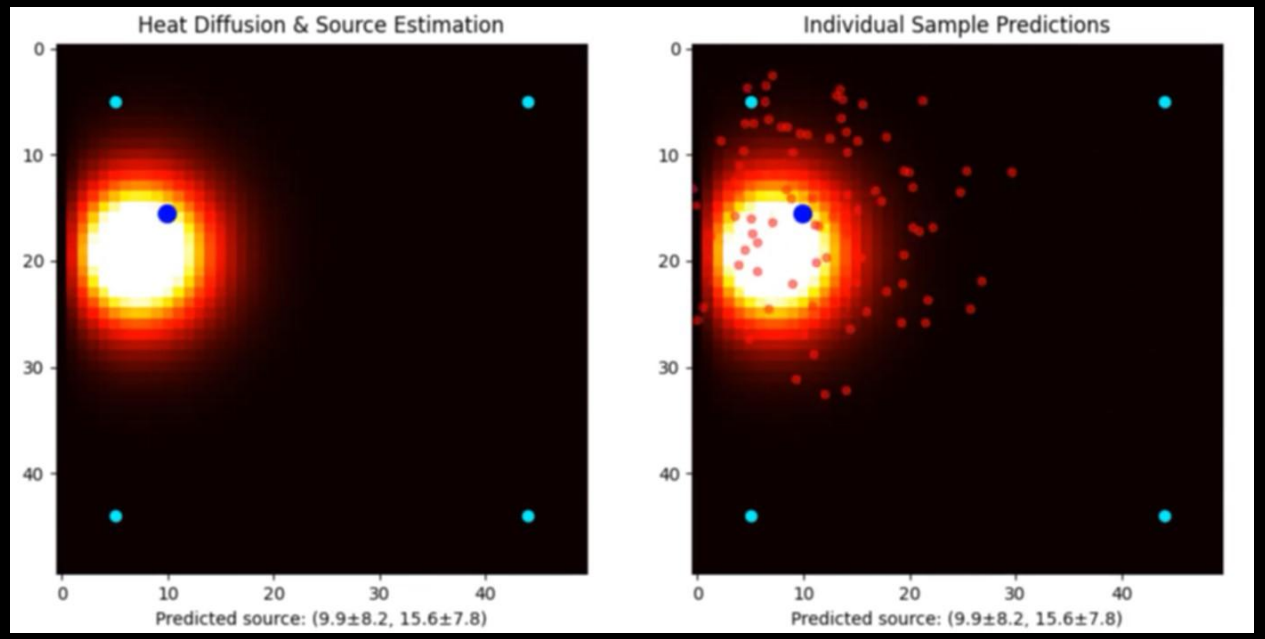
- Idea: use probability distributions that model the whole set of possible weights and/or outputs
- The distribution is assigned by the user/engineer (prior)
- The posterior distribution representing the weights is learned in a Bayesian way from the data:
  - Markov Chain Monte Carlo: the most Bayesian method, sample from the posterior distribution of a model's parameters. Very slow and hard to use in real applications. No assumptions on form of posterior. Overkill
  - **Variational Inference**: posterior computation as an optimization problem. Instead of directly calculating the posterior, it approximates it by selecting a simpler, parameterized distribution (variational distribution) and adjusts its parameters to be as close as possible to the true posterior. Surrogate of real posterior!
  - Monte Carlo Dropout: not really Bayesian, use Dropout in training and inference to generate random outputs
- Random variables can be used for internal weights, outputs, or both!
- Outputs as random variables quantify the aleatoric uncertainty
- Internal weights as random variables quantify the systemic uncertainty

