The Astrophysical Journal, 940:26 (13pp), 2022 November 20 https://doi.org/10.3847/1538-4357/ac88cc

Physics-informed Machine Learning for Modeling Turbulence in Supernovae

Platon I. Karpov[1,2], Chengkun Huang[2], Iskandar Sitdikov[3], Chris L. Fryer[2], Stan Woosley[1], and Ghanshyam Pilania[2]
[1] Department of Astronomy & Astrophysics, University of California, Santa Cruz, CA 95064, USA
[2] Los Alamos National Laboratory, Los Alamos, NM 87545, USA
[3] Provectus IT Inc., Palo Alto, CA 94301, USA
Received 2022 June 21; revised 2022 August 8; accepted 2022 August 9; published 2022 November 16

Abstract

Turbulence plays an important role in astrophysical phenomena, including core-collapse supernovae (CCSNe), but current simulations must rely on subgrid models, since direct numerical simulation is too expensive. Unfortunately, existing subgrid models are not sufficiently accurate. Recently, machine learning (ML) has shown an impressive predictive capability for calculating turbulence closure. We have developed a physics-informed convolutional neural network to preserve the realizability condition of the Reynolds stress that is necessary for accurate turbulent pressure prediction. The applicability of the ML subgrid model is tested here for magnetohydrodynamic turbulence in both the stationary and dynamic regimes. Our future goal is to utilize this ML methodology (available on GitHub) in the CCSN framework to investigate the effects of accurately modeled turbulence on the explosion of these stars.

Unified Astronomy Thesaurus concepts: Core-collapse supernovae (304); Computational methods (1965); Computational astronomy (293); Supernovae (1668)

1. Introduction

Turbulence plays a key role in many astrophysical phenomena (Schekochihin & Cowley 2007; Brandenburg & Lazarian 2013; Beresnyak 2019), a prominent example being a core-collapse supernova (CCSN): the bright, energetic, dynamic explosion of a highly evolved massive star of at least eight times the mass of the Sun. At the end of its life, the iron core of such a massive star can no longer generate energy by fusion reactions, and yet it is subject to enormous energy losses in the form of neutrinos. As the core of about 1.5 solar masses contracts and heats up, looking for a new source of energy generation, additional instabilities instead accelerate the collapse until it is in almost freefall. These instabilities include electron capture and photodisintegration—heavy nuclei splitting into lighter elements, due to high-energy photon absorption. The collapse continues until the density of the inner core exceeds that of the atomic nucleus ($\sim 2 \times 10^{14}$ g cm^{-3}), and then it abruptly halts, due to the repulsive component of the nuclear force. The outer part of the core rains down on the nearly stationary inner core and bounces, creating a powerful outward-bound shock wave. It was once thought that this "prompt shock" might propagate through the entire star, exploding it as a supernova (Baron et al. 1987). Now we know that this does not happen. The shock stalls in the face of prodigious losses to neutrinos and photodisintegration, and becomes an accretion shock outside the edge of the original iron core. All positive radial velocity is gone from the star. The evolution slows, taking hundreds of milliseconds instead of milliseconds. At this stage, the core is a hot protoneutron star, radiating a prodigious flux of neutrinos, surrounded by an accretion shock through which the rest of the star is falling. The success or failure of the explosion then depends on the efficiency of the neutrinos in depositing some fraction of their energy outside the protoneutron star (outside the neutrinosphere and inside the accretion shock), and the distribution of the pressure that energy deposition creates. A failed explosion will lead to a black hole and no supernova (Woosley & Janka 2005; Burrows & Vartanyan 2021).

Over the past three decades, the community has focused on the importance of turbulence and convection in improving the efficiency with which the energy released in the collapse of the core is converted into explosion energy (Herant et al. 1994; Blondin et al. 2003; Fryer & Young 2007; Melson et al. 2015; Burrows et al. 2018). Most of these studies have focused on the large-scale convective motions that transport both matter and energy. If the pressure in this convective region, including turbulence, becomes large, the accretion shock will be pushed outward, ultimately achieving positive velocity and exploding the star. A recent study attributes up to ∼30% of the gas pressure to turbulence aiding the CCSN explosion (Nagakura et al. 2019). Turbulence in this region has three origins: the primordial turbulence that is present because the star was convective in these zones before it collapsed; the turbulence that is generated by the convective overturn, driven by neutrino energy deposition beneath the shock; and, if the star is rotating, by magnetic instabilities in the differentially rotating layers (especially the "magnetorotational instability," or MRI). Multidimensional solutions exist to the CCSN problem, both with and without rotation and magnetic fields. Some explode, some do not, and this has been a problem for at least the past 60 years (Colgate & White 1966). A major difficulty is a physically correct description of the turbulence and its effective pressure in a multidimensional code that is unable to resolve the relevant length scales.

Here, we focus on magnetically generated turbulence. This introduces additional variables and uncertainties that are not contained in the non-magnetohydrodynamic (MHD) case, but has the merit of using conditions that are locally generated and the existence of a high-resolution training set (Mösta et al. 2015; detailed in Section 3.4). The framework that we derive can be used for both MHD and field-free turbulence, and it is

our intention to return to the non-MHD case in subsequent work. In this case, the MRI occurs in a setting of magnetized, differentially rotating fluid layers, i.e., stellar shells. The instability exponentially amplifies primordial perturbations, developing turbulence (Obergaulinger et al. 2009).

A flow can be considered turbulent if the Reynolds number (Re) is of order $\sim 10^3$, which corresponds to what we expect to see in CCSNe (Fryer & Young 2007). For direct numerical simulation (DNS), the per-axis 3D resolution scales as $N \sim 2\,\mathrm{Re}^{3/2}$ (Jiménez 2003), leading to 3D DNS of CCSNe requiring a grid size of 10^5 in each direction, which is extremely expensive (if possible) to achieve with today's high-performance computing (HPC) resources. Together with the vast spatial scale range needing to be resolved in CCSNe (a 200 km inner convective region, and out to a 10^9 km outer shell), and the complexity of the physical processes ongoing in CCSNe, DNS calculations are out of reach. Given the computational challenges, subgrid turbulence is often modeled using the following techniques (in astrophysics, these schemes are primarily used in 1D simulations):

1. Mixing Length Theory (MLT)—modeling turbulent eddies that transfer their momentum over some *mixing length* via eddy viscosity (Spiegel 1963); akin to molecular motion. MLT is used to study turbulence driven by convection, thus it is applicable to stellar convection (including SNe simulations; Couch et al. 2020) in 1D simulations. MLT performs well for small mixing length scales, while turbulence in CCSNe evolves over a wide range of scales, which is deemed problematic for MLT's accuracy (Joshi et al. 2019).
2. Reynolds-averaged Navier–Stokes (RANS)—time- (and ensemble-) averaged treatment of turbulence equations of motion. RANS is typically used for relatively low-Re, e.g., stellar evolution and consequently SNe, progenitors (Arnett et al. 2015), and requires a turbulent closure model.
3. Large Eddy Simulation (LES)—space-averaged turbulence treatment. Similar to RANS, a subgrid model substitutes the turbulence effects that are absent from small spatial scales. For *closure*, it is common to use Dynamic Smagorinsky (Lilly 1966) or gradient-type subgrid models (Miesch et al. 2015; Schmidt 2015) in LES simulations. Implicit LES (ILES)—similar to LES, but the small scales are assumed to be approximated by numerical artifacts (e.g., numerical diffusion and viscosity). ILES is typically employed by large global 3D simulations of CCSNe and other astrophysical events (Radice et al. 2015, 2018).

Even though these techniques have achieved some level of success when predicting HD turbulence, MHD poses a new set of challenges. With magnetic fields present, the magnetic/kinetic energy is primarily transferred at small scales through the dynamo process (Beresnyak & Lazarian 2014). The nonlinear behavior of MHD further exacerbates simulation challenges, leaving many open questions as to the nature of the turbulence, despite decades of focused studies (Beresnyak 2019). As an example of the changes due to the introduction of a magnetic field within a simulation, we present a comparison of a normalized Re stress tensor component distribution between HD and MHD simulations, given comparable initial conditions from the Johns Hopkins Turbulence Database (JHTDB), in Figure 1.

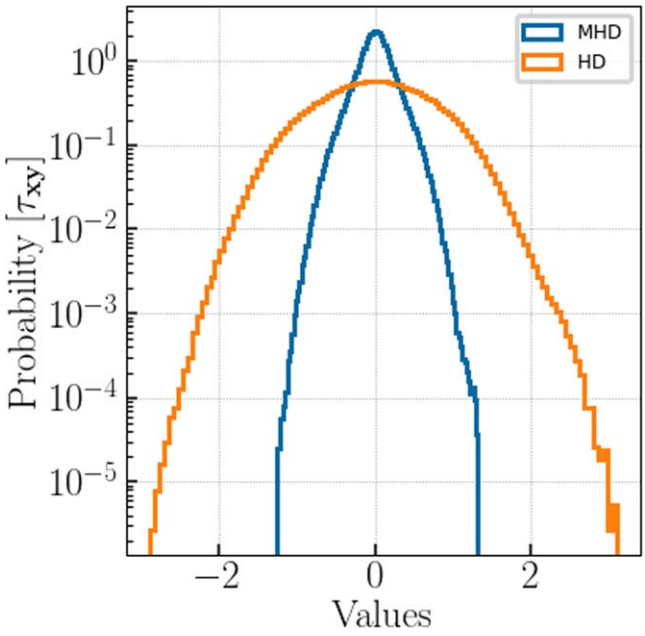


Figure 1. An example of the difference that a magnetic field can bring to the turbulence stress distribution. The data have been taken from JHTDB MHD and HD data sets (Li et al. 2008).

