ASTERIX

Adaptive Strategies Towards Expedient Recovery In eXascale

FAISER

Fast AI-based Space Environment Prediction

INNO4 SCALE

HELSINGIN YLIOPISTO

Research Council of Finland

Team

Participating here at Brussels:

- Kostis Papadakis [ASTERIX]
- Ioanna Bouri [FAISER]

- Juhani Kataja [ASTERIX] present at Brussels (but hacking away with the Elmer team).
- Markku Alho [ASTERIX] available remotely.

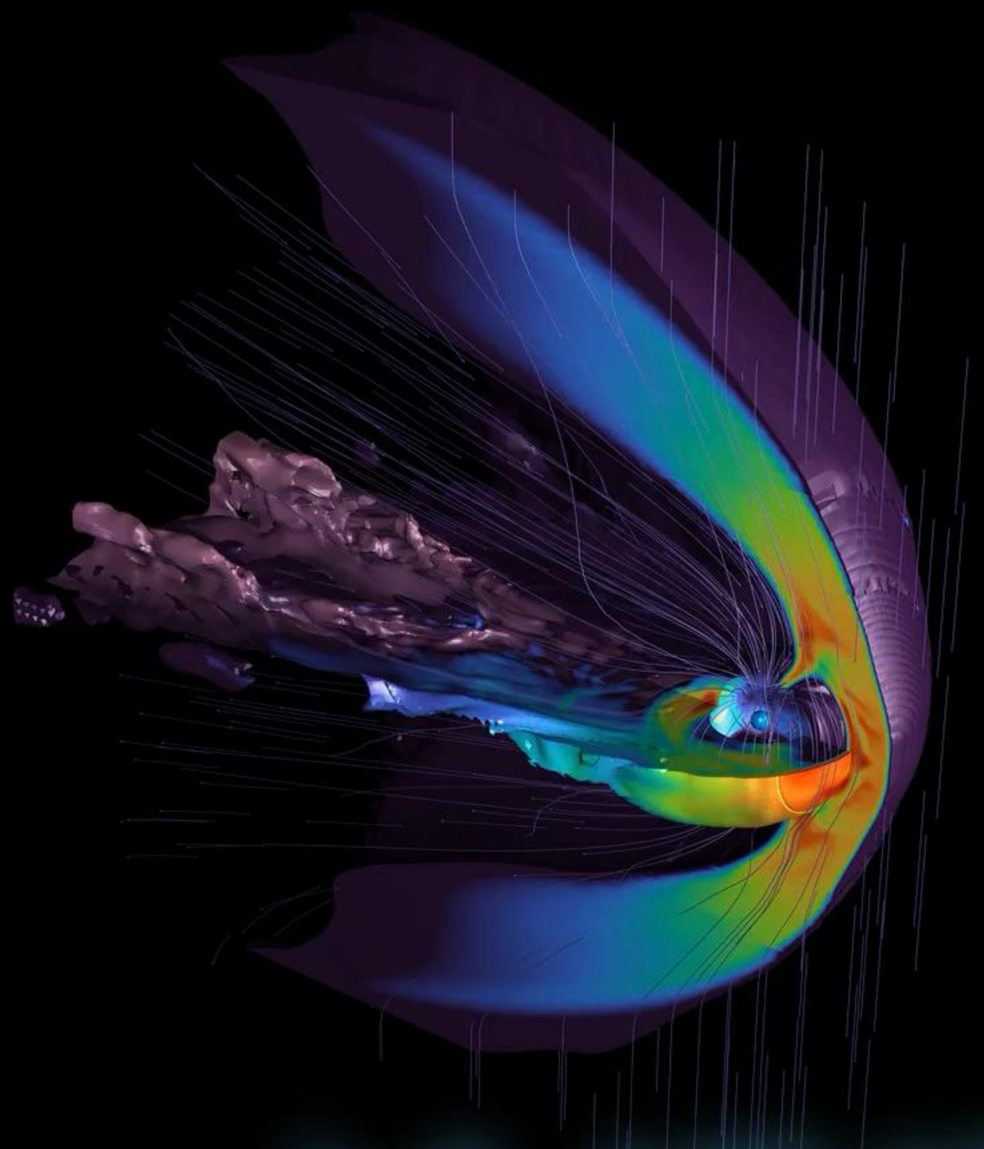
PIs:

