UNIVERSITY OF CALIFORNIA
SANTA CRUZ

TURBULENCE MODELING IN CORE-COLLAPSE SUPERNOVAE WITH MACHINE LEARNING

A dissertation submitted in partial satisfaction
of the requirements for the degree of

DOCTOR OF PHILOSOPHY

in

ASTRONOMY AND ASTROPHYSICS
with an emphasis in SCIENTIFIC COMPUTING

by

Platon Igorevich Karpov

September 2023

The Dissertation of Platon Igorevich Karpov
is approved:

Professor Stan Woosley, Chair

Professor Constance M. Rockosi

Professor J. Xavier Prochaska

Chris L. Fryer, Ph.D.

Chengkun Huang, Ph.D.

Peter Biehl
Vice Provost and Dean of Graduate Studies

Table of Contents

List of Figures

List of Tables

Abstract

Turbulence Modeling in Core-Collapse Supernovae with Machine Learning

by

Platon Igorevich Karpov

Chaotic fluid motion, known at the small scales as *turbulence*, can significantly alter the large-scale evolution of astrophysical events. For example, the growth and impact of convection to produce a successful core-collapse supernova (CCSN) depend upon the evolution of turbulence. The ideal way to investigate it in such an environment would be with high-resolution direct numerical simulations (DNS) that resolve the turbulent energy cascade down to some small scale where the energy could be safely assumed to dissipate as heat. Unfortunately, given the high Reynolds number and a large range of spatial scales, this is well beyond the current state-of-the-art computational 3D CCSN models. Since turbulence cannot be properly simulated in 1D or 2D, a subgrid-scale model (SGS) is needed to capture the unresolved 3D physics. Simple analytical SGS models often lack accuracy, though, and complex ones are difficult to tune, resulting in limited generalizability based on initial conditions. Given the recent successes in turbulence SGS modeling with Machine Learning (ML) in adjacent fields, this thesis develops an ML algorithm to analyze current simulations and study the features of turbulence in CCSN. To demonstrate its efficacy, we test our ML approach on modeling

dynamic 3D HD & MHD turbulence, integrating the former into a 1D code to study the role of one specific feature of turbulence (the effective turbulent pressure) on the fate of CCSN simulations. Furthermore, our ML tools can be used to study the broader effects of turbulence, explore other ML architectures, and be integrated into the outside 1- and multi-dimensional CCSN codes. The ML framework (Sapsan) and its implementation in a 1-dimensional code (COLLAPSO1D) are open-sourced and available for public use.

In memory of my grandmother

Tamara Mikhailovna Karpova

and my grandfather

Victor Antonovich Sakhonenko

Acknowledgments

The vast majority of this thesis was carried out at Los Alamos National Laboratory (LANL), with financial support from its Center for Space and Earth Science (CSES) and the Department of Energy's Scientific Discovery through Advanced Computing (SciDAC) program. I would like to thank my primary advisor at the laboratory, Chengkun Huang, for believing in me and pushing both of us to explore the uncharted territories of machine learning (ML) in physics. Also, this project could not have been possible without my second laboratory advisor, Chris Fryer, whose astrophysical input was instrumental to the success of this thesis.

Over the years, I met many scientists at the laboratory, interacting with whom gave me ample opportunity to grow my expertise. I especially would like to thank Jonah Miller, Arvind Mohan, Ghanshyam Pilania, and Daniel Livescu for insightful conversations directly contributing to this project. LANL also allowed me to build personal collaborations with industry partners, among whom I would like to recognize Iskandar Sitdikov, who played an integral role in developing my ML framework.

At Santa Cruz, I would like to thank my advisor Stan Woosley for taking me as a student to work on such an ambitious multidisciplinary project in astrophysics. In addition, I would like to recognize the feedback provided by my committee members, Constance Rockosi and Xavier Prochaska.

I could not have made it without the support of my friends. I would like to sin-

cerely thank Grecco Oyarzun, Zackery Briesemeister, and Mathew Siebert from Santa Cruz, and Alexander Kaltenborn, Oleksii Beznosov, Hyun Lim, Ryan Van Cleave from Los Alamos for all the banter and decompression sessions that pulled me through the tough times. A special thank you goes to Anna Llobet Megias and Tracy Forward, who welcomed me into their family and significantly contributed to my sanity during the COVID pandemic. The camaraderie built through the years in graduate school is bound to last a lifetime.

Anything I write here will not be enough to express my gratitude towards my wife, Joanna Piotrowska-Karpov. This journey could not have been done without her by my side. I am also thankful for the continuous support from my parents, Igor and Veronica Karpov.