To tackle the challenges of MHD turbulence in an astrophysical setting, we turned to machine learning (ML). In HD simulations, ML has shown promising results in the fields of applied mathematics, engineering, and industry as a whole. Over the last few decades, there have been significant technological and algorithmic advancements that have started a new era for the study of turbulence through the means of *Big Data*. In our context, by *Big Data* we mean the abundance of turbulence DNS data that resulted from the Moore's Law evolution of HPC. While the Re of those simulations is not to the natural level of CCSNe, it is still a significant improvement upon the resolution of the current CCSN models. Furthermore, DNS data in non-astronomy fields has been used as the ground truth, along with experimental data, for turbulence subgrid model optimization in large-scale 3D computational fluid dynamic simulations. *Big Data* presents opportunities for ML to help to augment and improve turbulence modeling. As shown by industry (e.g., facial recognition and self-driving cars), ML is highly flexible, and it has already shown its potential for turbulence modeling—both the direct prediction of turbulent fluxes and analytical (e.g., RANS-based) model optimization (King et al. 2016; Wu et al. 2018; Zhu et al. 2019; Rosofsky & Huerta 2020).

ML has been applied to quasistationary 2D ideal MHD turbulence in an astrophysical setting with promising results, as compared to the conventional analytic gradient subgrid turbulence model (Rosofsky & Huerta 2020). In this paper, we develop an ML model for 3D MHD turbulence for astrophysical simulations in reduced dimension, as well as perform a time-dependent prediction analysis. Considering the popularity and success of convolutional neural networks (CNNs) in the industry (Krizhevsky et al. 2012; He et al. 2016) for spatial pattern recognition, we seek to relate the large-scale eddy structure to small-scale turbulence distribution. Thus, we base our approach on CNNs to develop a physics-informed ML

(PIML) turbulence closure model, described in Section 3.3.2. It is applied to two regimes to test its generalizability: (1) statistically stationary homogeneous isotropic MHD turbulence from a general purpose data set and (2) dynamic MHD turbulence from a CCSN simulation. The former can be found in the interstellar medium, studying the stellar formation, and the latter is directly applicable to high-energy events, such as CCSNe. The model predicts the Reynolds stress tensor (τ), with the turbulent pressure ($P_{\rm turb}$) defined as:

$$P_{\rm turb} = tr(\tau), \tag{1}$$

where $tr(x)$ is a trace of x. Note that we neglect the $1/3$ coefficient in our definition of $P_{\rm turb}$, which has no effect on the ML prediction results. We will primarily focus on analyzing the statistical distributions, i.e., the probability density functions (PDFs), of time-dependent turbulence. In the case of stationary turbulence, we will be testing the stability of the ML model, ensuring that the prediction remains in the physical domain. For the dynamic case, we will check the model's ability to make future predictions within the limits of the available *ground truth* data, i.e., the DNS data that we assume to accurately represent the physical state of the system.

In Section 2, we will cover filtering, decomposition, and the resulting MHD formalism. Section 3 introduces the analytical subgrid turbulence model that we will be using for comparison, our ML pipeline, the specifics of the ML and PIML models that are used to treat various τ_{ij} components, and the data sets used for training and testing them. In Section 4, we provide an analysis of the stationary and dynamic results, with conclusion following in Section 5. Lastly, the Appendices include further details of the ML model developed and its training process.

2. Formalism

We begin by presenting the mathematical basis of our work, covering the fundamentals of MHD LES, including its unfiltered and filtered forms.

2.1. Filtering

A filtering operation is defined as an infinitely resolved—i.e., continuous—variable that is decomposed into *average* and *fluctuating* parts:

$$u = \bar{u} + u', \tag{2}$$

where $\bar{u}$ (an LES-simulated quantity) is the ensemble average of u (a DNS quantity), and u' are the fluctuations. By cutting out fluctuations of a specific size, what is left can be thought of as a filtered quantity. Then, $\bar{u}$ is defined by applying a filtering convolution kernel G:

$$\bar{u} = G * u. \tag{3}$$

In the context of LES, the simulation resolution is defined as a spatial filter of size Δ applied to a continuous variable. What we have done in this study is typical for the LES community: taking high-resolution DNS data and applying a filter of size Δ_f, where $\Delta_f > \Delta$, to decrease ("blur") the fidelity of the data to mimic a low-resolution simulation.

We applied a Gaussian filter to all of the data, with a 1D Gaussian kernel as G:

$$G(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{x^2}{2\sigma^2}}, \tag{4}$$

where σ is the standard deviation of the Gaussian, controlling the amount of "blur," and x is the data. The filter can be applied in 3D via the product of 1D Gaussian functions, covering each direction.

2.2. MHD Equations—Unfiltered

In order to bridge the gap between filtered and unfiltered values, i.e., a stress tensor that we will model, let us first establish the basis of the ideal MHD Navier–Stokes equations. The evolution equations for continuity, momentum, and induction follow the notation from Grete (2017)

$$\begin{gathered}\partial_t \rho + \nabla \cdot (\rho \mathbf{u}) = 0 \\ \partial_t(\rho \mathbf{u}) + \nabla \cdot (\rho \mathbf{u} \otimes \mathbf{u} - \mathbf{B} \otimes \mathbf{B}) + \nabla\left(P + \frac{B^2}{2}\right) = 0 \\ \partial_t \mathbf{B} - \nabla \times (\mathbf{u} \times \mathbf{B}) = 0\end{gathered} \tag{5}$$

where ρ is density, P is pressure, u is velocity, and B is a magnetic field that incorporates the units of $1/\sqrt{4\pi}$.

2.3. MHD Equations—Filtered

For the filtered LES equations, we need to apply Equation (4) to Equation (5). As a result, we get:

$$\begin{gathered}\partial_t \bar{\rho} + \nabla \cdot (\bar{\rho} \tilde{\mathbf{u}}) = 0 \\ \partial_t(\bar{\rho} \tilde{\mathbf{u}}) + \nabla \cdot (\bar{\rho} \tilde{\mathbf{u}} \otimes \tilde{\mathbf{u}} - \overline{\mathbf{B}} \otimes \overline{\mathbf{B}}) + \nabla\left(\bar{P} + \frac{\bar{B}^2}{2}\right) = -\nabla \cdot \tau^{mom} \\ \partial_t \overline{\mathbf{B}} - \nabla \times (\tilde{\mathbf{u}} \times \overline{\mathbf{B}}) = \nabla \times \mathcal{E}\end{gathered} \tag{6}$$

where $\tilde{\mathbf{u}} = \bar{\rho}\tilde{\mathbf{u}}/\bar{\rho}$, which is the mass-weighted filtering, i.e, Favre filtering; τ^{mom} stands for the momentum subgrid-scale (SGS) stress and $\mathcal{E}$ is the turbulent electromotive force. These are defined as follows:

$$\tau_{ij}^{mom} = \bar{\rho}\tau_{ij}^{kin} - \tau_{ij}^{mag} + (\overline{B^2} - \bar{B}^2)\frac{\delta_{ij}}{2} \tag{7}$$

$$\mathcal{E} = \overline{\mathbf{u} \times \mathbf{B}} - \tilde{\mathbf{u}} \times \bar{\mathbf{B}} \tag{8}$$

where δ_{ij} is the Kronecker delta, τ^{kin} is the stress due to the turbulent motion, i.e., Reynolds stress, and τ^{mag} is the Maxwell stress:

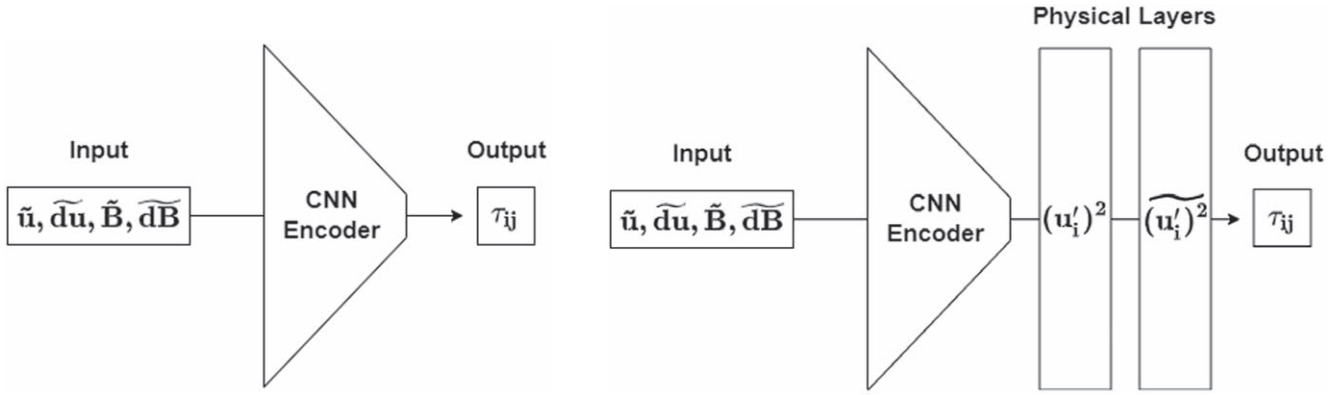
$$\tau_{kin}^{ij} = \widetilde{u^i u^j} - \tilde{u}^i \tilde{u}^j \tag{9}$$

$$\tau_{ij}^{mag} = \overline{B_i B_j} - \overline{B}_i \overline{B}_j \tag{10}$$

In this paper, we will focus on τ^{kin}, which will be further referred to as τ_{ij}, to simplify the notation.

3. Subgrid Modeling

We will be comparing our ML results with a conventional turbulence subgrid model that is used widely in astronomy—the gradient model (Liu et al. 1994).



(a) Off-diagonal terms (3D CNN) (b) Diagonal terms (PIML)

Figure 2. Model schematics to calculate the Reynolds stress (τ_{ij}) components.