- ASTERIX & Vlasiator & FAISER PI Minna Palmroth
- FAISER Co-I & CosCo PI Teemu Roos
- CSC ASTERIX Co-I Jussi Heikonen

Vlasiator

- 6D Global Hybrid Vlasov model that simulates the near-Earth space.
- 3D Velocity Distribution Functions (VDF) at every simulation cell.
- Massive resources for 6D simulations. Current runs are performed on 500 LUMI nodes.
- ~15-20 MCPUh or more for productions runs.
- Checkpoint mechanism for resilient restarting.
- Checkpoint files are 5-7 TiB for 3D production runs.
- Cannot have frequent restart files lying on disk.
- Fewer checkpoint files leads to wasted computational effort.

INNO4 SCALE HELSINGIN YLIOPISTO Research Council of Finland CSC



VLASIATOR

Vlasiator source code & details

- Vlasiator is written in modern C++.
- Parallelized over ~all available levels
  - MPI (domain decomposition, Zoltan)
  - OpenMP
  - Vectorization
- Currently our GPU branch with support for NVIDIA/AMD hardware is under development.
- Source code is hosted on GitHub at https://github.com/fmihpc/vlasiator.

- Solves the Vlasov equation of ion species:
  - 6D distribution (3D space, 3D velocity) propagated in time by shear transformations
  - Spatial AMR
  - Sparse representation of velocity space per spatial cell for memory efficiency (95% reduced memory footprint)

INNO4 SCALE
HELSINGIN YLIOPISTO
Research Council of Finland
CSC

Goals of Projects

ASTERIX

Compression of Vlasiator VDF data (with ML/AI methods, on GPUs) so that:

1) Compression is physically sensible (recovery from lossy restart)
2) Compression is fast enough (so that we can store lossy restarts often)

FAISER

1) 6D compressed/latent-space presentation of VDFs
2) Offloading Vlasov eq. propagation to happen solely in the latent space to obtain a fast solver
3) Bootstrap an AI "forecast model" with training data from fast simulations via 2)

Status

ASTERIX

Prototyping resulted in using Multilayer Perceptrons (MLPs) with Fourier features with Octree compression as a fallback; inconclusive results for vector-quantized variational autoencoder (VQ-VAE)

- MLP compression implemented in Vlasiator.
- GPU version of MLP under development.
- Octree method to be implemented.
- VQ-VAE inference model under development.

FAISER

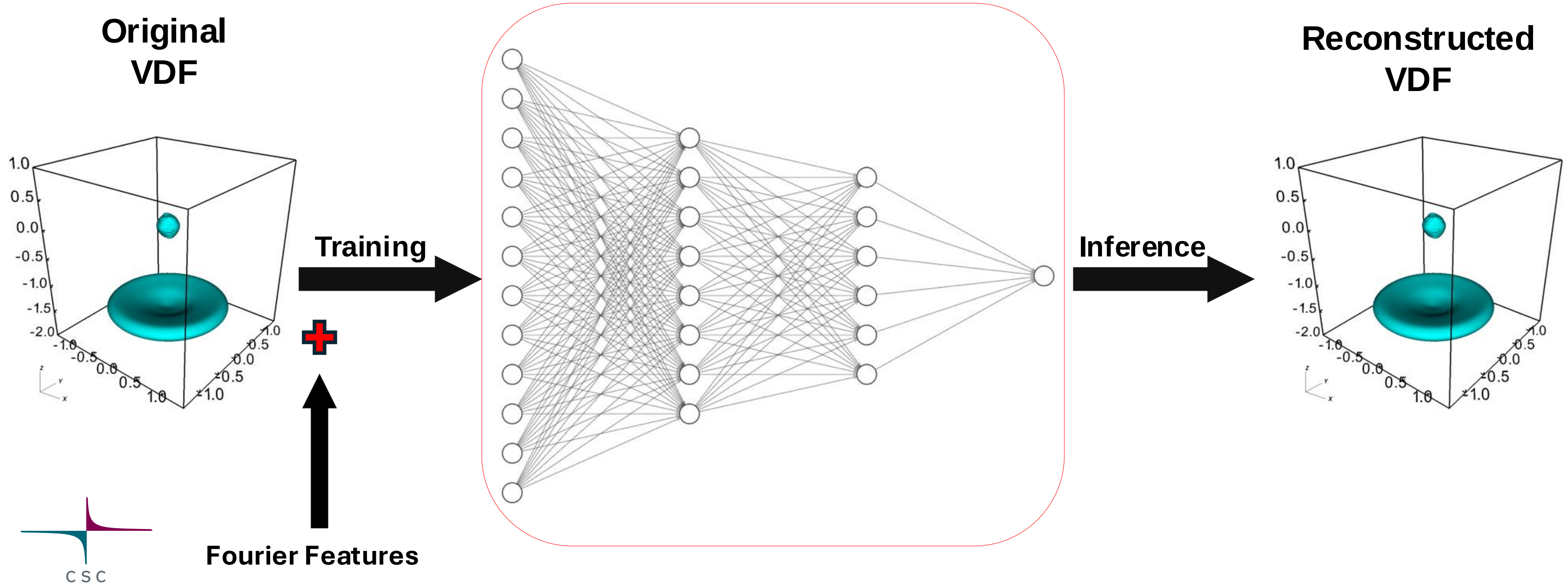
Goals align with using autoencoders such as VQ-VAE, but see above

