Learning Richtmyer-Meshkov Instability Fields from Parametrized Hydrodynamic Simulations

JOWOG 34 ACS

Charles F. Jekel, Dane M. Sterbentz, Sylvie Aubry, Youngsoo Choi, Daniel A. White, Jon L. Belof

This work was supported by the LLNL-LDRD Program under Project No. LDRD 21-SI-006.

February 28 – March 4, 2022

This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under contract DE-AC52-07NA27344. Lawrence Livermore National Security, LLC

Lawrence Livermore National Laboratory

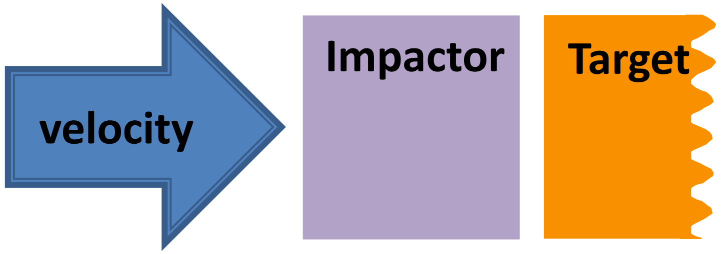
What are Richtmyer-Meshkov or Rayleigh-Taylor instabilities?

- Rayleigh–Taylor instability occurs at an interface of two different densities [2]
  - Water suspended above oil
- Richtmyer-Meshkov Instability (RMI) is impulsively accelerated
  - Two substances with different density
  - Some initial small perturbation between materials
  - Shock wave through interface causes large “jet-like” growths
  - Various importance and interest (e.g. ICF at NIF [1] [3])
- The Darkstar SI seeks to ‘control’ RMI (PI Jon Belof)
  - State of the art experiments and computations
  - **Machine Learning to predict RMI**

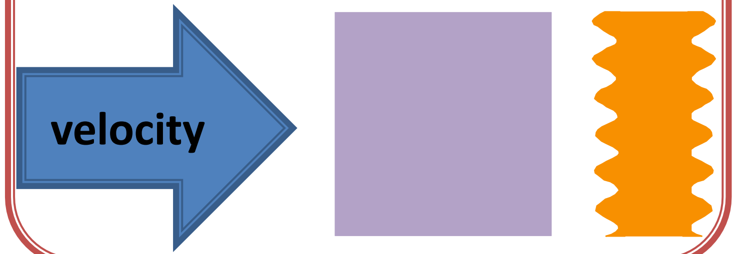
Snapshots of density in time increments of 0.1µs from left to right as an RMI forms.

Various Impact experiments to design for RMI

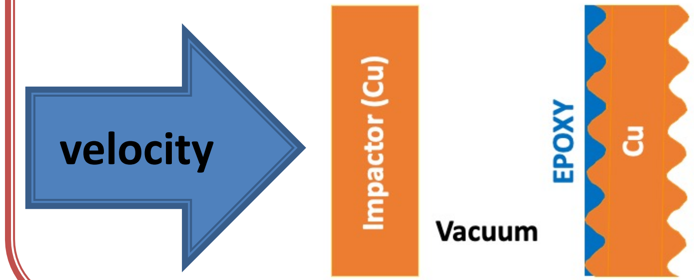
- Seeking designs that maximize RMI
- Also attempting to mitigate known RMI



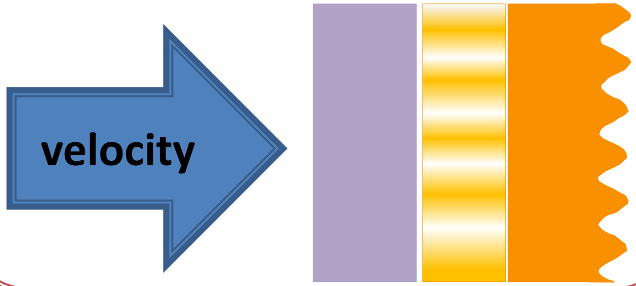
Double wave targets



Double wave targets

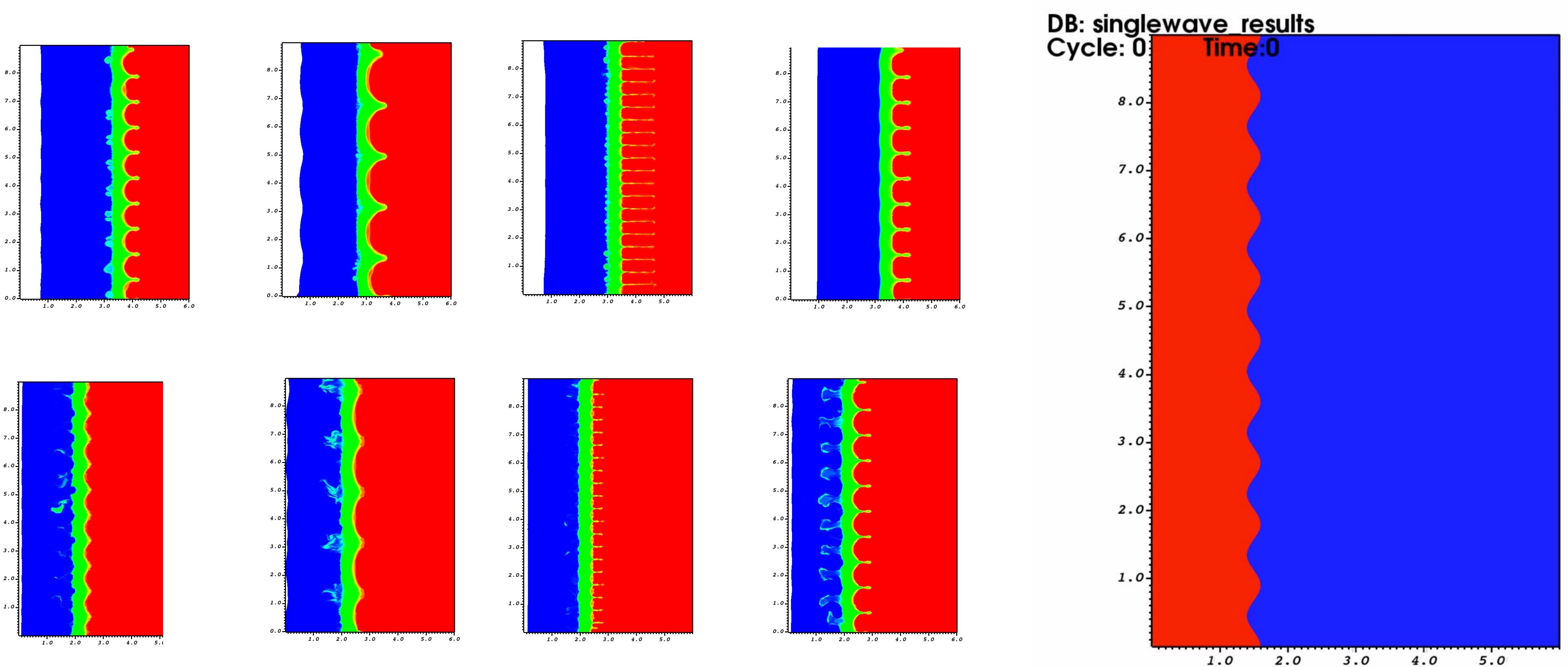


Graded buffer

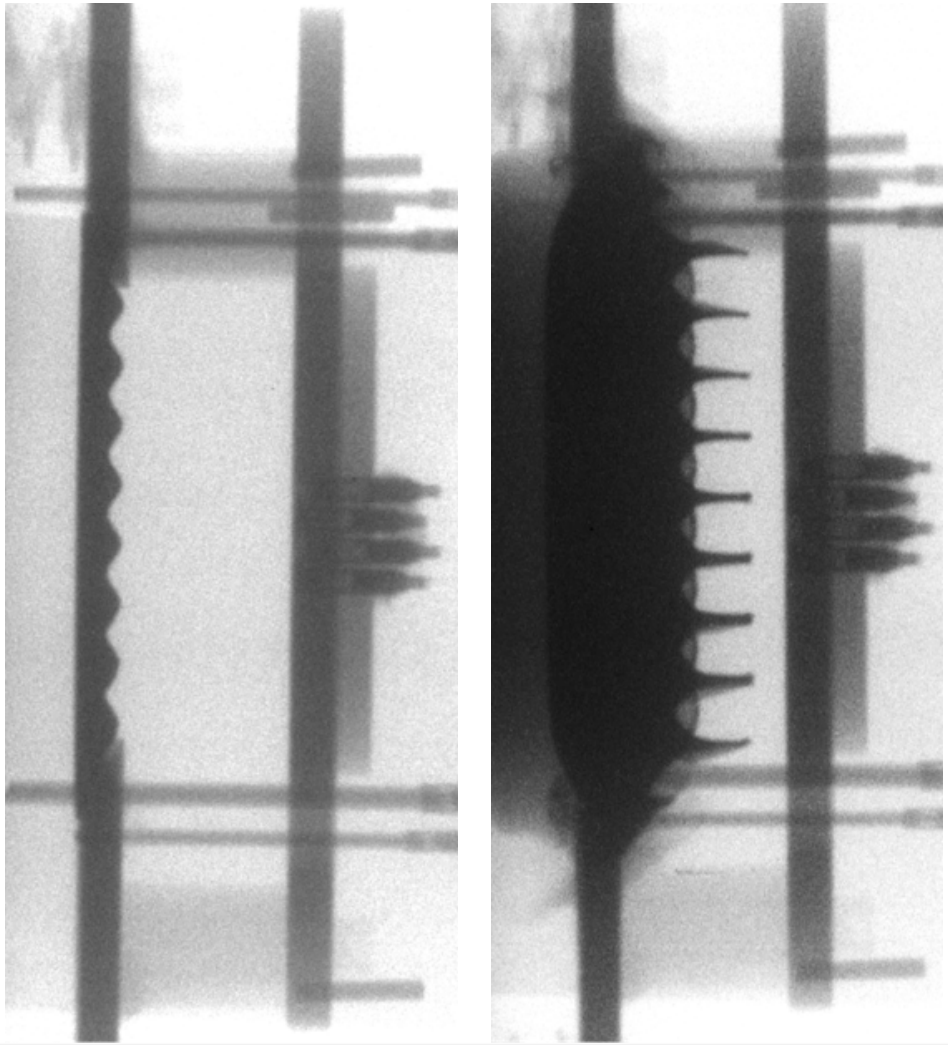


Simulated RMI at the same impact velocity

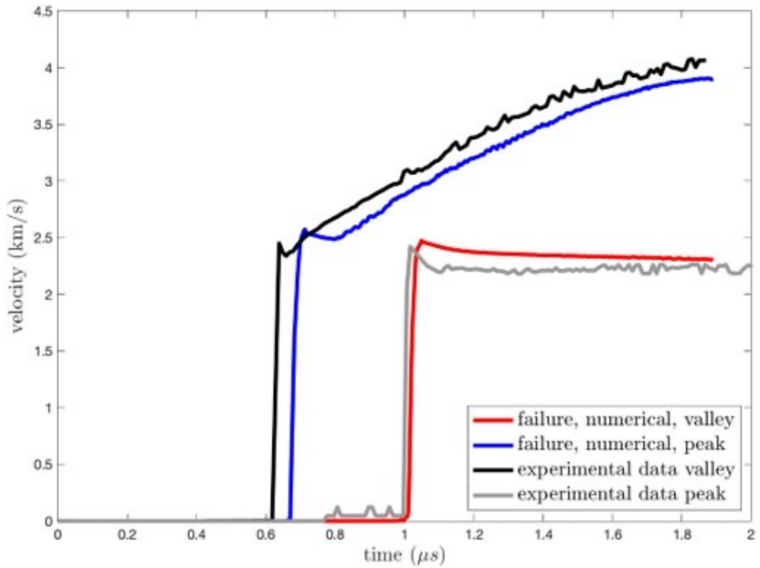
Changing impact materials and initial amplitude



How well do simulations agree with experiments?



