Visualize, Monitor and Control the Training Process of a Deep Surrogate Model in ParaView

François Mazen, Antoine Schieb, Alejandro Ribes, Lucas Meyer

Visualize, Monitor and Control the Training Process of a Deep Surrogate Model in ParaView

Poster · June 2022

CITATIONS
0

READS
143

4 authors, including:

Antoine Schieb
École d'Ingénieurs en Chimie et Sciences du Numérique
1 PUBLICATION **0** CITATIONS
SEE PROFILE

Alejandro Ribes
Électricité de France (EDF)
58 PUBLICATIONS **593** CITATIONS
SEE PROFILE

Visualize, Monitor and Control the training process of a deep surrogate model in ParaView

François Mazen
Kitware Europe
francois.mazen@kitware.com

Antoine Schieb
Kitware Europe, CPE Lyon
antoine.schieb@cpe.fr

Alejandro Ribes
EDF Lab Paris-Saclay
91120 Palaiseau, France
alejandro.ribes@edf.fr

Lucas Meyer
EDF Lab Paris-Saclay, Univ. Grenoble Alpes, Inria, CNRS, Grenoble INP, LIG
lucas.meyer@inria.fr

1.Context and targeted problem

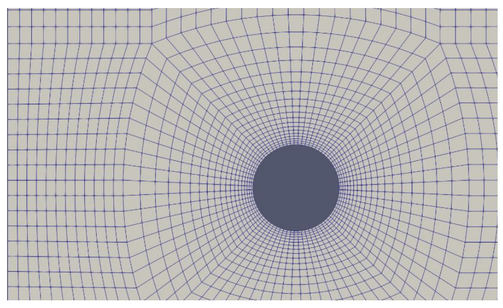
In the context of numerical simulation, a **surrogate model** approximates the outputs of a solver with a low computational cost. Solvers of differential equations, for instance those based on finite elements methods, often require long runs. Thus, they are not well-suited for real-time applications, for prediction or for the resolution of inverse problems requiring multiple executions. Surrogates can be deep-learning models that learn from simulation results and/or from experimental data. **Training** these models can be challenging for several reasons:

- The model often has to operate on an **irregular mesh**, so the architecture has to be thought beforehand and adapted to the mesh
- Intermediate inferences results can be **heavy** to store on disk, and **tedious** to load and visualize.
- Standard metrics like Mean Squared Error do not convey enough information on which aspects of the simulation are **harder to learn**.

To overcome this challenge, Kitware combines ParaView's complex data **visualisation** capabilities with ***In Situ*** data analysis though the Catalyst library[1]. The objective is to be able to flexibly **visualize** intermediary inferences during training, **monitor** the progress, and let the user **control** a set of parameters or hyperparameters.

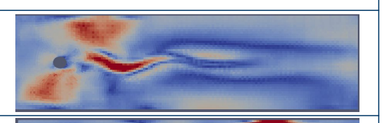
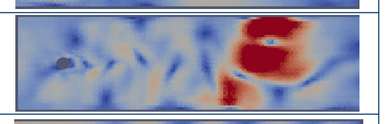
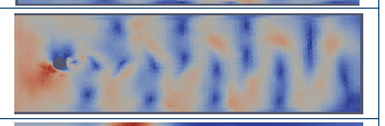
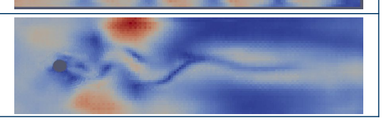
2. Deep surrogate model and training dataset

As a proof of concept, we present results on a simulation of the Von Karman Vortex Street; where a fluid (entering with an **imposed velocity** on the left of a rectangular domain) flows towards a cylindrical obstacle and a set of **vortices** forms behind it. An ensemble of simulations was performed by EDF's CFD solver Code_Saturne. Each member of the ensemble corresponds to a different imposed input velocity.