- RCF project online from 1st of September 2024
- Building on ASTERIX

INNO4 SCALE HELSINGIN YLIOPISTO Research Council of Finland CSC

Using Neural Networks to compress VDFs [ASTERIX MLP]

- We are developing a Multilayer Perceptron to train on our VDFs during runtime.
- Fourier features encoding of input space.
- Network weights are stored instead of the original VDF voxel mesh.
- Network weights are updated at regular simulation intervals.
- MLP is trained on a subsampled version of the VDF to speed up training.
- VDF is recovered via inference.

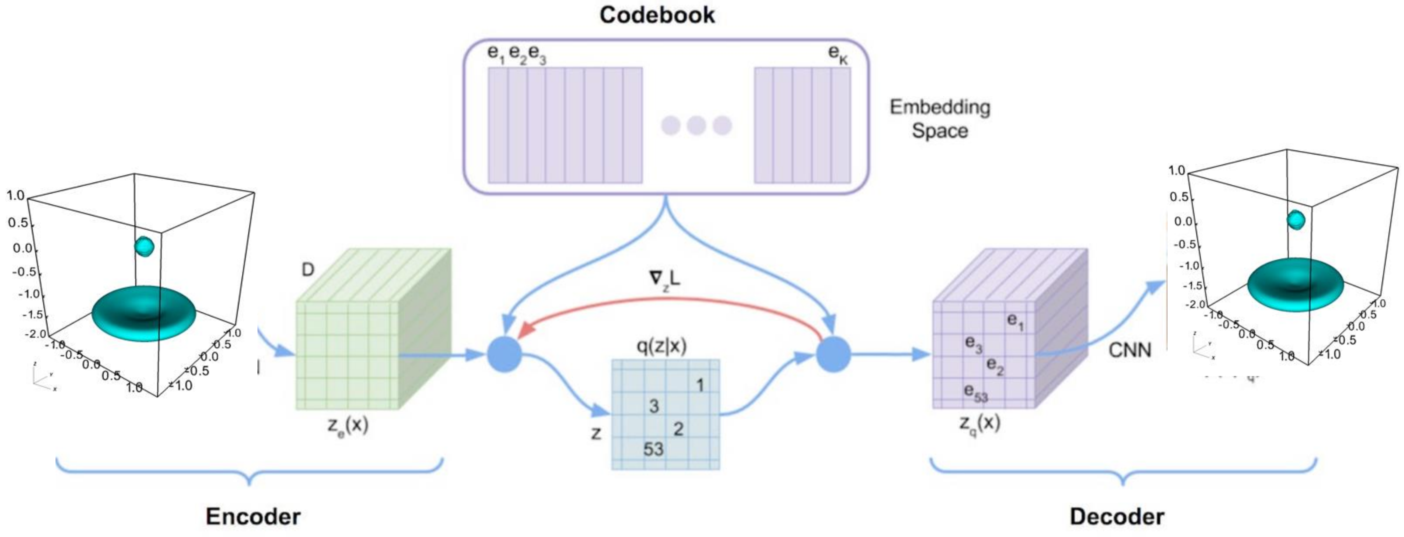


Using VQ-VAEs to compress VDFs [ASTERIX, FAISER]

- Objective: scaling a 3D VQ-VAE to train on existing restart files.
- Objective: extend the 3D VQ-VAE for time-dependent 6D data

- VQ-VAEs learn a discrete latent space representation of the input by incorporating a vector quantization (VQ) module at the bottleneck.
- Motivation: We developed a 3D VQ-VAE that can provide effective compressed state representations leveraging the sparsity of the input VDFs.