3.1. Gradient Model

The gradient model is defined by the Taylor series expansion of the filtering operation. The tensor has the form of:

$$\tau_{ij} = \frac{\widetilde{\Delta}^2}{12} \partial_k \tilde{u}_i \partial_k \tilde{u}_j \tag{11}$$

where $\tilde{\Delta}$ is the filter size and u is velocity.

3.2. ML Pipeline

In its essence, ML is a sophisticated fitting routine of a multidimensional data set against a target feature. However, unlike a fitting routine, ML does not require a theoretical understanding of the underlying statistical form of the data. Thus, the exact relationship of a feature to the target does not need to be defined, in contrast to conventional fitting routines. ML methods are capable of *learning* the data structure solely from the input data, with the model tuning being based on a validation data set (Bishop 2006; LeCun et al. 2015). This opens up the possibility of learning new links between the input variables, potentially leading to new physics and functional forms (Carleo et al. 2019). While we will not delve deeper into the latter topic in this paper, we will discuss how to use physics to inform and then further analyze the model training.

Lastly, ML models can learn iteratively, hence they improve themselves as new data become available. This signals the potential to achieve accurate interpolation/classification and, more importantly, extrapolation results (Carleo et al. 2019). This applies to both spatial and temporal data.

Currently, there are ML models that are based on CNNs. They are used as generalizable solutions and are standard in the industry (e.g., AlexNet Krizhevsky et al. 2012, ResNet He et al. 2016). However, those models are not optimized to solve problems in physical sciences, including astrophysics. Considering the lack of standardized packages for ML in astrophysics, we built our own pipeline, Sapsan[4] (Karpov et al. 2021). Here is a high-level procedure overview:

1. Data: choose a relevant high-fidelity data set. In our case, the data come from the DNS simulations that we consider to be the ground truth.
2. Data augmentation: filter and augment the data to mimic an LES simulation, in which the ML pipeline would be used. For example, the turbulent features in the CCSN LES simulations are severely underresolved. Hence, the filtering applied to high-resolution DNS simulation data needs to account for that adequately.
3. Data splitting: split the data into training, validation, and testing portions.
4. Optimization and training: optimizing the hyperparameters of the ML model via cross-validation and testing against the unseen data. In this context, *unseen* data is defined by the data not used in the training or validation of the ML model. This procedure strives to achieve the best efficiency and accuracy of the model.
5. Validation, testing, and analysis: test the trained ML model to confirm the predictions as being representative of the relevant physics, as well as estimating the efficiency and uncertainty of the ML scheme.

Next, we will discuss how we adopted and augmented conventional ML methods for CCSN turbulence prediction, enforcing physical principles to aid our studies.

3.3. ML Models

We used two ML models to calculate all nine components of the Reynolds stress τ_{ij}: a conventional CNN encoder for off-diagonal terms and a custom, physics-informed (PI) CNN encoder for diagonal terms. The schematics of the models are shown in Figures 2(a) and 2(b), with discussions of each in Sections 3.3.1 and 3.3.2, respectively. Both models have been trained on a dual-GPU system, equipped with NVIDIA Quadro RTX 5000 cards.

3.3.1. Off-diagonal Terms (3D CNN)

The idea behind a neural network is illustrated in Figure 2(a) as a pipeline schematic. We first need to input the data, which is represented by the input layer. Then the data will need to be manipulated in some way, as represented by the hidden layer (s), e.g., the CNN Encoder. At the end, there is an output layer, predicting our target quantity.

We based our model on the off-diagonal tensor components of a 3D CNN, with some modifications, while keeping it conventional. In a CNN, a convolution layer is applied as a hidden layer. In this case, a given kernel is used to parse through the data set to identify the spatial patterns needed for

[4] https://github.com/pikarpov-LANL/Sapsan

the given problem. The kernel has the form:

$$out(N_i, C_{\rm out}) = bias(C_{\rm out} + \sum_{k=0}^{C_{\rm in}-1} weight(C_{\rm out,k}) input(N_i, k)), \quad (12)$$

where N is the number of features, i is the feature index, and C is the number of channels. The input data size is defined by [N, $C_{\rm in}$, D, H, W] and the output is [N, $C_{\rm out}$, $D_{\rm out}$, $H_{\rm out}$, $W_{\rm out}$], where [D, H, W] are the [depth, height, width], i.e., [x,y,z]. The notation is in agreement with the PyTorch documentation. The reason for choosing a CNN as our core ML algorithm was the goal of relating the spatial structure of the turbulent eddies to the small-scale structure.

We utilized the PyTorch built-in modules, with slight modifications for our workflow, with the following parameters:

1. Model: a classical approach for CNN architectures, where the convolution and pooling layers are stacked up, and consequently followed by fully connected dense layers, as is shown in Figure 11, based around a 3D CNN.[5]
2. Optimizer: Adam Optimizer[6]—an extension of the stochastic gradient descent; it was picked due to its good performance on sparse gradients.
3. Activation function: LogSigmoig[7]—a nonlinear activation function to select the values to pass from layer to layer. The function is defined by $\log(\frac{1}{1_0+\exp(-x)})$.
4. Loss function: Custom SmoothL1Loss[8]—an L_1 loss that is smooth if $|x-y|<\beta$, where $\beta=1\sigma$ and σ is the standard deviation. The loss for $|x-y|<\beta$ was further increased by a factor of 10, to aid the efficiency of the training convergence of the model. It can be viewed as a combination of L_1 and L_2 losses (i.e., it behaves as L_1 if the absolute value is high or as L_2 if the absolute value is low).

Besides the network itself, the reasons for the choices of the Optimizer, Activation Function, and the Loss Function were the broad applicability, availability, and success of these techniques. In addition, we performed cross-validation over the available PyTorch functions to solidify our choices. These parameters were sufficient for off-diagonal terms of τ_{ij} for 3D MHD turbulence. However, the physical conditions had to be enforced in order to model the diagonal terms and ultimately predict $P_{\rm turb}$.

3.3.2. Diagonal Terms (PIML)

If calculated directly, the Reynolds stress is defined by:

$$\tau_{ij} = \widetilde{u_i' u_j'} = \widetilde{u_i u_j} - \tilde{u}_i \tilde{u}_j, \quad (13)$$

where u_i' is a velocity fluctuation component and $\tilde{x}$ is the spatial average. Thus, for diagonal terms, a realizability condition is defined as $\tau_{ii}>0$ (Schumann 1977), making the distribution of the diagonal tensor components asymmetric. While the model in Section 3.3.1 excelled at quasi-symmetric distribution prediction, it struggled with asymmetric distributions. Further analysis of this will be covered in Section 4.2.

1. Model: 3D CNN encoder, as described in Section 3.3.1, with PI layers. The encoder implicitly predicts the velocity fluctuations (u_i'), then the PI layers calculate $u_i'^2$, to enforce $\tau_{ii}>0$ and filter to find the mean, as per Equation (12).
2. Optimizer: Adam Optimizer—the same as in Section 3.3.1.
3. Activation function: LogSigmoid (stationary) and Tanhshrink[9] (dynamic)—the latter showed a faster model convergence rate for the dynamic turbulence. The function is defined by
$$f(x) = x - tanh(x).$$
4. Loss function: Custom—a dynamic combination of SmoothL1Loss (point to point) and Kolmogorov–Smirnov (KS) statistical (Massey 1951) losses.

The combined loss function was designed to compound the advantages of the point-to-point and statistical losses. Further discussion can be found in the discussion of the results in Section 4.2.

The PyTorch implementation of this PIML model, used for diagonal terms, along with the 3D CCSN data sampled down to 17^3, is provided as part of the Sapsan package.[10]

Table 1
Parameters of the Statistically Stationary (JHTDB) and Dynamic CCSN (Mösta et al. 2015) Turbulence Data Sets

	Stationary	Dynamic
Resolution	1024^3	347^3
$t_{\rm tot}$	2.56	10.3 [ms]
δt	2.5×10^{-4}	5×10^{-4} [ms]
Δt	2.5×10^{-3}	2.4×10^{-2} [ms]
$k_{\rm max}$	482	348
$\overline{KE/E_{\rm tot}}$	~0.5	~0.9
$\overline{E_B/E_{\rm tot}}$	~0.5	~0.1
σ	16	9

Note. The time values for the stationary data set are in normalized numerical units, amounting to the turnover time of one large eddy. $t_{\rm tot}$ is the total simulation time, δt is a time step, Δt is the checkpoint time separation, $k_{\rm max}$ is the maximum Fourier mode, and $\overline{KE/E_{\rm tot}}$ and $\overline{E_B/E_{\rm tot}}$ are the time-averaged fractions of kinetic and magnetic energy, with respect to the total energy. σ is the Gaussian filter standard deviation that we applied during the data preparation. Lastly, the spatial resolution of the dynamic data set is $\Delta x = 200\ m$.

3.4. Data Sets

In ML, the predictions will only be as good as the training data. With the goal of testing our algorithm on a broader range of physical conditions, we diversified by using stationary and dynamic turbulence data sets. A quick parameter overview of both data sets can be found in Table 1.