Aleatoric and Systemic Uncertainty

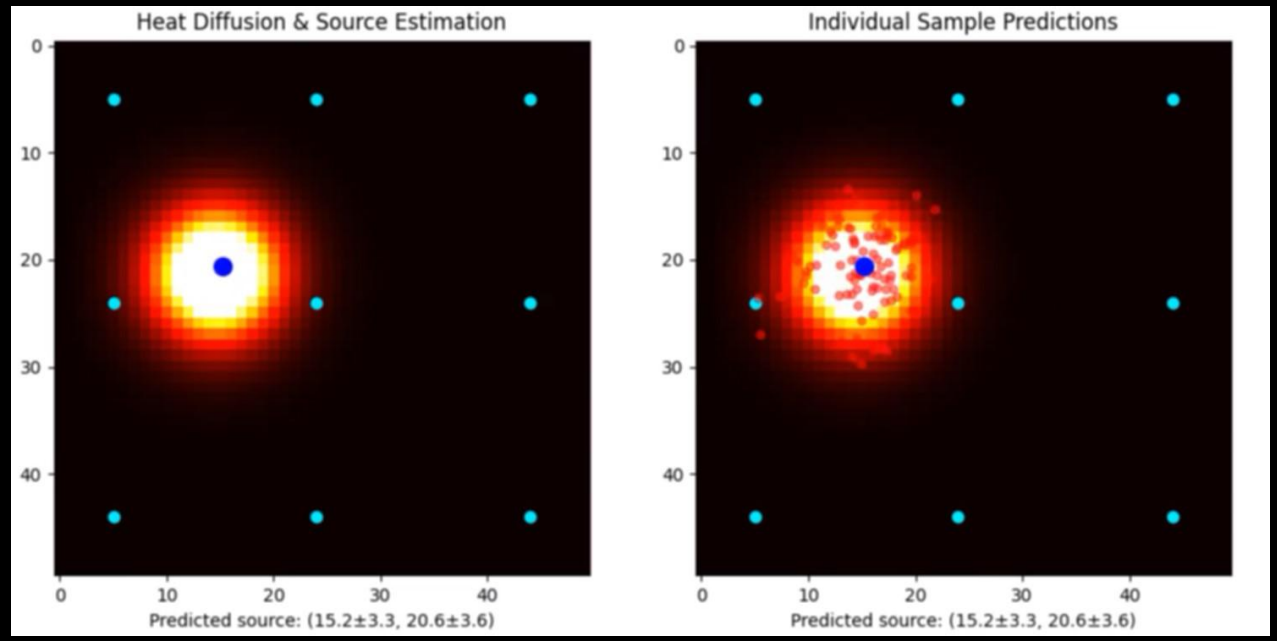
- Aleatoric uncertainty is caused by inherent randomness or variability in the system. This uncertainty **cannot** be reduced by gathering more data, as it is part of the system itself
  - Rolling a die: it is intrinsically random, no matter how much data is gathered
  - Weather prediction: even with perfect model and sensors, minimal variations cause major changes in prediction
- For our model, aleatoric uncertainty represents a fixed level of measurement or model error regardless of conditions. The stddev in the output variable quantifies this uncertainty. Sampling only the output variable
- Systemic uncertainty caused by lack of knowledge or insufficient data. This type of uncertainty can be reduced by improving the model or gathering more data
  - AI models trained on biased data: unseen data will have a greater uncertainty
  - For our model, it represents how “sure” the model is about a certain prediction, based on the data seen during training. Multiple runs of the same model, sampling the model parameters