- The 3D VQ-VAE is a PyTorch DDP implementation.



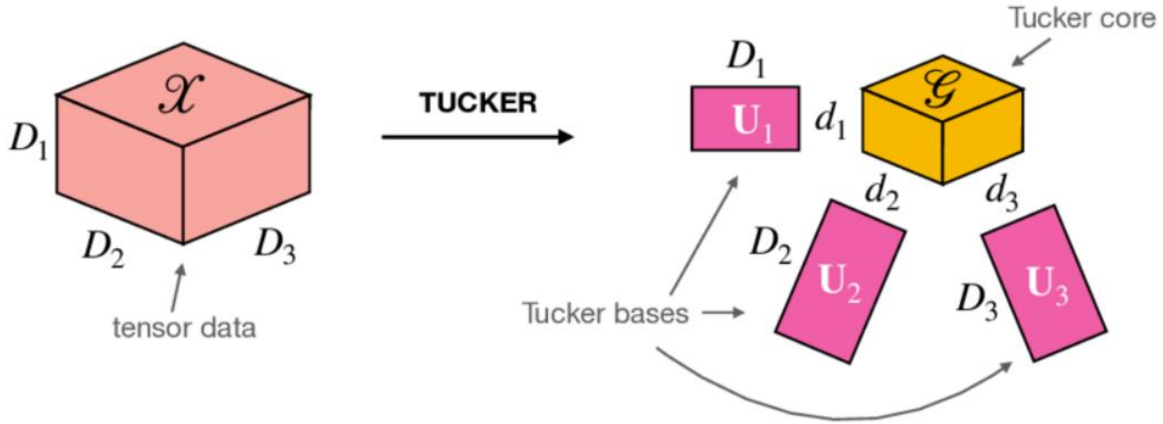
Adapted from: Neural Discrete Representation Learning [van den Oord et al.]

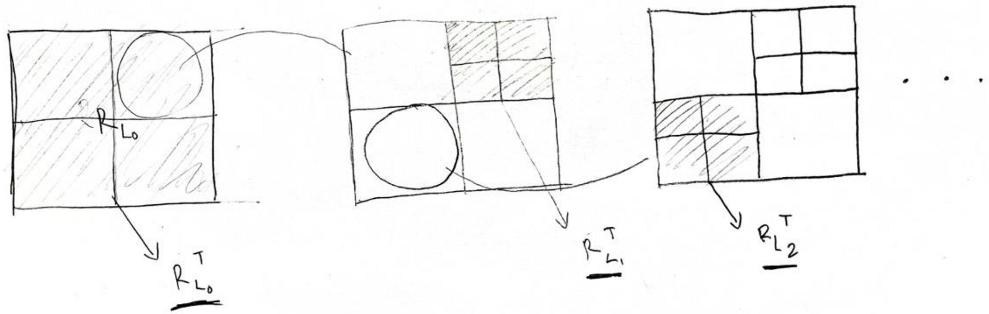
INNO4SCALE
HELSINGIN YLIOPISTO
Research Council of Finland
CSC

Multi-res Octree-Tucker Approximation of Gridded Data

... in case ML/AI methods don't work...

- Efficient compression and approximation of large 3D gridded data.
- Adaptive subdivision of the data into smaller cubical regions (leaf).
- Tensor factorization within cube with biggest residual using Tucker decomposition.
- Store factorizations and corresponding cube information.



