LAWRENCE
LIVERMORE
NATIONAL
LABORATORY

LLNL-JRNL-865692

Machine Learning Visualization Tool for Exploring Parameterized Hydrodynamics

C. F. Jekel, D. M. Sterbentz, T. M. Stitt, P. Mocz, R. N. Rieben, D. A. White, J. L. Belof

June 18, 2024

Machine Learning: Science and Technology

Disclaimer

MACHINE LEARNING VISUALIZATION TOOL FOR EXPLORING PARAMETERIZED HYDRODYNAMICS

C. F. Jekel* **D. M. Sterbentz** **T. M. Stitt** **P. Mocz** **R. N. Rieben** **D. A. White** **J. L. Belof**

Lawrence Livermore National Laboratory, PO Box 808, Livermore, CA,94551, USA†

November 25, 2024

ABSTRACT

We are interested in the computational study of shock hydrodynamics, i.e. problems involving compressible solids, liquids, and gases that undergo large deformation. These problems are dynamic and nonlinear and can exhibit complex instabilities. Due to advances in high performance computing it is possible to parameterize a hydrodynamic problem and perform a computational study yielding $\mathcal{O}(\text{TB})$ of simulation state data. We present an interactive machine learning tool that can be used to compress, browse, and interpolate these large simulation datasets. This tool allows computational scientists and researchers to quickly visualize "what-if" situations, perform sensitivity analyses, and optimize complex hydrodynamic experiments.

1 Introduction

For hydrodynamics, it can be very difficult to understand the sensitivity of physical instabilities to small perturbations in initial conditions. It is possible for a human to understand the impact of one or two inputs on the temporal evolution of an instability. However, as the number of system parameters grow, so does the complexity of the system. Consider the Rayleigh-Taylor instability (RTI) and Richtmyer-Meshkov instabilities (RMI) which can have several inputs that influence the transient state. In these cases, ensembles of simulations are required to understand the sensitivity of these instabilities with respect to their initial states. Often these simulation results are computationally expensive, and it is difficult for researchers to look at every simulation result. The Cinema project [1] set out to aid researchers in understanding ensemble calculations by providing tools to quickly look through simulation results. The tools provided an intuitive interface for researchers to explore pictures of simulation results. While this tool is quite useful, the results are limited to only the performed calculations. Our work builds upon this concept of allowing researchers to quickly explore ensemble calculations, with the main advantage being the ability to quickly visualize results that were not previously calculated. This interpolation is accomplished with a machine learning (ML) model that allows a user to view the temporal evolution of instabilities by seamlessly changing initial conditions.

The RTI and RMI are closely related. A RTI occurs at the interface of two fluids mixing with different densities. A RMI occurs when a shock wave amplifies perturbations at a material interface, causing large jet-like growths [2, 3, 4, 5]. The use and understanding of the transient behavior of these instabilities is important in many applications. For example, experimentally measuring RMI formations is useful for calibrating high strain rate material models [6, 7]. Additionally, in inertial confinement fusion (ICF) experiments, where lasers are used to heat and compress a fuel capsule to the point of starting a self-sustaining fusion reaction [8]. Unfortunately, RMI have been known to form within ICF capsules. The

*jekel1@llnl.gov

RMI often destroys the fuel target before fusion ignition is achieved [9]. Increasing our ability to design and control RMI would have profound impacts in fusion research.

Our ability to perform simulations with the onset of exascale computing[10] may exceed our ability to comprehend those results. Several tools, for instance Merlin[11], currently exist to orchestrate millions of simulations with high performance computing (HPC). Typically, a researcher processes simulation results one at a time. The tool Cinema [1] can be used to quickly swap between images of the ensemble of simulation results, but is limited to only those results. Our work proposed to use ML to interpolate and quickly infer from datasets of several simulations, while still allowing researchers to quickly build their own intuition for these complex systems.

There happen to be several applications of ML work applied specifically to the flow around airfoils. These simulations are typically 2D and are utilized to understand the flow properties (e.g. lift, drag) of the airfoil. Wu et al. [12] and Wu et al. [13] trained General Adversarial Networks (GAN) to airfoil flow fields. The airfoil shape was represented as a vector of 14 values, which the generator model then learned the mapping from the parameterized geometry to the CFD solution. For more detail on GANs we refer the reader to the review by Chakraborty et al. [14]. Li et al. [15] used the same methodology to learn force fields of a hypersonic vehicle from parameterized flight conditions. Wang et al. [16] then goes on to demonstrate how transfer learning can be used on subsequent datasets in a related domain. Hariansyah and Shimoyama [17] used Deep Convolutional Generative Adversarial Networks (DCGAN) model to produce fake airfoils in an optimization. Nandal et al. [18] trained a GAN model to pressure fields around parameterized airfoils. Hou et al. [19] predicted hydrodynamic solutions around a submarine using a time series of previous fluid flow states. A prediction at an arbitrary time requires iteratively feeding past ML predictions into the model. Kashefi and Mukerji [20] used neural networks to predict the permeability from 2D or 3D point clouds of the boundary surface of porous media. These studies all had very specific applications in mind, and our work will build upon these methods to show how to approach a general hydrodynamics problem.

It is also important to mention that ML has not been limited to just simulation results, as there has been recent work applying to experiments. Chen et al. [21] trained a neural network to predict how a flame would develop within a scramjet from pictures of the previous states of the flame. Li et al. [22] trained a system of neural networks to learn the mapping of pressure sensors to images of the shockwave structure. The structure of the network also follows the DCGAN model. Fernandez-Grande et al. [23] used GANs for the reconstruction of acoustic fields. The work shows that the potentially machine learning can be used to enhance bandwidth limited acoustic sensors by being able to recover some lost sound energy at high frequencies.

A number of the previous literature utilized the Deep Convolutional Generative Adversarial Networks (DCGAN)[24] architecture on physics based datasets. Akkari et al. [25] trained DCGAN models to CFD solutions of flow fields around a parameterized square obstacle. Instead of using the parameterized position of the obstacle, the DCGAN model learned an unsupervised latent representations of the obstacle placement. Cheng et al. [26] used a DCGAN to predict fluid flow solutions. Drygala et al. [27] trained a DCGAN model to fluid flow around a cylinder and a low pressure turbine stator, with the goal focusing on the model being able to quickly generate several synthetic fluid flows. Belinchon and Gallucci [28] used GANs to investigate 1D stochastic fields for multiscale physics of turbulence, and expressed a desire to extend their methods to 2D. 3D ML methods are currently a bit more expensive in inference time. For 3D applications, we refer the reader to the work of Wiewel et al. [29], who used an autoencoder to compress 3D Eulerian solutions of dropping liquid and buoyant smoke. Then a LSTM network was used to learn concurrent time sequences in the compressed latent space. The two models working together allowed for the generation of quick 3D fluid predictions.

The previous DCGAN work focused on steady-state fluids, while our work focuses on using the model in regression for large outputs of hydrodynamic instabilities. Additionally, the resolution and dataset sizes both exceed previous works. The focus of this work is largely limited to 2D simulations to demonstrate how the real-time tool can be used to intuitively explore the complex transient space of these instabilities. The DCGAN architecture being fast in inference also enables optimization, which our work demonstrates can be used to find initial conditions leading to particular RMI behavior and characteristics. Lastly, our work also explores how an ensemble of these models may be used to estimate the uncertainty of the predictions.

The generic multi-material hydrodynamic problem of interest consists of materials (in the solid, liquid, or gas state) that undergo large deformations. These problems are driven by high velocity impacts or rapid energy deposition e.g. chemical explosives or incident laser pulses. These problems are dynamic and nonlinear and exhibit complex behavior such as RTI and RMI. A key aspect of this work is studying the control, e.g. enhancement or suppression, of these hydrodynamic instabilities, while most of the previous literature applied ML to steady-state fluid dynamics. Several recent publications have demonstrated this is possible with experiments [30, 31, 32], and our work explores how ML models in this regime can be used to further explore the phenomena.

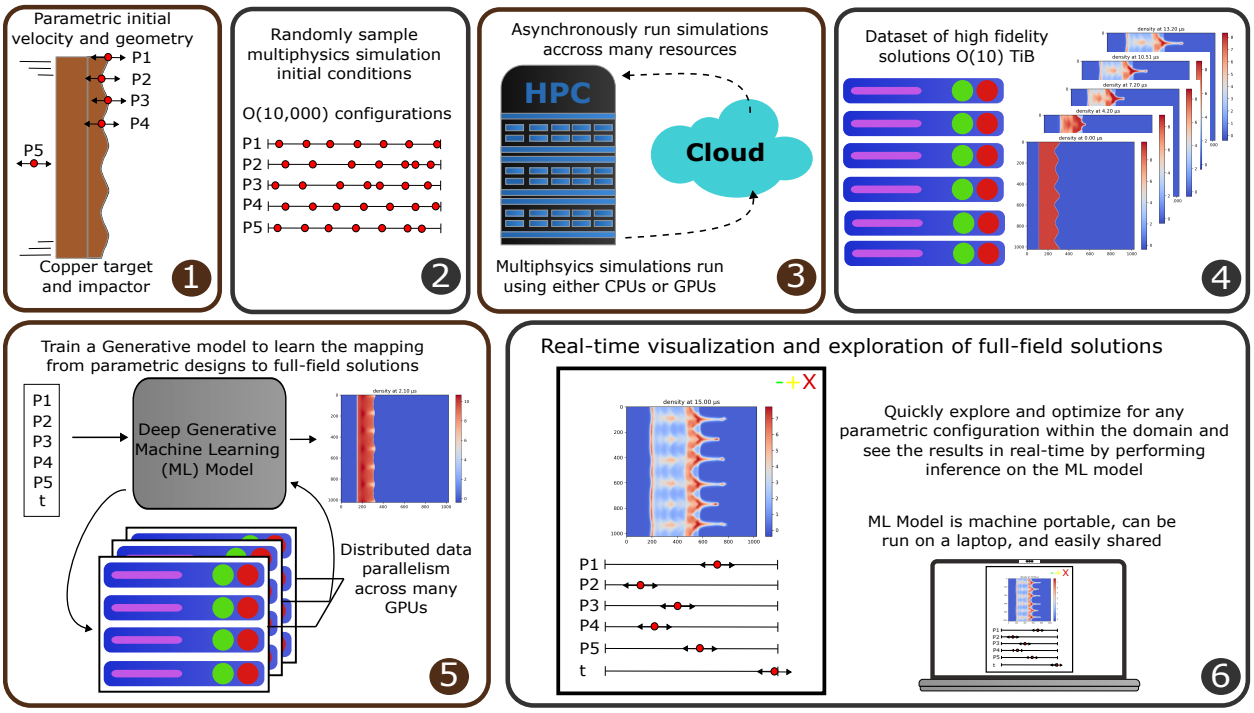

Figure 1: This is an overview of the proposed method. 1) A hydrodynamic simulation is parameterized to study the effect of these parameters on the resulting instabilities. 2) These parameters are randomly sampled. 3) Simulations are performed on HPC using an asynchronous queue. 4) This yields a large dataset of full-field hydrodynamic solutions. 5) A generative machine learning model is then trained to learn the temporal mapping from the parameters to the full-field solutions. 6) This machine learning model can be used for real-time visualization, dissemination of results, optimization, and more.

Physics Informed Neural Networks (PINNs) [33] have proven to be very popular for machine learning of fluid flow. In PINNs, the equations of fluid dynamics (the partial differential equations) are implemented in the ML loss function. Conversely, the DCGAN approach presented here has no hard-wired knowledge of conservation laws, equations-of-state, etc. Instead it is trained soley with images of physics fields e.g. density, pressure, velocity, etc. This is extremely important for our hydrodynamics applications, becuase re-implementing formulas for complex equation-of-state, material failure and fracture, explosive burn, etc. in a ML loss function is prohibitive. In order to demonstrate the general applicability of the proposed DCGAN approach, a non-fluids example using an analytical test function is presented.

The outline is as follows: Section 2 reviews the hydrodynamic simulation methodology, Section 3 reviews specific applications (high-velocity impacts, RTI mixing, linear shaped charges), and Section 4 details the machine learning approach. The key results of the trained models are discussed in Section 5. Several applications are presented in Section 6.1, which include real-time visualization, an analytical problem outside of hydrodynamics, optimization to find desirable RMI characteristics, and results on estimating the error in model predictions. Finally, we will discuss some of the limitations of the approach.

2 Hydrodynamic Simulations

A general overview of the methodology used to create the machine learning model is shown in Figure 1. The key steps are: (1) parameterize initial conditions of a hydrodynamic problem, (2) define sample points in parameter space for which to perform high-fidelity hydrodynamic simulations, (3) asynchronously run the independent hydrodynamic simulations on large computational resources, (4) create a dataset of the full-field solutions, which is typically $O(TB)$ of data, (5) train a generative deep convolutional neural network to "learn" the hydrodynamic response. The final step (6) is to use the ML model to quickly predict time dependent full-field results for arbitrary parameters, including parameters *not included* the training space.

The hydrodynamic model consists of the Euler Equations (conservation of mass, momentum, and energy) combined with equation of state formula and strength models such as Steinberg-Guinan[34]. The initial condition consists of the geometry of the materials and the initial density, velocity, and internal energy. Boundary conditions such as wall, periodic, or outflow are enforced. Source terms such as energy deposition may be applied. The Euler Equations are solved using an Arbitrary Lagrangian-Eulerian (ALE) Finite Element Method (FEM). The Euler Equations are integrated in time using explicit conditionally stable Runge-Kutta with adaptive time stepping. At fixed intervals, e.g. 1 μs, the fields are projected from the high order ALE-FEM mesh onto a uniform Cartesian mesh, i.e. an image, and are recorded to disk. A single hydrodynamics simulation thus results in a sequence of images of l physical fields (e.g. density, velocity, energy) at fixed snapshots of time. The images are of a constant $n \times m$ size, and we have k temporal instances of these images. Each value in the image arrays is a single precision float.

A hydrodynamic study consists of a hydrodynamic problem with an initial condition that is parameterized via d real-valued parameters. Clearly these parameters must have bounds and it is assumed these parameters can be normalized to the interval $[0,1]$. The overall goal is to construct an accurate and compact model of the form

$$f_{ML} : R_{\square}^{d+1} \rightarrow R^{l \times n \times m} \tag{1}$$

where R is the set of real numbers in a specified dimension. The input space consists of the d initial condition parameters plus time (where time has also been normalized to $[0,1]$. The output consists of images of hydrodynamic fields e.g. density, velocity, energy. The model function f_{Ml} has millions of free parameters, the optimal value of these parameters are "learned" from simulation data without knowledge of the Euler Equations, material properties, etc.

The machine learning tool that we have developed has three major advantages that have the potential to revolutionize the way analysis and postprocessing of large sets of hydrodynamic simulations. One key point is that the size of this model (in bytes) is many orders of magnitude smaller than the raw image data, this enables efficient archiving and sharing of large ensembles of hydrodynamic studies with colleagues. A second key point is that the model can be evaluated at some new point in parameter space many orders of magnitude faster than performing a new hydrodynamic simulation, enabling a researcher to quickly visualize a hydrodynamics simulation that was not actually performed. A third point is that since by construction the model f_{Ml} is continuous, the derivative can be computed (exactly, using the automatic differentiation capability of ML libraries) enabling sensitivity analysis and inverse design optimization. The architecture of the model function f_{Ml}, and the training process, is described in more detail in Section 4.

The hydrodynamic simulations were performed using an Arbitrary Lagrangian-Eulerian (ALE) Finite Element Method (FEM) simulation code called MARBL. Generally speaking, Lagrangian formulations allow for precise tracking of material interfaces, but the computational mesh can become excessively distorted. On the other hand, Eulerian methods do not attempt to maintain sharp interfaces between material interfaces, and materials are allowed to mix. In the ALE method the simulation begins as pure Lagrangian but at later time allows mesh relaxation and remap as the material deformation becomes large, the flow becomes turbulent, or material mixing occurs. A particular feature of MARBL is higher order elements [35],[36],[37], this allows for greater accuracy for a given mesh, and results in high efficiency on Graphical Processing Units (GPU) due to high ratio of flops per mesh element [38]. Other considerations include the need for high order artificial viscosity [39] and of course the important ALE remap step [40]. The Livermore Equation of State Library (LEOS) [41] is used for the equation of state for all materials, and Steinberg-Cochran-Guinan strength model [42] is used for solid materials. The MARBL simulation code is not in the public domain, but a more limited high-order Lagrangian-FEM code restricted to ideal gases and single materials, named Laghos [3], is publicly available.

3 Parameterized hydrodynamic examples

Four parameterized hydrodynamic problems are presented here. An ensemble of simulations is performed to understand the influence of the parameters on the complex time-dependent instabilities. Latin Hypercube Sampling [44] (LHS) was used to generate samples from the bounded parameters. The workflow utility Merlin[11] was used to manage the ensemble of simulations. The physics simulations were either run in a single large allocation, or multiple smaller allocations. Merlin executes the simulations asynchronously as soon as either resources became available, or a simulation completed. The studies were performed on the LLNL Lassen[4] HPC.

An overview of the datasets generated is shown in Table 1. The full-field solutions were stored as float 32 data using hdf5 [45]. All datasets were on order of a hundred billion pixels, with the Rayleigh-Taylor dataset having the fewest number of pixels at 120 billion. Lossless compression with gzip was used to reduce storage requirements of the data. The Rayleigh-Taylor dataset is 328 GB on disk.

[3] Available online at https://github.com/CEED/Laghos

[4] Lassen is a HPC with 795 number of nodes. Each node contains 2 power 9 ppc64 CPU sockets and 4 NVidia 16 GB V100 GPUs. `https://hpc.llnl.gov/hardware/compute-platforms/lassen`

Table 1: An overview of the datasets created.

Name	# of sims.	Time Steps	Fields	Pixels	# of pixels (millions)
PCHIP impact	2,985	51	6	1024 x 1024	957,779
Double sine wave	1,626	51	3	1024 x 1024	260,862
Linear shaped-charge	2,299	41	7	512 x 1664	566,153
Rayleigh-Taylor	2,000	51	6	768 x 256	120,324

3.1 High Velocity Impact Study

The high velocity impact studies consist of an initially stationary copper target with a perturbation machined into the right-hand side (the copper-air interface), and a copper impactor with velocity of $2km/s$. As the shock wave reaches the interface perturbations, vorticity deposition occurs along the interface due to misalignments between pressure and density gradients at the perturbations. This creates the RMI that generally results in the jetting of the copper target material. The impactor is 1×9 cm and the target is nominally 0.5×9 cm. These dimensions and velocities are chosen to be compatible with the two-stage gas gun at LLNL's High Explosive Applications Facility (HEAF)[46, 47]. Whereas in actual HEAF experiments the impactor/target are circular with 9cm diameter, the simulations are 2D. The simulations begin at time $t = 0$ with the impactor and target in contact with a discontinuous velocity. An example figure of one of these impacts is shown in Figure 2.

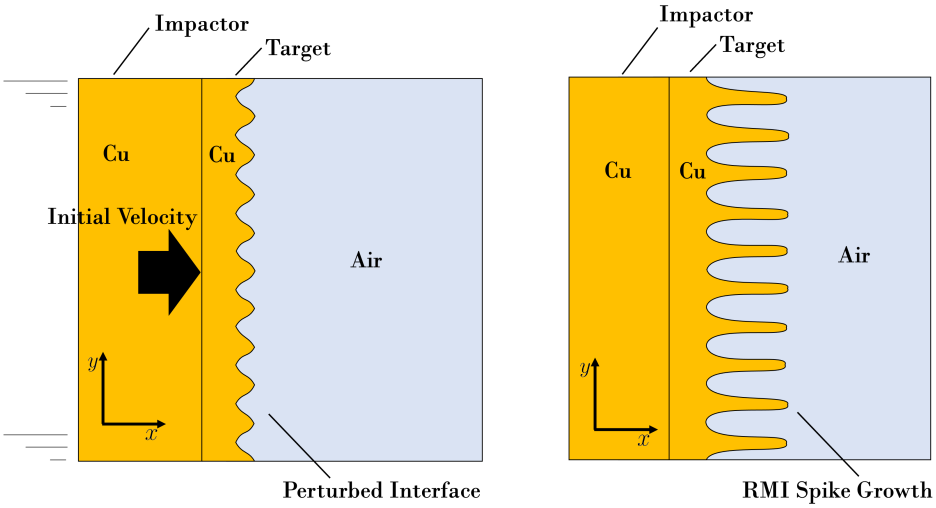

Figure 2: Illustration of the high velocity impact. The left figure is at $t = 0$, the right figure is at some later time i.e. $t = 10\mu s$.

Two different parameterizations of these high velocity impacts were studied to investigate the formation of Richtmyer-Meshkov Instabilities (RMI) that occur at the target-air interface, as small perturbations evolve into a sharp jet. The first study is the PCHIP impact which looks at how changes in the perturbation in the target-air interface influence the RMI. The second study is the double sine wave study, which looks at how a sine wave can be placed on the impactor side of the target to influence RMI, when there is a fixed sine wave along the target-air interface.