Aleatoric Uncertainty

Using 4 sensors

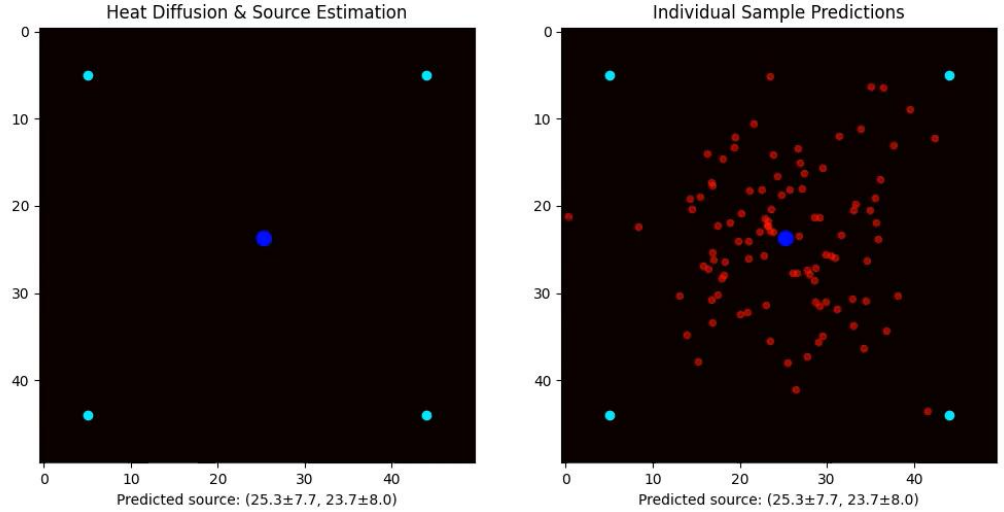


Using 9 sensors

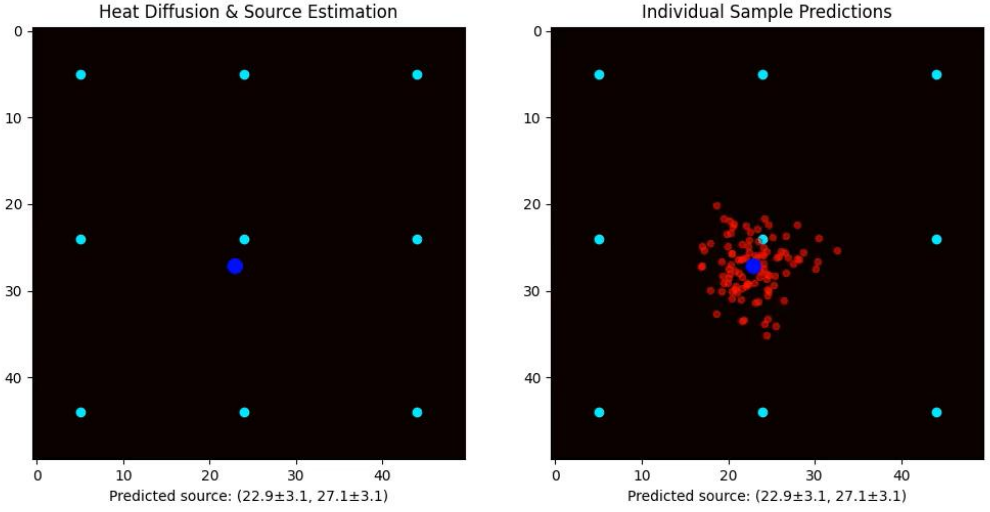


Aleatoric Uncertainty (video)