- HEAF gas gun experiments
  - 9cm diameter
  - Hector Lorenzana, Jeff Nguyen, Mike Armstrong

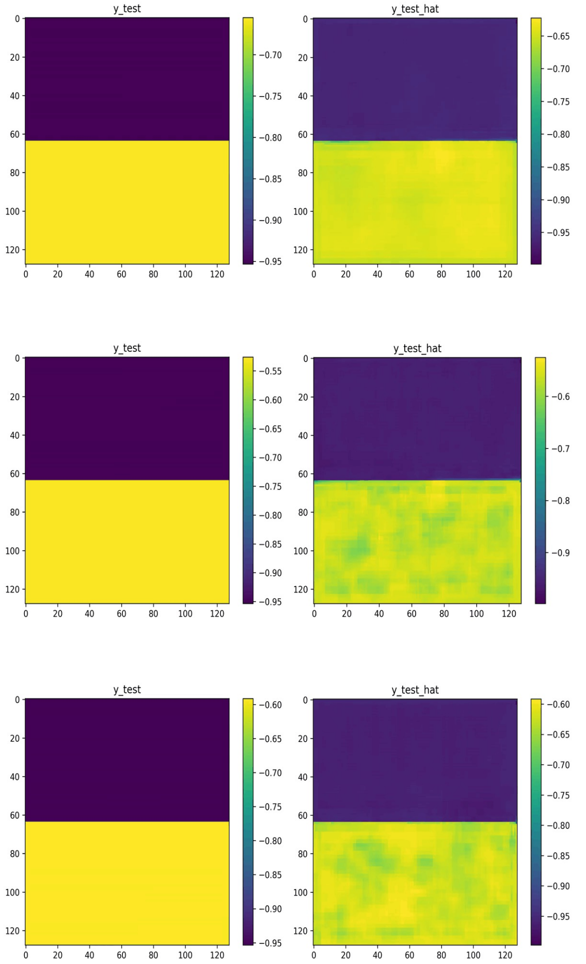


Comparison with sinusoidal wave.

Previous work to model Rayleigh–Taylor instability

- Generator portion of DCGAN model [4]
  - Fake celebrity faces
  - Trained in Regression
    - Not using GAN or Auto Encoder
    - **Thomas Stitt** and **Dan White**
- Prediction of Rayleigh–Taylor instability
  - 2 parameter input
  - 128 x 128 'images'

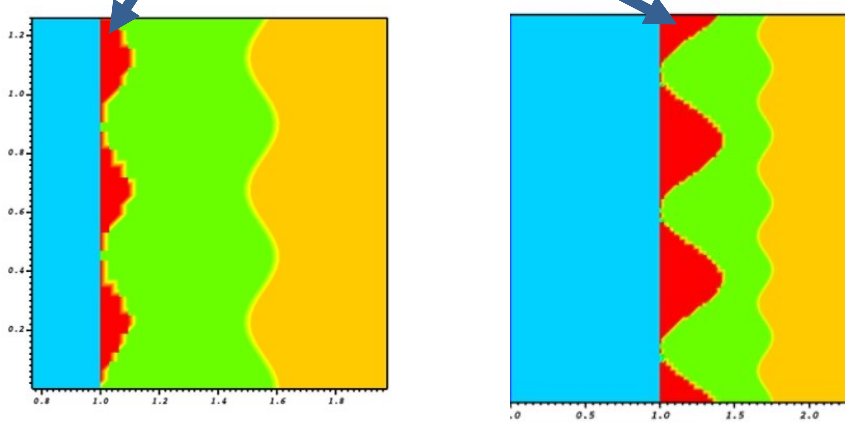
Fake celebrity images from
https://pytorch.org/tutorials/beginner/dcgan_faces_tutorial.html



Left MARBL Simulation, Right ML prediction

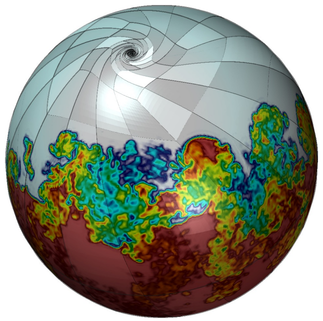
A parameterized simulation to study RMI

- 3 parameters to change
  - Changes impactor side front
  - B, Q, S



$$x = B\cos\left(\frac{2\pi Q y}{9} - s\pi\right)$$

- Machine learning ready tools!
  - MARBL
    - ALE Hydrodynamics
  - Ascent
    - Fast ray tracing ‘images’
  - Merlin
    - HPC workflow management



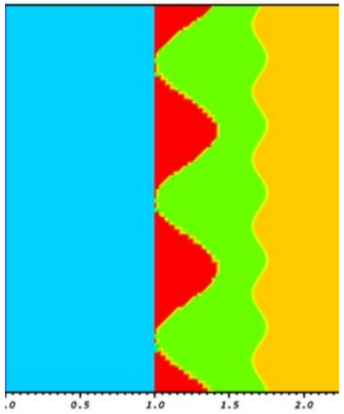
Merlin

Ascent

Machine learning model overview

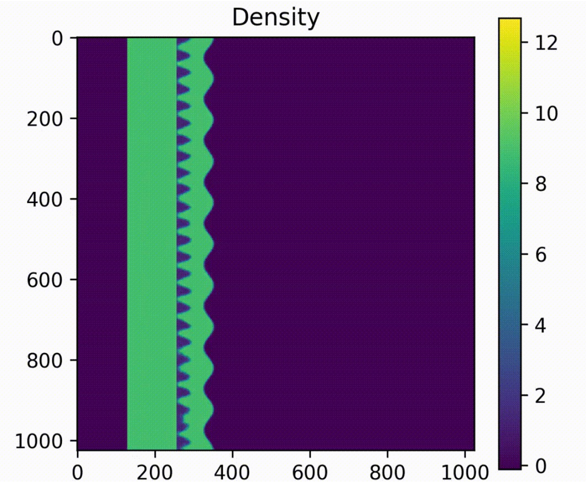
- Model predicts full RMI formation
  - **Input**: Initial conditions
  - **Output**: Full field response
- Why do this?
  - Use ML model to quickly explore designs
  - Optimization on the ML model is fast

3 input parameters
defining initial conditions



Machine
Learning Model

Entire time dependent
density field prediction



Machine learning dataset at a glance

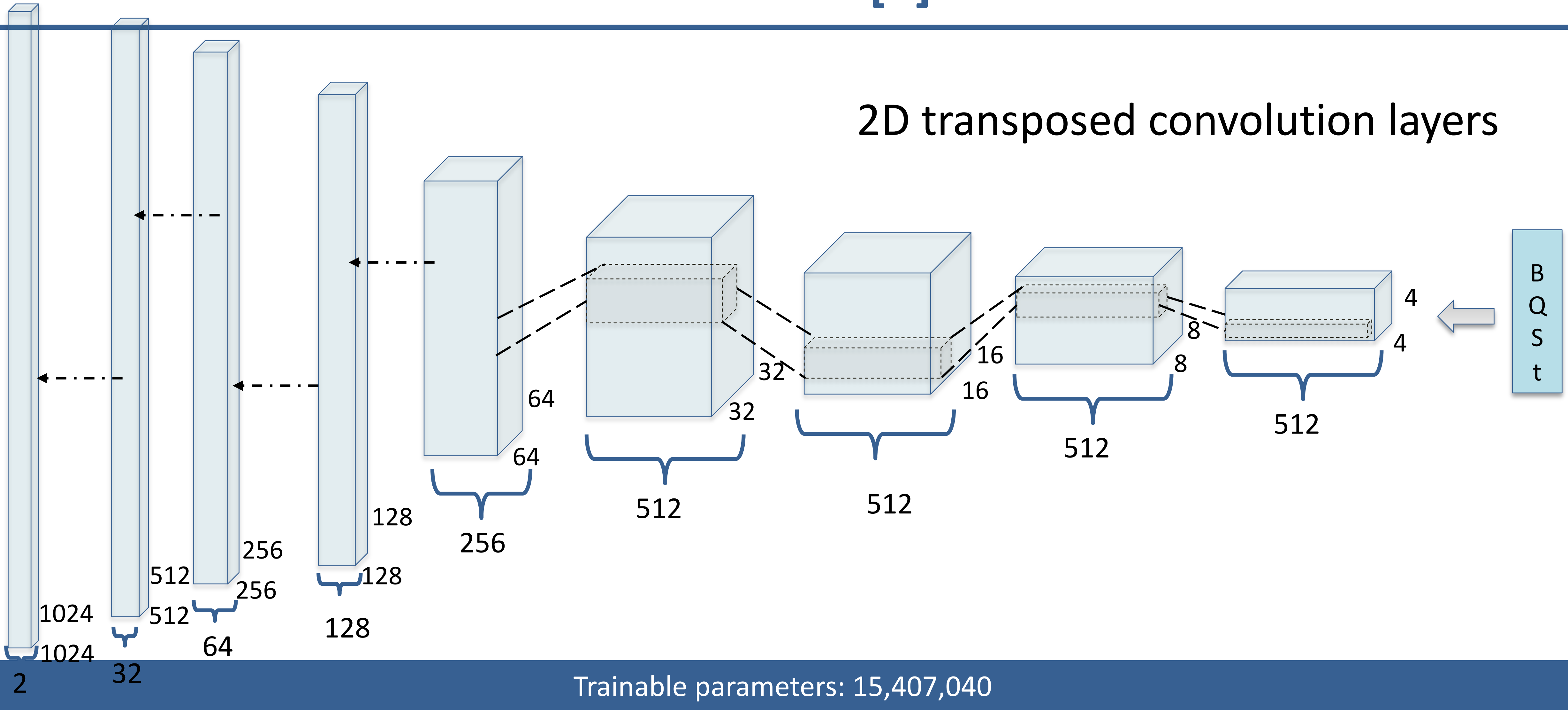
- For the three parameter study
  - 1,600 simulations
    - 30 hours with 20 Lassen/Sierra nodes
  - 51 times steps per simulation
  - 5 output fields
    - Density
    - Velocity X & Y
    - Energy
    - Materials
  - 1024 x 1024 "pixels"
  - 427,819,008,000 single precision floats
- Larger studies in the works
  - More parameters
  - More complicated physics

```
12G     dataset_000.h5
12G     dataset_001.h5
12G     dataset_002.h5
12G     dataset_003.h5
12G     dataset_004.h5
12G     dataset_005.h5
12G     dataset_006.h5
12G     dataset_007.h5
12G     dataset_008.h5
12G     dataset_009.h5
12G     dataset_010.h5
12G     dataset_011.h5
12G     dataset_012.h5
12G     dataset_013.h5
12G     dataset_014.h5
11G     dataset_015.h5
11G     dataset_016.h5
11G     dataset_017.h5
11G     dataset_018.h5
11G     dataset_019.h5
11G     dataset_020.h5
```