3.1.1 PCHIP impact

An example of the experimental setup of the PCHIP impact is shown in Figure 3. The target-air interface is parameterized with four parameters defining a Piecewise Cubic Hermite Interpolating Polynomial (PCHIP) [48]. The PCHIP parameter range was $[-0.25, 0.25]$ cm.

The hydrodynamic solutions were computed using a nominal mesh of 144×144 quadratic (Q_2Q_1) elements, the mesh was morphed to have conformal interfaces between air and copper. During the simulation the fields density, velocity x, velocity y, energy, pressure, and the material indicator are projected onto a 1024×1024 Cartesian image and exported at 51 uniform timesteps from 0 to 15 μs.

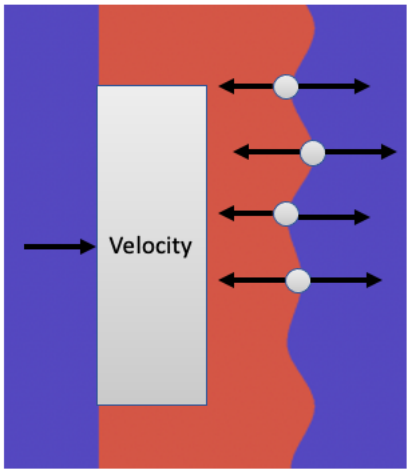

Figure 3: An illustrative depiction of the parameterized PCHIP impact.

3.1.2 Double sine wave

The other high velocity impact involved sinusoidal waves on both sides of the impactor. The impactor side of the target was parameterized with the sinusoidal wave

$$B\cos\left(\frac{2\pi Qx - s\pi}{9.0}\right) \tag{2}$$

to seed initial RMI growths. The free side of the target (copper-air interface) utilized a fixed wave of

$$0.5 + 0.1\cos\left(\frac{2\pi 10x}{9.0}\right) \tag{3}$$

which also seed RMI growth. The purpose of this parameterization was to study how parameterized RMI growths interact with a known RMI seed (on the free side), the notion is that an optimized sinusoidal perturbation can initiate vorticity that "cancels out" the primary RMI. The following bounds were placed on the three parameters of the impactor side wave: B from $[0.1, 0.25]$, Q from $[5.0, 25.]$, and s from $[0.0, 3.14]$.

The copper impactor was 1×9 cm and traveling at 2 km/s. Lucite was used to fill in the material between the target and the impactor, creating a flush interface for an initial impact. The simulations were ran out for 7 μs after the initial impact. An overview of the simulation setup is seen in Figure 4.

The solutions were computed using quadratic (Q_2Q_1) elements on a nominal grid of 144×144. The mesh was morphed to have conformal interfaces. The simulation results were saved on a 1024×1024 uniform grid for the following fields: density, velocity x, velocity y. These fields were exported along 51 ideally uniform timesteps from 0 to 7 μs.

3.2 Linear Shaped Charge Study

Linear shaped charges have a variety of industrial use cases such as structural demolition [49], geo-engineering [50], [51], and aerospace [52]. They utilize an explosive which propels a liner (typically copper) into a high velocity jet that will penetrate or cut into materials. Like the previous high velocity impactor study, the shaped charge jet is initiated by RMI. Design exploration of shaped charge jet formation can be non-intuitive and require many thousands of hydrodynamic simulations to explore the parameter space [53]. As a case study, this work proposes a parameterized linear shaped charge design involving liner shape and detonator locations.

An overview of the parameterized linear shaped charge is shown in Figure 5. The shape of the copper liner facing the explosive is parameterized with four spline parameters ranging from $[0.05, 0.3]$ cm, while jet side of the liner is fixed at a 60°angle. One additional parameter controls the location of two detonation points, which are kept symmetric about the center of the liner. The detonator location was parameterized along the steel case, where 0 represents a placement along the center of the liner, and 1 represents a placement against the liner.

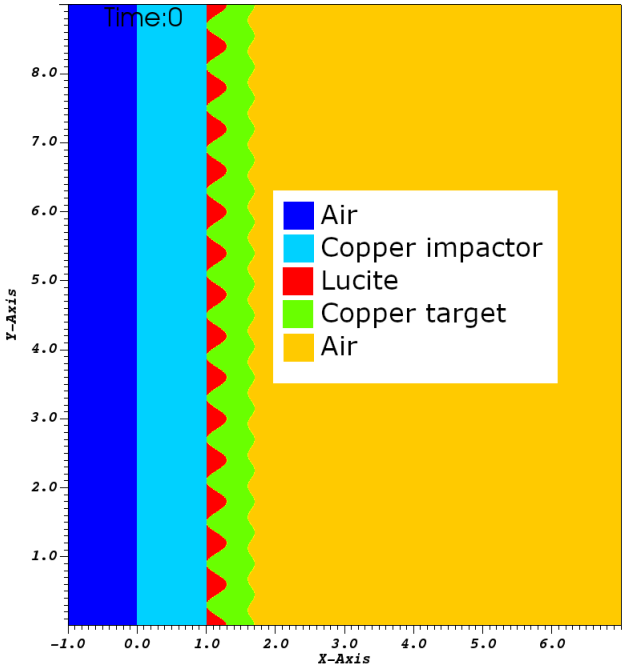

Figure 4: This shows the geometry and materials of the double sine wave impact.

The linear shaped charge hydrodynamic simulations used a mesh of 11,000 quadratic (Q_2Q_1) elements. The full field solutions were saved on a 512×1664 uniform grid. The results contain the following fields: density of the liner, density, velocity x, velocity y, energy, pressure, and the volume fraction of the liner. The simulations were run from initial detonation to t=20 μs, and results were saved every 0.5 μs. The high explosive was modeled using the Cochran-Tarver ignition and growth reactive flow model.

3.3 Rayleigh-Taylor Study

The final study is a single-mode Rayleigh-Taylor instability (RTI). The setup for the initial RTI was based off of the example in Athena++[54] and the work of Liska and Wendroff [55]. The problem has a x domain of $[-1/6, 1/6]$ cm and a y domain of $[-0.5, 0.5]$ cm. A heavier ideal gas is placed on-top of a lighter ideal gas. An initial velocity was applied in y direction to seed the instability growth as

$$v_y = v_{\text{init}}(u_0 * (1 + \cos(6\pi x))(1 + \cos(2\pi y))/4) \tag{4}$$

with $u_0 = 0.01$ cm/s. There is a constant gravitational acceleration of 1.0 cm/s^2.

The simulations were parameterized for three physical parameters: density ratio of the two gases, the heat capacity ratio of γ for both gases, and the initial velocity of v_{init}. The parameters were randomly sampled in the following ranges respectively: $[1.1, 6.7]$, $[1.1, 1.6]$, and $[0.6, 10.0]$.

It is well known that this RTI problem becomes turbulent with decreasing feature size as time progresses. Thus, the simulation based on the Euler equations is never fully resolved. Our solutions were computed on a 192×64 grid using cubic (Q_3Q_2) elements.

The full field solutions were saved on a 768×256 uniform grid. The results contain the following fields: density, velocity x, velocity y, energy, pressure, and the materials. The simulations were run to 10 s, and results were saved every 0.2 s (with 51 total snapshots per simulation).

4 Machine Learning

The purpose of the proposed Machine Learning (ML) model is to learn the mapping Eq 1 which maps the parameterized simulation to 2D image arrays of physics fields. The ML architecture originates from a generative model based on deep, but sparse, convolutional layers. The typical application of these deep convolutional networks is in unsupervised

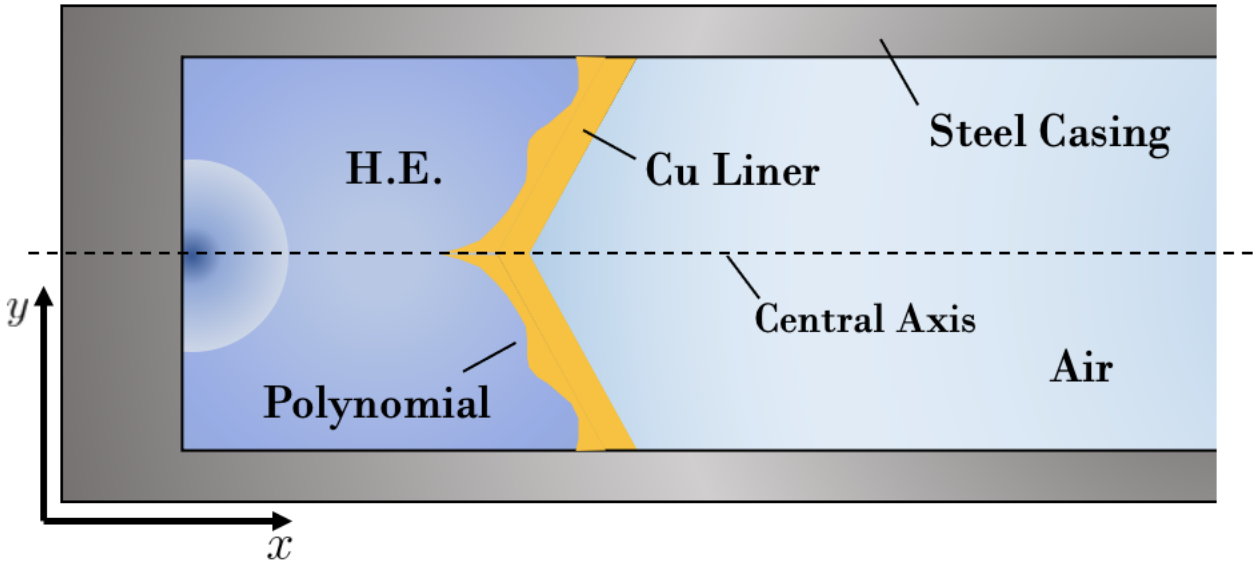

Figure 5: The planar shaped charge in this study consists of a steel case, explosive, and a copper liner. One face of the copper liner is fixed, the other face is parameterized with a polynomial; the polynomial coefficients can be optimized to enhance performance.

learning of pictures e.g. human faces[24]. In these applications, the faces have no labels and no known *a-prior* parameterization. In our work, neural network layers are added to the generative model to connect the simulation parameters to the layers that have the smallest image representation. This allows us to take just about any generative model architecture and perform regression to learn the parameterized simulation solutions.

The DCGAN [24] generator architecture is perhaps one of the simplest generative models to learn the mapping of parameters to full-field solutions. The models require only a couple milliseconds for inference which is approximately a million times faster than a full hydrodynamic solution. The model uses transposed convolution layers (sometimes also called inverse convolutional, or deconvolutional [56]). The first transposed convolutional layer creates an initial kernel representation (e.g. 4×4 pixels) of the entire field. Then, each subsequent transposed convolutional layer doubles the previous layer's full field representation (e.g.$4 \times 4 \rightarrow 8 \times 8$). These layers can be stacked until the output is the correct size of the final images.

Our use of the DCGAN model differs in a couple key characteristics from [57]. We discovered that we could achieve superior accuracy by using the image channels (normally used for e.g. red, green, blue) to learn multiple physical fields (e.g. density, velocity, pressure) rather than using an entirely separate generator models for each physical field. The model takes advantage of the many correlations between fields. Each physical field has different units that may be orders of magnitude different, which would cause issues with the transposed convolution layers. To address the imbalance of units within the physical fields, we propose the model learn the following linear mapping as the final layer,

$$\alpha \boldsymbol{X} + \beta \tag{5}$$

where α and β are vectors that are each the size of the number of l physical fields. This allows to output the physical fields of different orders of magnitude into the correct units, which would be important when applying physics informed constraints[58].

A visual overview of the model architecture for the PCHIP impact is shown in Figure 6, and ML models for the other datasets was similar. The initial kernel was 4×4, and the number of channels ranged from 1024 down to the final 6 fields. Each transposed convolution layer is followed by batch norm [59] and ReLU activations [60]. The layer by layer[5] breakdown with learnable parameters for each ML model is shown in Appendix A. All models generally have around 80 million learnable parameters.

The number of features in the model were limited to between 128 through 1024 (in the dimension where RGB is typically used in images). Reducing to 64 through 512 reduces model quality while also reducing the total number of

[5]The PyTorch layers, shapes, and learnable parameters are outputs of a packaged called torchinfo available online at `https://github.com/TylerYep/torchinfo`

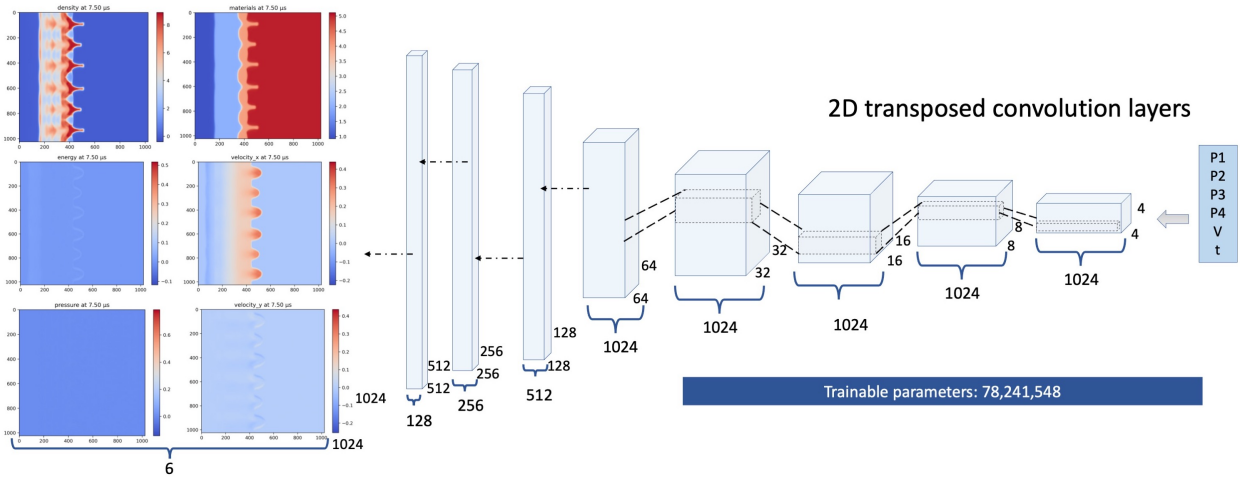

Figure 6: The convolution neural network architecture of the generator model starts with a 4×4 kernel (right). Each subsequent layer doubles pixels until the final 1024×1024 fields are created (left). This is actual ML output for the PCHIP impact, showing fields of density, energy, pressure, materials, velocity x, and velocity y.

learnable parameters. Increasing these features to 256 through 2048 greatly increases the total number of learnable parameters while making a negligible impact on accuracy, however this remains an area of active investigation. It is still not well understood how the sizes in this dimension affect the accuracy of the model, as well as which datasets would benefit from the additional features.

The models were implemented using PyTorch [61]. Training was performed using Adam[62] to minimize the mean absolute error. All models were trained using distributed data-parallel training and a zero redundancy optimizer [63]. There is a copy of the model on each GPU which processes some unique n mini-batch fraction of the data. The mini-batch size was chosen as the largest value that maximizes the available GPU memory. The effective batch size for k GPUs is given by nk. An epoch represents one complete training iteration through the entire dataset. The learning rate η was selected to be $1e-5$. The learning rate scaling is inspired by Goyal et al. [64] and follows

$$\eta_{\text{effective}} = \eta nk \tag{6}$$

which should result in similar models when trained on different resources. The models were trained on NVIDIA 16GB V100 GPUs on the Lassen HPC using 12 hour allocations.

5 Results

Machine learning (ML) models were trained on the four datasets. The training details including batch size, number of GPUs, learning rates, and effective batch size are reported in Table 2. A single image array from the Rayleigh-Taylor dataset was roughly a third of the size of the images from the other datasets, and thus could support approximately three times the batch size on a single GPU compared to the others.

The training errors for each epoch are shown in Figure 7. All models were trained on mean absolute error, and there is strong correlation with mean absolute error, mean squared error, and L-infinity error. Diminishing training returns with respect to additional epochs is observed with all models. This is most pronounced in the Rayleigh-Taylor case which shows negligible training improvements for the last 100 epochs.

One necessary aspect that enables the ML models to be machine portable is having a reasonable file size. Table 3 shows the file sizes of the ML model weights compared to the dataset size. The ML models were generally around 0.9

Table 2: Machine learning training hyperparameters for each hydrodynamics study..

Name	Batch size	# of GPUs	η	$\eta_{\text{effective}}$	effective batch size
PCHIP impact	14	160	1e-5	2.24e-2	2240
Double sine wave	14	40	1e-5	5.6e-3	560
Linear shaped-charge	14	60	1e-5	8.4e-3	840
Rayleigh-Taylor	48	32	1e-5	1.536e-2	1536

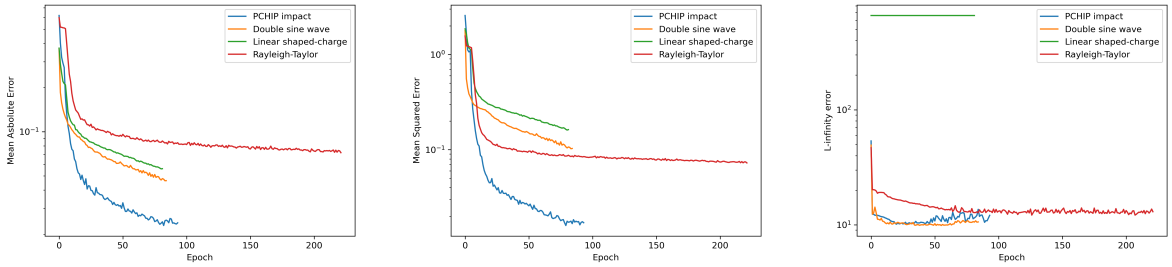

Figure 7: The mean absolute error, mean squared error, and L-infinity training errors for each dataset.

Table 3: Dataset vs machine learning model size in gigibytes.

Name	Dataset size (GiB)	ML model size (GiB)
PCHIP impact	2098	0.900
Double sine wave	729	0.896
Linear shaped-charge	1151	0.709
Rayleigh-Taylor	328	0.899

gigibytes, while the dataset were on the order of a tebibyte[6]. In this aspect, the ML model can be thought of as a lossy compression of the data. In these cases, the achieved compression factor was around 1,000 times. The dataset sizes discussed here is the size on disk of the hdf5 files which utilize gzip lossless compression.

For all models, a simulation not in the training set is compared to the ML model's density predictions in Figures 8-11. Density is just one of the multi-field outputs from these models. The density field provides a good illustration on how the materials are mixing in the hydrodynamic simulations. All of the ML predictions appear to be an excellent representation of the actual field. It is nearly impossible to notice the errors in the ML predictions of the PCHIP impact. The double sine wave impact has much finer Richtmyer-Meshkov instabilities than the PCHIP impact, and the ML model blurs a high-low density interface of the finer instabilities (e.g. at the copper-lexan-copper interface). The ML predictions of the linear shaped charge look excellent, with only some minor fine details missing on the jet as it forms.

The Rayleigh-Taylor results (shown in Figure 11) are perhaps the most interesting. For all times the ML prediction is tracking the overall interface between the high-low density fluids. However, as the simulation time progresses the overall interface becomes significantly more complicated with many very fine perturbations along the mixing interface. The ML model appears incapable of modeling these finer features and appears to blur the many very fine high-low density features with a smooth "middle" density prediction. This spatial averaging behavior of the ML predictions is similar to the expected behavior of a RANS solver [65].

Additional results showing predictions and errors for each field are shown in Appendix B. Predictions are compared to the simulation results for a simulation not in the training set. All model fields are shown at different times within the simulation domain. The mean absolute error for each field is roughly two orders of magnitude better than the ML model initiated with random weights. Additional results showing predictions and errors for each field are shown in Appendix B. Predictions are compared to the simulation results for a simulation not in the training set. All model fields are shown at different times within the simulation domain. The mean absolute error for each field is roughly two orders of magnitude better than the ML model initiated with random weights.

6 Applications

The trained machine learning (ML) models have several valuable applications. The ML model enables real time visualization, which provides novel ways for users to explore parameterized solutions. An analytical expression is presented as a demonstrator of applying this methodology to further problems outside of hydrodynamics. This demonstrator is very cheap to generate both training and validation data. Optimization algorithms can be used with ML surrogate models to search for initial conditions that lead to desirable RMI characteristics at later times. This was demonstrated with the PHCIP ML model to find initial conditions that lead to either the shortest or longest RMI interface length. Lastly, it is demonstrated that an ensemble of models is useful to estimate the error in ML predictions on the linear shaped charge problem.

[6]One tebibyte is 1.099511627776 terabytes.

6.1 Real time visualization

A desired application that the ML models enable is real-time visualization and exploration. The ML model can be queried at any place within the parameterized space. These full-field solutions out of the ML model can be thought of as an interpolation in the parameterized space. While it would be possible to quickly view results from an ensemble dataset in a similar fashion, the ML model offers the ability to quickly explore anywhere in the high dimensional space. Additionally, inference from the ML model can be performed on a portable laptop computer, while both the dataset and simulations are solidly in the realm of HPC. The full-field inference for the PCHIP impact model takes 0.2 s using an Apple M2 Max on battery power.