Using 4 sensors

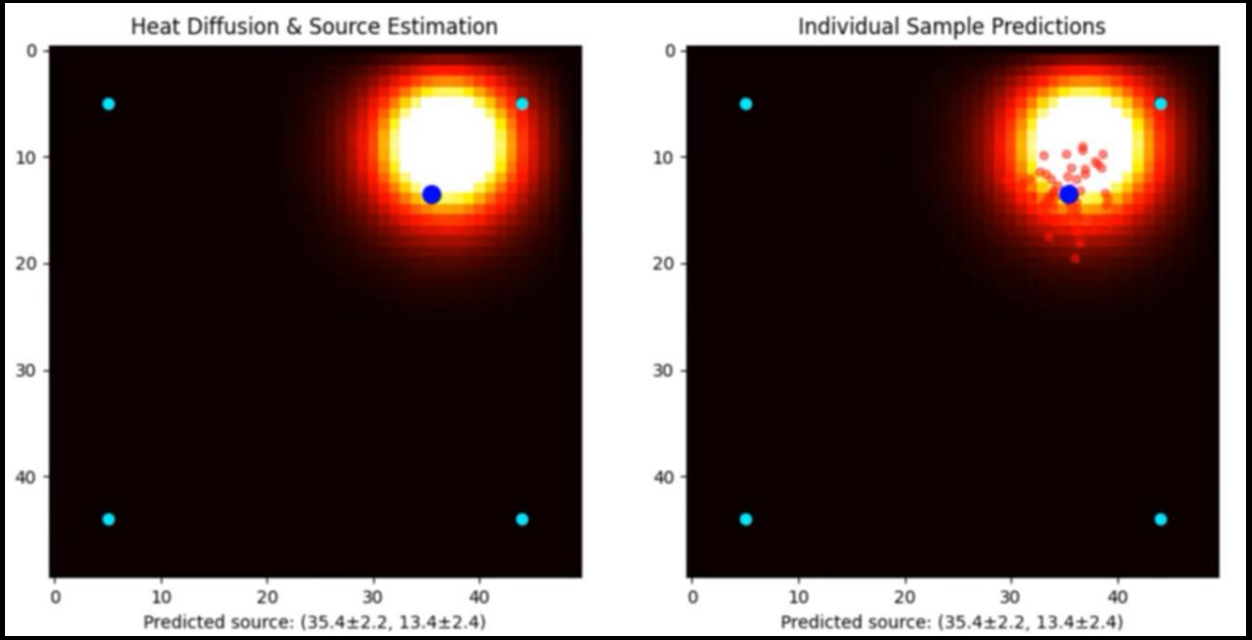


Using 9 sensors

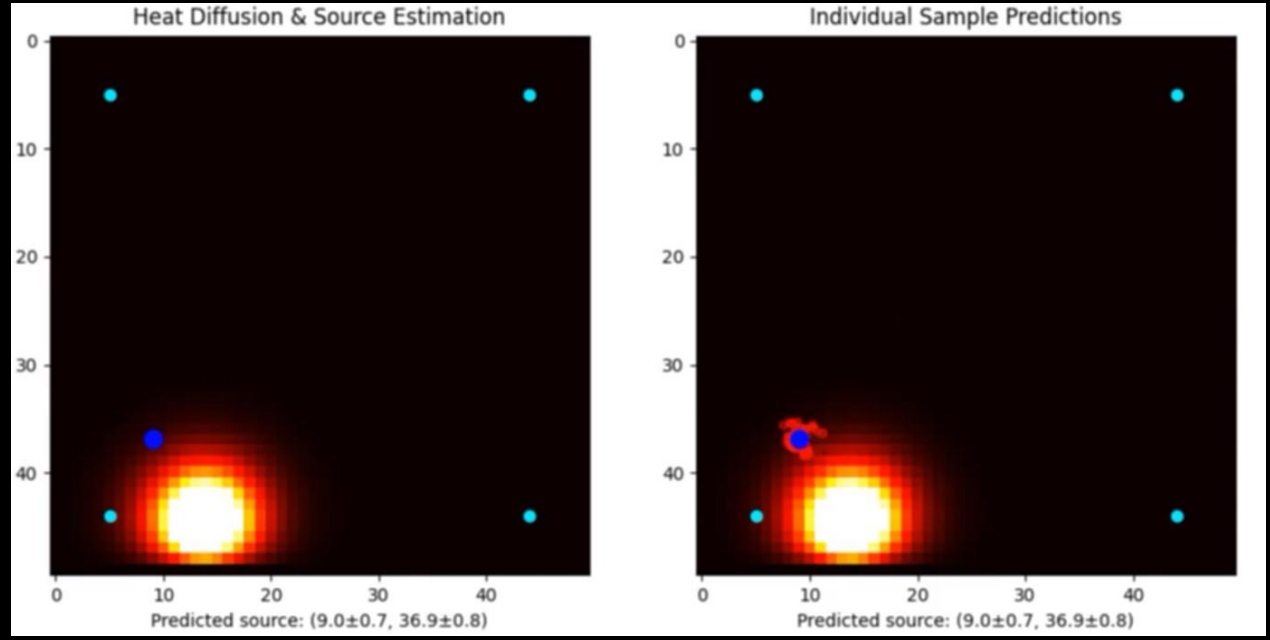


Systemic Uncertainty

10 training samples

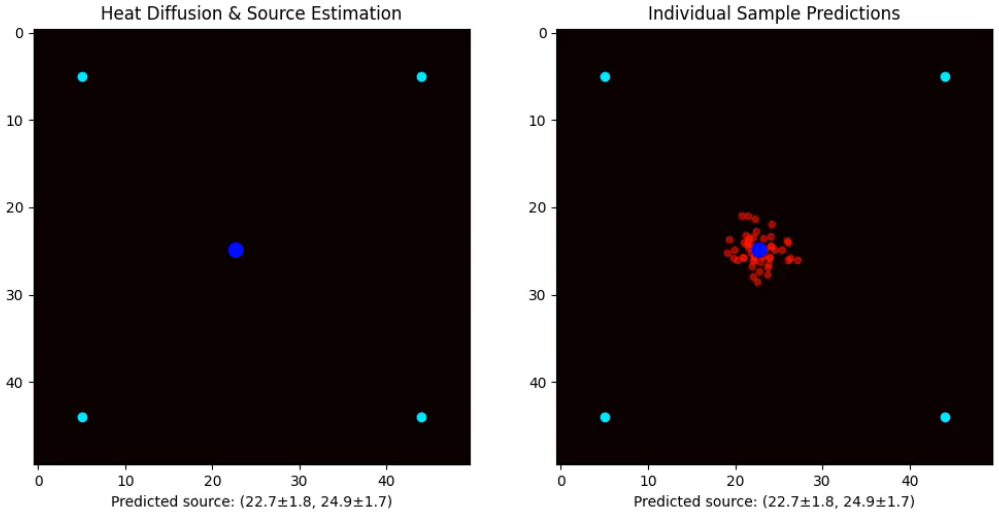


50 training samples

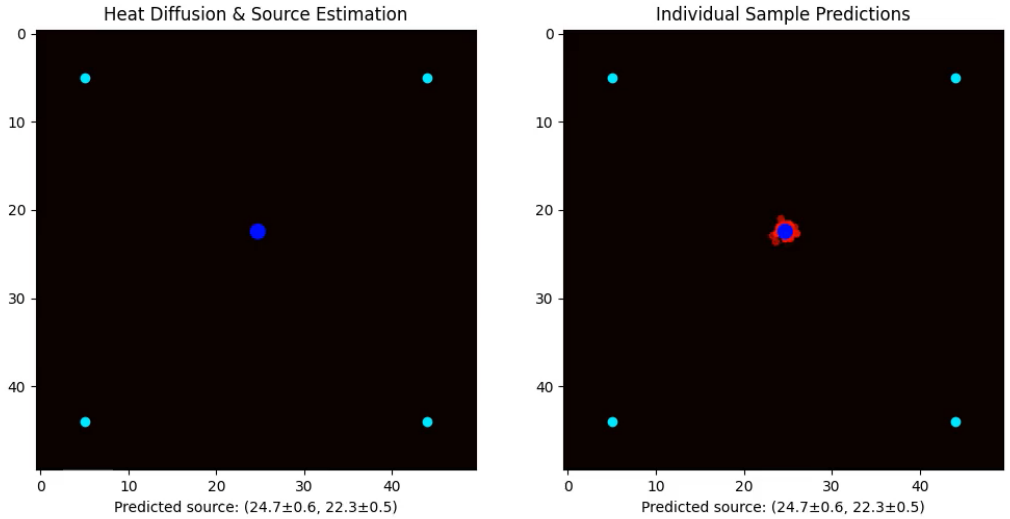


Systemic Uncertainty (video)

10 training samples



50 training samples



Knowledge distillation

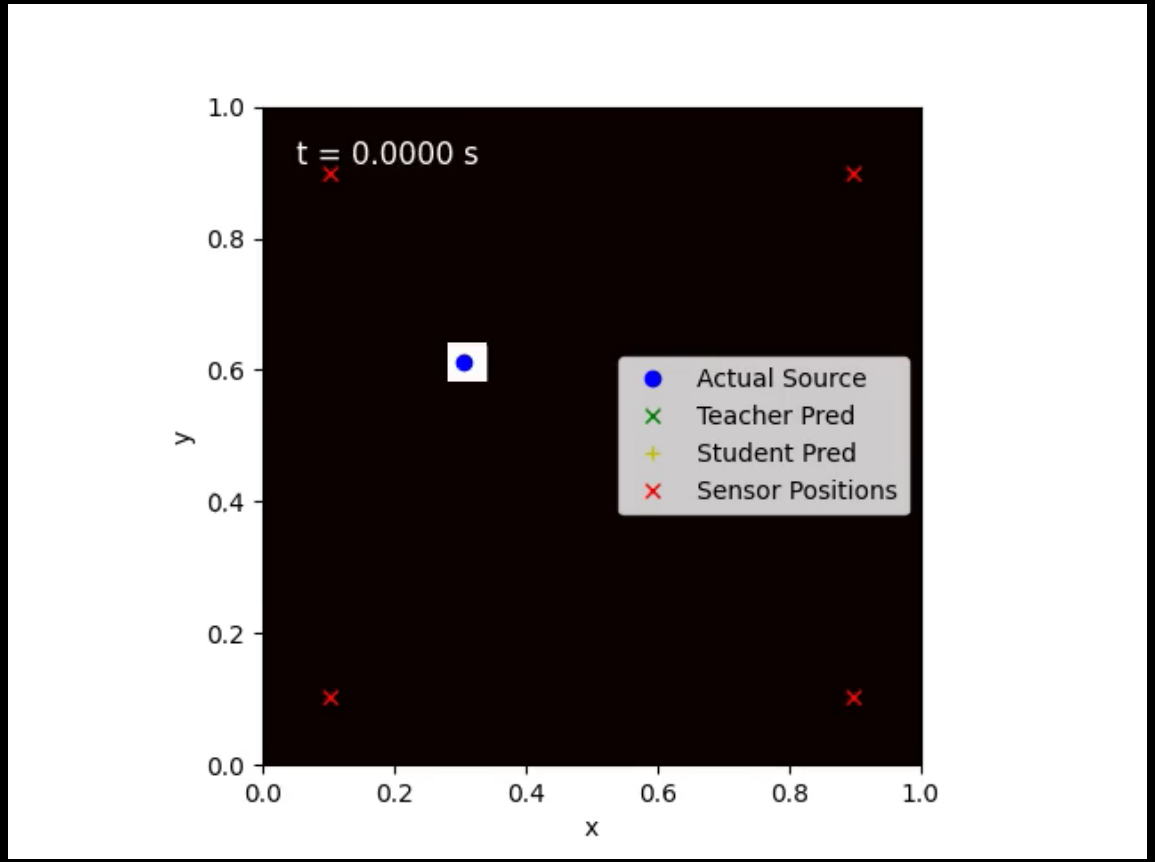
- Knowledge distillation is a technique for transferring the “knowledge” learned by a large model (teacher) into a simpler model (student) without sacrificing too much predictive power
- The teacher has learned an approximation of a complex function, trained on the original data
- The student is NOT trained on the original data but on the teacher’s outputs (mimic the teacher)
- If the student model is “easy” to understand for humans, we can interpret what the main model learned and/or we can use the human readable rules
- Good student models for interpretability:
  - Decision Trees/Random Forests: we can extract rules directly in the form of if-then-else statements
  - Linear models: easy to understand
  - Symbolic Regression: non-linear complex functions found via genetic optimization algorithms
- Run feature importance methods like LIME (Local Interpretable Model-agnostic Explanations) and SHAP (SHapley Additive exPlanations)