143 - 12 GB h5 files

The ML model in this work
See ‘Generator’ model from DCGAN [4]

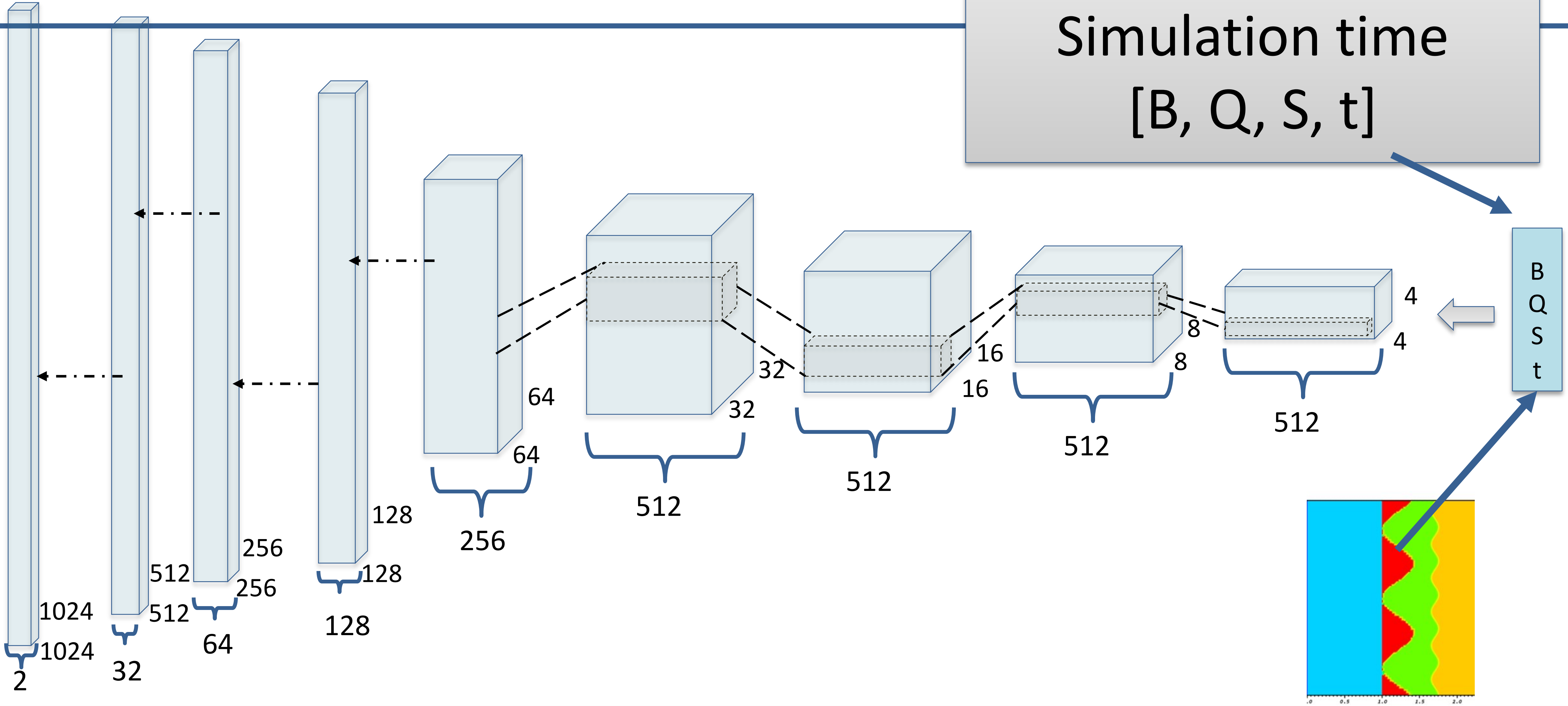
2D transposed convolution layers



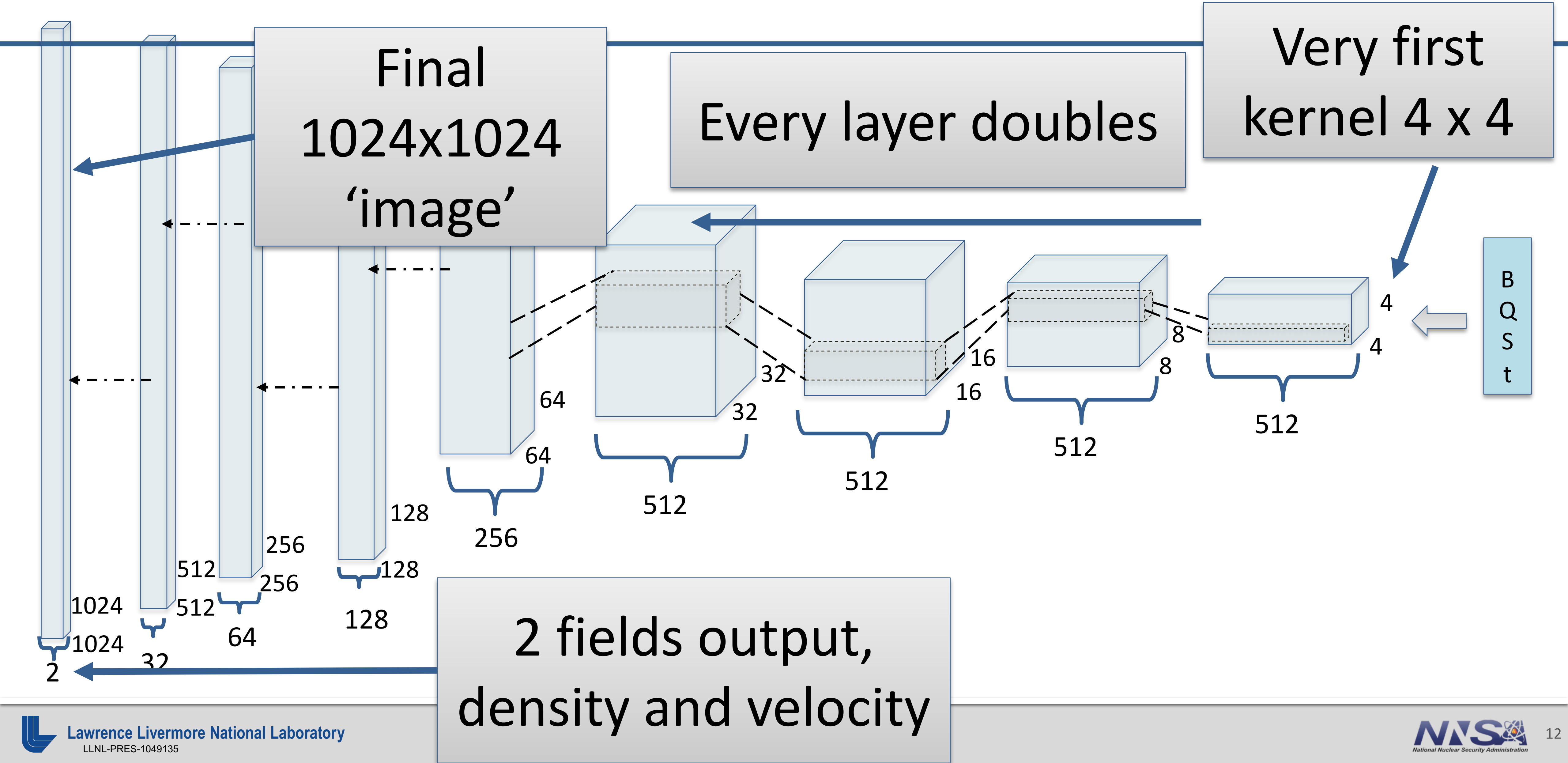
Trainable parameters: 15,407,040

What's the input?