A real-time visualization tool was created using napari[66]. Static images taken from the tool are shown in Figure 12. The user is presented with 4 inputs to the Rayleigh-Taylor ML model (density ratio, heat capacity ratio γ, initial velocity, and time) as a slider interface. The user can either click or slide through the input parameters and view the resulting full-field solutions in real-time. There is a dropdown menu to switch between which output field is displayed (e.g. density, velocity, pressure). Inference was done using an AMD MI250X, which takes an average time of 0.002 ms using float16 precision. Having fast ML inference is key to visualize the full-field solutions in real-time.

Videos of the slider interface while doing live ML inference are included in the supplemental material for all four ML models. The interface allows for the interpolation of the physical fields at any point within the ensemble domain. The predictions from the tool look excellent. With the linear shaped charge model, users can interactively explore how the liner shape influences the long-term jet development, to build intuition on how these parameters work together to enhance or mitigate the jet. With the double sine wave and PCHIP models, users can gain intuition on how the initial shape of the copper target influences the development of the RMI. This interactive ability allows users to explore and interpolate an ensemble of hydrodynamic simulations to further their understanding of complex physical instabilities.

The real-time visualization tool also makes it easy to explore how well ML models behave when extrapolating outside of the ensemble domain. While it would be incredibly desirable for a fast-running hydrodynamics tool to extrapolate, unfortunately these ML models possess no ability to extrapolate the hydrodynamic instabilities. As all input parameters are extrapolated, the ML model's predictions go to zero everywhere. Even with subtle extrapolations like 10%, users will see predictions quickly begin to look unrealistic as interfaces begin to blur with cloud-like pockets of material disappearing. The material interfaces do not move when extrapolating in simulation time, but rather begin to disappear. In summary as the model begins to extrapolate then the predictions quickly look both nonphysical and unbelievable.

6.2 Analytical expression outside of hydrodynamics

The focus on this work has been mostly on applications in hydrodynamics, however the ML methodology is applicable to other domains. As a demonstrator, we applied this methodology on a test dataset based on an analytical expression.

Consider the following equation

$$F(x,y,\boldsymbol{\beta}) = \big(\beta_0 \sin(2\pi x) + \beta_1 \sin(4\pi x)\big)\big(\beta_2 \sin(2\pi y) + \beta_3 \sin(4\pi y)\big) \tag{7}$$

where x and y were sampled on a fixed grid of 512×512 from zero to one. The ML model attempts to learn F as a function of the four $\boldsymbol{\beta}$ parameters. The dataset was constructed by taking eleven samples per dimension such that

$$\boldsymbol{\beta} \in [-1.0, -0.8, -0.6, -0.4, -0.2, 0.0, 0.2, 0.4, 0.6, 0.8, 1.0] \tag{8}$$

for a total of 14,641 image arrays. This analytical expression is interesting because it is both spatially smooth while exhibiting topology changes with respect to the four $\boldsymbol{\beta}$ parameters.

A model was trained on this dataset of 3.8 billion floating point values. To demonstrate how well the model interpolates, random points were selected from

$$\boldsymbol{\beta} \in [-0.9, -0.7, -0.5, -0.3, -0.1, 0.1, 0.3, 0.5, 0.7, 0.9] \tag{9}$$

and Figure 13 shows the results. The ML model appears to have learned the analytical expression quite well, as it is difficult to spot the differences between the ML predictions and the actual solution. A video of running the real-time visualization tool on a laptop for this ML model is included in the supplementary material.

6.3 Optimization of RMI

It is possible to quickly solve optimization problems on the ML model to explore how parameters influence the complicated physical phenomena. Consider the interesting case of maximizing or minimizing the RMI on the parameterized PCHIP high velocity impact. The arc length of the copper and air interface can be used to quantify the

size of the RMI. The smaller the length of the interface (when perturbed) is an indication of a smaller influence of RMI. The larger the length of the interface indicates larger or more RMI. Figure 14 shows how the interface can be constructed with a threshold on ML predictions, which is then used to quantify the magnitude of RMI.

Optimization problems were set up on the PCHIP ML model to suppress and enhance the RMI. The optimization problems can be solved quickly on the ML model as exploring the surrogate is significantly faster than running the hydrodynamics code. The objective of the optimization problems was either to minimize or maximize the arc length of the copper-air interface at the final time, by changing the initial geometry perturbations. For simplicity the initial velocity was fixed at 0.2 cm/μs. For these described conditions, the minimal RMI should occur when there is no perturbation on the initial geometry (e.g. $\boldsymbol{x} = [x_0 = x_1 = x_2 = x_3]$). Likewise, the largest RMI should occur with the largest perturbations in initial geometry (e.g. $\boldsymbol{x} = [-0.25, 0.25, -0.25, 0.25]$ cm, $\boldsymbol{x} = [0.25, -0.25, 0.25, -0.25]$ cm). As there is expected to be multiple local optima with the optimization formulation, we run 100 L-BFGS-B [67] optimizations from different random starting points. The geometry was bounded to the same domain as the original sampling.

The ML predictions from minimizing the arc length of the copper-air interface on the PCHIP ML model are shown in Figure 15. The best found solution was $\boldsymbol{x} = [-0.0593, -0.0726, -0.0617, -0.0550]$ cm which is close to the theoretical best solution of $\boldsymbol{x} = [x_0 = x_1 = x_2 = x_3]$. The minuscule geometry perturbation results in no RMI.

The ML predictions from maximizing the arc length of the copper-air interface is shown in Figure 16. It is seen that the largest possible initial perturbation resulted in many large RMI. The best found solution was at $\boldsymbol{x} = [0.25, -0.25, 0.25, -0.25]$ cm which was one of the anticipated outcomes from this optimization.

6.4 Error estimations via cross-validation ensemble

It is possible to estimate the generalization error of the ML model with a cross-validation ensemble [68]. Different models (of the same architecture) are trained on different subsets of the training data. The disagreement in the predictions from the ensemble can be used to approximate the error in the ML models. Note that the different models can be in agreement, but still be wrong, as this cross-validation ensemble does not measure *bias* of the model.

An ensemble of ML models were trained on the linear shaped charge dataset. Using 5-fold cross validation, each model was trained on a different fraction of 4/5 the entire dataset. Figure 17 shows plots of the average and standard deviation from the ensemble, using a simulation that none of the models were trained on. The absolute error was taken between the average ensemble prediction and the actual simulation result. The standard deviation from the ensemble correlates very well to the absolute error, indicating that the disagreement in the ensemble is a reasonable approximation of the error. At early times, the standard deviation is an excellent representation, highlighting just a few pixels around the copper interface. As the jet grows, the standard deviation is largest along the interface boundaries, which corresponds visually well to where the errors are the largest.

The ensemble results appear to estimate the error well when interpolating, however this begins to quickly breakdown when extrapolating. Figure 18 shows the ensemble average and standard deviation while extrapolating in time. As the temporal extrapolation increases, the quality of the ensemble prediction begins to quickly deteriorate, becoming blurry and nonphysical. In these cases the ensemble's standard deviation significantly increases in much of the domain. There are non-physical regions where the average prediction is showing the density of air increasing. This is non-physical and should be an error in the ML predictions. However, the standard deviation is small despite this mistake. The error estimation begins to fail when the entire ensemble wrongly predict the same values.

7 Conclusion

Performing ensembles of simulations is one way to understand the complex sensitivity of physical instabilities to initial conditions. We present results from four ensemble datasets involving the Richtmyer-Meshkov Instability (RMI) and Rayleigh-Taylor instabilities. A machine leaning (ML) model framework was proposed to learn the full-field solutions as functions of initial physical and geometric conditions. The ML model weights were a thousand times smaller than the datasets (as a form of lossy compression). Additionally inference from the ML model can be millions of times faster than a full hydrodynamic solution, while also not requiring HPC resources demanded by the ensemble. The ML model enables real-time visualization of the instabilities by sliding through both initial conditions and time. This enables users to explore and gain intuition about complicated hydrodynamic relationships. Such tools also enable the dissemination of hydrodynamic simulations and results to additional audiences (e.g. experimentalists) with reduced barriers of entry (e.g. high performance computing). Lastly, the output of the ML model is fully differentiable with respect to the input, which can be useful for applications in design optimization or uncertainty quantification.

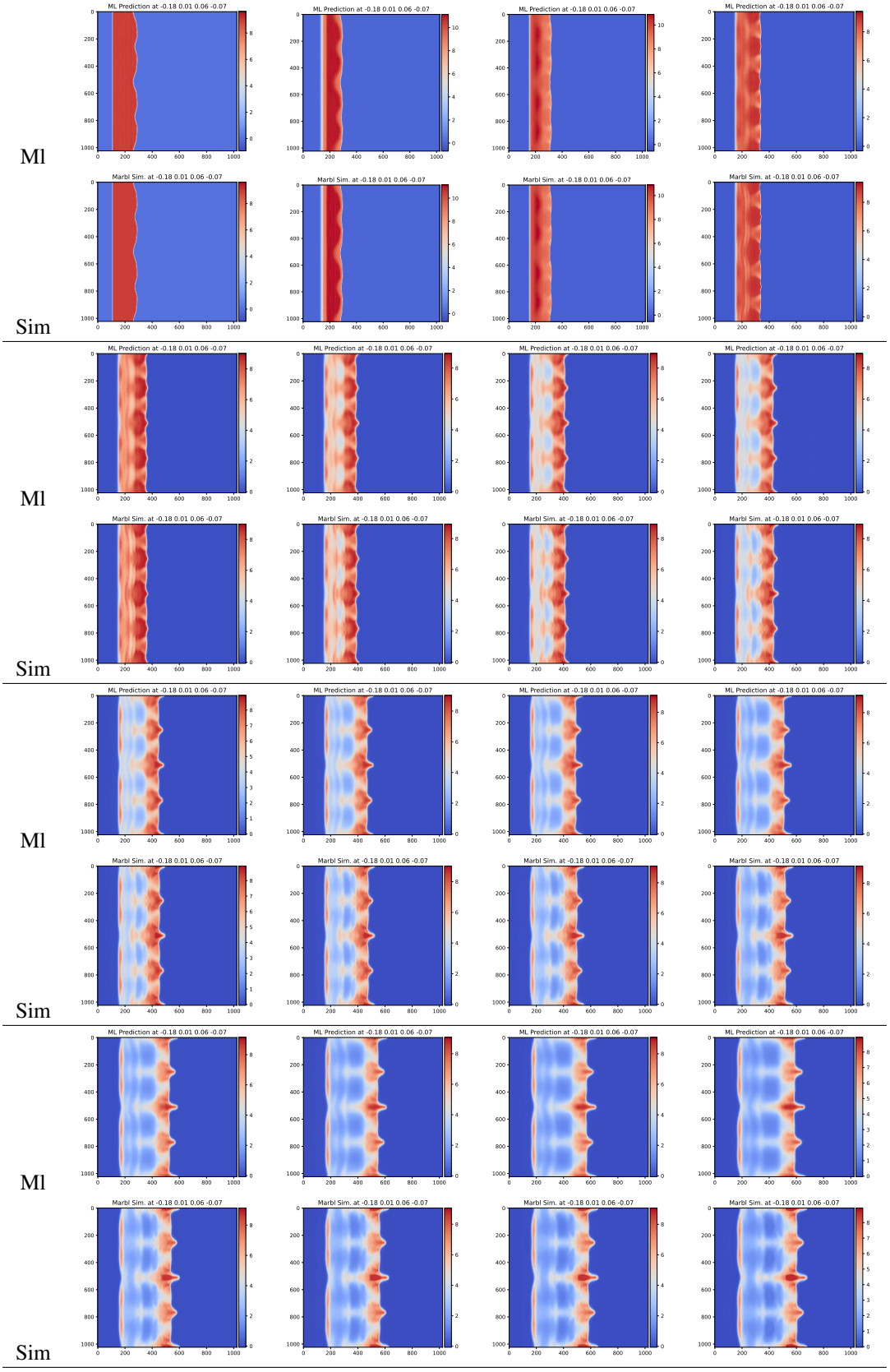

Figure 8: Figure of density predictions and truth for the PCHIP impact. Rows show machine learning model (ML) next to simulation (Sim) results.

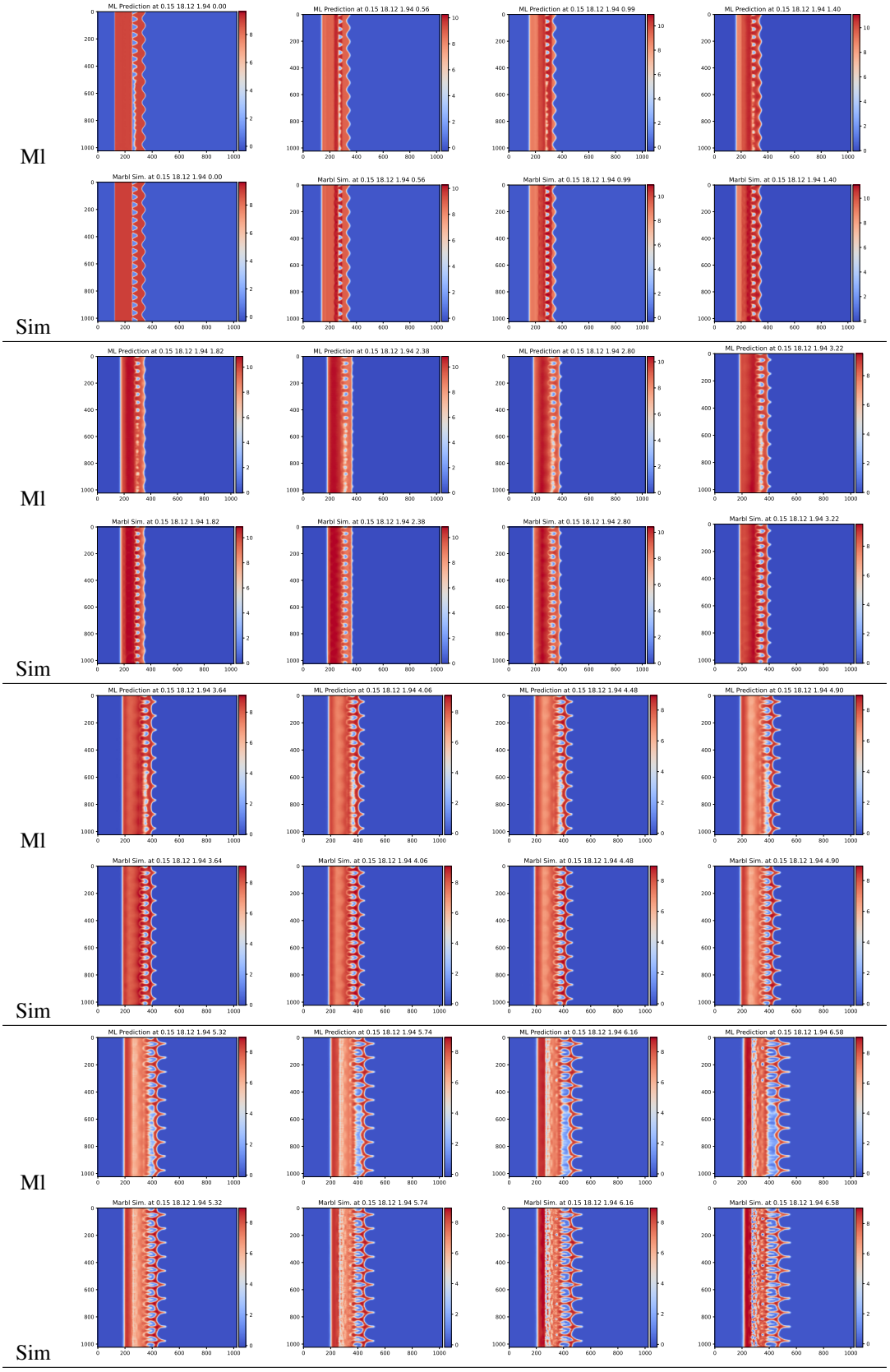

Figure 9: Figure of density predictions and truth for the double sine wave impact. Rows show machine learning model (ML) next to simulation (Sim) results.

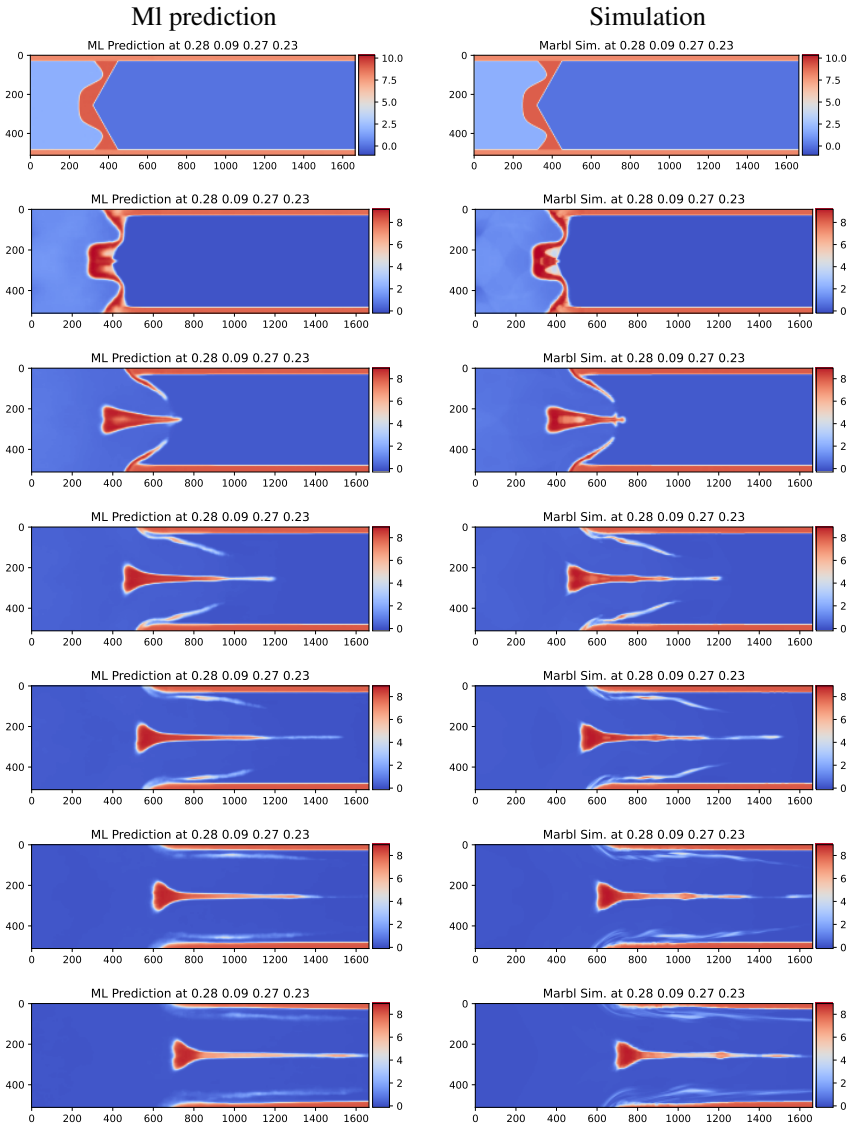

Figure 10: Figure of density predictions and truth for the linear shaped charge.