Decision Tree (DT) – Rules Extraction

- Train a DT based on the outputs from the Neural Network
- Extract the rules driving the DT in text format. Highly interpretable
- Rules improve trust, debugging, and regulatory compliance
- Very useful for fast (hard coded) inference in HPC libraries
- More complex tree-based model can be used:
  - Random Forests
  - Gradient Boosting
- Some branches/outcomes might repeat, tree simplification needed
- Parameters like depth and split need to be tuned for optimal results

```
Printing rules
|--- sensor0_t9 <= 0.07
|   |--- sensor1_t9 <= 0.30
|   |   |--- sensor3_t9 <= 0.08
|   |   |   |--- sensor2_t9 <= 0.01
|   |   |   |   |--- sensor3_t9 <= 0.00
|   |   |   |   |   |--- sensor1_t9 <= 0.00
|   |   |   |   |   |   |--- sensor0_t9 <= 0.00
|   |   |   |   |   |   |   |--- sensor2_t9 <= 0.00
|   |   |   |   |   |   |   |   |--- sensor0_t9 <= 0.00
|   |   |   |   |   |   |   |   |   |--- sensor1_t9 <= 0.00
|   |   |   |   |   |   |   |   |   |   |--- value: [24.44, 23.54]
|   |   |   |   |   |   |   |   |   |--- sensor1_t9 >  0.00
|   |   |   |   |   |   |   |   |   |   |--- value: [27.82, 22.80]
|   |   |   |   |   |   |   |   |--- sensor0_t9 >  0.00
|   |   |   |   |   |   |   |   |   |--- sensor0_t9 <= 0.00
|   |   |   |   |   |   |   |   |   |   |--- value: [23.40, 22.18]
|   |   |   |   |   |   |   |   |   |--- sensor0_t9 >  0.00
|   |   |   |   |   |   |   |   |   |   |--- value: [22.36, 20.89]
|   |   |   |   |   |   |   |   |--- sensor2_t9 >  0.00
|   |   |   |   |   |   |   |   |   |--- sensor2_t9 <= 0.00
|   |   |   |   |   |   |   |   |   |   |--- sensor2_t9 <= 0.00
|   |   |   |   |   |   |   |   |   |   |   |--- value: [21.95, 25.66]
|   |   |   |   |   |   |   |   |   |   |--- sensor2_t9 >  0.00
|   |   |   |   |   |   |   |   |   |   |   |--- value: [20.62, 26.45]
|   |   |   |   |   |   |   |   |   |--- sensor2_t9 >  0.00
|   |   |   |   |   |   |   |   |   |   |--- sensor0_t9 <= 0.00
|   |   |   |   |   |   |   |   |   |   |   |--- value: [19.29, 27.17]
|   |   |   |   |   |   |   |   |   |   |--- sensor0_t9 >  0.00
|   |   |   |   |   |   |   |   |   |   |   |--- value: [16.70, 26.35]
|   |   |   |   |   |   |   |--- sensor0_t9 >  0.00
|   |   |   |   |   |   |   |   |--- sensor1_t5 <= 0.00
|   |   |   |   |   |   |   |   |   |--- sensor2_t6 <= 0.00
|   |   |   |   |   |   |   |   |   |   |--- sensor0_t9 <= 0.01
|   |   |   |   |   |   |   |   |   |   |   |--- value: [18.20, 16.93]
|   |   |   |   |   |   |   |   |   |   |--- sensor0_t9 >  0.01
|   |   |   |   |   |   |   |   |   |   |   |--- value: [16.22, 13.45]
|   |   |   |   |   |   |   |   |   |--- sensor2_t6 >  0.00
```

Decision Tree (video)



Symbolic Regression

- Symbolic regression aims to discover mathematical expressions that best fit a given dataset, capturing both linear and nonlinear relationships among variables
- The expressions derived through symbolic regression are typically composed of a combination of mathematical operators and functions, such as:
  - Arithmetic Operations: addition (+), subtraction (−), multiplication (×), and division (÷)
  - Transcendental Functions: exponential (exp), logarithmic (log), trigonometric functions (sin, cos, tan), among others
  - Power Functions: raising variables to constant or variable exponents
- Symbolic Regression employs **genetic programming (GP)** to discover mathematical expressions that best fit a given dataset