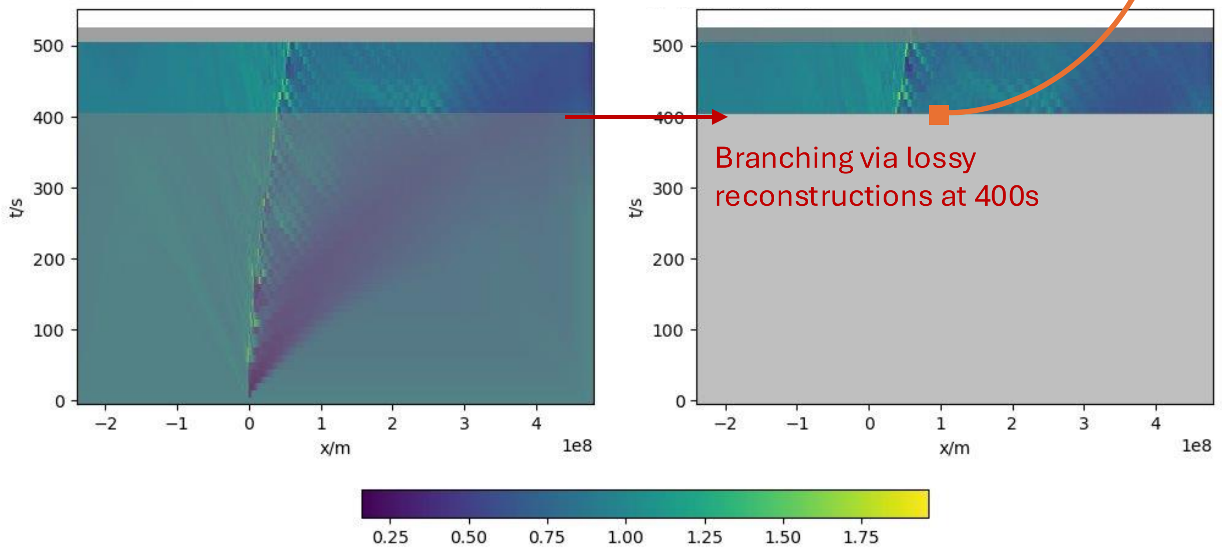


INNO4 SCALE
HELSINGIN YLIOPISTO
Research Council of Finland
CSC

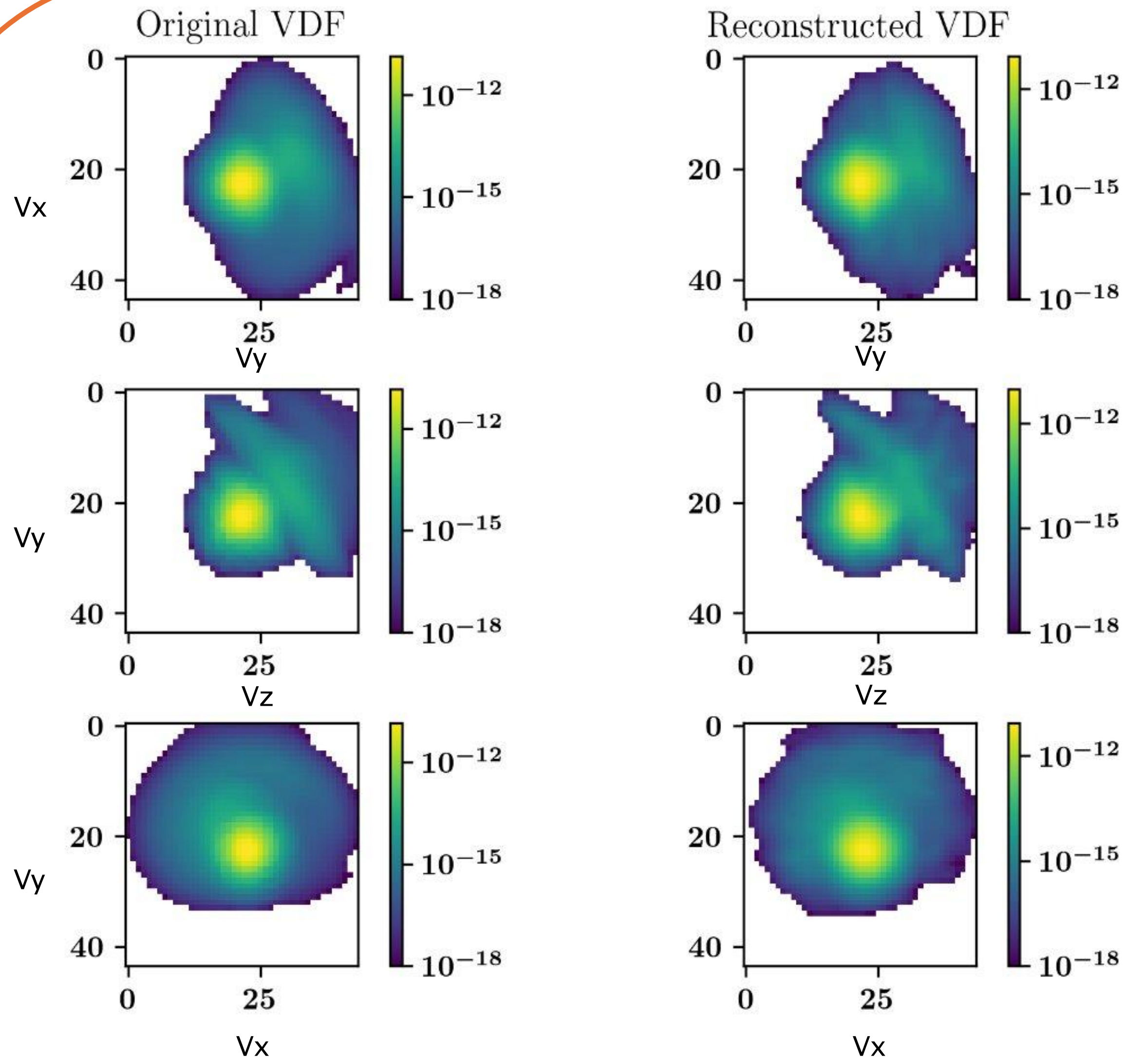
Restarting Vlasiator from a compressed state - proof of concept