3 Parameters +
Simulation time
[B, Q, S, t]

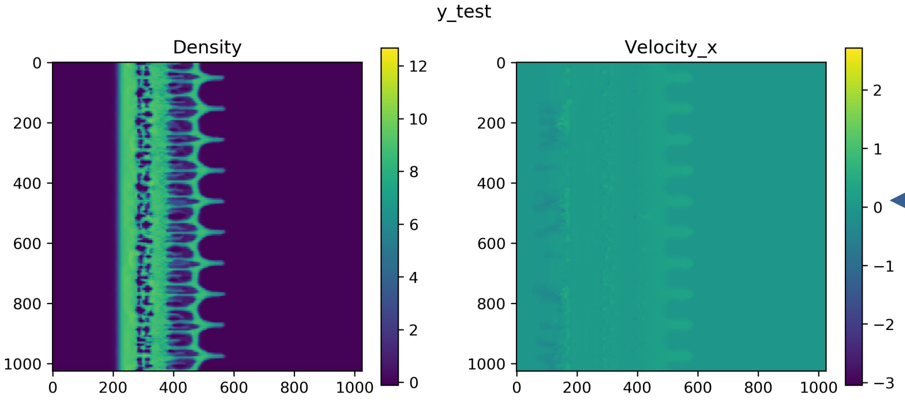


Layer by layer progression

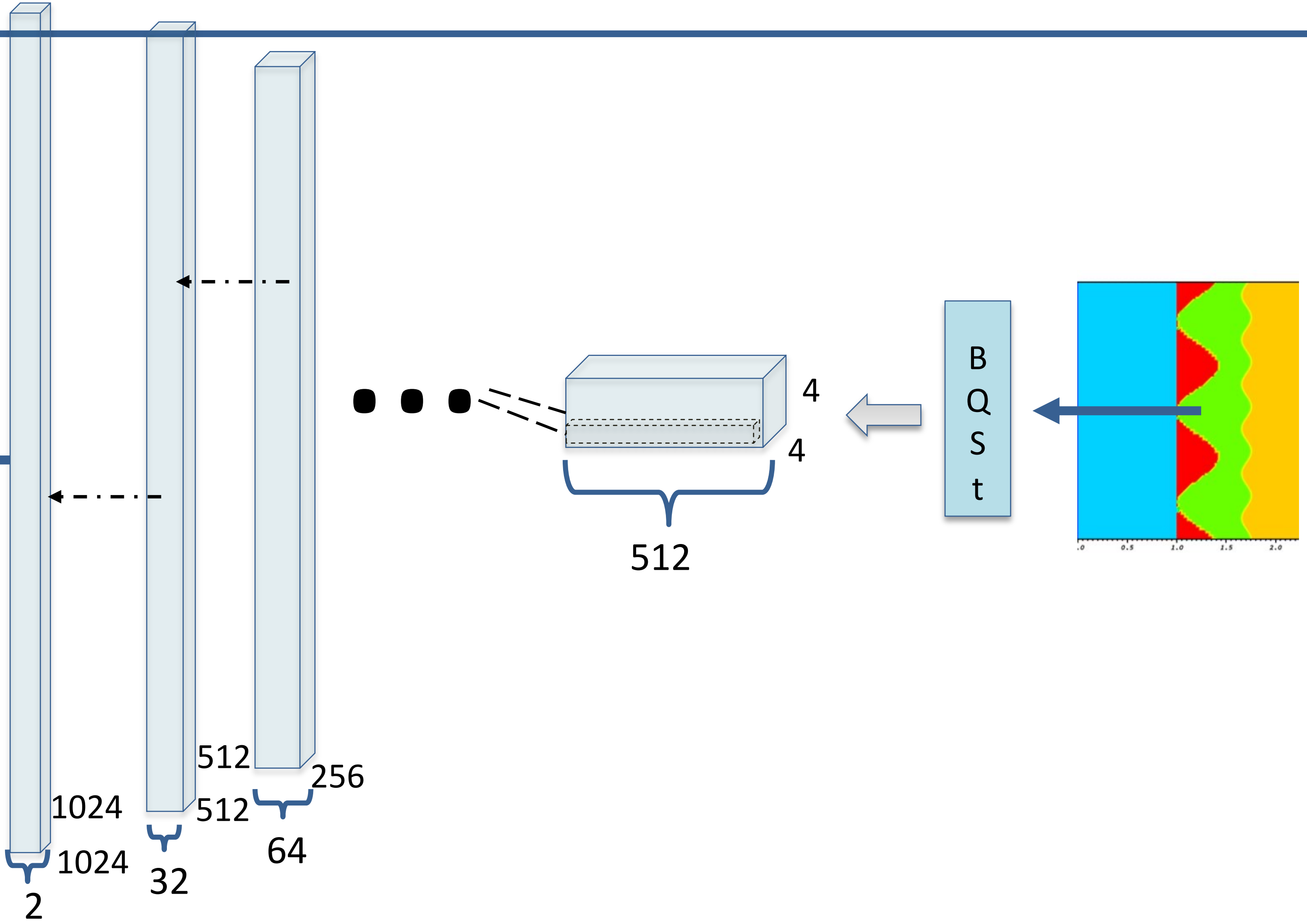


Input and Output

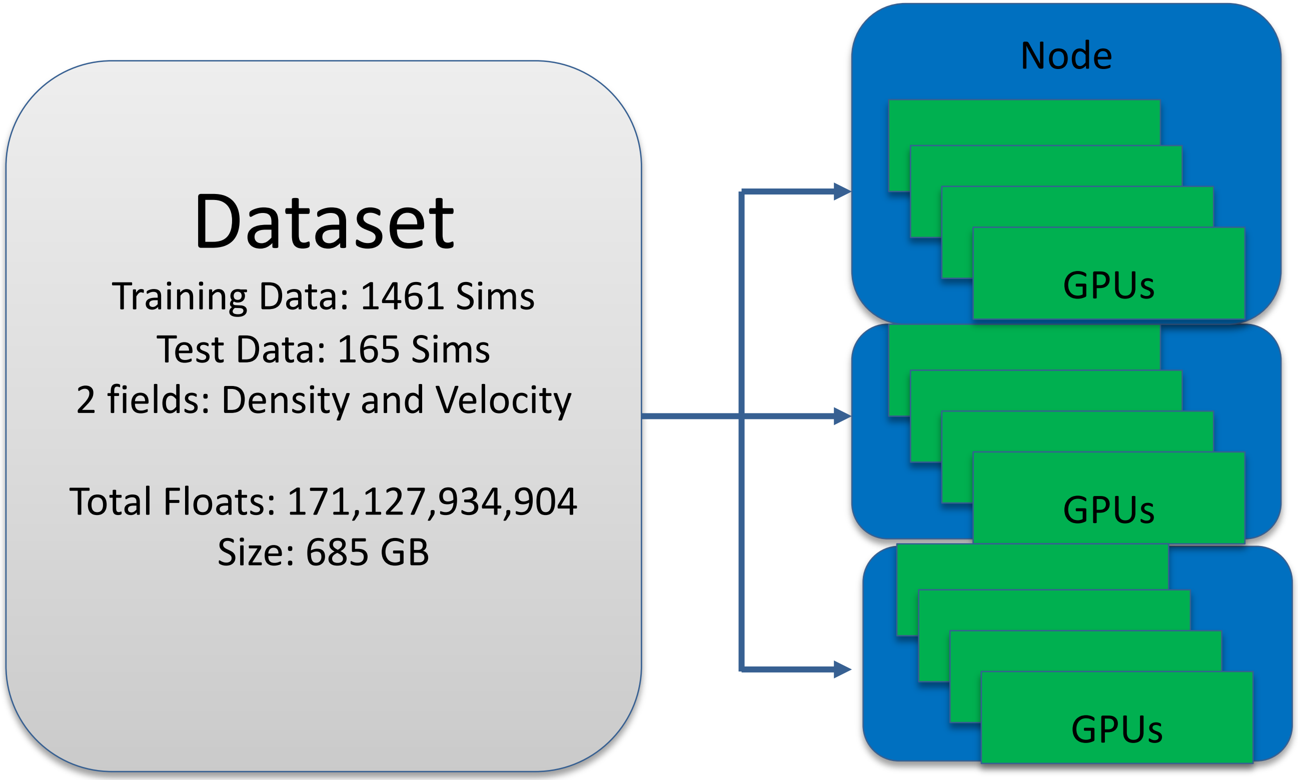
Density and Velocity
at time = t



2 x 1024 x 1024



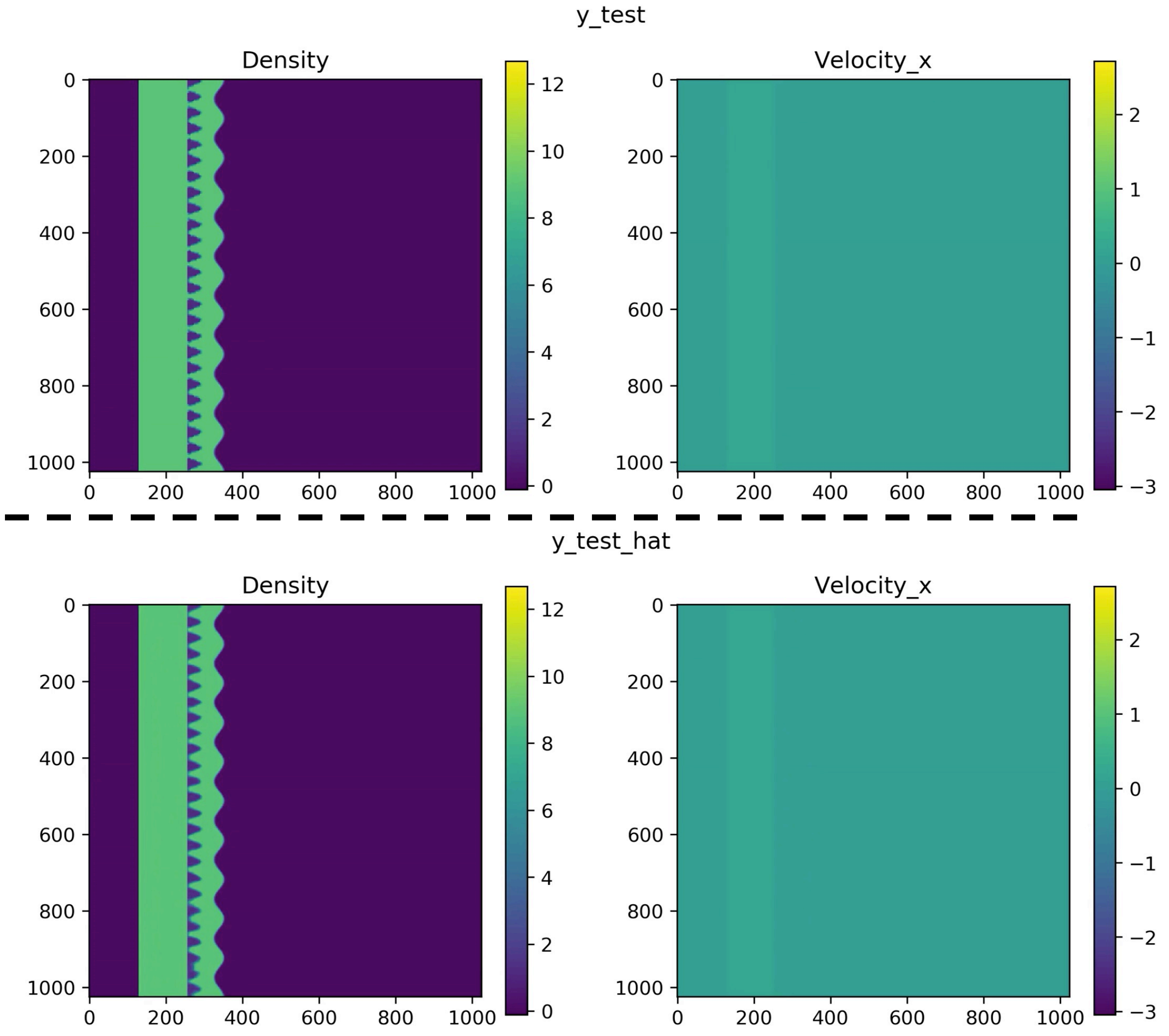
Distributed data model training paradigm



- Dataset split among multiple nodes
- Each GPU
  - receives unique fraction of dataset
  - Duplicate copy of model and optimizer
  - MPI syncs model and optimizer states
- GPU memory limited
  - Can only generate N number of 2x1024x1024 'images' at a time
  - More GPUs -> faster training and inference throughput

Best left-out ‘test’ simulation comparison

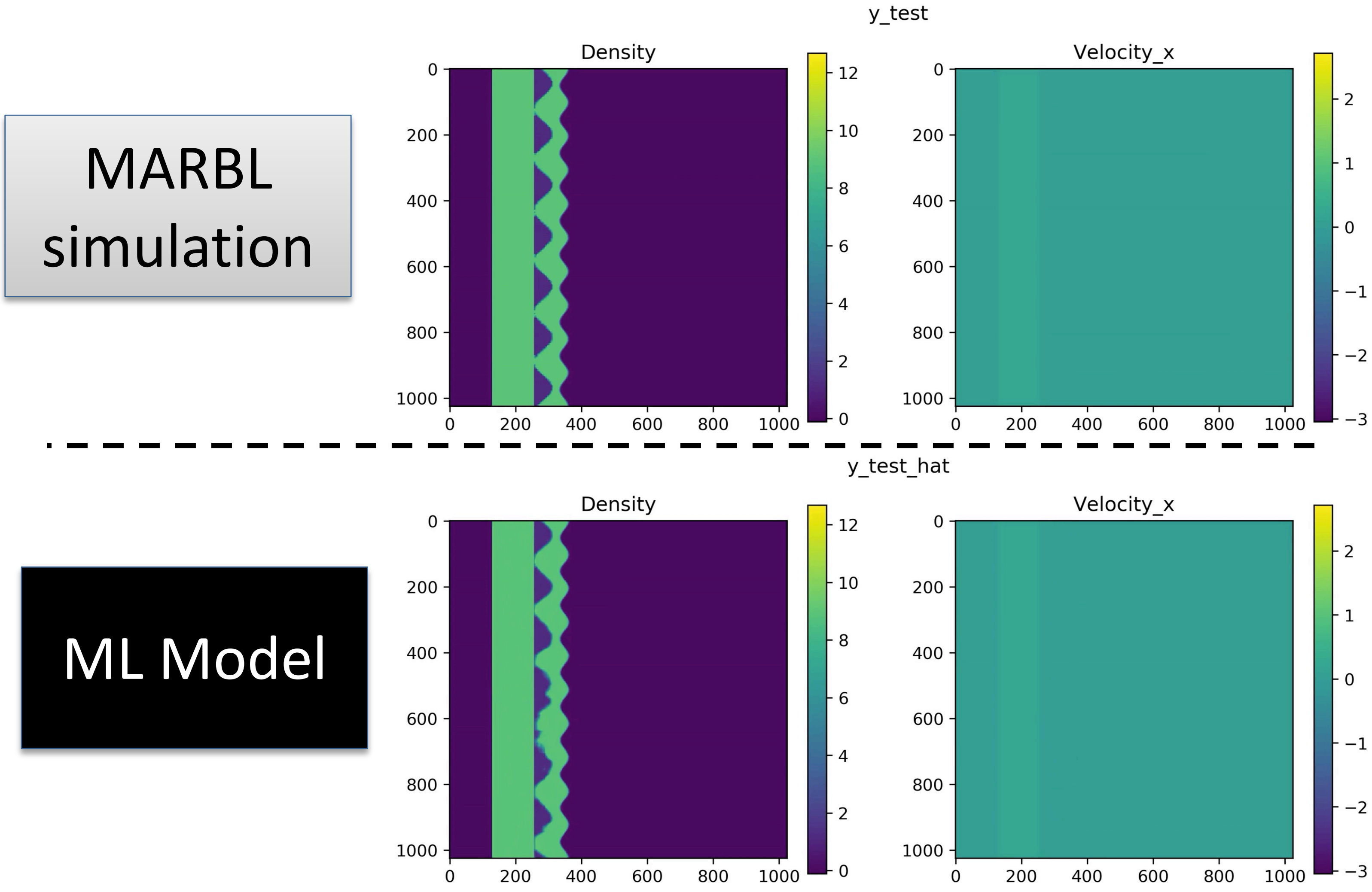
MARBL simulation



ML Model

- Lowest L1 error in test set
- Epoch 400
- MARBL simulation **top**
- ML prediction **bottom**