[5] https://pytorch.org/docs/stable/generated/torch.nn.Conv3d.html
[6] https://pytorch.org/docs/stable/generated/torch.optim.Adam.html
[7] https://pytorch.org/docs/stable/generated/torch.nn.LogSigmoid.html
[8] https://pytorch.org/docs/stable/generated/torch.nn.SmoothL1Loss.html
[9] https://pytorch.org/docs/stable/generated/torch.nn.Tanhshrink.html
[10] https://sapsan-wiki.github.io/details/estimators/#physics-informed-cnn-for-turbulence-modeling-pimlturb

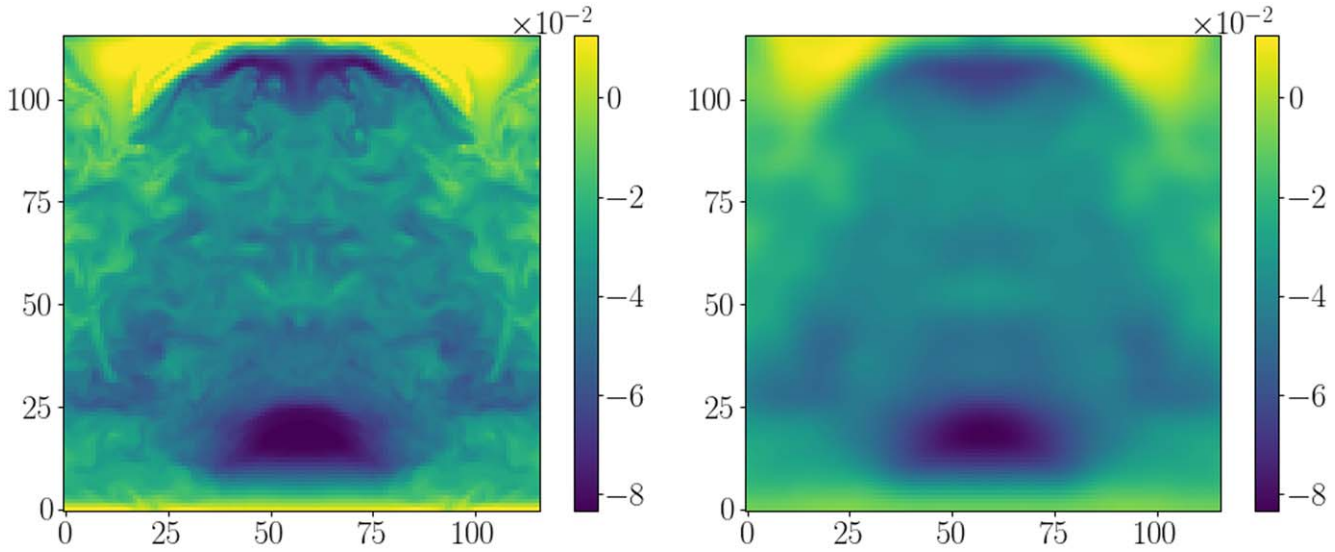


Figure 3. A box filter with a Gaussian kernel ($\sigma = 9$) was used to filter the data. The figure shows a slice of sampled u_x, down to (116,116,116); the filter was applied prior to sampling. Left: original; right: filtered.

1. Stationary: a high-resolution, statistically stationary, isotropic, forced, incompressible MHD turbulence data set[11] from the JHTDB (Li et al. 2008). It has a low Reynolds number fluctuating around $Re \sim 186$.
2. Dynamic: an evolving, highly magnetized CCSN turbulence data set by Mösta et al. (2015). It is a 3D DNS of an MHD CCSN. The data set has been developed to study the effects of MRI in growing the turbulence in a CCSN scenario to aid the explosion, aiming to prove the plausibility of CCSNe being progenitors of LGRBs and magnetars (Mösta et al. 2015). The needed resolution is high, so it is not a global simulation, only tracking the first ∼10 ms after the core bounce to see the development of turbulence. In addition, only a quarter of the star close to the core has been simulated (66.5 km in x and y and 133 km in z), maintaining a 90° rotational symmetry in the xy plane. To fit our memory constraints, we used the data set with $\Delta x = 200$ m resolution, which exhibits mild turbulence.

3.4.1. Data Preparation

In order to reflect a realistic low-fidelity environment of the CCSN simulations, we chose to apply a Gaussian filter (Carati et al. 2001), as described in Section 2.1. The standard deviation σ of the filter (Equation (4)) for each data set was chosen, to provide similar levels of filtering for both the stationary and dynamic data sets, with the exact σ specified in Table 1. An example of the filtering used is shown in Figure 3. For both data sets, derivatives were calculated, and the filter was applied at the highest resolution available. Then, the data were equidistantly sampled down to 116^3, to fit the hardware memory constraints. The exact data preparation procedures are summarized below:

1. Calculate the derivatives of $[u, B]$.
2. Calculate the Reynolds stress tensor τ using Equation (9).
3. Apply a Gaussian filter to the original data, to get $[\tilde{u}, \widetilde{du}, \tilde{B}, \widetilde{dB}]$.
4. Sample the data equidistantly down to 116^3.
5. Use the sampled quantities of $[\tilde{u}, \widetilde{du}, \tilde{B}, \widetilde{dB}]$ as the model input.
6. Use each τ_{ij} component as the model output.

[11] https://turbulence.pha.jhu.edu/Forced_MHD_turbulence.aspx

4. Results and Discussion

4.1. Stationary Turbulence

Though the data set is evolving spatially, the statistics remain stationary in the JHTDB MHD data set that we used. Our CNN and PIML models for off-diagonal and diagonal terms, respectively, outperform the traditional gradient subgrid model. Figure 4 presents predictions of the x components of the Reynolds stress: $[\tau_{xx}, \tau_{xy}, \tau_{xz}]$ at $\Delta t = 1000$, which is near the end of the simulation. The prediction performance of the y and z components remained comparable to the x components; hence those plots were omitted.

For training, we used the first four checkpoints (Δt), separated by 10 time steps (δt) each. The exact data preparation was performed as per Section 3.4.1. Our PIML method especially excelled at predicting τ_{xx}, which is consequently important for the P_{turb} calculation, while the gradient model completely missed the peak and the overall turbulent distribution. Next, τ_{xy} matches the bulk of the data, but overpredicts the outliers. Note that the y-axis is on a log scale; hence the actual error remains minimal. As for τ_{xz}, the CNN model does not show any particular weaknesses.

The point of this exercise was to test the reliability of the CNN algorithm when it was applied to a changing spatial distribution. Since CNNs parse a kernel through the data cube, it is not trivial to assume statistical consistency in the predictions, based on the evolving spatial distribution. Nonetheless, the statistically stationary data set did not require significant tuning to achieve its results, and served more as a consistency check for our algorithms before moving to the dynamic data set.

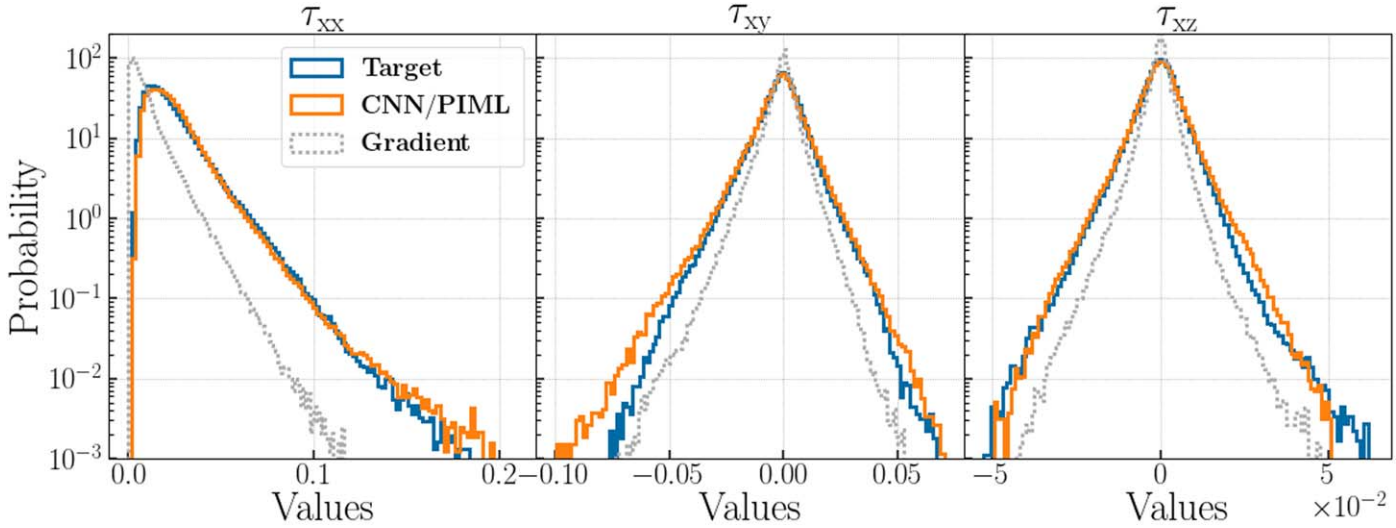


Figure 4. Prediction of x components of τ at $t = 1000$ of the JHTDB MHD data set in normalized numerical units. Blue shows the original target data, orange shows the predictions of the CNN or PIML models, for off-diagonal and diagonal terms, respectively, and gray shows the result of the gradient subgrid model, as per Section 3.1.