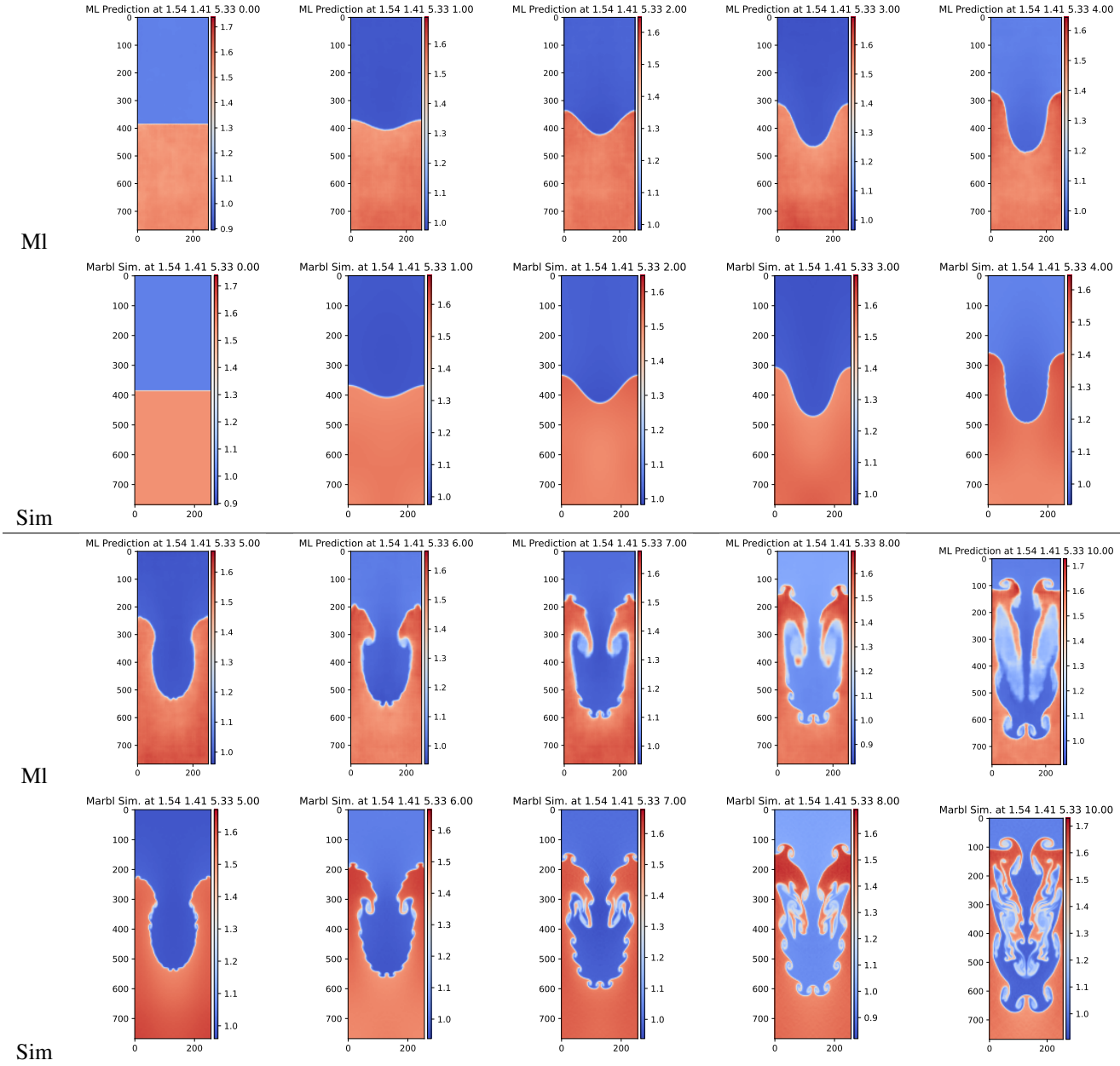

Figure 11: Figure of density predictions and truth for the Rayleigh-Taylor simulation. Rows show machine learning model (ML) next to simulation (Sim) results.

Figure 12: Real-time visualization of the Rayleigh-Taylor model showing the full-field density solution. The first two rows show a user sliding through time for a particular parameterization that the model was not trained on. The last row shows the user sliding through the input velocity at a fixed time. These are still frames from a video of a user interacting with the tool.

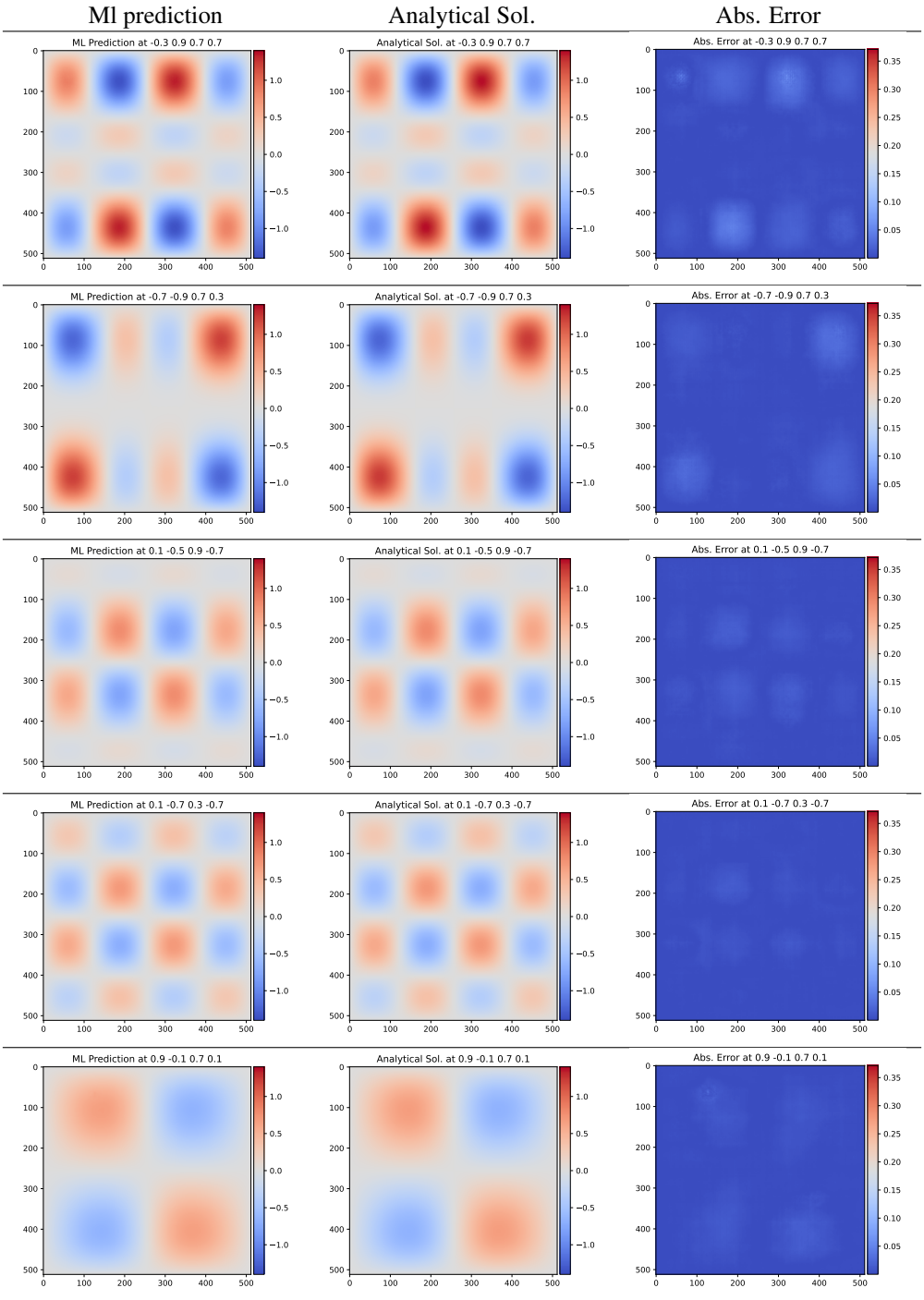

Figure 13: Figure of predictions and analytical solutions for the analytical expression. Predictions shown are interpolations, meaning the model was not trained on those exact β values. The model appears to be an excellent representation of the expression.

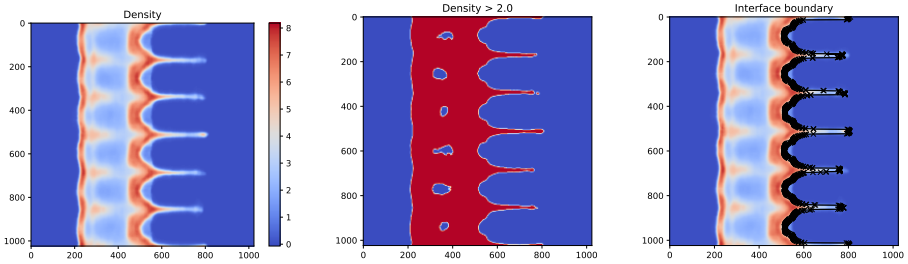

Figure 14: Demonstration on how a threshold is applied to find the interface of copper and air from a ML prediction on the PCHIP model. Left shows ML prediction of density field, middle shows threshold on density greater than 2, and right shows how the edge interface is constructed using the boundary from the threshold.

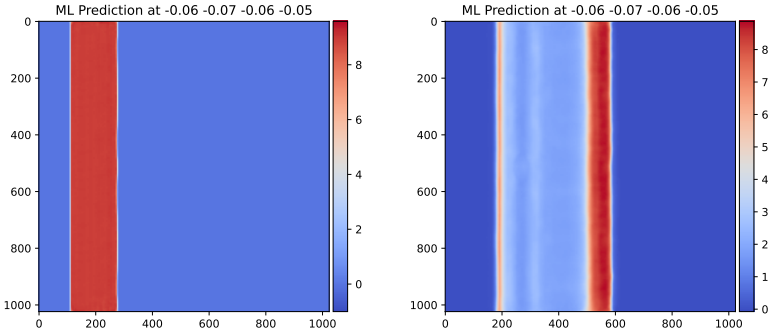

Figure 15: ML predictions from the optimization results of minimizing the RMI show a flat initial interface (left) results in no RMI at latter time (right).

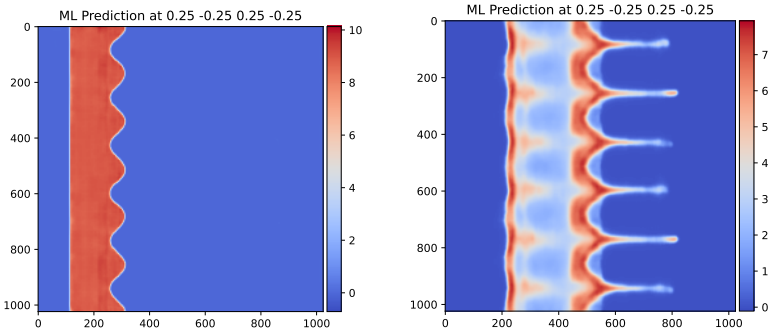

Figure 16: ML predictions from the optimization results of maximizing the RMI shows a large initial interface (left) results in large RMI at latter time (right).

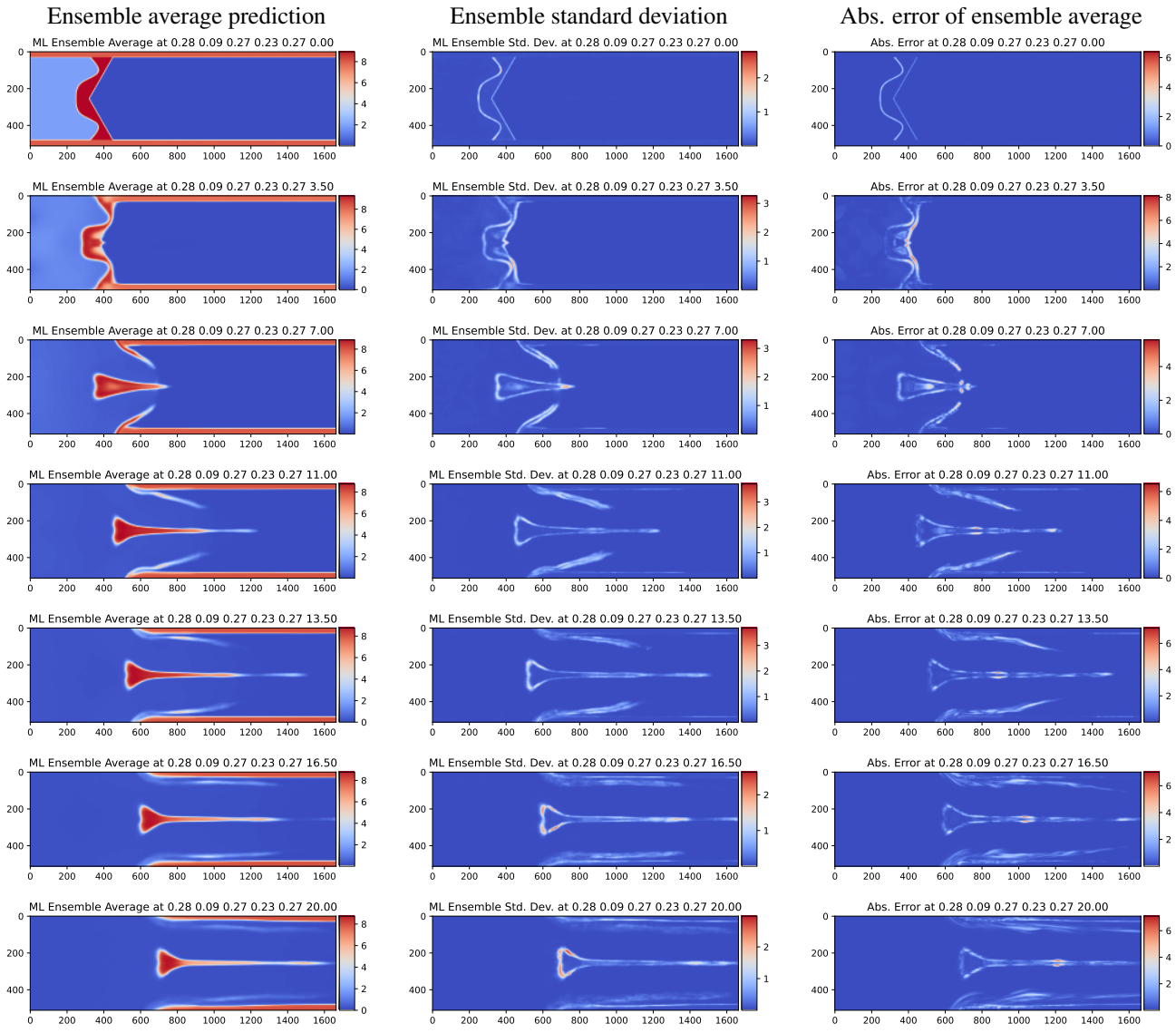

Figure 17: Ensemble average and standard deviation predictions for a simulation not in the training set. The standard deviation appears to reasonably correlate to the absolute error of the ensemble average.

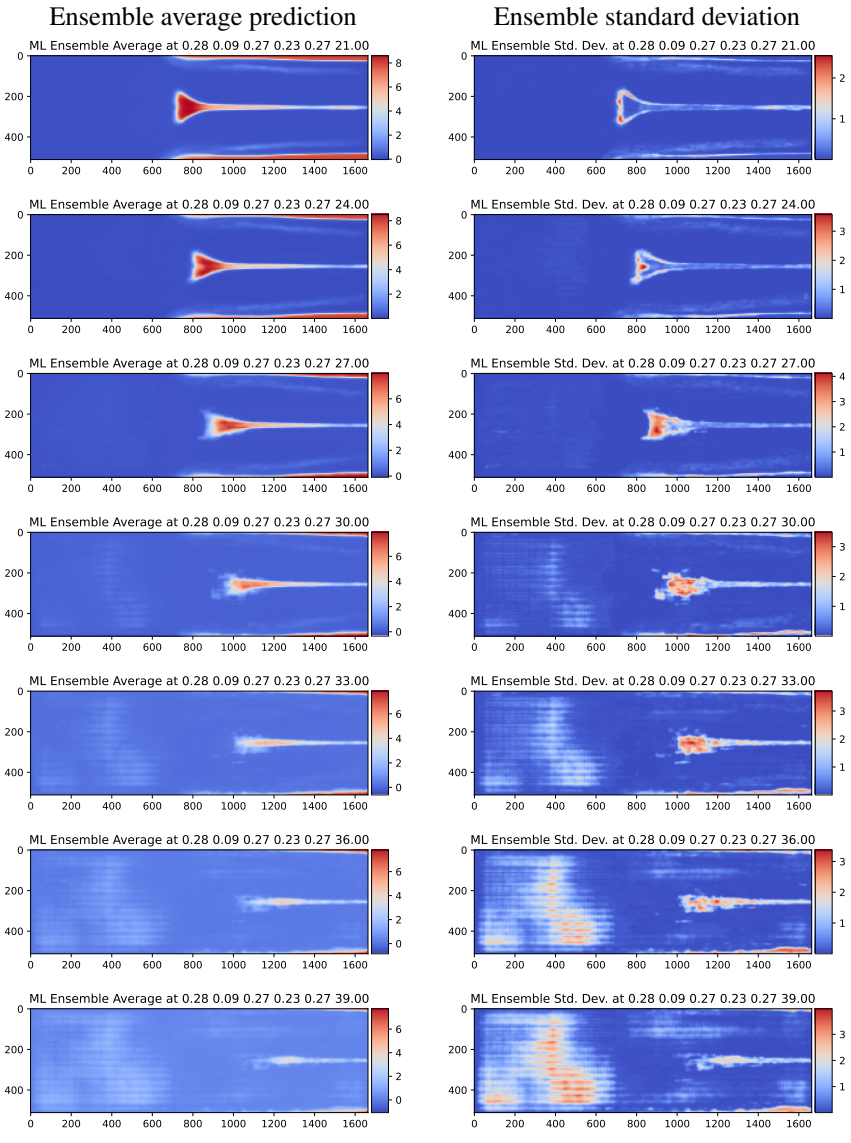

Figure 18: Ensemble average and standard deviation when extrapolating in time. The end time of simulations in the training set was 20, and the extrapolations shown go from 21 to 39.

Acknowledgments

This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344 and was supported by the LLNL-LDRD Program under Project No. 21-SI-006.

We would like to acknowledge Kareem Hegazy for pointing out that placing the multiple physics fields into the image channel would enable the model to learn the correlations between physical fields, which thus resulted in more accurate ML models.

Data Availability Statement

The Rayleigh-Taylor and PCHIP datasets and ML model weights will be available upon request.

A Model architectures

Tables **??**-7 shown the layer-by-layer breakdown of the number of learnable parameters for each ML model, in addition to the architecture of each layer.

Table 4: PCHIP impact ML model architecture.

Layer name	Output shape	# of learnable parameters
Input parameters	[6, 1, 1]	-
ConvTranspose2d → BatchNorm2d → ReLU	[1024, 4, 4]	100,352
ConvTranspose2d → BatchNorm2d → ReLU	[1024, 8, 8]	16,779,264
ConvTranspose2d → BatchNorm2d → ReLU	[1024, 16, 16]	16,779,264
ConvTranspose2d → BatchNorm2d → ReLU	[1024, 32, 32]	16,779,264
ConvTranspose2d → BatchNorm2d → ReLU	[1024, 64, 64]	16,779,264
ConvTranspose2d → BatchNorm2d → ReLU	[512, 128, 128]	8,389,632
ConvTranspose2d → BatchNorm2d → ReLU	[256, 256, 256]	2,097,664
ConvTranspose2d → BatchNorm2d → ReLU	[128, 512, 512]	524,544
ConvTranspose2d	[6, 1024, 1024]	12,288
Linear map	[6, 1024, 1024]	12
Total number of learnable parameters		78,241,548

Table 5: Double sine wave ML model architecture.

Layer name	Output shape	# of learnable parameters
Input parameters	[4, 1, 1]	-
ConvTranspose2d → BatchNorm2d → ReLU	[1024, 4, 4]	67,584
ConvTranspose2d → BatchNorm2d → ReLU	[1024, 8, 8]	16,779,264
ConvTranspose2d → BatchNorm2d → ReLU	[1024, 16, 16]	16,779,264
ConvTranspose2d → BatchNorm2d → ReLU	[1024, 32, 32]	16,779,264
ConvTranspose2d → BatchNorm2d → ReLU	[1024, 64, 64]	16,779,264
ConvTranspose2d → BatchNorm2d → ReLU	[512, 128, 128]	8,389,632
ConvTranspose2d → BatchNorm2d → ReLU	[256, 256, 256]	2,097,664
ConvTranspose2d → BatchNorm2d → ReLU	[128, 512, 512]	524,544
ConvTranspose2d	[3, 1024, 1024]	6,144
Linear map	[3, 1024, 1024]	6
Total number of learnable parameters		78,202,630

Table 6: Linear shaped charge ML model architecture.

Layer name	Output shape	# of learnable parameters
Input parameters	[6, 1, 1]	-
ConvTranspose2d → BatchNorm2d → ReLU	[1024, 4, 13]	321,536
ConvTranspose2d → BatchNorm2d → ReLU	[1024, 8, 26]	16,779,264
ConvTranspose2d → BatchNorm2d → ReLU	[1024, 16, 52]	16,779,264
ConvTranspose2d → BatchNorm2d → ReLU	[1024, 32, 104]	16,779,264
ConvTranspose2d → BatchNorm2d → ReLU	[512, 64, 208]	8,389,632
ConvTranspose2d → BatchNorm2d → ReLU	[256, 128, 416]	2,097,664
ConvTranspose2d → BatchNorm2d → ReLU	[128, 256, 832]	524,544
ConvTranspose2d	[128, 512, 1664]	14,343
Linear map	[6, 1024, 1024]	14
Total number of learnable parameters		61,685,518

Table 7: Rayleigh Taylor ML model architecture.

Layer name	Output shape	# of learnable parameters
Input parameters	[4, 1, 1]	-
ConvTranspose2d → BatchNorm2d → ReLU	[1024, 3, 1]	14,336
ConvTranspose2d → BatchNorm2d → ReLU	[1024, 6, 2]	16,779,264
ConvTranspose2d → BatchNorm2d → ReLU	[1024, 12, 4]	16,779,264
ConvTranspose2d → BatchNorm2d → ReLU	[1024, 24, 8]	16,779,264
ConvTranspose2d → BatchNorm2d → ReLU	[1024, 48, 16]	16,779,264
ConvTranspose2d → BatchNorm2d → ReLU	[512, 96, 32]	8,389,632
ConvTranspose2d → BatchNorm2d → ReLU	[256, 192, 64]	2,097,664
ConvTranspose2d → BatchNorm2d → ReLU	[128, 384, 128]	524,544
ConvTranspose2d	[6, 768, 256]	12,288
Linear map	[6, 768, 256]	12
Total number of learnable parameters		78,155,532

B Model results

B.1 PCHIP impact results

The mean absolute error for each field separately in training of the PCHIP high velocity impact dataset is shown in figure 19. Samples of ML predictions for multiple fields are plotted in figures 20–22.