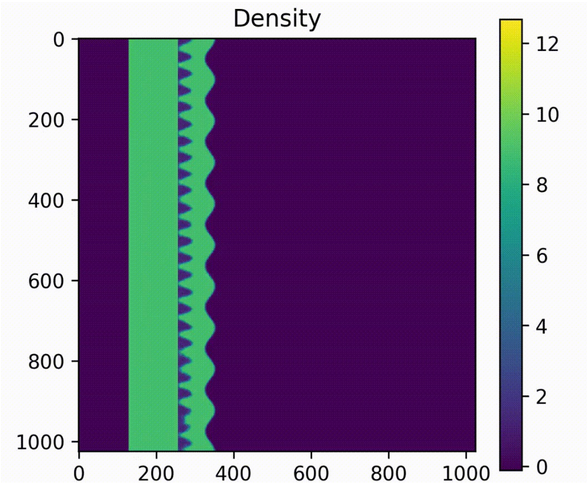
Worst left-out ‘test’ simulation comparison

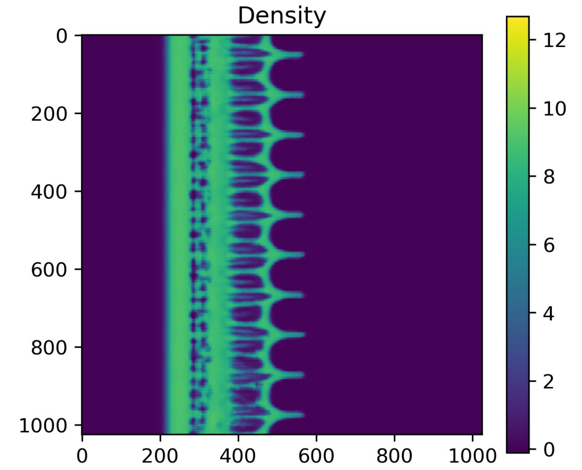


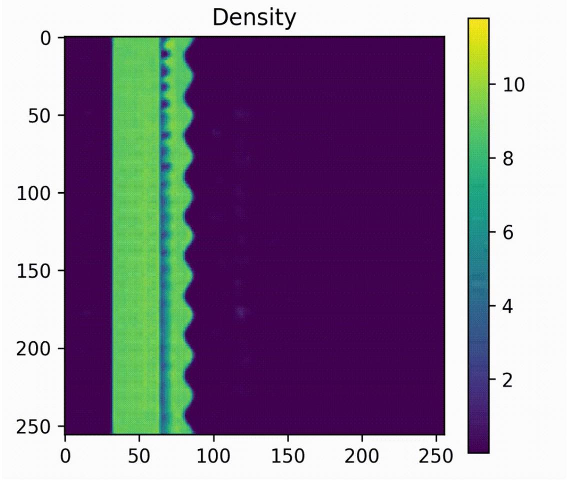
- Highest L1 error in test set
- Epoch 400
- MARBL simulation **top**
- ML prediction **bottom**

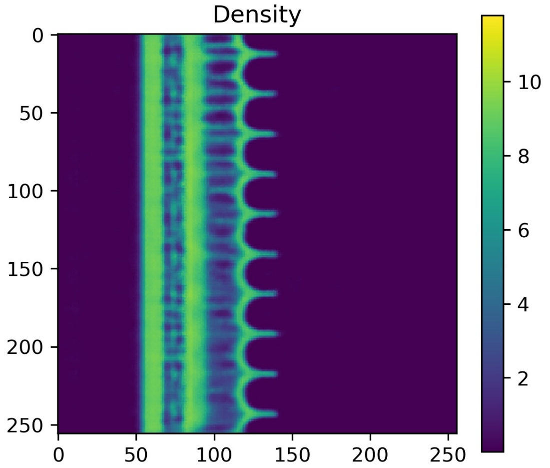
More pixels gave us much more detail
but significantly increase computation demand

1024x1024







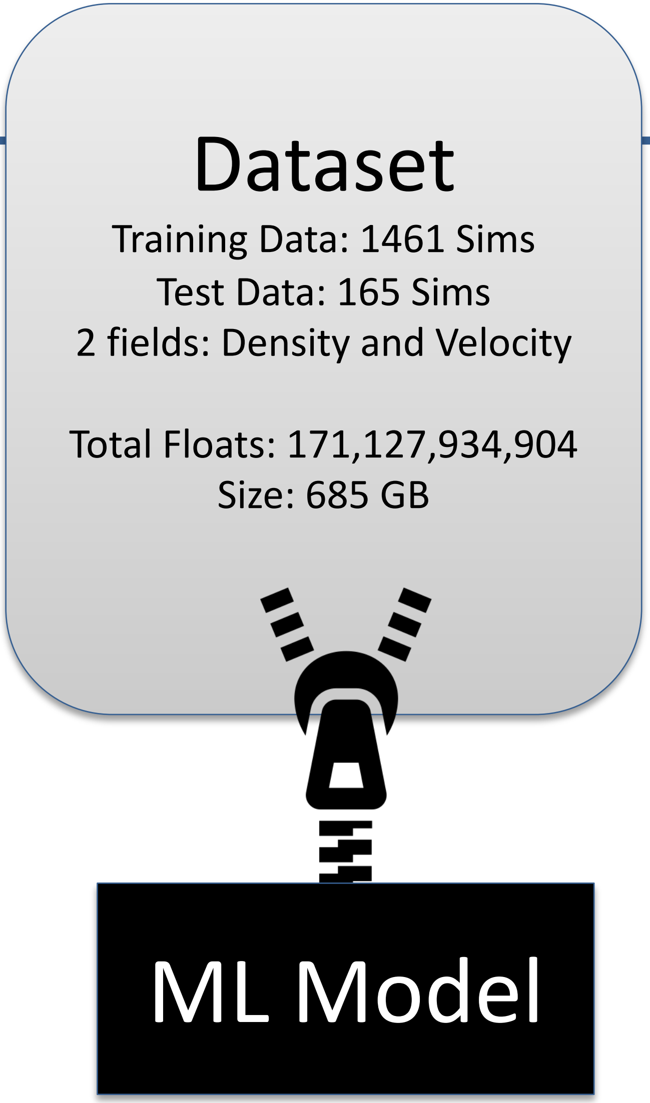


256x256

Lowest MAE from each left-out 'test' set shown

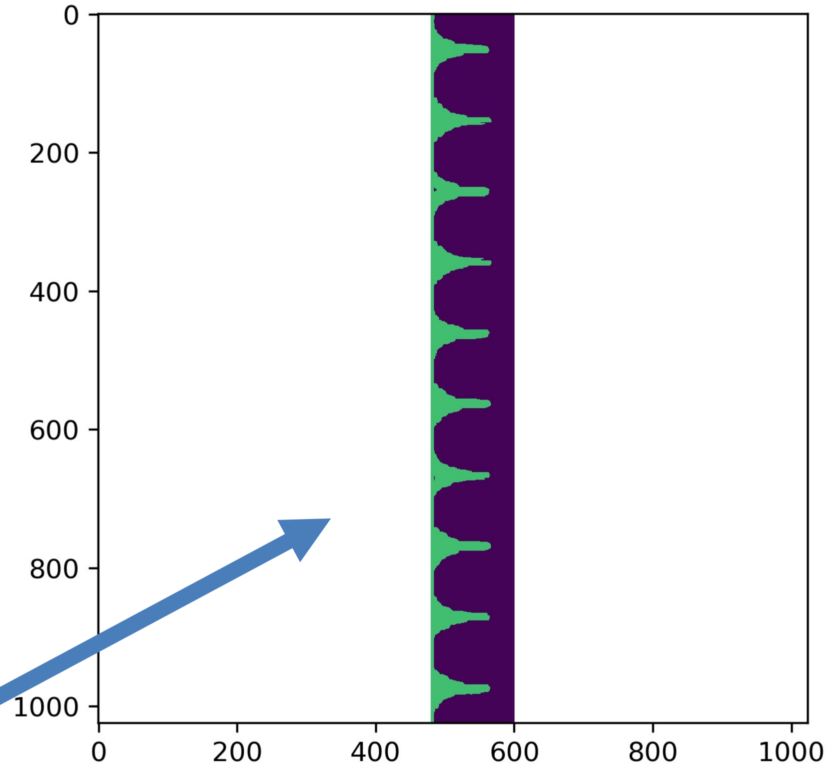
Data compression of the ML model

- 1626 simulations
- 171 billion floats
- Exported model is 178 MB
- **4,000 to 1 compression**
- Brings data visualization from HPC world to laptop world
- With losses to accuracy/detail



Using the ML model to do an inverse analysis

- Use ML model to find [B, Q ,S] that give us the time profile on the right
- Ignore whitespace
- No perfect solution, I drew this by hand and code