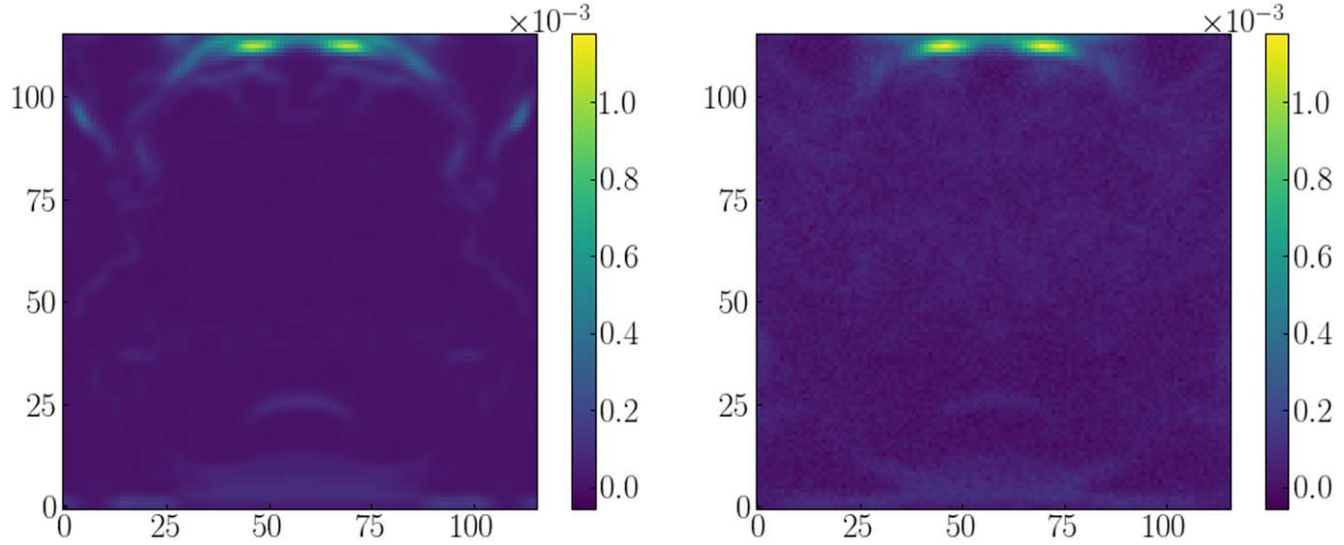


Figure 5. Slices presenting the spatial distribution of the 3D stress tensor component at $t = 7.55$ ms, sampled down to 116^3. Left: target τ_{xx}; right: PIML prediction of τ_{xx}. The statistical distribution of the above slices can be found in Figure 6 (first row, second column).

4.2. Dynamic Turbulence

The ultimate goal of our study was to test the models on a more realistic astrophysical data set. While the DNS CCSN simulation from Mösta et al. (2015) has its limitations, it is the best-resolved turbulence data set investigating CCSNe. Figure 6 presents predictions of τ_x components in the second half of the simulation, $t = [5.62,\ 7.55,\ 9.48]$ ms. The results remain consistent with the statistically stationary JHTDB predictions. The gradient model continuously underpredicts the Reynolds stress distribution, performing especially poorly at capturing the outliers. The predictions of the τ_y and τ_z components are comparable in accuracy across all time steps, hence they were omitted from the plot. Figure 5 presents an example of a spatial distribution prediction, i.e., a slice of the data cube at $t = 7.55$ ms.

The key to capturing the dynamics was to train across a wide range of checkpoints covering the first half of the simulation, $t < 5$ ms. We were able to optimize the results using nine equally distant checkpoints; the evolution of the normalized PDFs of τ_{xy} from the exact checkpoints used in training can be seen in Figure 9. Such a diversified training data set helped to prevent overfitting, while maintaining the flexibility of the model in predicting future time steps ($t > 5$ ms).

Thus far, the CNN methods have worked well for off-diagonal terms, and our PIML has enforced the realizability condition ($\tau_{ii} > 0$) for the diagonal terms. However, the diagonal terms of the dynamic turbulence data set presented another challenge—asymmetric statistical distributions. A CNN with a point-to-point loss such as SmoothL1Loss has shown its robust performance in predicting quasi-symmetric distributions. This includes predictions of the diagonal terms in the JHTDB stationary data (τ_{xx} in Figure 4), where, due to the shift of the peak, the distribution can be classified as quasi-symmetric. However, in the dynamic data set, the peak is near the origin, making the distribution acutely asymmetric. As a result, while accurately capturing the outliers, the previous approach failed

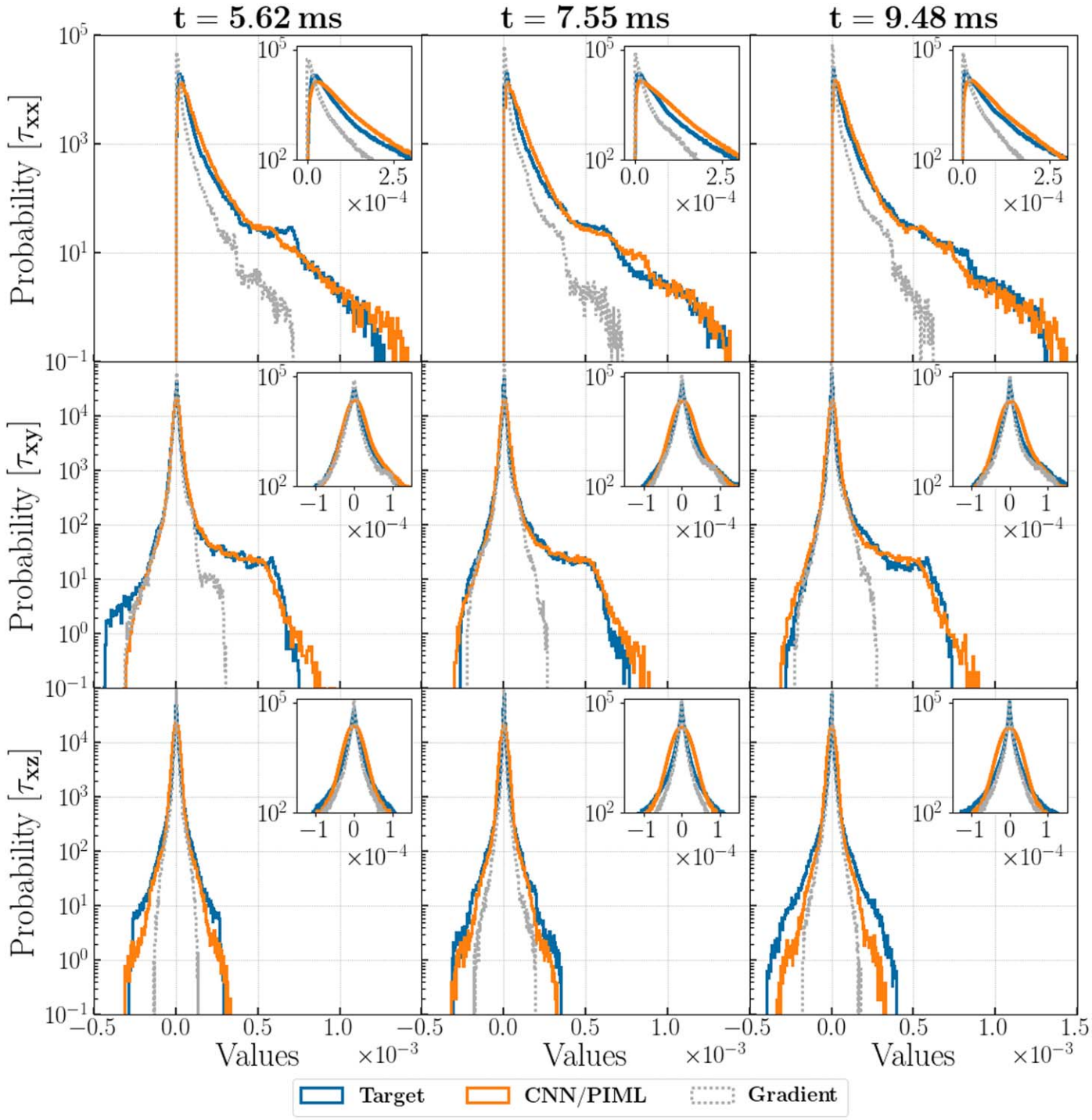


Figure 6. Statistical distributions of the stress tensor component, where the values are in units of u^2. Rows: [τ_{xx}, τ_{xy}, τ_{xz}] components; columns: simulation times at [5.62, 7.55, 9.48] ms. Blue lines: target τ distribution; orange lines: CNN prediction; gray dotted lines: gradient turbulence subgrid model of the form $\tau_{ij} = 1/12\Delta^2\delta u_{ik}\delta u_{jk}$, using the Einstein notation, where Δ is the filter width and u is the velocity.

to predict the correct peak position, i.e., the bulk of the data. To remedy this behavior, we developed a custom loss function, combining a point-to-point SmoothL1Loss with a loss based on the KS statistic (KS_{stat}). The latter metric is the maximum distance between the two cumulative distribution functions (CDFs), i.e., how far apart the two distributions are from one another.

SmoothL1Loss excelled at predicting the distribution outliers, while it struggled to determine the peak position. On the other hand, the KS loss excelled at predicting the bulk of the data, including the peak position, by minimizing the distance between the input and target distributions, but it struggled with the outliers. As a result, the two losses were combined in a dynamic fashion. The model first minimized SmoothL1Loss, to get the overall distribution shape, particularly the outlier portion, and then it minimized the KS loss, to shift the peak into the right position. The results can be seen in the top row of Figure 6, with the detailed peaks presented in a separate box