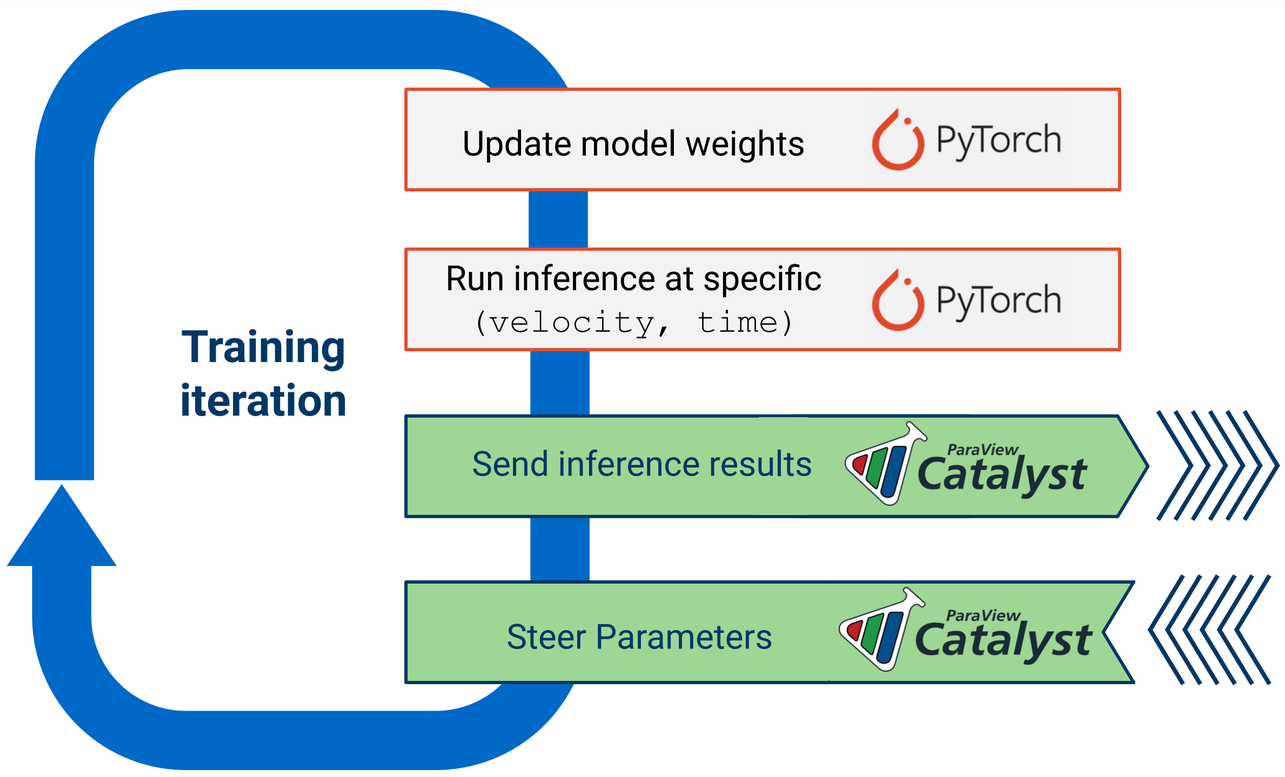


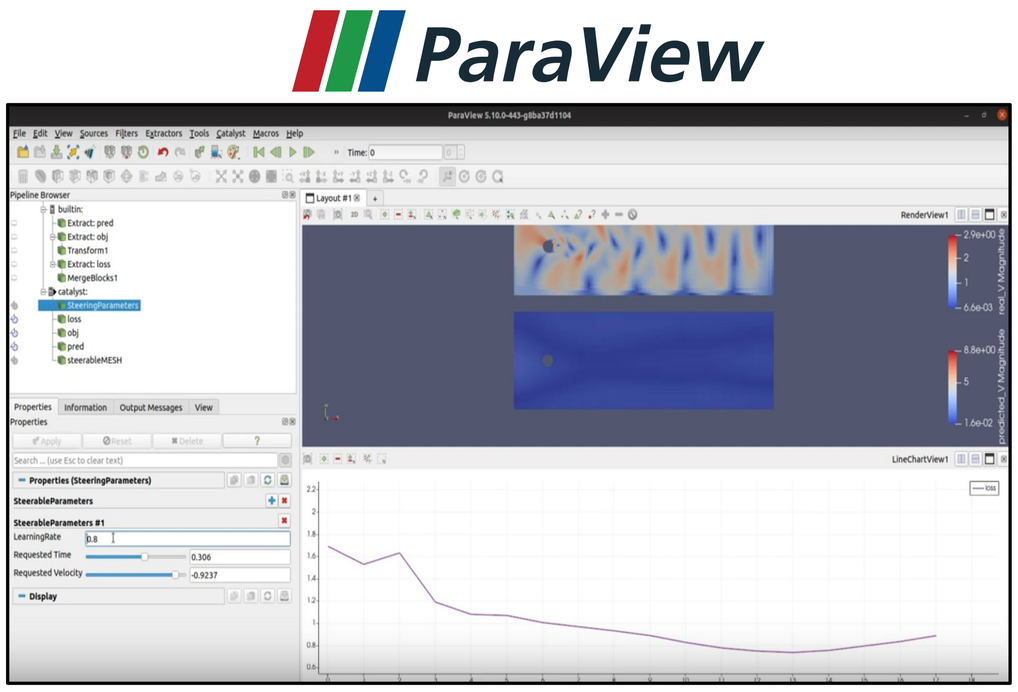
The mesh is **irregular** since we need higher levels of detail around the obstacle.

The Deep Neural Network described in [2] was trained on the ensemble of simulation results by using the **Pytorch**[3] framework. The constructed surrogate model takes two parameters as input, the **imposed velocity** of the fluid and the **time** of the simulation. This surrogate then outputs a discretized velocity vector field.