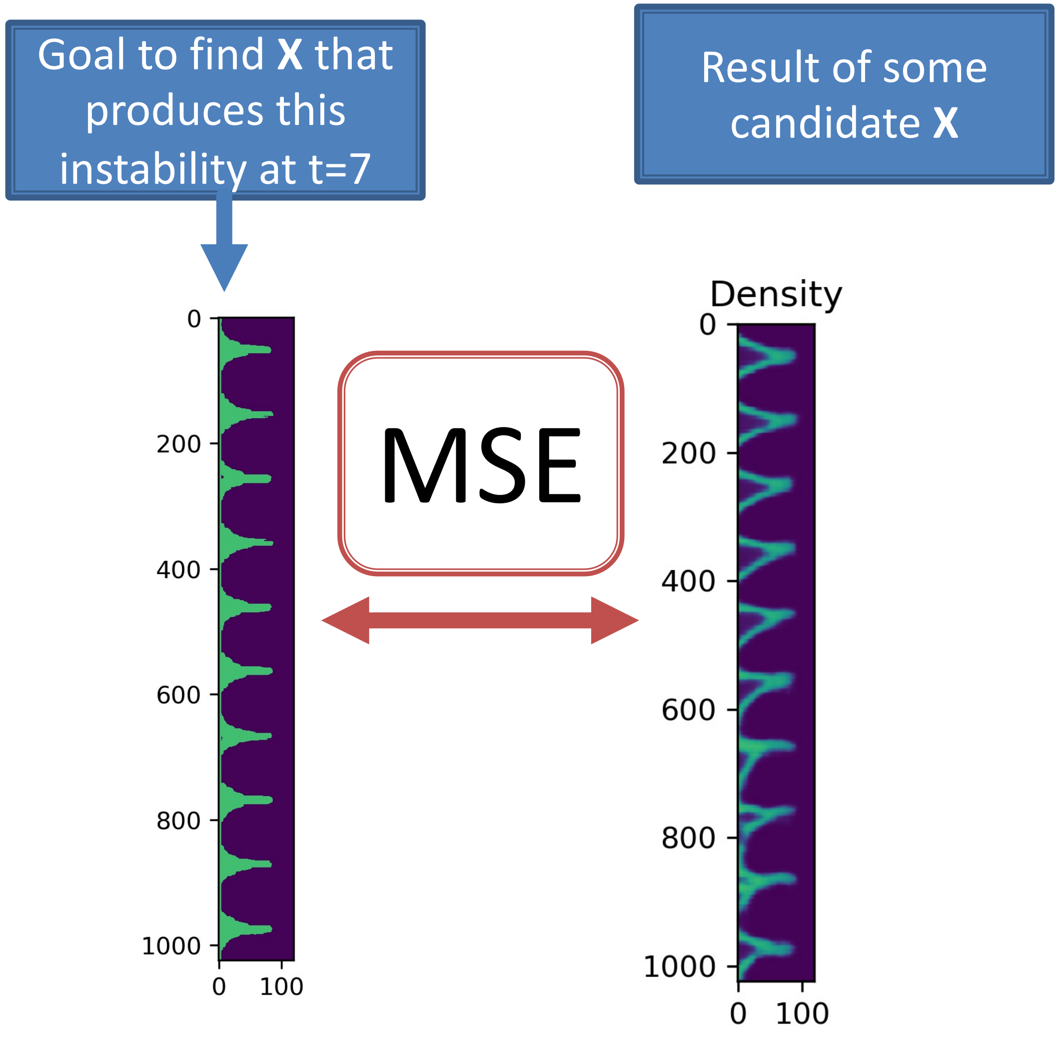
I want to find this at t=7 within my simulation domain

Formulate an optimization problem

- Find **X** that minimizes

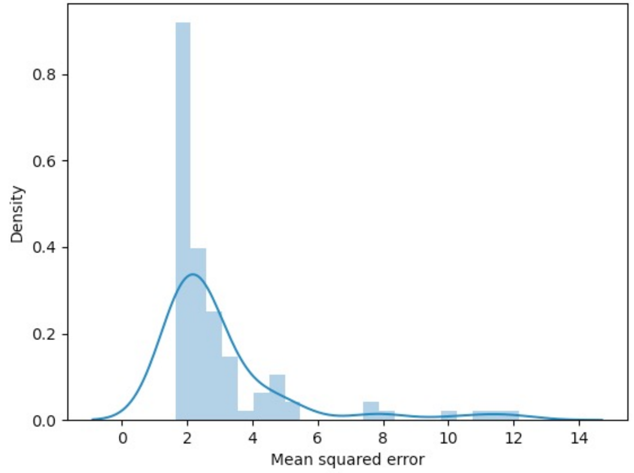
$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2$$

- **X =** $[B, Q, \boldsymbol{S}]$
  - These are the input parameters to the ML model!
- L-BFGS-B (scipy) on ML model
- Derivatives available from ML model!



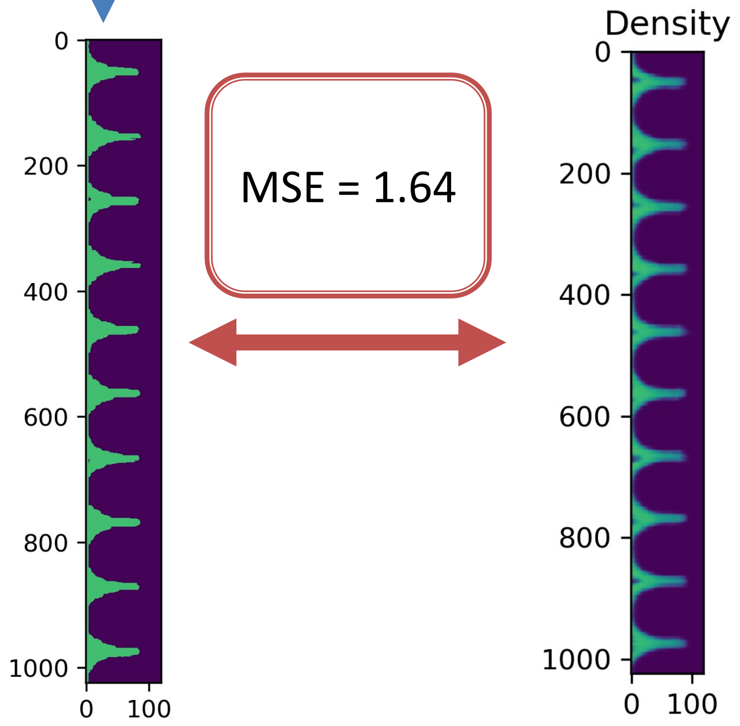
Inverse optimization results

- Optima from 100 runs shown on right
- Lots of local minima shown in histogram
- Single optimization could run on a laptop
  - 1 minute for 120 function evaluations
  - CPU only



Goal to find **X** that produces this instability at t=7

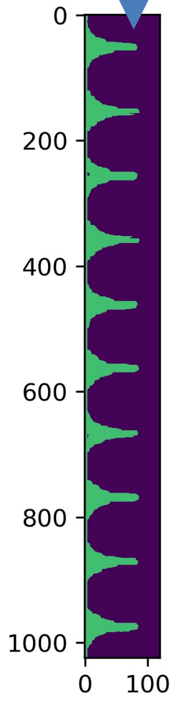
Optimization result
X = [0.162, 24.9, 2.08]
Prediction from ML



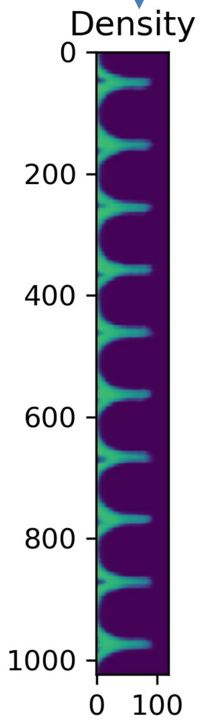
Full ML prediction of inverse optimization

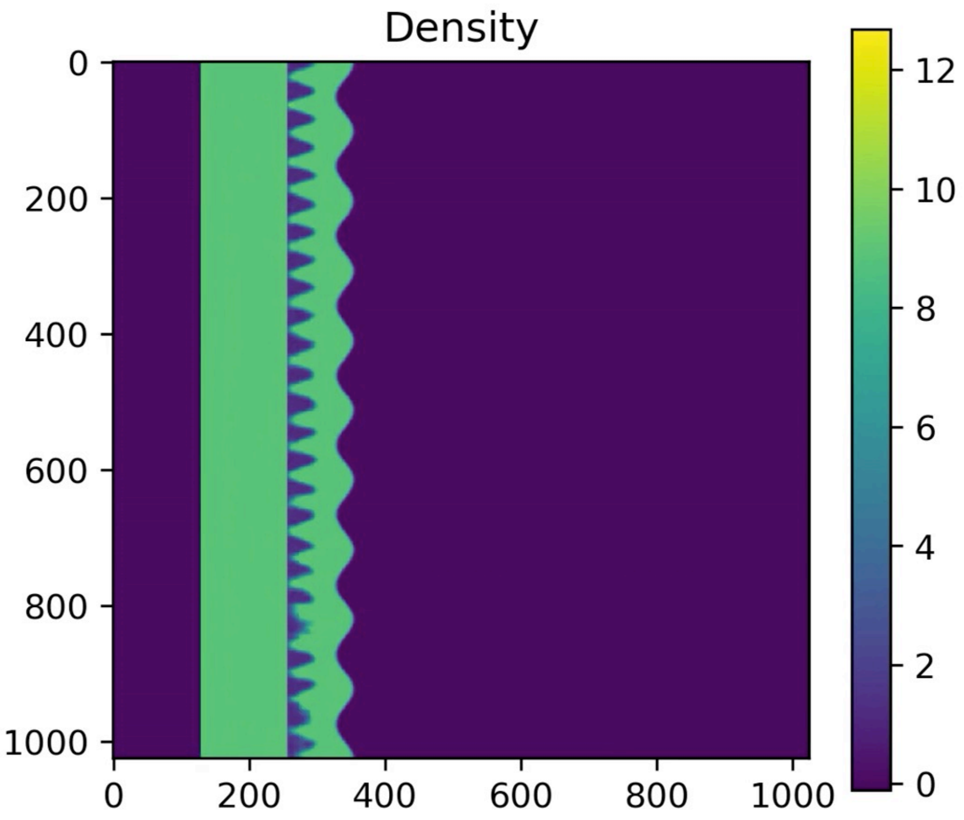
Goal to find **X** that produces this instability at t=7

Optimization result
X = [0.162, 24.9, 2.08]
Prediction from ML



MSE = 1.64





How can you trust your ML model's predictions?

- Trying to use first principles to infer the accuracy of our predictions
- These metrics can be calculated without running a simulation
- Simulations are all closed domain, so these equations should be preserved

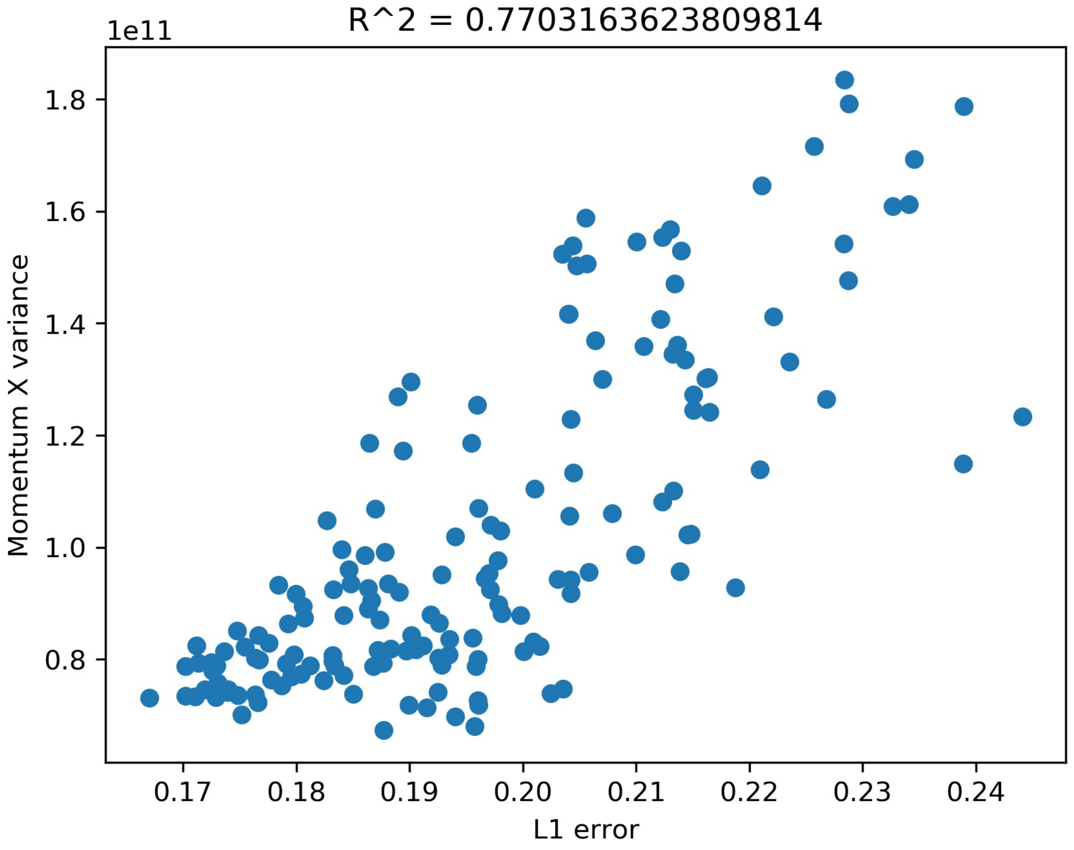
How can you trust your ML model's predictions?