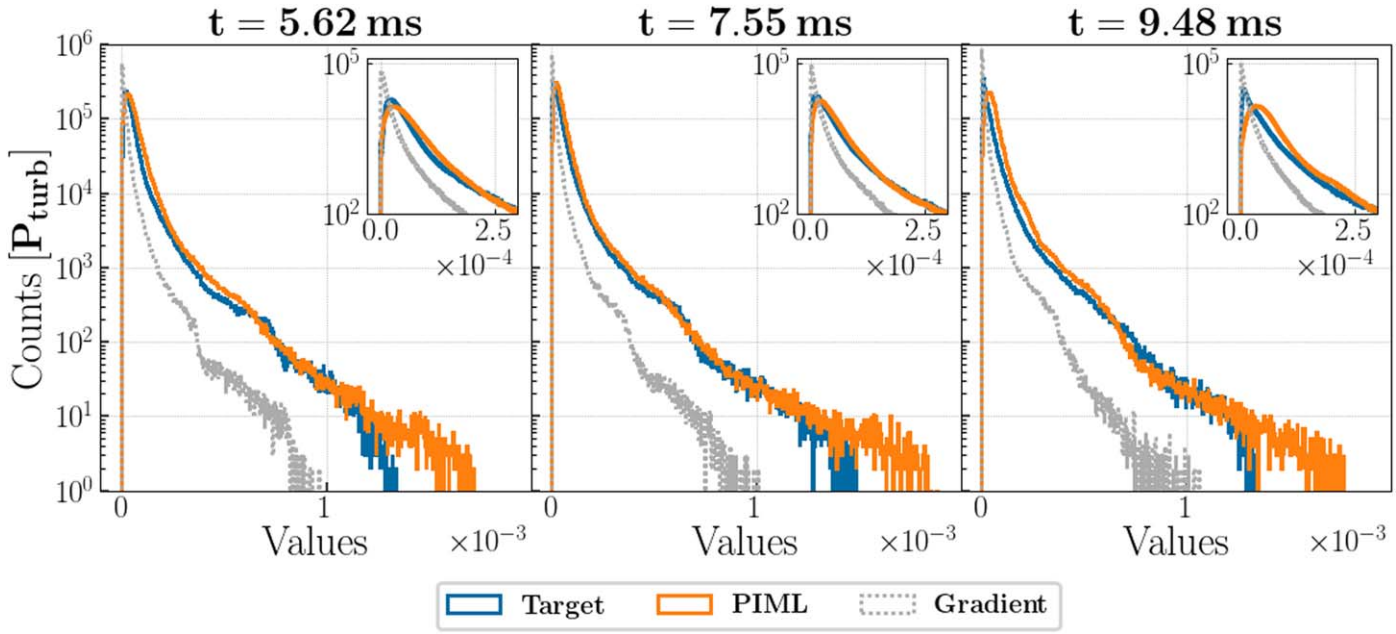


Figure 7. The plot presents an unnormalized distribution of $P_{\rm turb}$ as time evolves. $P_{\rm turb}$ is in units of u^2. We compare the performance of our PIML model with the gradient turbulence subgrid model.

within each plot. Further details of the training loss behavior are provided in Appendix B. While we primarily stressed an accurate prediction of the statistical distribution, another benefit of not using an exclusively statistical loss is a sound spatial prediction, as shown in Figure 5.

The leading deliverable of the τ_{ij} predictions is the ability to calculate the turbulence pressure via Equation (1). Thus, any deviation in the peak of τ_{ii} is further exacerbated when computing $P_{\rm turb}$. As an example of our PIML model's performance, we present the $P_{\rm turb}$ prediction calculations at $t = [5.62, 7.55, 9.48]$ ms, as shown in Figure 7. There, the trace was taken from the sorted τ_{ii} components. During this operation, the spatial distribution of $P_{\rm turb}$ is lost, though it is not required for our main goal: an accurate prediction of the total pressure due to turbulence in the region and its statistical distribution. This is due to the convection region being extremely underresolved, while it is responsible for supplying $P_{\rm turb}$ to the stalled shock for the potential explosion in the global CCSN simulations. Thus, the astrophysical question is reduced to a binary one: will the stalled shock move outward (explosion) or inward (black hole)? Consequently, the accuracy of the total $P_{\rm turb}$ in the convection region becomes the most significant, while alleviating the need for an accurate prediction of the spatial distribution.

The performance of the PIML method shows significant advantages over the gradient model in predicting the distribution, the outliers, and the peak position. Its performance does deteriorate with time, as can be seen by the slight peak shift in the right plot of Figure 7, at $t = 9.48$ ms. Quantitatively, the performance metrics for comparing PIML and the gradient model predictions are summarized in Figure 8. The top panel shows that the total $P_{\rm turb}$ calculated from the PIML predictions overpredicts the target ground truth (the dynamic 3D DNS CCSN data) by ∼5%–35%, depending on the future prediction time, resulting in a ∼19% deviation on average. This is a significant advantage over the ∼63% underprediction error of the gradient model, which will fail to supply sufficient $P_{\rm turb}$ to

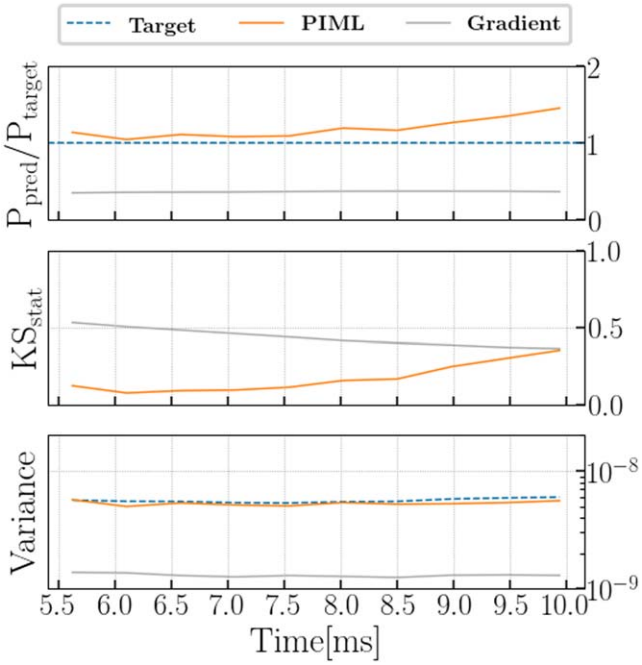


Figure 8. Performance metrics of the PIML vs. gradient subgrid models. The top panel shows the ratio of the total turbulent pressure calculated from the model prediction to the target dynamic 3D DNS CCSN data; the middle panel shows the KS statistic; and the bottom panel shows the variance. In total, metrics at 10 checkpoints equally separated in time are presented in the plots.

reenergize the stalled shock to explode the star. This means that by using the PIML method, $P_{\rm turb}$ could reach on average ∼36% of the gas pressure, instead of the estimated ∼30% in Nagakura et al. (2019), making it easier for the star to explode.

The middle panel of Figure 8 shows the $\rm KS_{stat}$ for the PIML method to degrade to the level of the gradient model at checkpoints far in the future. This large discrepancy in the target and PIML CDFs is due to the slight peak shift of the prior PIML prediction. While $\rm KS_{stat}$ is an important metric used

in our custom loss function, it does not disqualify PIML's advantages over the conventional gradient turbulence model.

Last, the bottom panel presents consistent variance between the target and PIML results. The predicted distribution stays consistent in its dispersion, i.e., its bulk and outlier distribution, which cannot be said for the gradient model results. Thus, the PIML approach has an advantage in modeling the small-scale eddies that, in turn, can grow to larger scales, to provide the dominant fraction of the $P_{\rm turb}$ to reenergize the stalled shock as the simulation evolves.

5. Conclusion

The study of CCSNe requires an accurate treatment of turbulence, yet conventional subgrid turbulence approaches are unreliable. A DNS treatment of the turbulence in global 3D CCSN simulations is not achievable with current computational resources, thus the calculations are typically done via ILES. Although they can capture the effects of large-scale flows with relative accuracy, these simulations neglect the turbulent pressure ($P_{\rm turb}$) entirely, relying on numerical artifacts to represent its effect. Building upon prominent ML techniques used in industry, we have developed PIML networks to increase the predictive accuracy of the Reynolds stress (τ_{ij}), the trace of which is $P_{\rm turb}$. $P_{\rm turb}$ is thus the main deliverable of this paper, which can be used in a CCSN simulation in aid of reenergizing the stalled shock and exploding the star.

Our PIML approach has consistently outperformed a conventional gradient subgrid model for both stationary and dynamic turbulence data sets. It resulted in a $\sim$19% PIML average error of the total $P_{\rm turb}$, in comparison to $\sim$63% of the gradient model. In addition, our method has excelled at predicting the outliers of both τ_{ij} and $P_{\rm turb}$, which are important for dynamic simulations, to investigate the turbulent growth. Given the flexibility of the ML algorithms used, these results should be reproducible across HD and MHD CCSN simulations, which we are currently investigating for our next publication. That being said, the performance of the ML models deteriorates the further in time that the predictions are made, which is to be expected with a CNN-based approach. While, at its worst, it continued to take the lead over the gradient model, the temporal and overall performance can be further improved in our future work, with the inclusion of recurrent neural network layers in the models or by utilizing PI neural operators (PINOs; Li et al. 2021; Rosofsky & Huerta 2022).

Furthermore, the broader application of the ML model could suffer from data–model inconsistencies when integrating a trained model within CCSN codes. The distribution discrepancies between the training data set and the newly simulated grids, as well as the numerical errors, could lead to an exponential error growth in the predictions (Beck & Kurz 2021). Regularization of the model predictions can improve its stability. Our future work will investigate the approaches to tackling this potential issue.

This paper has been our first attempt at studying the generalizability of ML methods for studying turbulence over different physical regimes. In the future, we would like to delve deeper into this topic, employing other 3D MHD CCSN data sets. Specifically, we here used a DNS MHD CCSN data set of a single star, while we would like to expand the study to both HD and MHD models over a wide range of CCSN progenitor masses (from 8 $M_{\odot}$ to 25 $M_{\odot}$), which exhibit great variation in their physical engines. In our next paper, we will present our current implementation of the evolving turbulent pressure term, trained on 3D simulation data, into 1D CCSN models, to study a large parameter space of progenitors and understand its impact on CCSN explosion rates.

The research presented in this paper was supported by the Laboratory Directed Research and Development program of Los Alamos National Laboratory (LANL), a LANL Center of Space and Earth Science student fellowship, and the DOE ASCR SciDAC program. This research has used resources provided by the LANL Institutional Computing Program, which is supported by the U.S. Department of Energy National Nuclear Security Administration, under Contract No. 89233218CNA000001. We would also like to thank Philipp Mösta for providing the 3D MHD CCSN data set used throughout this paper, and Jonah Miller for insightful discussions that helped develop the ML methods.