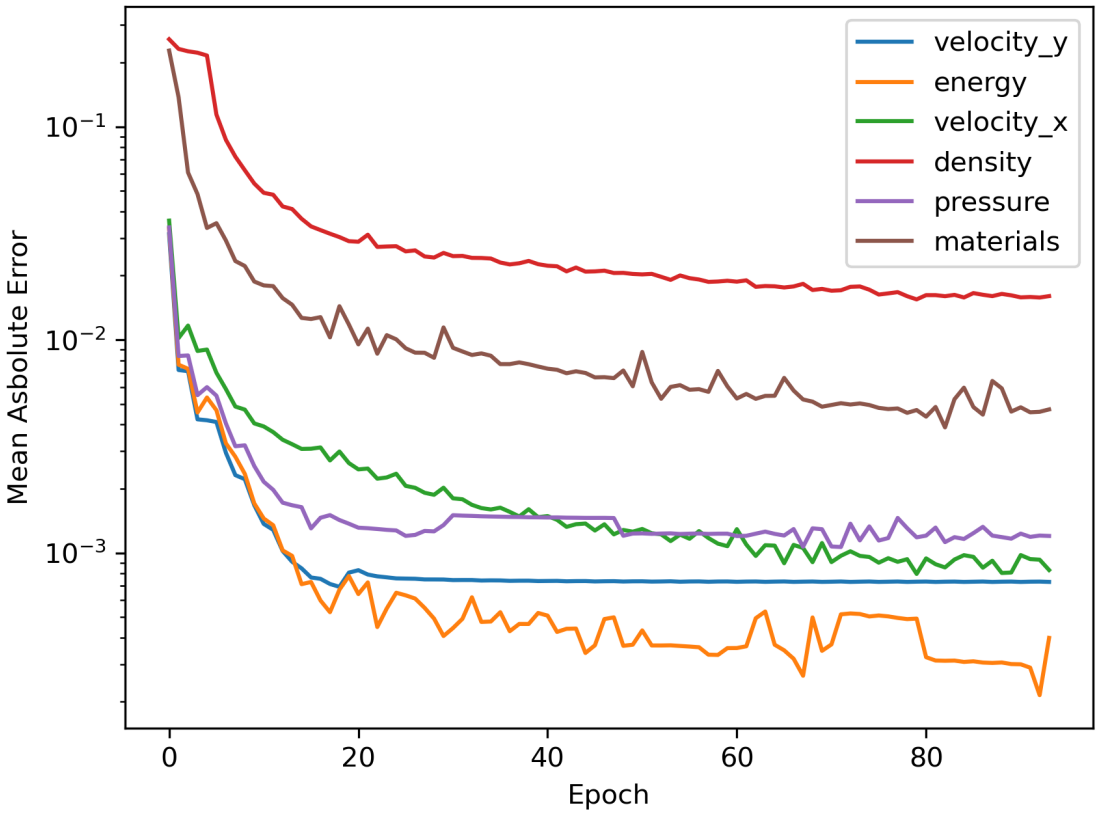

Figure 19: The epoch vs mean absolute error for each field while training the PCHIP impact model.

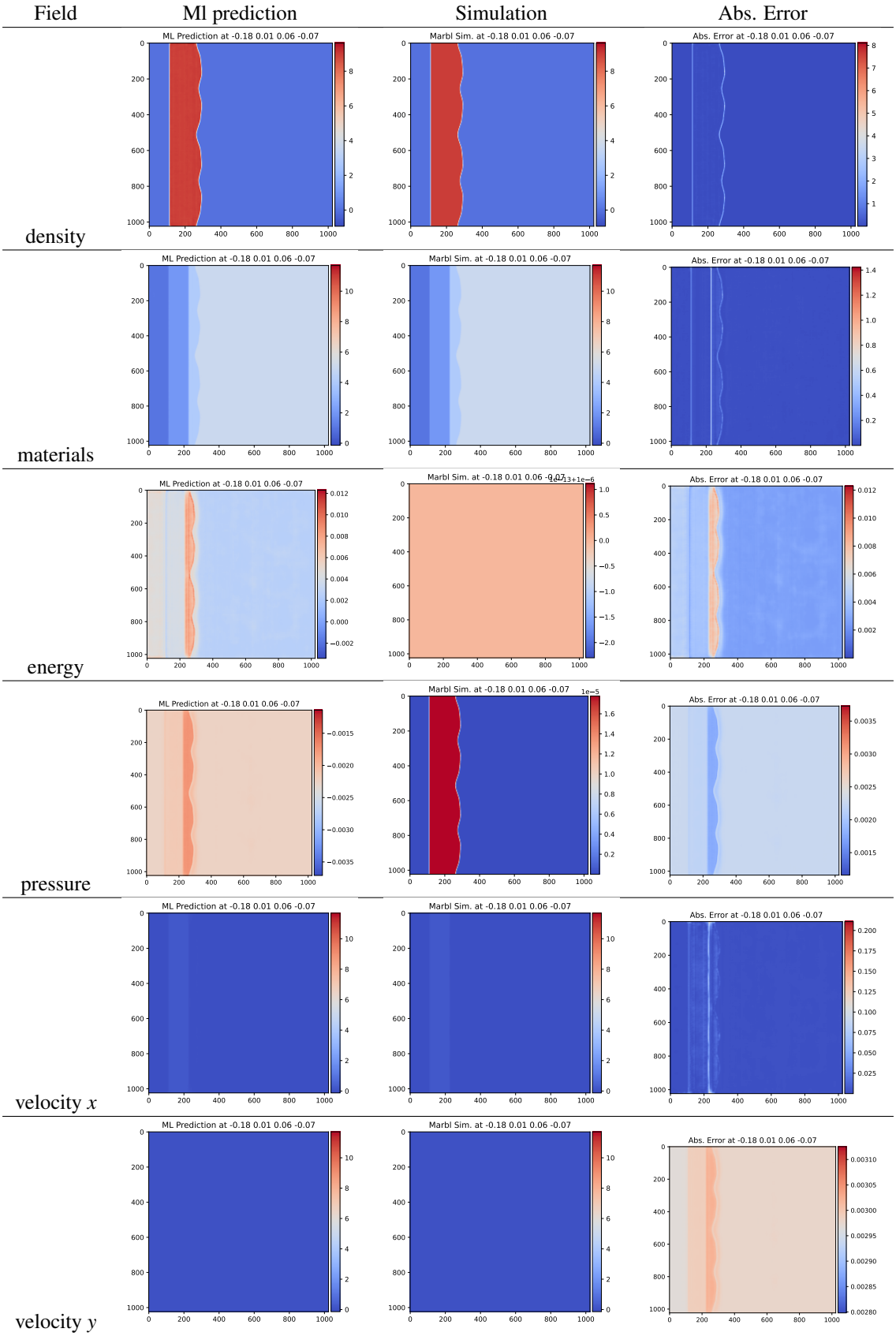

Figure 20: Figure of predictions, truth, and absolute error for linear shaped charge predictions at t=0.

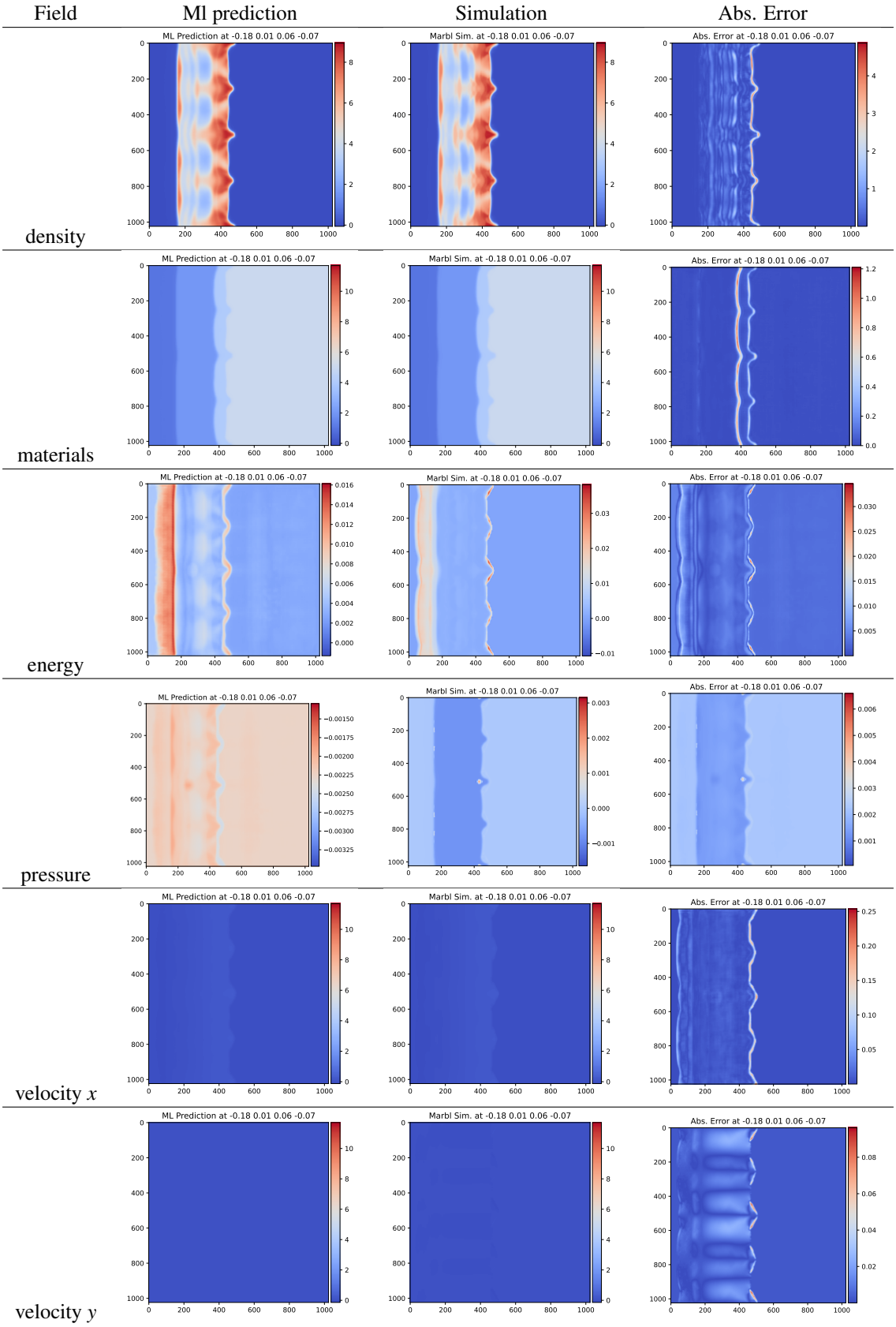

Figure 21: Figure of predictions, truth, and absolute error for linear shaped charge predictions at t=7.

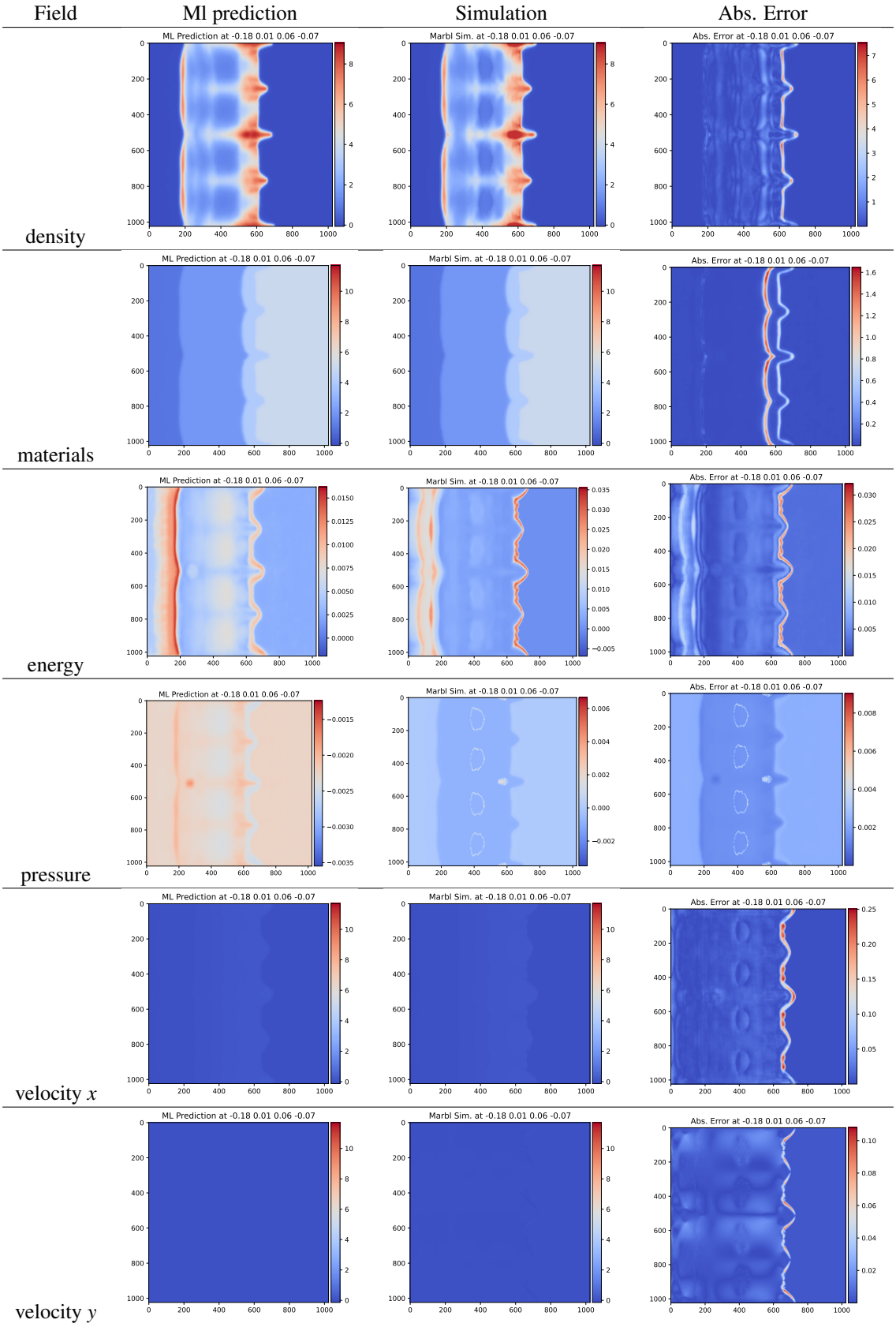

Figure 22: Figure of predictions, truth, and absolute error for linear shaped charge predictions at t=15.

B.2 Double sine wave results

The mean absolute error for each field separately in training of the double sine wave high velocity impact dataset is shown in figure 23. Samples of ML predictions for multiple fields are plotted in figures 24-26.

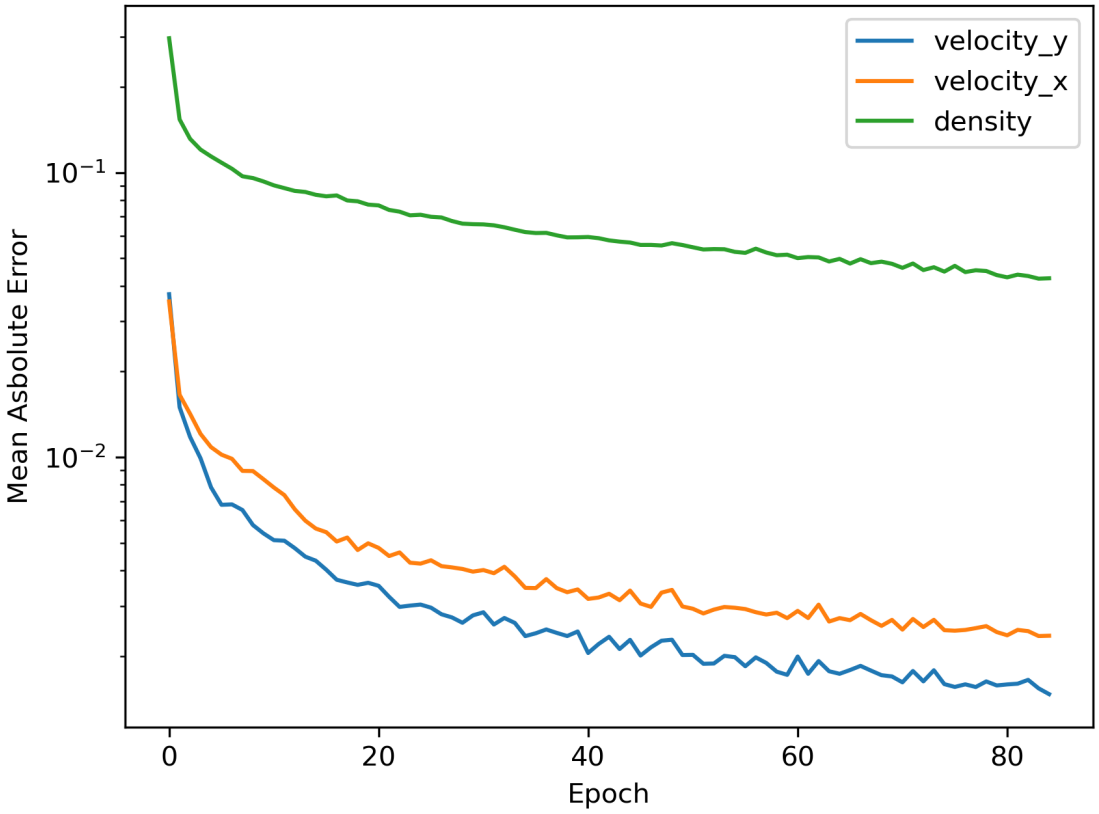

Figure 23: The epoch vs mean absolute error for each field while training the double sine wave model.

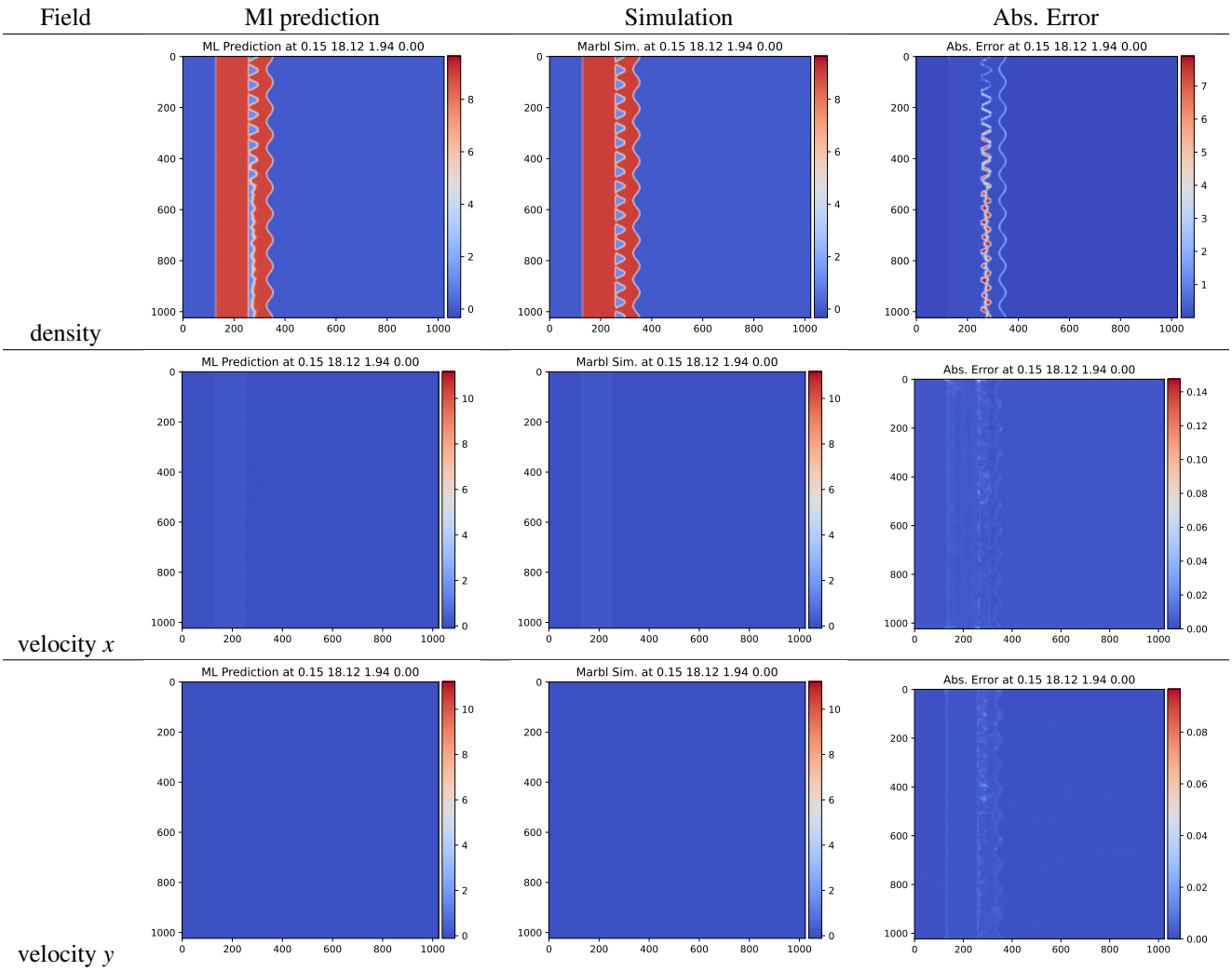

Figure 24: Figure of predictions, truth, and absolute error for double sine wave predictions at t=0.