1D shock tube simulation as a test case:
- develops nontrivial VDFs
- Dynamics dependent on reconstruction quality

Evolution of temperature anisotropy in a shock test run.



Single VDF from the shock run compressed 83 times



Hackathon goals

- Task 1: Scaling, tuning and training the 3D VQ-VAE prototype (ASTERIX)
- Task 2: Vlasiator restart data compression to GPUs (ASTERIX)
- Task 3: Scaling, tuning and training of the 6D VQ-VAE with temporal propagation (FAISER)
- Task 4: Vlasiator runtime hooks / online training of 6D VQ-VAE (FAISER)

Task 1) 3D VQ-VAE training at scale

- Problematic scaling so far. Implemented on PyTorch DDP.
- Not able to run on more than 10 LUMI-G nodes. Situation post LUMI-update?

- Goal: Training run on Vlasiator restart file(s), á 5 TiB

- Current implementation:
- Custom dataloader w/ cached disk reads
- https://github.com/kstppd/asterix/tree/vdf_replace/src/assessment/vqvae
- PyTorch support on AMD hardware & over MPI?
- Dataloader optimizations? Dep. on analysator for reading in data

INNO4 SCALE HELSINGIN YLIOPISTO Research Council of Finland CSC

Task 2) Enable GPU training for MLP

- Use GPU partition to train the MLP during Vlasiator runtime.
- Use stream events to synchronize training.
- Evaluate performance against CPU training, test scaling.

Currently:

- Prototype running on CUDA, HIP-supported.
- Written in modern C++.
- Custom implementation using a Matrix class that supports CUblas/HIPblas operations.

Task 3) 6D VQ-VAE

- Toy-model dataset to be produced for hackathon
- Prototype autoencoder to be produced for hackathon

- Task: Expand Task 1 to enable training on the 6D dataset and with 6D autoencoders – at least at “small scale” without online training
- Will require more involved dataloaders, optimization, tuning...

Task 4) Runtime hooks in Vlasiator

- Interface to expose VDF data from Vlasiator runtime to training process. Below are the current ideas for approaching the topic

- Potentially implement a client server interface in Vlasiator.
- Split communicator probably using MPDP.
- Use MPI's dynamic process management.
- Create a synchronization scheme between communicators.
- Query VDFs and send them to parent communicator for online training?

- To be worked/discussed on as time allows during the hackathon.

Restarting Vlasiator from a compressed state extra example

Growth Rate of a 2D Kelvin Helmholtz Instability in Vlasiator.