Last but not least, I would like to acknowledge all of my foster graduate dogs from the Espanola Humane Animal Shelter. Aurora, Adele, Twix, Bernice, JoJo, Rufus, Daisy, Capitan, Diamond, Bella, Arlekino, Besos, Ded, Ryzhyk, and Anya. Thank you for getting me through the pandemic, and I hope you are all happy in your forever homes!

Chapter 1

Introduction

Some of the most energetic events in the Universe are stellar explosions known as supernovae (SNe). They can be roughly separated into two categories: thermonuclear and core-collapse SNe (CCSNe). The former result from a degenerate white dwarf stellar remnant exceeding the Chandrasekhar mass of 1.4 $M_{\odot}$, resulting in a deflagration-to-detonation supernova, commonly known as Type Ia. The latter CCSN event marks the death of a massive star. A spectacular explosive event is actually not guaranteed for a star undergoing core-collapse at the end of its life. While more details behind the CCSN mechanism will be discussed in Section 1.1, let us introduce the main stages here. Shortly after the core bounce, the shock is stalled within $\sim 200\,km$, requiring it to be re-energized for a successful explosion to avoid a further collapse into a black hole. In this process, turbulence has been speculated to play a significant role by supply-

ing additional pressure, energy, and increasing its dissipation to heat below the shock. These contributions further amplify the importance of including turbulence in the global 3D CCSN simulations. However, this is a very demanding computational task where current analytical models do not have a definitive answer for turbulence. This work investigated the applicability of Machine Learning (ML) modeling for turbulence in CCSN as an alternative to the current approaches, which can be used either on its own or in conjunction with the analytical turbulence models.

In this thesis, later sections of Chapter 1 present an overview of CCSN, characteristics of turbulence, and its modeling implementations and goes over ML methods that can be applied to benefit current global astrophysical simulations. Chapter 2 discusses the open-source pipeline called Sapsan that was built for designing, training, and evaluating ML models for turbulence modeling in astrophysical applications. Next, Chapter 3 goes over a physics-informed ML (PIML) method to train and predict 3D magnetohydrodynamic (MHD) turbulence in CCSN. Chapter 4 adopts the PIML model to capture 3D dynamics to introduce into 1D CCSN simulations. It discusses the effects of the turbulent pressure term, increasing the accretion shock radius, which in turn ameliorates explodability of the CCSN models. Lastly, Chapter 5 summarizes the ML applications in computational astrophysics, outlining its future in CCSN modeling and beyond.

1.1 Core-Collapse Supernova Mechanism

The fusion process in a massive ($> 8\ M_{\odot}$) star eventually leads to a formation of an iron core. At this point, enough energy from the fusion reactions cannot be generated to support the star's gravitational force. Along with energy losses from escaping neutrinos radiating from the core, the star undergoes a collapse. Colgate and White (1966) proposed that the inner bound shock then reaches the core, compressing it to nuclear densities ($\sim 2 \times 10^{14}\, g\, cm^{-3}$), at which point it must bounce back and explode the star. However, its fate is not as determined to result in a successful SN. As the star collapses, the shock loses energy due to photodisintegration, i.e., undoing the heavy nuclei through high-energy photon absorption. After the bounce, the shock is launched outwards but quickly loses energy due to continued neutrino losses and photodisintegration, stalling within several milliseconds at the radius of $100 - 200\, km$. The shock just became an accretion shock, losing its positive momentum, and needs to be re-energized in order to avoid a failed SN, collapsing into a black hole (Woosley and Janka, 2005; Janka, 2012; Müller, 2020).

At the core, the newly formed proto-neutron star (PNS) continues to supply a flux of neutrinos. It cools the PNS while getting quickly re-absorbed in the gain region right above, providing the negative entropy gradient to power convection in the region (Herant et al., 1994b; Fryer and Heger, 2000; Fryer and Warren, 2002). Then, convection carries the heat generated from neutrinos out to the shock, providing the