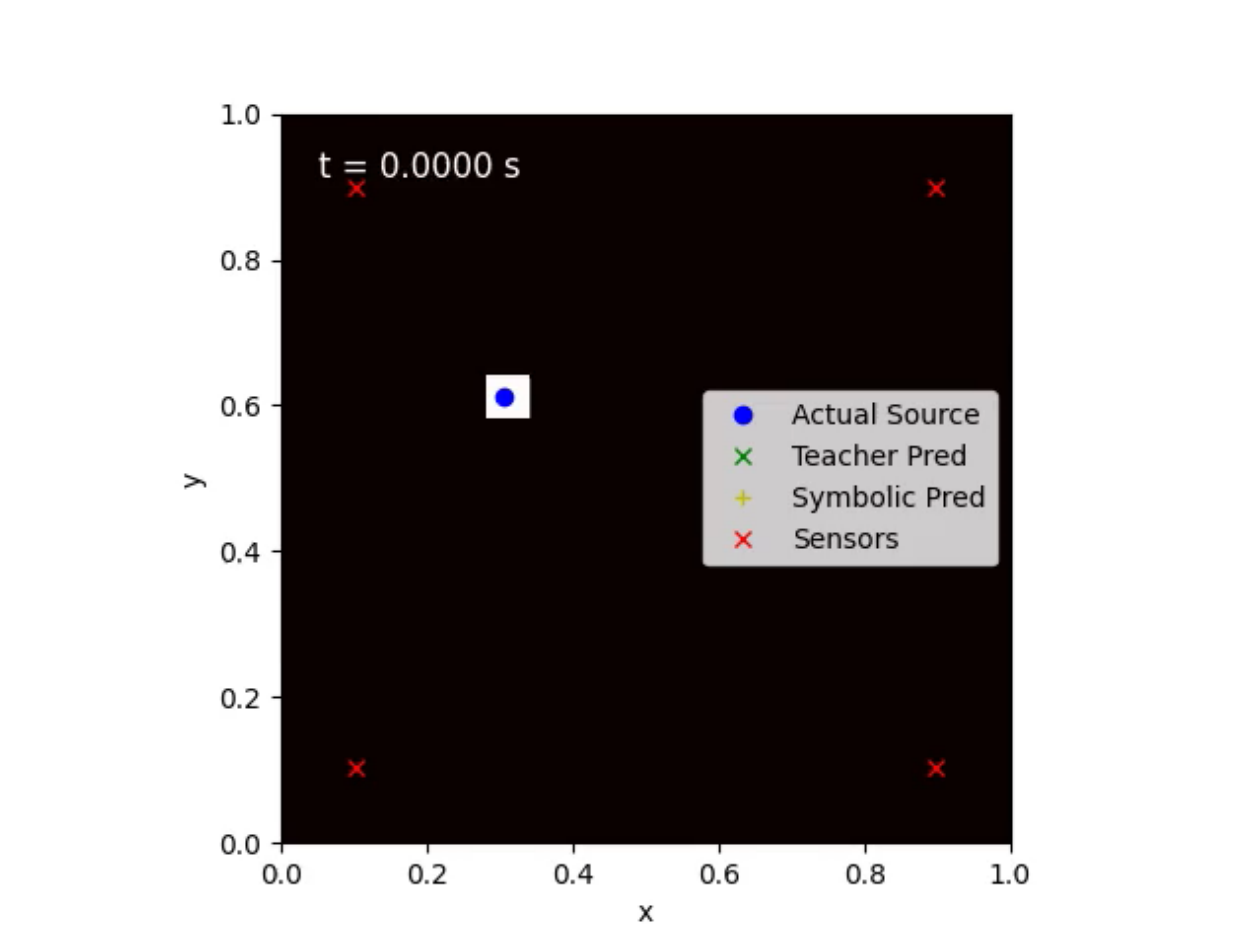
Distilled from initial model:

Formula for X coordinate: -1.8512824*log(x36 + x38 + 0.0036557764) + 30.320702*cos(exp(-0.45548022*x37 - 0.45548022*x39))

Formula for Y coordinate: 23.8731941922807 - 1.839306081316 09*log((x36 + x37 + 0.002661277)/(x38 + x39 + 0.008379506))

This shows that the model only cares about the latest value coming from the sensors, window length is irrelevant

Symbolic Regression (video)