- Trying to use first principles to infer the accuracy of our predictions
- These metrics can be calculated without running a simulation
- Simulations are all closed domain, so these equations should be preserved

- Continuity Equation
  - $\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \boldsymbol{u}) = 0$
- Conservation of Mass
  - Variance of Mass
    - $M(t) = \frac{1}{n} \sum_i^n \rho_i(t) \quad \mathrm{Var}(M(t)) = \frac{1}{n_t} \sum_i^{n_t} (M(i) - \mu_m)^2$
  - Rate of change of Mass
    - $M(t) = \frac{1}{n} \sum_i^n \rho_i(t) \qquad \frac{dM(t)}{dt} = 0$
- Conservation of Momentum
  - Variance of Momentum
    - $M_x(t) = \frac{1}{n} \sum_i^n \rho_i v_i^x \quad \mathrm{Var}(M_x(t)) = \frac{1}{n_t} \sum_i^{n_t} (M_x(i) - \mu_m)^2$
  - Rate of change of Momentum
    - $M_x(t) = \frac{1}{n} \sum_i^n \rho_i v_i^x \qquad \frac{dM_x(t)}{dt} = 0$

Momentum conservation vs L1 error at 'early' epoch

- A model with random shows strong correlation
- This is a 'reasonable' ml model that shows strong correlation!
- As model training continues, sometimes these correlations get worse
- Active research in progress



Correlation value: 0.77

Conclusions

- ML modeling of RMI from MARBL simulations
- ML model allows for quick visualization of a design space
- ML models can be 'run backwards' and inverted
- Demonstrated ML model to interpolate between simulations
- This is just another tool to further our understanding of complicated physics phenomena

- Dataset generation
  - 1,600 simulations
  - 600 Node hours (Lassen/Sierra)
- ML model training
  - 40 GPUs
  - 85 Node hours (Lassen/Sierra)
- ML model vs MARBL sims
  - 1,000 times faster
  - 4,000 to 1 data compression
  - Derivative information

References

1. Zylstra, A.B., Hurricane, O.A., Callahan, D.A. *et al.* Burning plasma achieved in inertial fusion. *Nature* **601,** 542–548 (2022). https://doi.org/10.1038/s41586-021-04281-w

2. Park HS, Lorenz KT, Cavallo RM, Pollaine SM, Prisbrey ST, Rudd RE, Becker RC, Bernier JV, Remington BA. Viscous Rayleigh-Taylor instability experiments at high pressure and strain rate. Physical review letters. 2010 Apr 2;104(13):135504.

3. T.R. Desjardins, C.A. Di Stefano, T. Day, *et al.* A platform for thin-layer Richtmyer-Meshkov at OMEGA and the NIF, *High Energy Density Physics*, Volume 33, 2019, 100705, ISSN 1574-1818, https://doi.org/10.1016/j.hedp.2019.100705

4. Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. arXiv:1511.06434, 2015.

5. Raissi, Maziar, Paris Perdikaris, and George E. Karniadakis. "Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations." Journal of Computational Physics 378 (2019): 686-707.

Lawrence Livermore National Laboratory

The 'Generator' of the DCGAN from [4] used in this work

```
==========================================================================================
Layer (type:depth-idx)                    Output Shape              Param #
==========================================================================================
Generator                                 --                        --
├─Sequential: 1                           --                        --
│    └─Identity: 2-1                      [30, 4, 1, 1]             --
│    └─ConvTranspose2dMod: 2-2            [30, 512, 4, 4]           33,792
│    └─ConvTranspose2dMod: 2-5            [30, 512, 8, 8]           4,195,328
│    └─ConvTranspose2dMod: 2-8            [30, 512, 16, 16]         4,195,328
│    └─ConvTranspose2dMod: 2-11           [30, 512, 32, 32]         4,195,328
│    └─ConvTranspose2dMod: 2-14           [30, 256, 64, 64]         2,097,664
│    └─ConvTranspose2dMod: 2-17           [30, 128, 128, 128]       524,544
│    └─ConvTranspose2dMod: 2-20           [30, 64, 256, 256]        131,200
│    └─ConvTranspose2dMod: 2-23           [30, 32, 512, 512]        32,832
│    └─ConvTranspose2d: 2-26              [30, 2, 1024, 1024]       1,024
├─Tanh: 1-3                               [30, 2, 1024, 1024]       --
```

Trainable parameters: 15,407,040
Total mult-adds: 1.23 (T)

What’s the input?

Batch Size

3 Parameters + Simulation time
[B, Q, S, t]

```
=============================================================
Layer (type:depth-idx)                    Output Shape
=============================================================
Generator                                 --
├─Sequential: 1                           --                         --
│    └─Identity: 2-1                      [30, 4, 1, 1]              --
│    └─ConvTranspose2dMod: 2-2            [30, 512, 4, 4]            33,792
│    └─ConvTranspose2dMod: 2-5            [30, 512, 8, 8]            4,195,328
│    └─ConvTranspose2dMod: 2-8            [30, 512, 16, 16]          4,195,328
│    └─ConvTranspose2dMod: 2-11           [30, 512, 32, 32]          4,195,328
│    └─ConvTranspose2dMod: 2-14           [30, 256, 64, 64]          2,097,664
│    └─ConvTranspose2dMod: 2-17           [30, 128, 128, 128]        524,544
│    └─ConvTranspose2dMod: 2-20           [30, 64, 256, 256]         131,200
│    └─ConvTranspose2dMod: 2-23           [30, 32, 512, 512]         32,832
│    └─ConvTranspose2d: 2-26              [30, 2, 1024, 1024]        1,024
├─Tanh: 1-3                               [30, 2, 1024, 1024]        --
```

Layer by layer progression

```
==================================================                    ==============
Layer (type:depth-idx)                    Output Shape
==================================================                    ==============
Generator                                 --                          --
├─Sequential: 1                           --                          --
│    └─Identity: 2-1                      [30, 4, 1, 1]
│    └─ConvTranspose2dMod: 2-2            [30, 512, 4, 4]
│    └─ConvTranspose2dMod: 2-5            [30, 512, 8, 8]
│    └─ConvTranspose2dMod: 2-8            [30, 512, 16, 16]
│    └─ConvTranspose2dMod: 2-11           [30, 512, 32, 32]           4,195,328
│    └─ConvTranspose2dMod: 2-14           [30, 256, 64, 64]           2,097,664
│    └─ConvTranspose2dMod: 2-17           [30, 128, 128, 128]         524,544
│    └─ConvTranspose2dMod: 2-20           [30, 64, 256, 256]          1
│    └─ConvTranspose2dMod: 2-23           [30, 32, 512, 512]          3
│    └                                    [30, 2, 1024, 1024]         1
├─Tanh                                    [30, 2, 1024, 1024]         -
```

Very first kernel 4 x 4

Every layer doubles

Final 1024x1024 ‘image’

2 fields output, density and velocity

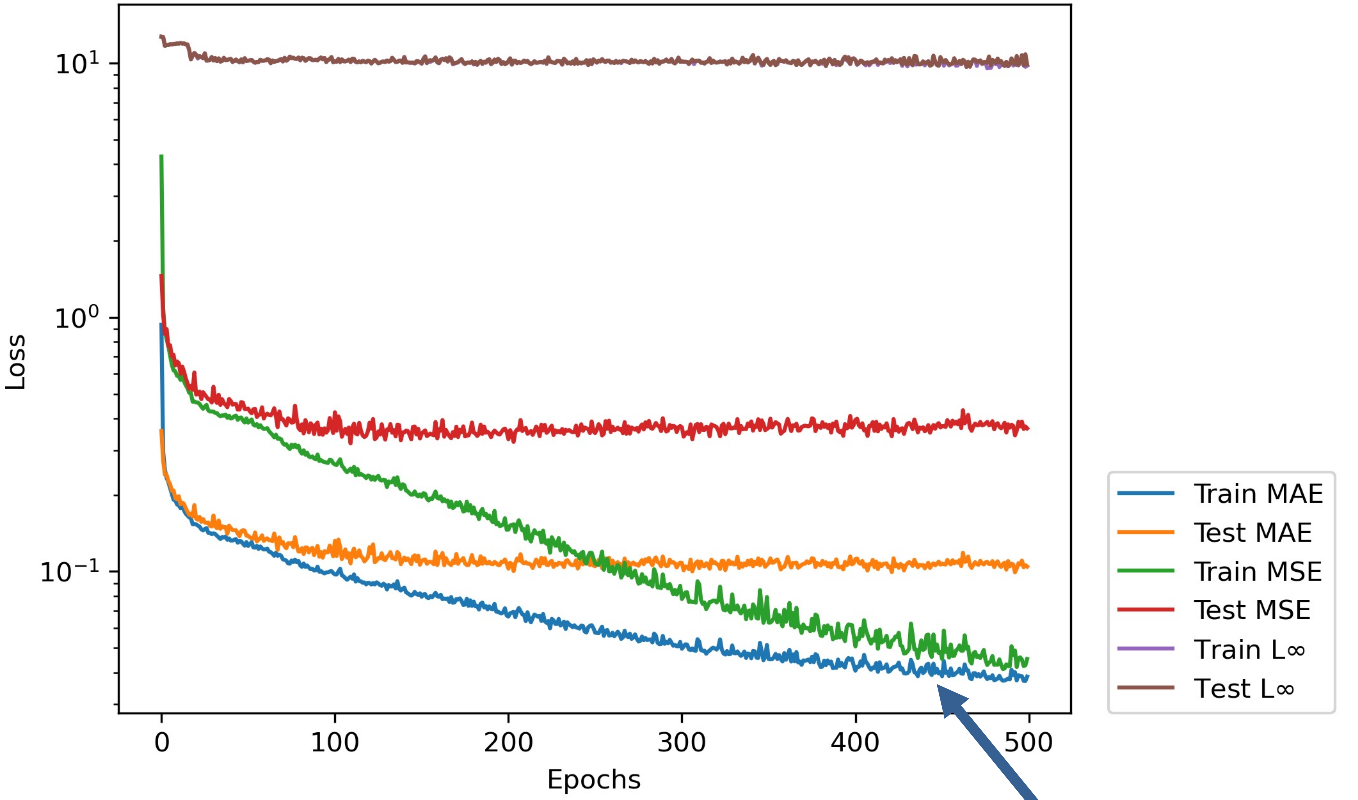
What is “ConvTranspose2dMod”

```
========================================
Layer (type:depth-idx)
========================================
ConvTranspose2dMod
├─Sequential: 1
│    └─Identity: 2-1
│    └─ConvTranspose2d: 2-2
│    └─BatchNorm2d: 2-3
│    └─ReLU: 2-4
```

Just a standard ConvTranspose2d with Batch Norm and activation layer!