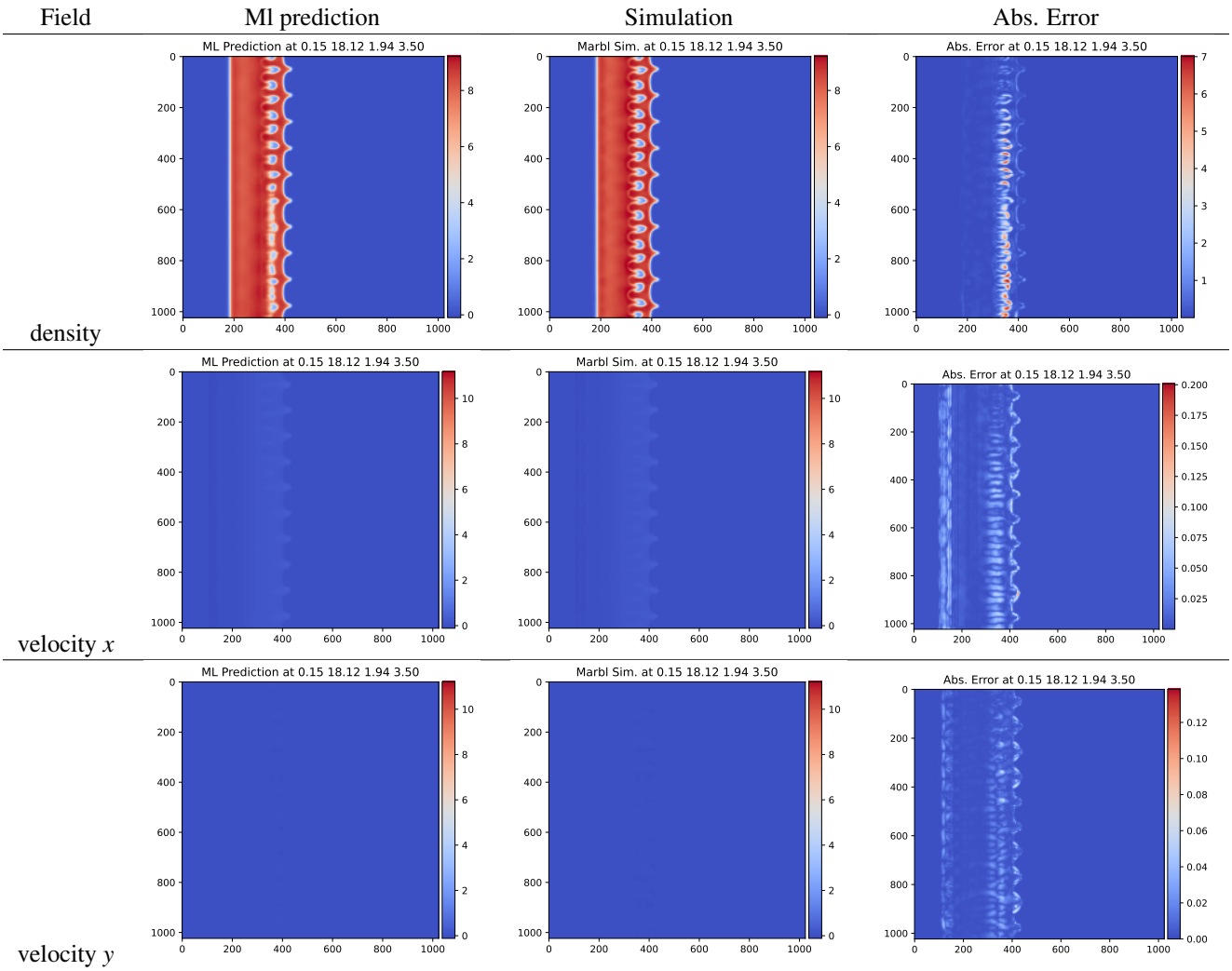

Figure 25: Figure of predictions, truth, and absolute error for double sine wave predictions at t=3.5.

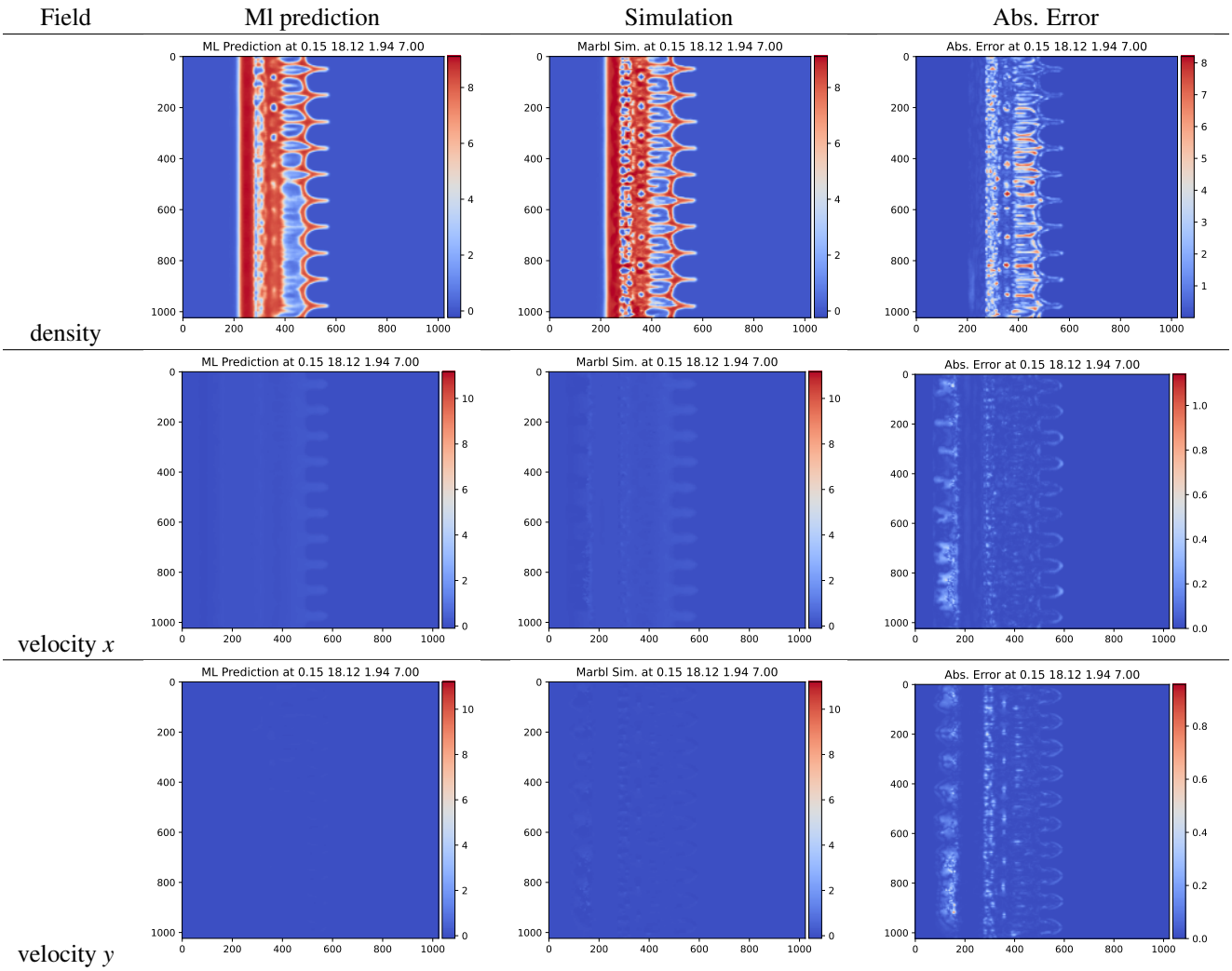

Figure 26: Figure of predictions, truth, and absolute error for double sine wave predictions at t=7.

B.3 Linear shaped charge results

The mean absolute error for each field separately in training of the linear shaped charge dataset is shown in figure 27. Samples of ML predictions for multiple fields are plotted in figures 28-30.

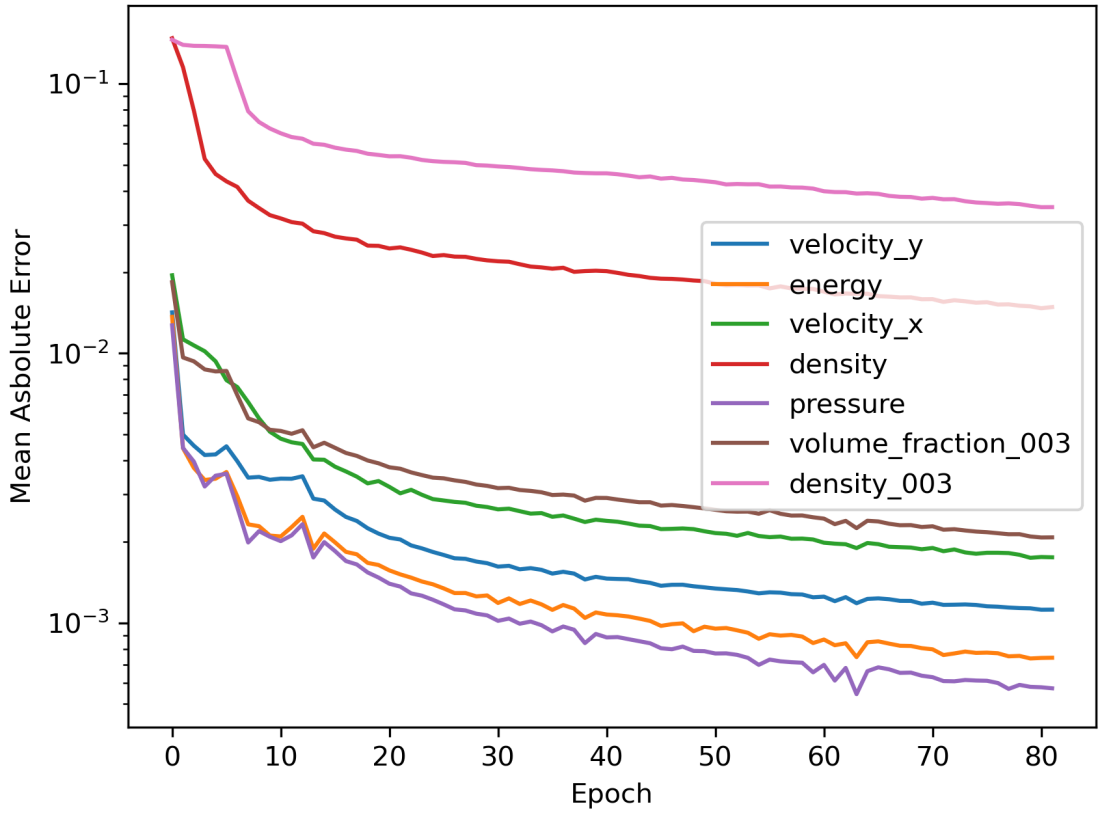

Figure 27: The epoch vs mean absolute error for each field while training the linear shaped charge model.

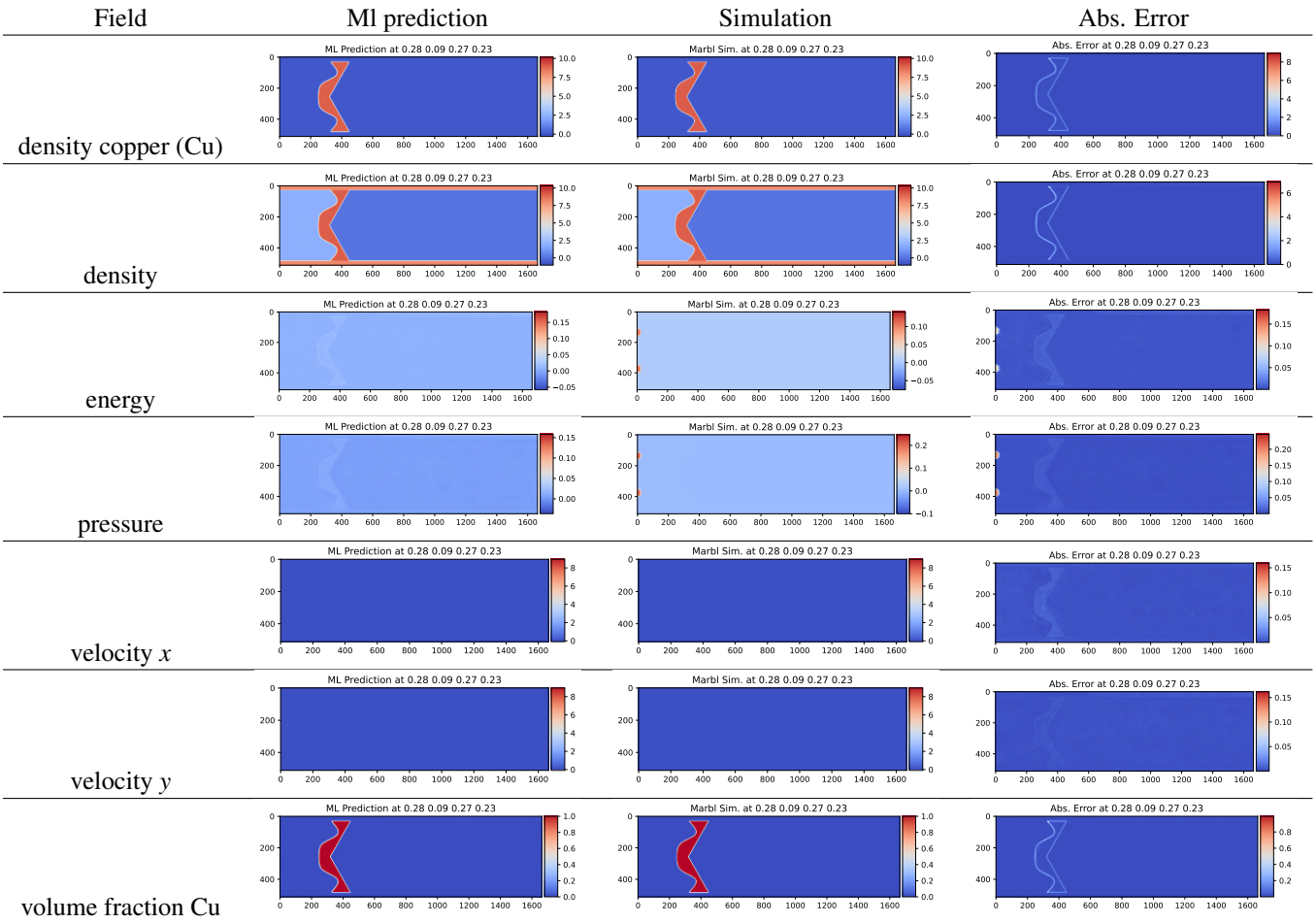

Figure 28: Figure of predictions, truth, and absolute error for linear shaped charge predictions at t=0.

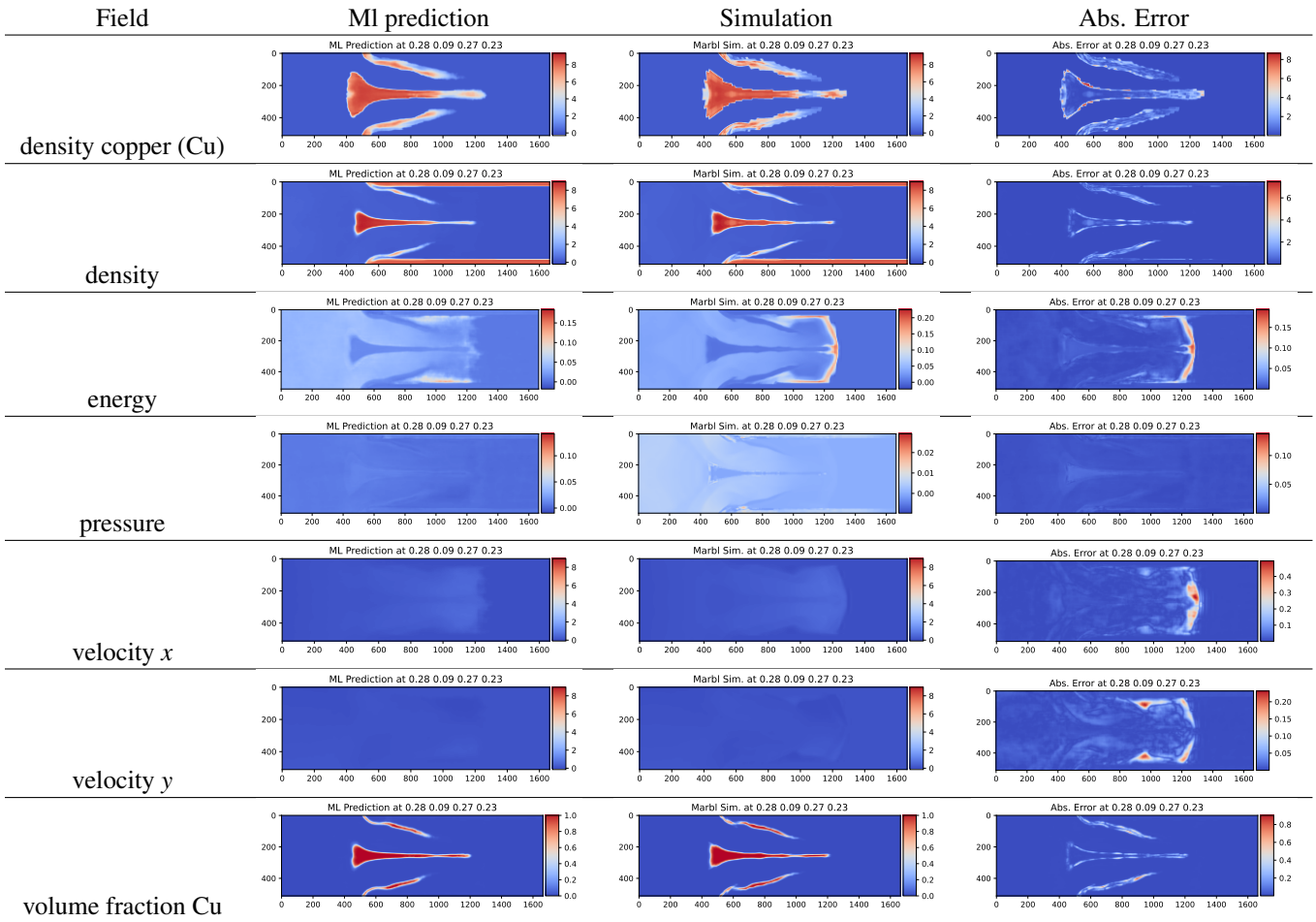

Figure 29: Figure of predictions, truth, and absolute error for linear shaped charge predictions at t=0.