Appendix A
Training Features

The ML models were trained on u, du, B, and dB as the input features, and τ_{ij} as the target feature: a model per tensor component. Figure 9 presents an example of how τ_{xy} evolves at $t < 0.5$ ms, following the exact checkpoints used for training. The other τ_{ij} components follow a similar level of dynamics. This provided a sufficient level of diversity in the training data set to prevent model overfitting and aid the flexibility of the model.

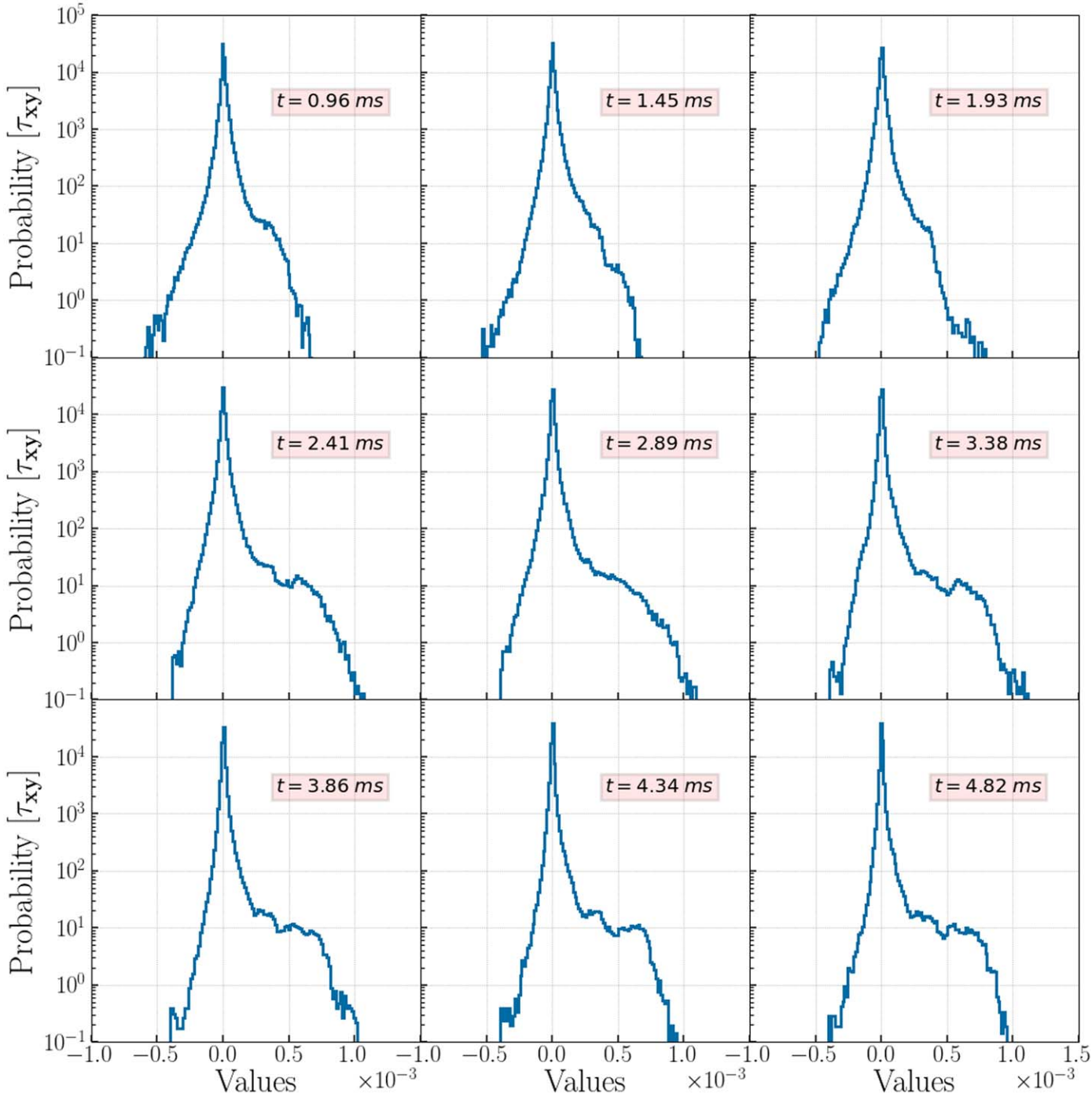


Figure 9. Evolution of the statistical distribution of the stress tensor component (τ_{xy}) in the first half of the simulation ($t < 5$ ms). These exact checkpoints, nine in total, were used as a target for training the CNN network.

Appendix B
Training Loss

We developed a custom loss function (l) that combines a point-to-point loss (SmoothL1Loss, i.e., $L1$) with a statistical loss (the KS statistic, i.e., KS_{stat}) in a dynamic fashion. The goal was to force the model to minimize the $L1$ loss first, to get the overall distribution shape. Then, the model was to prioritize minimizing KS_{stat}, to shift the peak into its correct position.

The two losses operated on different scales during training: the $L1$ loss can span a range of $\sim 10^4$ (from 10^{-4} down to 10^{-8}), while KS_{stat} ranges $\sim 10^1$ (from 10^0 down to 10^{-1}), before overfitting. To account for such differences, we first normalized the losses and then applied a scaling factor α, to prioritize $L1$ until the general PDF shape had been learned. Given our training data, at L_1 loss $< 10^{-6}$ this condition was satisfied, making $\alpha = 10^6$. Thus, $L1$ loss is heavily prioritized for the first several orders of magnitude, decreasing the combined loss (l), then sharing an equal weight with KS_{stat}. This results in l following $L1$ loss's training dynamic, as shown by the top and middle plots of Figure 10. In summary, the two

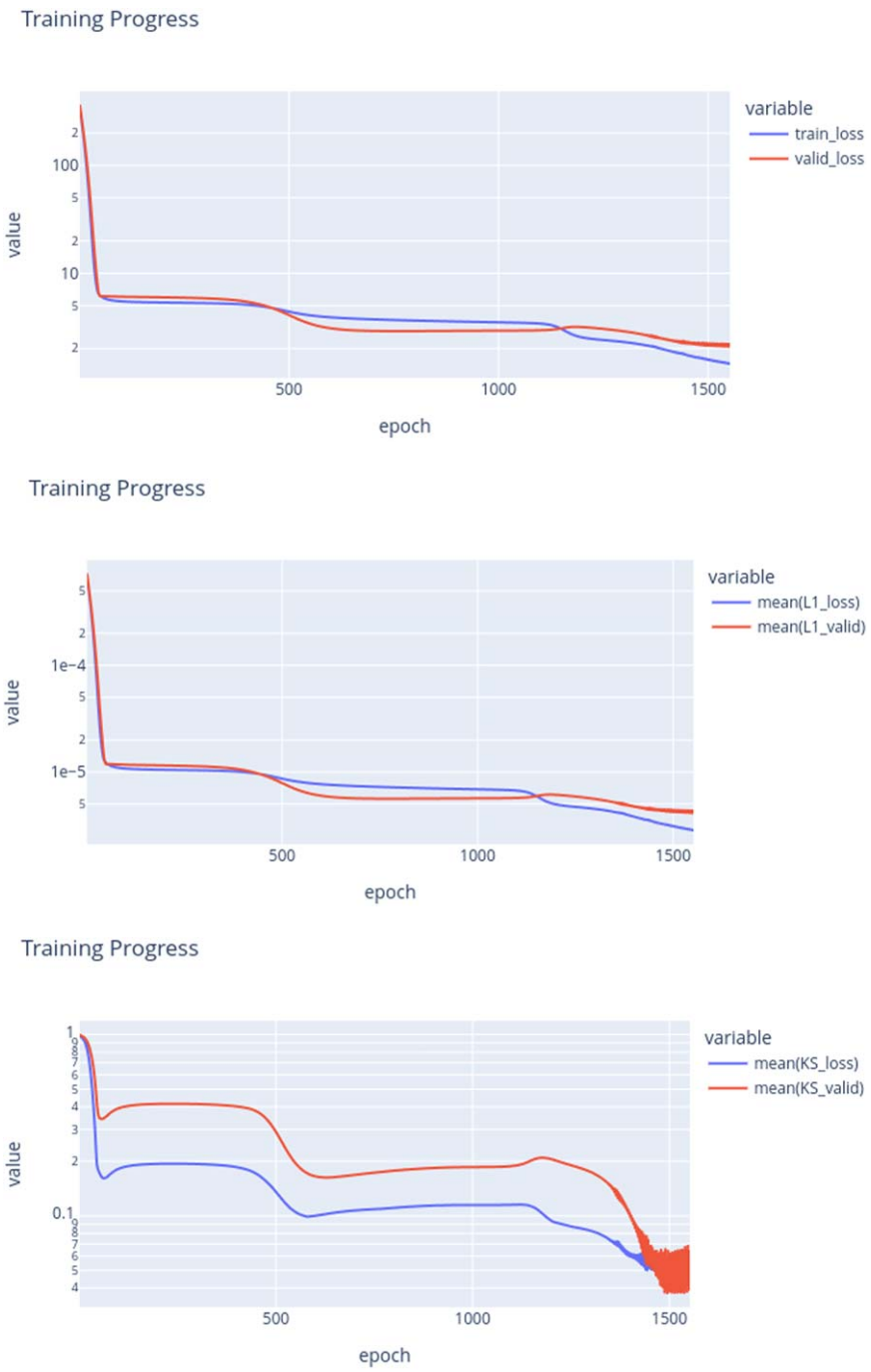


Figure 10. Training loss evolution: the top panel shows the actual loss of the model consisting of combined L_1 and KS loss components; the middle panel shows the L_1 loss component, and the bottom panel shows the $\mathrm{KS}_{\mathrm{stat}}$ loss component.

losses are combined as follows:

$$l = 0.5(\alpha\widetilde{L1}) + 0.5\widetilde{KS}_{stat} \tag{B1}$$

where $\widetilde{X}$ is a spatial average of X.