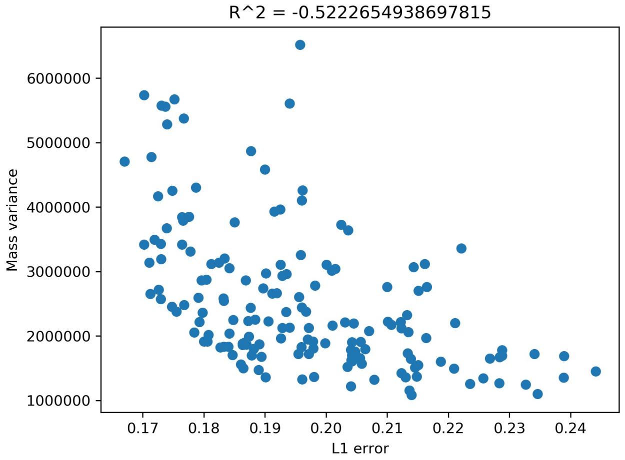
Training the model from scratch

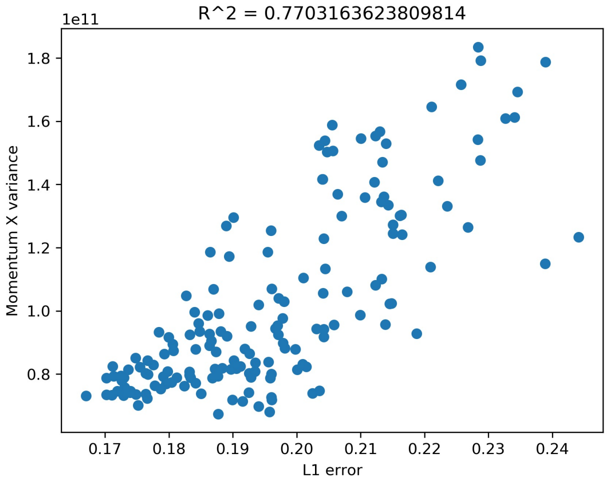
- 40 GPUs in total
  - 10 Lassen Nodes
  - 8.5 hours for 500 epochs
- Minimize Mean Absolute Error (MAE)
  - Showing MSE and L-infinity as well
- Test / Train split
  - 165 simulations / 1461 simulations
- Adam learning rate of 1e-3

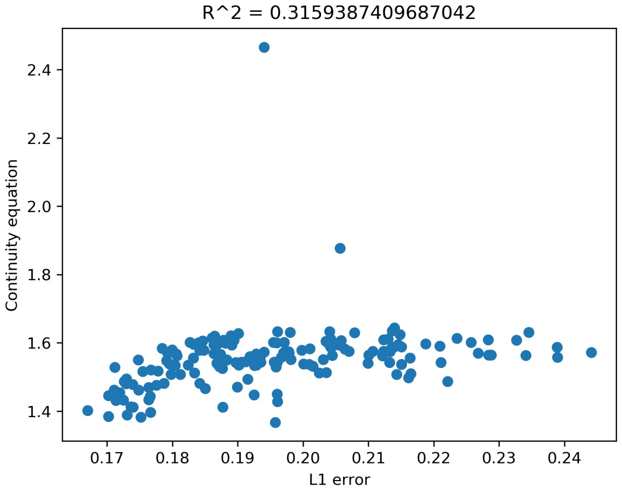


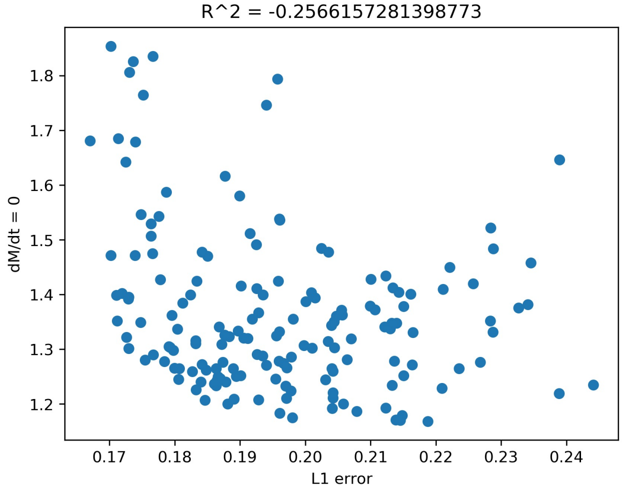
Objective Function

Correlations between first principles and L1 error ‘early’ epoch



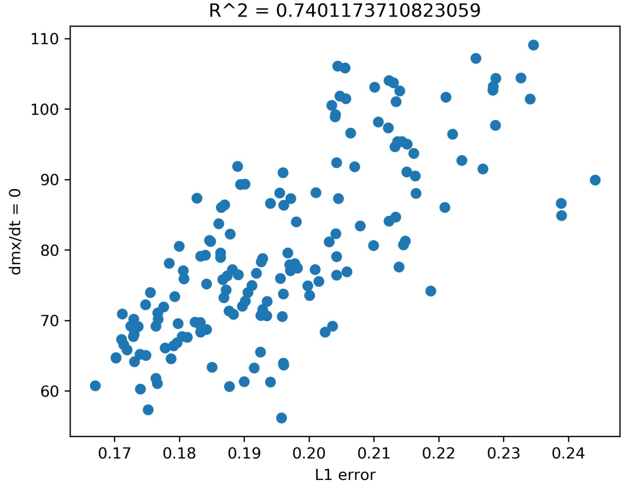




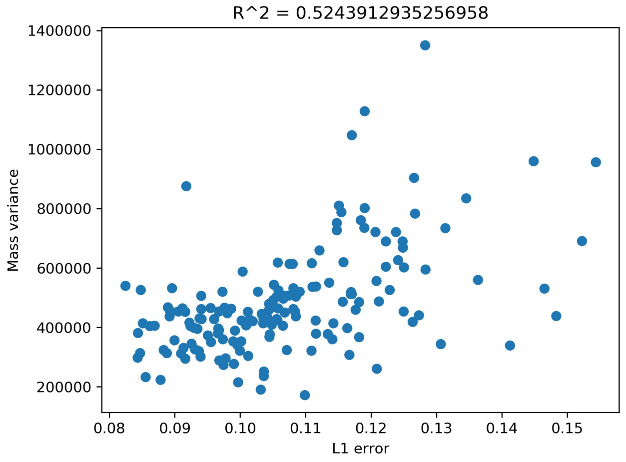


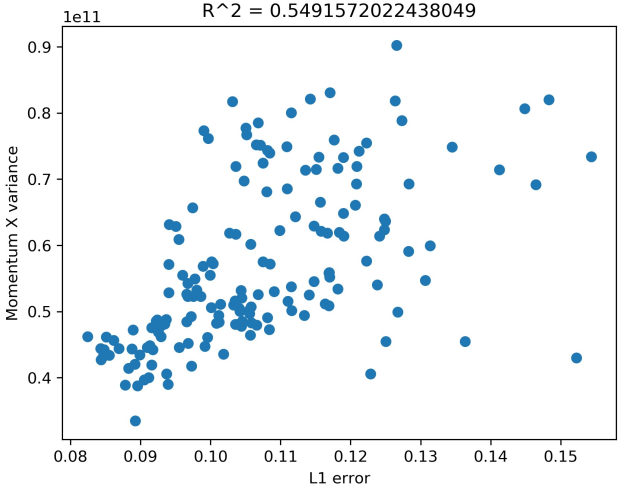
Correlation values

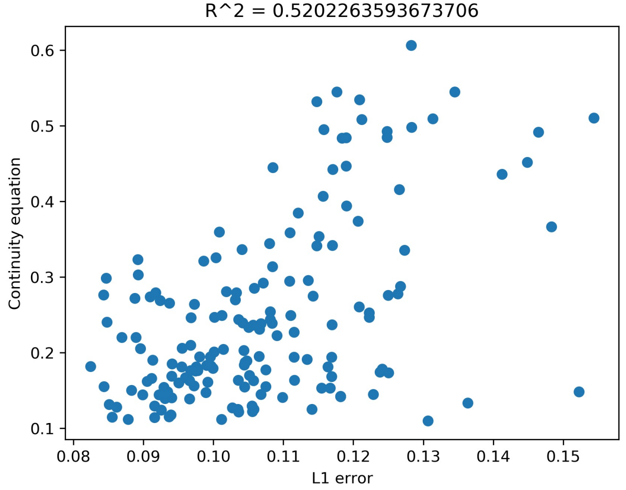
[-0.5, 0.77, 0.31, -0.256, 0.74]

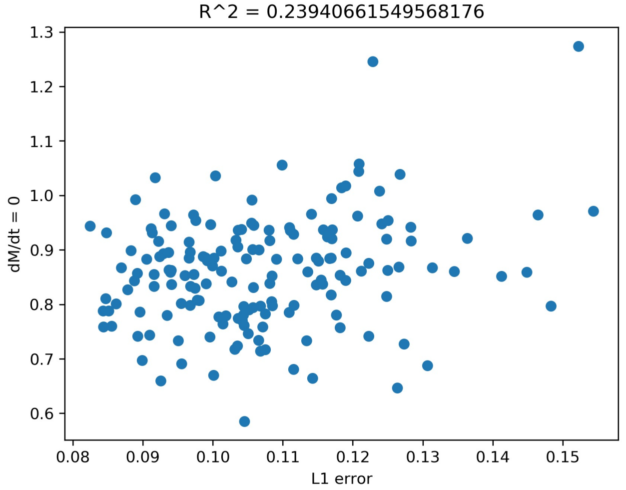


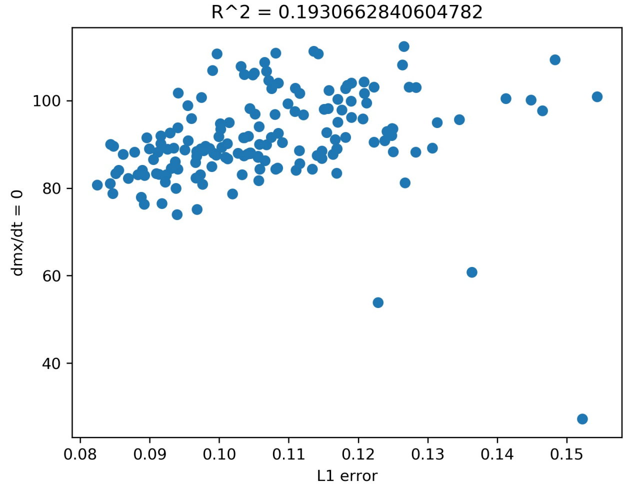
Correlations between first principles and L1 error at ‘final’ epoch











Correlation values
[0.52, 0.54, 0.52, 0.24, 0.19]

What to make of the physics based error indicators?

- Simple physics based errors can be used to infer ML accuracy
- ML Momentum violations do correlate to ML accuracy
  - Other metrics show promise too
- Included some in loss function for PINN [5] ML model
  - Makes the training very difficult
  - Unclear how to balance equations
- Very much active research in progress
  - Believe this can have profound impacts in our field