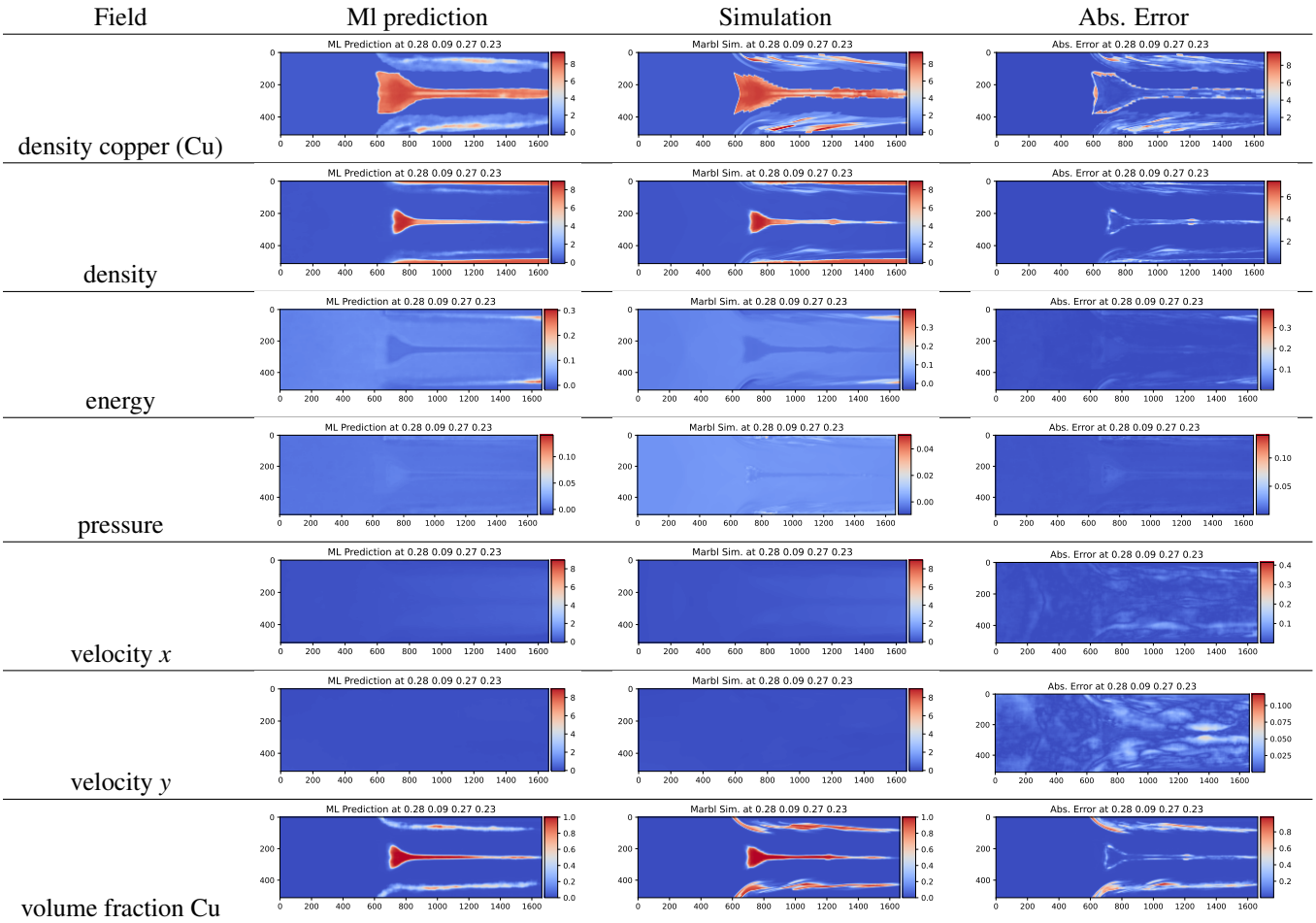

Figure 30: Figure of predictions, truth, and absolute error for linear shaped charge predictions at t=0.

B.4 Rayleigh-Taylor results

The mean absolute error for each field separately in training the Rayleigh Taylor simulation is shown in Figure 31. The final fields errors (from highest to lowest) are density, energy, materials, velocity y, velocity x, and pressure. The error in density was roughly an order of magnitude larger than pressure and velocity x.

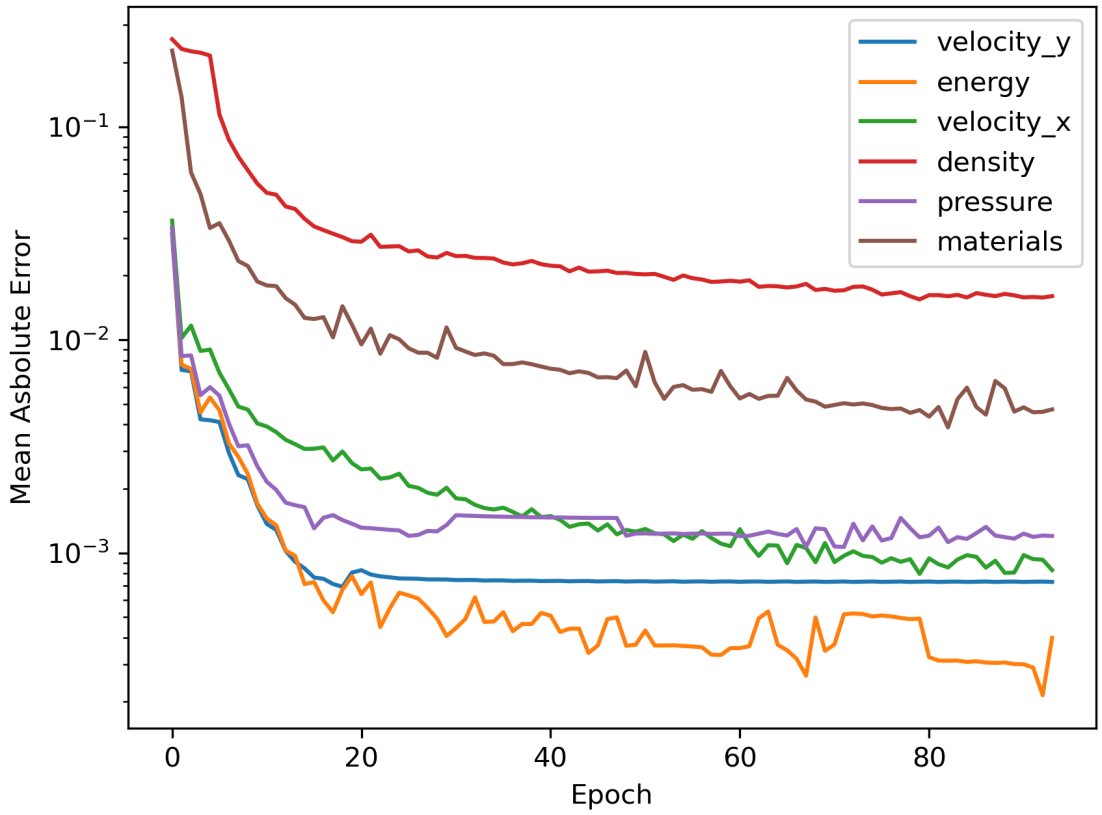

Figure 31: The epoch vs mean absolute error for each field while training the Rayleigh-Taylor model.

Samples of ML predictions for multiple fields are plotted in Figures 32-34 with the simulation results and the absolute error (between simulation and prediction). This was a simulation within the training set, that is representative of the general trends within the Rayleigh-Taylor results. Essentially at early to middle times (Figure 32 and Figure 33) we see the ML predictions are excellent at tracking the Rayleigh-Taylor instability. The errors are largest at the interface, which is generally tracked well. However, at late times as shown in Figure 34, a large number of eddies are formed along the interface of the two gases where the ML predictions are less accurate at maintaining a crisp interface. For these high detail and complex predictions, it appears as if the ML model is locally averaging or smoothing the complicated interface.

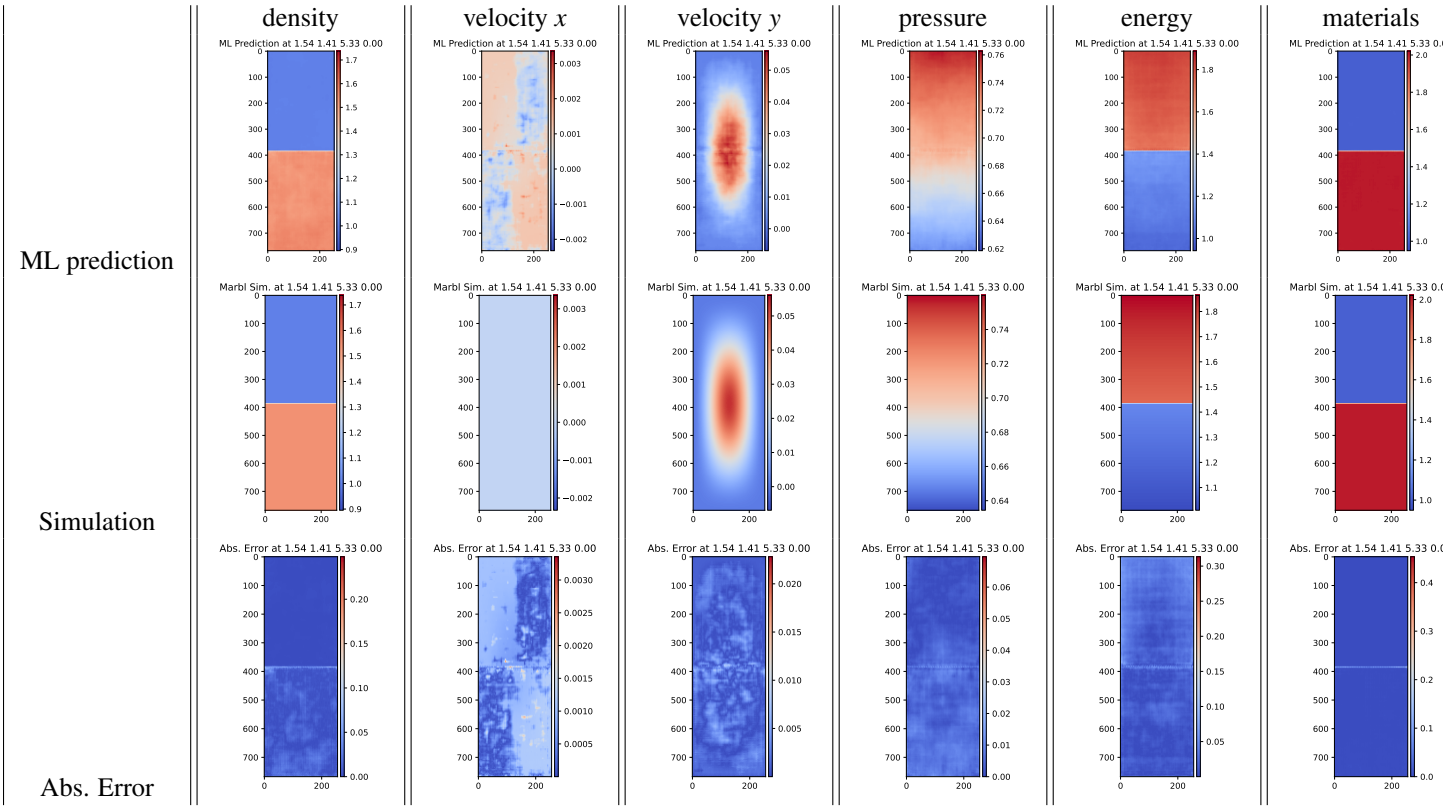

Figure 32: Figure of predictions, truth, and absolute error for Rayleigh-Taylor predictions at t=0.

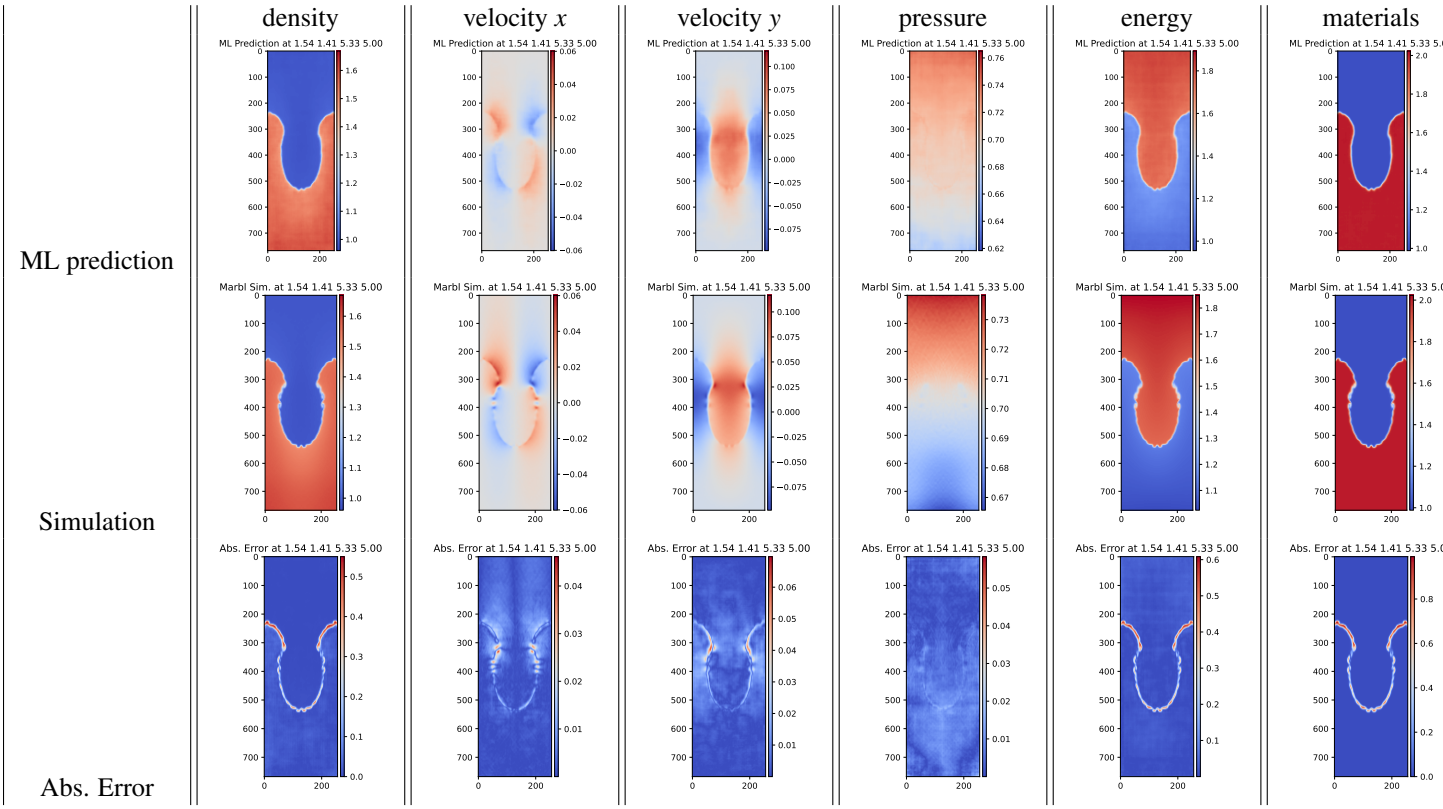

Figure 33: Figure of predictions, truth, and absolute error for Rayleigh-Taylor predictions at t=5.

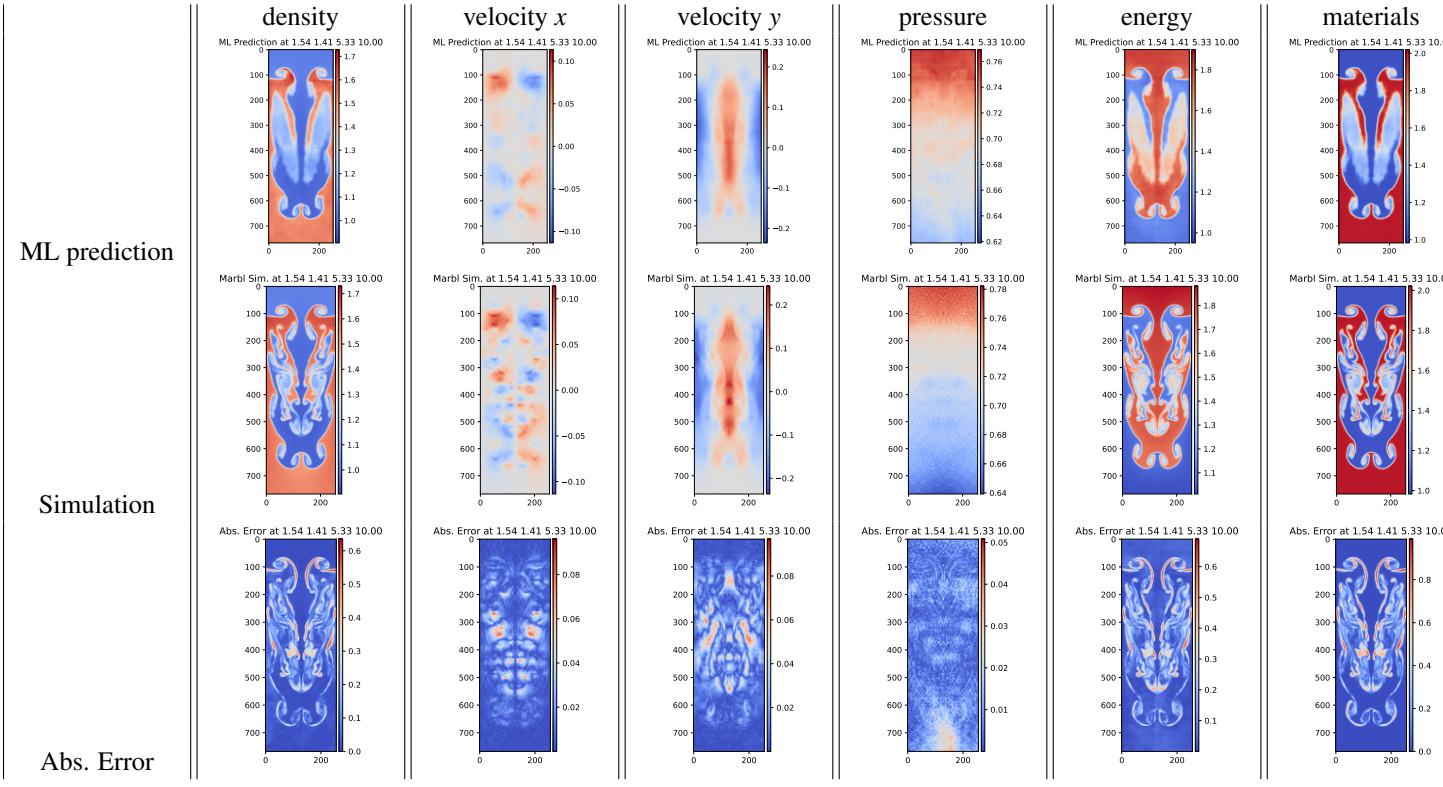

Figure 34: Figure of predictions, truth, and absolute error for Rayleigh-Taylor predictions at t=10.

References

[1] James Ahrens, Sebastien Jourdain, Patrick O'Leary, John Patchett, David H. Rogers, and Mark Petersen. An image-based approach to extreme scale in situ visualization and analysis. In *SC '14: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 424–434, 2014. doi:10.1109/SC.2014.40.

[2] Qian Chen, Li Li, Yousheng Zhang, and Baolin Tian. Effects of the atwood number on the richtmyer-meshkov instability in elastic-plastic media. *Phys. Rev. E*, 99:053102, May 2019.

[3] Hye-Sook Park, K. T. Lorenz, R. M. Cavallo, S. M. Pollaine, S. T. Prisbrey, R. E. Rudd, R. C. Becker, J. V. Bernier, and B. A. Remington. Viscous rayleigh-taylor instability experiments at high pressure and strain rate. *Phys. Rev. Lett.*, 104:135504, Apr 2010. doi:10.1103/PhysRevLett.104.135504. URL `https://link.aps.org/doi/10.1103/PhysRevLett.104.135504`.

[4] Z. Sternberger, B. R. Maddox, Y. P. Opachich, C. E. Wehrenberg, R. G. Kraus, B. A. Remington, G. C. Randall, M. Farrell, and G. Ravichandran. A comparative study of rayleigh-taylor and richtmyer-meshkov instabilities in 2d and 3d in tantalum. *AIP Conference Proceedings*, 1793(1):110006, 2017. doi:10.1063/1.4971669. URL `https://aip.scitation.org/doi/abs/10.1063/1.4971669`.

[5] W. T. Buttler, D. M. Oró, D. L. Preston, K. O. Mikaelian, F. J. Cherne, R. S. Hixson, F. G. Mariam, C. Morris, J. B. Stone, G. Terrones, and et al. Unstable richtmyer–meshkov growth of solid and liquid metals in vacuum. *Journal of Fluid Mechanics*, 703:60–84, 2012. doi:10.1017/jfm.2012.190.

[6] Guy Dimonte, G. Terrones, F. J. Cherne, T. C. Germann, V. Dupont, K. Kadau, W. T. Buttler, D. M. Oro, C. Morris, and D. L. Preston. Use of the richtmyer-meshkov instability to infer yield stress at high-energy densities. *Phys. Rev. Lett.*, 107:264502, Dec 2011. doi:10.1103/PhysRevLett.107.264502. URL `https://link.aps.org/doi/10.1103/PhysRevLett.107.264502`.

[7] Jonathan L Belof, R. M. Cavallo, Russel T. Olson, Robert S. King, George Thompson Gray III, David Holtkamp, Shuh-Rong Chen, R. E. Rudd, Nathan Barton, Athanasios Arsenlis, B. Remington, Hye-Sook Park, S. T. Prisbrey, Peter Vitello, Grant Bazan, Karnig O. Mikaelian, A. J. Comley, Brian Maddox, and M. J. May. Rayleigh-taylor strength experiments of the pressure-induced phase transition in iron. *AIP Conference Proceedings*, 1426(1):1521–1524, 2012. doi:10.1063/1.3686572. URL `https://aip.scitation.org/doi/abs/10.1063/1.3686572`.