Once $\mathrm{KS}_{\mathrm{stat}}$ becomes important, the peak is being shifted, and we introduce an early stopping condition to prevent overfitting. The bottom plot of Figure 10 presents the training evolution of the $\mathrm{KS}_{\mathrm{stat}}$ loss component of l. After the training loss (blue) and the validation loss (red) cross at $\sim 4 \times 10^{-2}$, the training loss decreases exponentially, while the validation loss increases exponentially. This indicates that the model is overfitting. Thus, the early stopping condition was set to $\sim 4 \times 10^{-2}$, based on the crossing value of the training and validation losses.

More details of the application and reasoning behind the combined loss can be found in Section 4.2.

Appendix C
CNN Encoder

We present a graph of the CNN encoder that we used in Figure 11. The data shape is noted at each arrow, akin to what was used to produce our results throughout this paper. For the

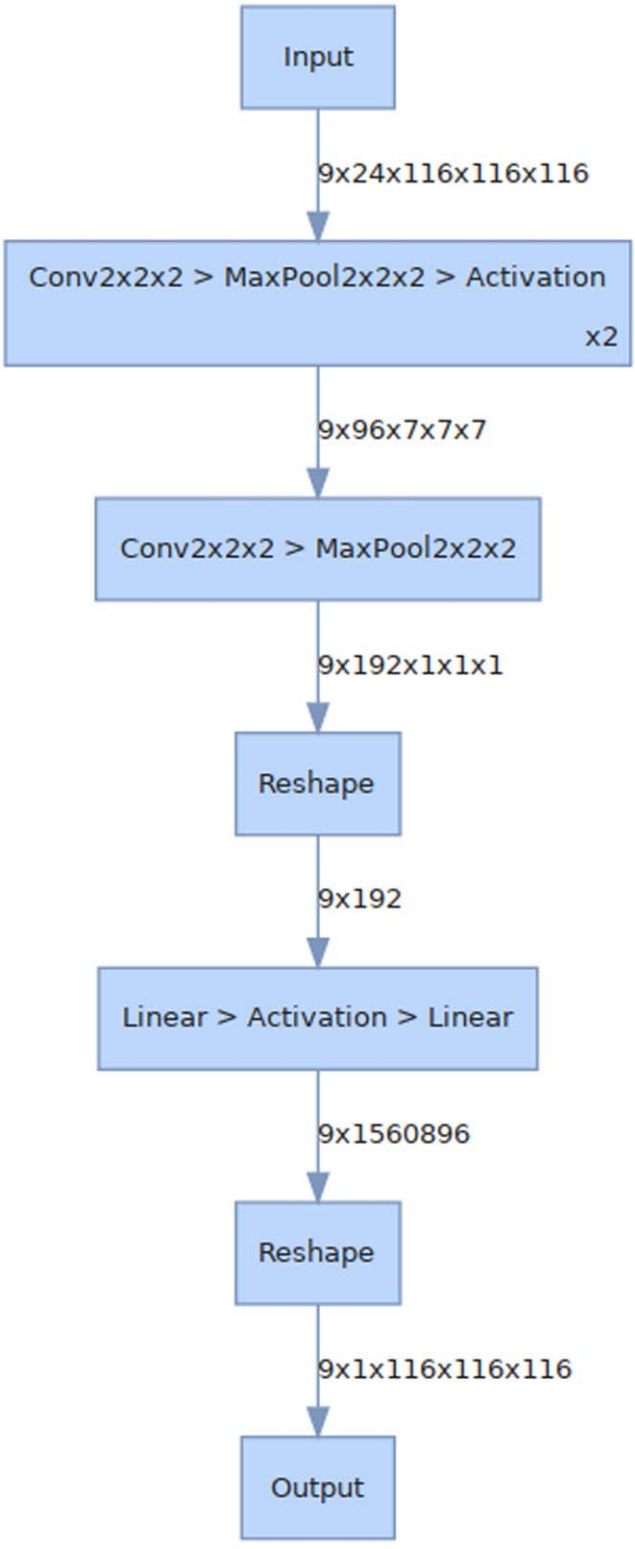


Figure 11. Graph of the CNN encoder used in all models for tensor component prediction. The activation function is either LogSigmoid or ShrinkTanh, depending on the tensor component type. Further information can be found in Section 3.3.1

input and output, the shape is formatted as [N, C, D, H, W], where N is the number of batches, C represents the channels, i.e., features, and [D,H,W] stand for the depth, height, and width of the data. The notation is in agreement with the PyTorch documentation. The graph was produced with Sapsan.[12]

ORCID iDs

Platon I. Karpov https://orcid.org/0000-0003-4311-8490
Chengkun Huang https://orcid.org/0000-0002-3176-8042
Iskandar Sitdikov https://orcid.org/0000-0002-6809-8943
Chris L. Fryer https://orcid.org/0000-0003-2624-0056
Stan Woosley https://orcid.org/0000-0002-3352-7437
Ghanshyam Pilania https://orcid.org/0000-0003-4460-1572

References

Arnett, W. D., Meakin, C., Viallet, M., et al. 2015, ApJ, 809, 30
Baron, E., Bethe, H. A., Brown, G. E., Cooperstein, J., & Kahana, S. 1987, PhRvL, 59, 736
Beck, A., & Kurz, M. 2021, GAMM-Mitteilungen, 44, e202100002
Beresnyak, A. 2019, LRCA, 5, 2
Beresnyak, A., & Lazarian, A. 2014, Magnetic Fields in Diffuse Media (Berlin: Springer), 163
Bishop, C. M. 2006, Pattern Recognition and Machine Learning (Information Science and Statistics) (Berlin: Springer)
Blondin, J. M., Mezzacappa, A., & DeMarino, C. 2003, ApJ, 584, 971
Brandenburg, A., & Lazarian, A. 2013, SSRv, 178, 163
Burrows, A., & Vartanyan, D. 2021, Natur, 589, 29
Burrows, A., Vartanyan, D., Dolence, J. C., Skinner, M. A., & Radice, D. 2018, SSRv, 214, 33
Carati, D., Winckelmans, G. S., & Jeanmart, H. 2001, JFM, 441, 119
Carleo, G., Cirac, I., Cranmer, K., et al. 2019, RvMP, 91, 045002
Colgate, S. A., & White, R. H. 1966, ApJ, 143, 626
Couch, S. M., Warren, M. L., & O'Connor, E. P. 2020, ApJ, 890, 127
Fryer, C. L., & Young, P. A. 2007, ApJ, 659, 1438
Grete, P. 2017, PhD thesis, Max-Planck-Institute for Solar System Research, Lindau
He, K., Zhang, X., Ren, S., & Sun, J. 2016, in 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (Piscataway, NJ: IEEE), 770
Herant, M., Benz, W., Hix, W. R., Fryer, C. L., & Colgate, S. A. 1994, ApJ, 435, 339
Jiménez, J. 2003, JTurb, 4, 22
Joshi, J. B., Nandakumar, K., Patwardhan, A. W., et al. 2019, in Advances of Computational Fluid Dynamics in Nuclear Reactor Design and Safety Assessment, ed. J. B. Joshi & A. K. Nayak (Sawston, UK: Woodhead Publishing), 21
Karpov, P. I., Sitdikov, I., Huang, C., & Fryer, C. L. 2021, JOSS, 6, 3199
King, R. N., Hamlington, P. E., & Dahm, W. J. A. 2016, PhRvE, 93, 031301
Krizhevsky, A., Sutskever, I., & Hinton, G. E. 2012, in Advances in Neural Information Processing Systems, Vol. 25, ed. F. Pereira et al. (Curran Associates Inc.), https://proceedings.neurips.cc/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf
LeCun, Y., Bengio, Y., & Hinton, G. 2015, Natur, 521, 436
Li, Y., Perlman, E., Wan, M., et al. 2008, JTurb, 9, N31
Li, Z., Zheng, H., Kovachki, N., et al. 2021, arXiv:2111.03794
Lilly, D. K. 1966, On the Application of the Eddy Viscosity Concept in the Inertial Sub-range of Turbulence (Vol. 231), 123 (Boulder, CO: NCAR), 1
Liu, S., Meneveau, C., & Katz, J. 1994, JFM, 275, 83
Massey, F. J. 1951, J. Am. Stat. Assoc., 46, 68
Melson, T., Janka, H.-T., Bollig, R., et al. 2015, ApJL, 808, L42
Miesch, M., Matthaeus, W., Brandenburg, A., et al. 2015, SSRv, 194, 97
Mösta, P., Ott, C. D., Radice, D., et al. 2015, Natur, 528, 376
Nagakura, H., Burrows, A., Radice, D., & Vartanyan, D. 2019, MNRAS, 490, 4622
Obergaulinger, M., Cerdá-Durán, P., Müller, E., & Aloy, M. A. 2009, A&A, 498, 241
Radice, D., Abdikamalov, E., Ott, C. D., et al. 2018, JPhG, 45, 053003
Radice, D., Couch, S. M., & Ott, C. D. 2015, ComAC, 2, 7
Rosofsky, S. G., & Huerta, E. A. 2020, PhRvD, 101, 084024
Rosofsky, S. G., & Huerta, E. A. 2022, arXiv:2203.12634
Schekochihin, A. A., & Cowley, S. C. 2007, Turbulence and Magnetic Fields in Astrophysical Plasmas (Dordrecht: Springer), 85
Schmidt, W. 2015, LRCA, 1, 1
Schumann, U. 1977, PhFl, 20, 721
Spiegel, E. A. 1963, ApJ, 138, 216
Woosley, S., & Janka, T. 2005, NatPh, 1, 147
Wu, J.-L., Xiao, H., & Paterson, E. 2018, PhRvF, 3, 074602
Zhu, L., Zhang, W., Kou, J., & Liu, Y. 2019, PhFl, 31, 015105

[12] https://sapsan-wiki.github.io/tutorials/model_graph