Conclusions

- Uncertainty Quantification (UQ) is essential for trust and robustness, with Bayesian Neural Networks and Aleatoric/Systemic UQ providing deeper insights
- Interpretability methods, such as Symbolic Regression and Decision Trees, enhance model transparency and usability for HPC applications
- While AI surrogates accelerate simulations, model reliability, generalization, and explainability remain critical challenges
- The simpler, the better. The smaller, the better. Multiple small/simple models are usually better than one big/complex model
- Methods presented here are not a must. They are new tools for model validation/calibration and to better understand what the model is doing
- AI surrogate models are not just accelerators but **decision-support tools.** Combining efficiency with trustworthiness is key for real-world deployment

References

- Blundell, C., Cornebise, J., Kavukcuoglu, K., & Wierstra, D. (2015). Weight Uncertainty in Neural Networks
- Depeweg, S., Hernández-Lobato, J. M., Doshi-Velez, F., & Udluft, S. (2017). Sensitivity Analysis for Predictive Uncertainty in Bayesian Neural Networks
- Ribeiro, M. T., Singh, S., & Guestrin, C. (2016). "Why Should I Trust You?": Explaining the Predictions of Any Classifier. Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining
- Koh, P. W., & Liang, P. (2017). Understanding Black-box Predictions via Influence Functions
- Zhou, Y., Xu, P., & Hooker, G. (2024). A Generic Approach for Reproducible Model Distillation. Machine Learning, 113(10), 7645–7688
- Lundberg, S. M., & Lee, S.-I. (2017). A Unified Approach to Interpreting Model Predictions. Advances in Neural Information Processing Systems 30
- Bomarito, G. F., Leser, P. E., Strauss, N. C. M., Garbrecht, K. M., & Hochhalter, J. D. (2022). Automated Learning of Interpretable Models with Quantified Uncertainty
- Brožová, A., Šmídl, V., Tichý, O., & Evangeliou, N. (2025). Spatial-Temporal Source Term Estimation Using Deep Neural Network Prior and Its Application to Chernobyl Wildfires
- Lopez-Ferber, R., Leirens, S., & Georges, D. (2020). Source Term Estimation: Variational Method Versus Machine Learning Applied to Urban Air Pollution. In Proceedings of CSC 2022
- Jimenez Rezende, D. et al. (2014). Stochastic Backpropagation and Approximate Inference in Deep Generative Models. Proceedings of the International Conference on Machine Learning
- Stenen Blakseth, S. et al. (2021). Deep Neural Network Enabled Corrective Source Term Approach to Hybrid Analysis and Modeling
- Rezende, D. J., Mohamed, S., & Wierstra, D. (2014). Variational Autoencoder
- Bingham, E., Chen, J. P., Jankowiak, M., Obermeyer, F., Singh, R., Szerlip, P., Horsfall, B., & Goodman, N. D. (2019). Pyro: Deep Universal Probabilistic Programming.Journal of Machine Learning Research, 20(28), 1–6
- Cranmer, M. (2023). Interpretable Machine Learning for Science with PySR and Symbolic Regression

Disclaimers and Trademark

The information presented in this document is for informational purposes only and may contain technical inaccuracies, omissions, and typographical errors. The information contained herein is subject to change and may be rendered inaccurate for many reasons, including but not limited to product and roadmap changes, component and motherboard version changes, new model and/or product releases, product differences between differing manufacturers, software changes, BIOS flashes, firmware upgrades, or the like.

Any computer system has risks of security vulnerabilities that cannot be completely prevented or mitigated. AMD assumes no obligation to update or otherwise correct or revise this information.

However, AMD reserves the right to revise this information and to make changes from time to time to the content hereof without obligation of AMD to notify any person of such revisions or changes.

THIS INFORMATION IS PROVIDED ‘AS IS.” AMD MAKES NO REPRESENTATIONS OR WARRANTIES WITH RESPECT TO THE CONTENTS HEREOF AND ASSUMES NO RESPONSIBILITY FOR ANY INACCURACIES, ERRORS, OR OMISSIONS THAT MAY APPEAR IN THIS INFORMATION. AMD SPECIFICALLY DISCLAIMS ANY

IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY, OR FITNESS FOR ANY PARTICULAR PURPOSE.

IN NO EVENT WILL AMD BE LIABLE TO ANY PERSON FOR ANY RELIANCE, DIRECT, INDIRECT, SPECIAL, OR OTHER CONSEQUENTIAL DAMAGES ARISING FROM THE USE OF ANY INFORMATION CONTAINED HEREIN, EVEN IF AMD IS EXPRESSLY ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

AMD, the AMD Arrow logo, and combinations thereof are trademarks of Advanced Micro Devices, Inc.

Other product names used in this publication are for identification purposes only and may be trademarks of their respective companies.


AMD