Input :		Output:
time	velocity	Velocity vector of the fluid at each cell
0.10	2.05	
0.50	2.05	
1.20	2.05	
0.10	1.75	

3. Training, monitoring and steering workflow





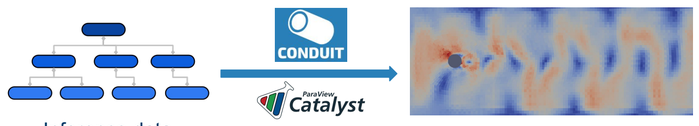
Inference result and loss graph shown in real time

4.Sending inference results to ParaView

ParaView **Catalyst**[1] is an In Situ library that retrieves data located in memory instead of reading it from disk. It lets you :

- **Visualize live** results within the ParaView application
- **Save images** on disk, typically to generate videos of the entire simulation (see section 6)

Here, the aim is to generate an inference at a specific time and send the result with Catalyst so that ParaView can display it in real time on an predefined appropriate mesh.



Inference data in Mesh Blueprint format

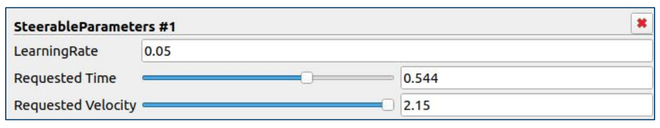
Inference data in VTK format

The data is sent through a Conduit[5] node in the form of a **generic**, hierarchical, and human-readable format called Mesh Blueprint[6] : no need to manually generate Visualization Toolkit (VTK) data structures

5.Parameter steering

Although Catalyst 1 only allowed data to be sent from the simulation to ParaView, the newer release **Catalyst 2** adds the ability to steer the simulation with input from the ParaView GUI. In this case, we have chosen to expose three parameters :

- domain specific parameters → **time** and imposed **velocity** of the inference being displayed
- training hyperparameters → **learning rate**

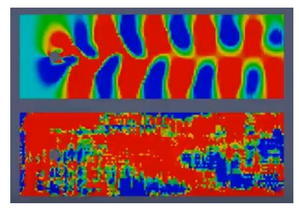


The ParaView GUI offers a convenient way to control the training hyperparameters, and request an inference for a specific `(velocity, time)` tuple.

6. Full training overview

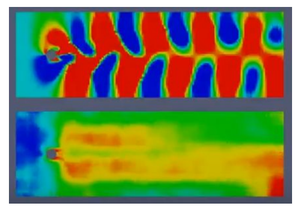
Let's define a **fixed** `(velocity, time)` tuple arbitrarily. For each image :

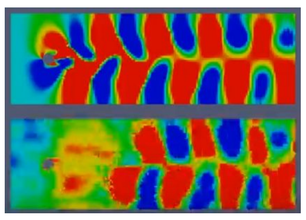
- Top mesh → dataset element corresponding to this tuple
- Bottom mesh → result of the inference of the partially trained model for this tuple.



During the **first** training iterations, not much has been learnt by the model.

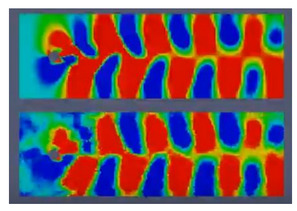
After a few epochs, the model understands that a seemingly continuous **trail** is forming behind the obstacle.





Training continues, and the trail fades out as **vortices** start forming. The vortices at the back, farther away from the obstacle, are **easier** for the model to learn.

After enough time, the model has learnt each vortex and can predict the result of the simulation with **high accuracy**.



7. Added value and perspectives

	Existing tools	ParaView-Catalyst training workflow	Added Value
Visualisation	Basic data structures	Supports more complex structures, custom pipeline, parallel rendering	More powerful, flexible and interactive way to view inference data
Monitoring	Basic metrics for loss or accuracy	Accurate representation of the performance on each dataset element	Better understanding of the behavior of the model
Tuning	Scheduling tools	Steering combined with scheduling	Interactive, real-time feedback, more control

Moving onwards, some **future work** related to this project may include :

- Working on bigger, more complex datasets
- Modeling real-use CFD simulations
- Improvements to the interface between Python and Catalyst
- Using this workflow for reinforcement learning methods

8. References

- [1] ParaView Catalyst : www.paraview.org/in-situ/
- [2] Lucas Meyer, Louen Pottier, Alejandro Ribes, Bruno Raffin : "*Deep Surrogate for Direct Time Fluid Dynamics*" (NeurIPS 2021 - Thirty-fifth Workshop on Machine Learning and the Physical Sciences) arXiv:2112.10296 [cs.LG]
- [3] Pytorch : pytorch.org
- [4] Integrating Geometric Deep-Learning models into Paraview : www.kitware.com/integrating-geometric-deep-learning-models-into-paraview/
- [5] Conduit documentation : llnl-conduit.readthedocs.io/en/latest/conduit.html
- [6] Mesh Blueprint Format : llnl-conduit.readthedocs.io/en/latest/blueprint_mesh.html