[8] A. B. Zylstra, O. A. Hurricane, D. A. Callahan, A. L. Kritcher, J. E. Ralph, H. F. Robey, J. S. Ross, C. V. Young, K. L. Baker, D. T. Casey, T. Döppner, L. Divol, M. Hohenberger, S. Le Pape, A. Pak, P. K. Patel, R. Tommasini, S. J. Ali, P. A. Amendt, L. J. Atherton, B. Bachmann, D. Bailey, L. R. Benedetti, L. Berzak Hopkins, R. Betti, S. D. Bhandarkar, J. Biener, R. M. Bionta, N. W. Birge, E. J. Bond, D. K. Bradley, T. Braun, T. M. Briggs, M. W. Bruhn, P. M. Celliers, B. Chang, T. Chapman, H. Chen, C. Choate, A. R. Christopherson, D. S. Clark, J. W. Crippen, E. L. Dewald, T. R. Dittrich, M. J. Edwards, W. A. Farmer, J. E. Field, D. Fittinghoff, J. Frenje, J. Gaffney, M. Gatu Johnson, S. H. Glenzer, G. P. Grim, S. Haan, K. D. Hahn, G. N. Hall, B. A. Hammel, J. Harte, E. Hartouni, J. E. Heebner, V. J. Hernandez, H. Herrmann, M. C. Herrmann, D. E. Hinkel, D. D. Ho, J. P. Holder, W. W. Hsing, H. Huang, K. D. Humbird, N. Izumi, L. C. Jarrott, J. Jeet, O. Jones, G. D. Kerbel, S. M. Kerr, S. F. Khan, J. Kilkenny, Y. Kim, H. Geppert Kleinrath, V. Geppert Kleinrath, C. Kong, J. M. Koning, J. J. Kroll, M. K. G. Kruse, B. Kustowski, O. L. Landen, S. Langer, D. Larson, N. C. Lemos, J. D. Lindl, T. Ma, M. J. MacDonald, B. J. MacGowan, A. J. Mackinnon, S. A. MacLaren, A. G. MacPhee, M. M. Marinak, D. A. Mariscal, E. V. Marley, L. Masse, K. Meaney, N. B. Meezan, P. A. Michel, M. Millot, J. L. Milovich, J. D. Moody, A. S. Moore, J. W. Morton, T. Murphy, K. Newman, J. M. G. Di Nicola, A. Nikroo, R. Nora, M. V. Patel, L. J. Pelz, J. L. Peterson, Y. Ping, B. B. Pollock, M. Ratledge, N. G. Rice, H. Rinderknecht, M. Rosen, M. S. Rubery, J. D. Salmonson, J. Sater, S. Schiaffino, D. J. Schlossberg, M. B. Schneider, C. R. Schroeder, H. A. Scott, S. M. Sepke, K. Sequoia, M. W. Sherlock, S. Shin, V. A. Smalyuk, B. K. Spears, P. T. Springer, M. Stadermann, S. Stoupin, D. J. Strozzi, L. J. Suter, C. A. Thomas, R. P. J. Town, E. R. Tubman, C. Trosseille, P. L. Volegov, C. R. Weber, K. Widmann, C. Wild, C. H. Wilde, B. M. Van Wonterghem, D. T. Woods, B. N. Woodworth, M. Yamaguchi, S. T. Yang, and G. B. Zimmerman. Burning plasma achieved in inertial fusion. *Nature*, 601(7894):542–548, 2022. doi:10.1038/s41586-021-04281-w. URL `https://doi.org/10.1038/s41586-021-04281-w`.

[9] T.R. Desjardins, C.A. Di Stefano, T. Day, D. Schmidt, E.C. Merritt, F.W. Doss, K.A. Flippo, T. Cardenas, B. DeVolder, P. Donovan, S. Edwards, F. Fierro, R. Gonzales, L. Goodwin, C. Hamilton, T. Quintana, R. Randolph, A.M. Rasmus, T. Sedillo, C. Wilson, and L. Welser-Sherrill. A platform for thin-layer richtmyer-meshkov at omega and the nif. *High Energy Density Physics*, 33:100705, 2019. ISSN 1574-1818. doi:https://doi.org/10.1016/j.hedp.2019.100705. URL `https://www.sciencedirect.com/science/article/pii/S1574181819300461`.

[10] Reuben D. Budiardja, Mark Berrill, Markus Eisenbach, Gustav R. Jansen, Wayne Joubert, Stephen Nichols, David M. Rogers, Arnold Tharrington, and O. E. Bronson Messer. Ready for the frontier: Preparing applications

for the world's first exascale system. In Abhinav Bhatele, Jeff Hammond, Marc Baboulin, and Carola Kruse, editors, *High Performance Computing*, pages 182–201, Cham, 2023. Springer Nature Switzerland. ISBN 978-3-031-32041-5.

[11] J Luc Peterson, Ben Bay, Joe Koning, Peter Robinson, Jessica Semler, Jeremy White, Rushil Anirudh, Kevin Athey, Peer-Timo Bremer, Francesco Di Natale, et al. Enabling machine learning-ready hpc ensembles with merlin. *Future Generation Computer Systems*, 131:255–268, 2022.

[12] Haizhou Wu, Xuejun Liu, Wei An, Songcan Chen, and Hongqiang Lyu. A deep learning approach for efficiently and accurately evaluating the flow field of supercritical airfoils. *Computers & Fluids*, 198:104393, 2020. ISSN 0045-7930. doi:https://doi.org/10.1016/j.compfluid.2019.104393. URL https://www.sciencedirect.com/science/article/pii/S0045793019303512.

[13] Haizhou Wu, Xuejun Liu, Wei An, and Hongqiang Lyu. A generative deep learning framework for airfoil flow field prediction with sparse data. *Chinese Journal of Aeronautics*, 35(1):470–484, 2022. ISSN 1000-9361. doi:https://doi.org/10.1016/j.cja.2021.02.012. URL https://www.sciencedirect.com/science/article/pii/S1000936121000728.

[14] Tanujit Chakraborty, Ujjwal Reddy K S, Shraddha M Naik, Madhurima Panja, and Bayapureddy Manvitha. Ten years of generative adversarial nets (gans): a survey of the state-of-the-art. *Machine Learning: Science and Technology*, 5(1):011001, jan 2024. doi:10.1088/2632-2153/ad1f77. URL https://dx.doi.org/10.1088/2632-2153/ad1f77.

[15] Ni Li, Yunqin Liu, Guanghong Gong, Luming Zhao, and Haitao Yuan. A generative deep learning approach for real-time prediction of hypersonic vehicles in fluid-thermo-structural coupling fields. *Aerospace Science and Technology*, 139:108398, 2023. ISSN 1270-9638. doi:https://doi.org/10.1016/j.ast.2023.108398. URL https://www.sciencedirect.com/science/article/pii/S127096382300295X.

[16] Zhipeng Wang, Xuejun Liu, Jian Yu, Haizhou Wu, and Hongqiang Lyu. A general deep transfer learning framework for predicting the flow field of airfoils with small data. *Computers & Fluids*, 251:105738, 2023. ISSN 0045-7930. doi:https://doi.org/10.1016/j.compfluid.2022.105738. URL https://www.sciencedirect.com/science/article/pii/S0045793022003309.

[17] Muhammad Alfiyandy Hariansyah and Koji Shimoyama. Deep learning techniques for high-dimensional surrogate-based aerodynamic design. In *33rd Congress of the International Council of the Aeronautical Sciences, ICAS 2022*, pages 164–182. International Council of the Aeronautical Sciences, 2022.

[18] Tanishk Nandal, Vaibhav Fulara, and Raj Kumar Singh. A synergistic framework leveraging autoencoders and generative adversarial networks for the synthesis of computational fluid dynamics results in aerofoil aerodynamics. *arXiv preprint arXiv:2305.18386*, 2023.

[19] Yuqing Hou, Hui Li, Hong Chen, Wei Wei, Jiayue Wang, and Yicang Huang. A novel deep u-net-lstm framework for time-sequenced hydrodynamics prediction of the suboff aff-8. *Engineering Applications of Computational Fluid Mechanics*, 16(1):630–645, 2022. doi:10.1080/19942060.2022.2030802. URL https://doi.org/10.1080/19942060.2022.2030802.

[20] Ali Kashefi and Tapan Mukerji. Point-cloud deep learning of porous media for permeability prediction. *Physics of Fluids*, 33(9):097109, 09 2021. ISSN 1070-6631. doi:10.1063/5.0063904. URL https://doi.org/10.1063/5.0063904.

[21] Erda Chen, Mingming Guo, Ye Tian, Yi Zhang, Hao Chen, Jialing Le, Fuyu Zhong, and Hua Zhang. Flame development prediction of supersonic combustion flow based on lightweight cascaded convolutional neural network. *Physics of Fluids*, 35(2):025134, 02 2023. ISSN 1070-6631. doi:10.1063/5.0140624. URL https://doi.org/10.1063/5.0140624.

[22] Yunfei Li, Juntao Chang, Ziao Wang, and Cheng Kong. Inversion and reconstruction of supersonic cascade passage flow field based on a model comprising transposed network and residual network. *Physics of Fluids*, 31(12):126102, 12 2019. ISSN 1070-6631. doi:10.1063/1.5124133. URL https://doi.org/10.1063/1.5124133.

[23] Efren Fernandez-Grande, Xenofon Karakonstantis, Diego Caviedes-Nozal, and Peter Gerstoft. Generative models for sound field reconstruction. *The Journal of the Acoustical Society of America*, 153(2):1179–1190, 02 2023. ISSN 0001-4966. doi:10.1121/10.0016896. URL https://doi.org/10.1121/10.0016896.

[24] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. In Yoshua Bengio and Yann LeCun, editors, *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, 2016. URL http://arxiv.org/abs/1511.06434.

[25] Nissrine Akkari, Fabien Casenave, Marc-Eric Perrin, and David Ryckelynck. Deep convolutional generative adversarial networks applied to 2d incompressible and unsteady fluid flows. In *Science and Information Conference*, pages 264–276. Springer, 2020.

[26] M. Cheng, F. Fang, C.C. Pain, and I.M. Navon. Data-driven modelling of nonlinear spatio-temporal fluid flows using a deep convolutional generative adversarial network. *Computer Methods in Applied Mechanics and Engineering*, 365:113000, 2020. ISSN 0045-7825. doi:https://doi.org/10.1016/j.cma.2020.113000. URL https://www.sciencedirect.com/science/article/pii/S0045782520301845.

[27] C. Drygala, B. Winhart, F. di Mare, and H. Gottschalk. Generative modeling of turbulence. *Physics of Fluids*, 34(3): 035114, 03 2022. ISSN 1070-6631. doi:10.1063/5.0082562. URL https://doi.org/10.1063/5.0082562.

[28] Carlos Granero Belinchon and Manuel Cabeza Gallucci. A multiscale and multicriteria generative adversarial network to synthesize 1-dimensional turbulent fields. *Machine Learning: Science and Technology*, 5(2):025032, may 2024. doi:10.1088/2632-2153/ad43b3. URL https://dx.doi.org/10.1088/2632-2153/ad43b3.

[29] S. Wiewel, M. Becher, and N. Thuerey. Latent space physics: Towards learning the temporal evolution of fluid flow. *Computer Graphics Forum*, 38(2):71–82, 2019. doi:https://doi.org/10.1111/cgf.13620. URL https://onlinelibrary.wiley.com/doi/abs/10.1111/cgf.13620.

[30] Dane M. Sterbentz, Dylan J. Kline, Daniel A. White, Charles F. Jekel, Michael P. Hennessey, David K. Amondson, Abigail J. Wilson, Max J. Sevcik, Matthew F. L. Villena, Steve S. Lin, Michael D. Grapes, Kyle T. Sullivan, and Jonathan L. Belof. Explosively driven Richtmyer–Meshkov instability jet suppression and enhancement via coupling machine learning and additive manufacturing. *Journal of Applied Physics*, 136(3):035102, 07 2024. ISSN 0021-8979. doi:10.1063/5.0213123. URL https://doi.org/10.1063/5.0213123.

[31] Dylan J. Kline, Michael P. Hennessey, David K. Amondson, Steve Lin, Michael D. Grapes, Massimiliano Ferrucci, Peggy Li, H. Keo Springer, Robert V. Reeves, Kyle T. Sullivan, and Jonathan L. Belof. Reducing Richtmyer–Meshkov instability jet velocity via inverse design. *Journal of Applied Physics*, 135(7):074902, 02 2024. ISSN 0021-8979. doi:10.1063/5.0180712. URL https://doi.org/10.1063/5.0180712.

[32] Michael P Hennessey, Finnegan Wilson, Grace I Rabinowitz, Max J Sevcik, Kadyn J Tucker, Dylan J Kline, David K Amondson, H Keo Springer, Kyle T Sullivan, Veronica Eliasson, et al. Modification of jet velocities in an explosively loaded copper target with a conical defect. *arXiv preprint arXiv:2404.06640*, 2024.

[33] M. Raissi, P. Perdikaris, and G.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*, 378:686–707, 2019. ISSN 0021-9991. doi:https://doi.org/10.1016/j.jcp.2018.10.045. URL https://www.sciencedirect.com/science/article/pii/S0021999118307125.

[34] JIANXIANG PENG, CHANGMING HU, YINGLEI LI, LIN ZHANG, and FUQIAN JING. Determination of parameters of steinberg-guinan constitutive model with shock wave experiments. *International Journal of Modern Physics B*, 22(09n11):1111–1116, 2008. doi:10.1142/S0217979208046396. URL https://doi.org/10.1142/S0217979208046396.

[35] Veselin A. Dobrev, T. E. Ellis, and Robert N. Rieben. Curvilinear finite elements for lagrangian hydrodynamics. *Int. J. Numer. Meth. Fluid*, 65:1295–1310, 2011.

[36] Veselin A. Dobrev, Tzanio V. Kolev, and Robert N. Rieben. High order curvilinear finite element methods for lagrangian hydrodynamics. *SIAM J. Sci. Comp*, 34(5):606–641, 2012.

[37] Robert W. Anderson, Veselin A. Dobrev, Tzanio V. Kolev, Robert N. Rieben, and Vladimir Z. Tomov. High-order multi-material ale hydrodynamics. *SIAM J. Sci. Comp.*, 40(1):B32–B58, 2018.

[38] Thomas Stitt, Kristi Belcher, Alejandro Campos, Tzanio Kolev, Philip Mocz, Robert N. Rieben, Aaron Skinner, Vladimir Tomov, Arturo Vargas, and Kenneth Weiss. Performance Portable Graphics Processing Unit Acceleration of a High-Order Finite Element Multiphysics Application . *Journal of Fluids Engineering*, 146(4):041102, 02 2024. ISSN 0098-2202. doi:10.1115/1.4064493. URL https://doi.org/10.1115/1.4064493.

[39] Tzanio V. Kolev and Robert N. Rieben. A tensor artificial viscosity using a finite element approach. *J. Comp. Phys.*, 228:8336–8366, 2009.

[40] R. W. Anderson, V. A. Dobrev, Tzanio V. Kolev, and Robert N. Rieben. Monotonicity in high-order curvilinear finite element arbitrary lagrangian–eulerian remap. *Int. J. Numer. Meth. Fluids*, 77:249–273, 2015.

[41] F N. Fritsch. The leos interpolation package. *LLNL Techical Report*, 2003. doi:doi:10.2172/15005830. URL https://www.osti.gov/biblio/15005830.

[42] D. J. Steinberg, S. G. Cochran, and M. W. Guinan. A constitutive model for metals applicable at high-strain rate. *J. Appl. Phys*, 41(3):1498–1504, 1980. doi:https://doi.org/10.1063/1.327799.

[43] URL https://github.com/CEED/Laghos.

[44] F. A. C. Viana. A tutorial on latin hypercube design of experiments. *Qual. Reliab. Engng.*, 32:1975–1985, 2016.

[45] Mike Folk, Gerd Heber, Quincey Koziol, Elena Pourmal, and Dana Robinson. An overview of the hdf5 technology suite and its applications. In *Proceedings of the EDBT/ICDT 2011 workshop on array databases*, pages 36–47, 2011.

[46] Michael Armstrong, Jeffrey Nguyen, Sylvie Aubry, William Schill, Jonathan Belof, and Hector Lorenzana. Use of shock wavefront curvature to modulate rmi jet growth. *Bulletin of the American Physical Society*, 67, 2022.

[47] Jeffrey Nguyen, Sylvie Aubry, Michael Armstrong, Andrew Hoff, Jonathan Belof, Hector Lorenzana, Matthew Staska, and Brandon LaLone. Modulation of richtmyer-meshkov instability in gas gun experiments. *Bulletin of the American Physical Society*, 67, 2022.

[48] Dane M. Sterbentz, Charles F. Jekel, Daniel A. White, Sylvie Aubry, Hector E. Lorenzana, and Jonathan L. Belof. Design optimization for Richtmyer–Meshkov instability suppression at shock-compressed material interfaces. *Physics of Fluids*, 34(8):082109, 08 2022. ISSN 1070-6631. doi:10.1063/5.0100100. URL https://doi.org/10.1063/5.0100100.

[49] B. Duan, Y. Zhou, S. Zheng, M. Bao, and J. Dong. Blasting demolition of steel structure using linear cumulative cutting technology. *Adv. Mech. Engng.*, 9(11):1–11, 2017.

[50] E. N. Sher. Development of a system of plane radial cracks during explosion of linear blasthole and borehole charges. *J. Appl. Mech. Tech. Phys.*, 58(5):933–938, 2017.

[51] B. Wu, G. Xu, and et. al. Meng. Study on dynamic evolution law of blasting cracks in elliptical bipolar linear shaped charge blasting. *Shock and Vibration*, pages 1–10, 2021. doi:https://doi.org/10.1155/2021/5272878.

[52] D. Novotny and M. Mallery. Historical development of linear shaped charge. In *Joint Propioulsion Conference*. AIAA/ASME, 2007. URL https://arc.aiaa.org/doi/pdf/10.2514/6.2007-5141.

[53] Dane M. Sterbentz, Charles F. Jekel, Daniel A. White, Robert N. Rieben, and Jonathan L. Belof. Linear shaped-charge jet optimization using machine learning methods. *Journal of Applied Physics*, 134(4):045102, 07 2023. ISSN 0021-8979. doi:10.1063/5.0156373. URL https://doi.org/10.1063/5.0156373.

[54] James M. Stone, Kengo Tomida, Christopher J. White, and Kyle G. Felker. The athena++ adaptive mesh refinement framework: Design and magnetohydrodynamic solvers. *The Astrophysical Journal Supplement Series*, 249(1):4, June 2020. doi:10.3847/1538-4365/ab929b. URL https://doi.org/10.3847%2F1538-4365%2Fab929b.

[55] Richard Liska and Burton Wendroff. Comparison of several difference schemes on 1d and 2d test problems for the euler equations. *SIAM Journal on Scientific Computing*, 25(3):995–1017, 2003. doi:10.1137/S1064827502402120. URL https://doi.org/10.1137/S1064827502402120.

[56] Augustus Odena, Vincent Dumoulin, and Chris Olah. Deconvolution and checkerboard artifacts. *Distill*, 1(10):e3, 2016.

[57] Charles F Jekel, Dane M Sterbentz, Sylvie Aubry, Youngsoo Choi, Daniel A White, and Jonathan L Belof. Using conservation laws to infer deep learning model accuracy of richtmyer-meshkov instabilities. *arXiv preprint arXiv:2208.11477*, 2022.

[58] Derek Hansen, Danielle C. Maddix, Shima Alizadeh, Gaurav Gupta, and Michael W. Mahoney. Learning physical models that can respect conservation laws. *Physica D: Nonlinear Phenomena*, 457:133952, 2024. ISSN 0167-2789. doi:https://doi.org/10.1016/j.physd.2023.133952. URL https://www.sciencedirect.com/science/article/pii/S0167278923003068.

[59] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. PMLR, 2015.

[60] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Icml*, 2010.

[61] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

[62] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[63] Samyam Rajbhandari, Jeff Rasley, Olatunji Ruwase, and Yuxiong He. Zero: Memory optimizations toward training trillion parameter models. In *SC20: International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 1–16. IEEE, 2020.

[64] Priya Goyal, Piotr Dollár, Ross Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, and Kaiming He. Accurate, large minibatch sgd: Training imagenet in 1 hour. *arXiv preprint arXiv:1706.02677*, 2017.

[65] Brandon E. Morgan. Large-eddy simulation and reynolds-averaged navier-stokes modeling of three rayleigh-taylor mixing configurations with gravity reversal. *Phys. Rev. E*, 106:025101, Aug 2022. doi:10.1103/PhysRevE.106.025101. URL https://link.aps.org/doi/10.1103/PhysRevE.106.025101.

[66] Napari Contributors. napari: a multi-dimensional image viewer for python. *Zenodo https://doi. org/10.5281/zenodo*, 3555620, 2019.

[67] Ciyou Zhu, Richard H. Byrd, Peihuang Lu, and Jorge Nocedal. Algorithm 778: L-bfgs-b: Fortran subroutines for large-scale bound-constrained optimization. *ACM Trans. Math. Softw.*, 23(4):550–560, December 1997. ISSN 0098-3500. doi:10.1145/279232.279236. URL https://doi.org/10.1145/279232.279236.

[68] Anders Krogh and Jesper Vedelsby. Neural network ensembles, cross validation, and active learning. *Advances in neural information processing systems*, 7, 1994.