extra ram pressure to revive the shock. Many studies have been done in the neutrino-powered paradigm (Fryer et al., 2007; Janka et al., 2016; Janka, 2017; Radice et al., 2018; O'Connor et al., 2018; Müller, 2020; Burrows and Vartanyan, 2021), but neutrinos alone do not appear to provide sufficient energy to explode the star. A critical missing piece of the CCSN engine appears to be attributed to turbulence in the bubble region between the PNS and the shock (Herant et al., 1994a; Blondin et al., 2003; Fryer and Young, 2007a; Melson et al., 2015; Burrows et al., 2018). However, its source also presents the biggest challenge in studying it. Turbulence can develop from the progenitor through the collapse, standing accretion shock instability (SASI) (Blondin et al., 2003), magneto-rotational instability (MRI) in MHD (Mösta et al., 2015), or other instabilities in the bubble. In general, CCSNe are asymmetric, requiring multi-D treatment, substantially raising computational costs (Figure 1.1). There is also strong evidence of asymmetry in CCSN from observations of neutron star and black hole kicks (Hobbs et al., 2005; Faucher-Giguère and Kaspi, 2006; Ng and Romani, 2007; Repetto et al., 2012), which in turn would be responsible for turbulence. In the recent literature, pressure due to turbulence alone has been estimated to reach up to $\sim 50\%$ of the gas pressure (Couch and Ott, 2015) aiding the shock. Furthermore, turbulence increases the energy flux and its dissipation to heat below the shock, which can contribute similarly in magnitude to the additional pressure term (Mabanta et al., 2019; Couch et al., 2020).

Typically the fate of the star is decided within a few 100 *ms* after the bounce. If

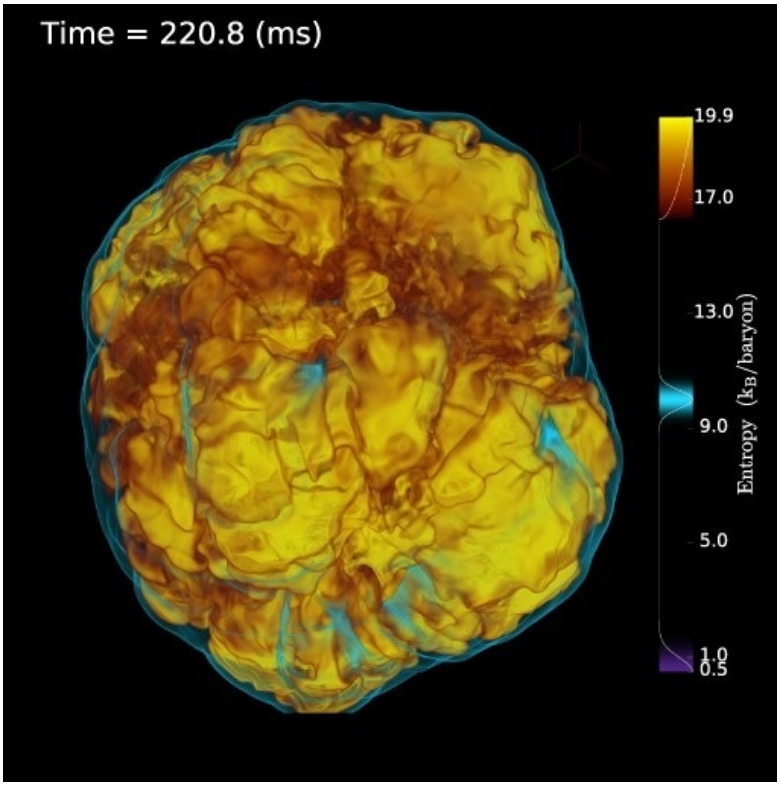


Figure 1.1: Entropy in the convective hot bubble below the accretion shock (outer cyan outline) at 220.8 ms after bounce of a 3D CCSN simulation performed with FLASH by O'Connor and Couch (2018). The snapshot presents the growing asymmetry of a CCSN.

the shock is successfully revived, it starts to move out through the star, mixing and sweeping material on its way. The shock can take hours to days to reach the outer shell. Besides the lighter elements, the explosion of $\sim 10^{51}\,erg$ will release a wide abundance of intermediate and Fe-group elements, formed through nucleosynthesis in the explosive burning (Woosley et al., 1995; Nomoto et al., 2006; Heger and Woosley, 2010). In addition, CCSNe are one of the sources of heavy elements beyond the Fe-group through the rapid neutron capture process, i.e., r-process (Woosley et al., 1994; Qian and Woosley, 1996; Nishimura et al., 2015).

1.2 Turbulence

Turbulence is a vital part not only of SNe or other astrophysical phenomena, but also of our everyday life. From blood flow to clouds to stellar formation, chaotic velocity fluctuations known as turbulence are found essential for those processes. Despite how common it is, turbulence is incredibly difficult, if not impossible, to predict exactly, given its random nature. As such, it is best to model turbulence through statistical and qualitative means, approximating its effect on larger scales.

A useful parameter to characterize a flow is Reynolds Number (Re), defined as

$$Re = \frac{uL}{\nu} \tag{1.1}$$

where u is characteristic velocity, L is characteristic length, and kinematic viscosity ν

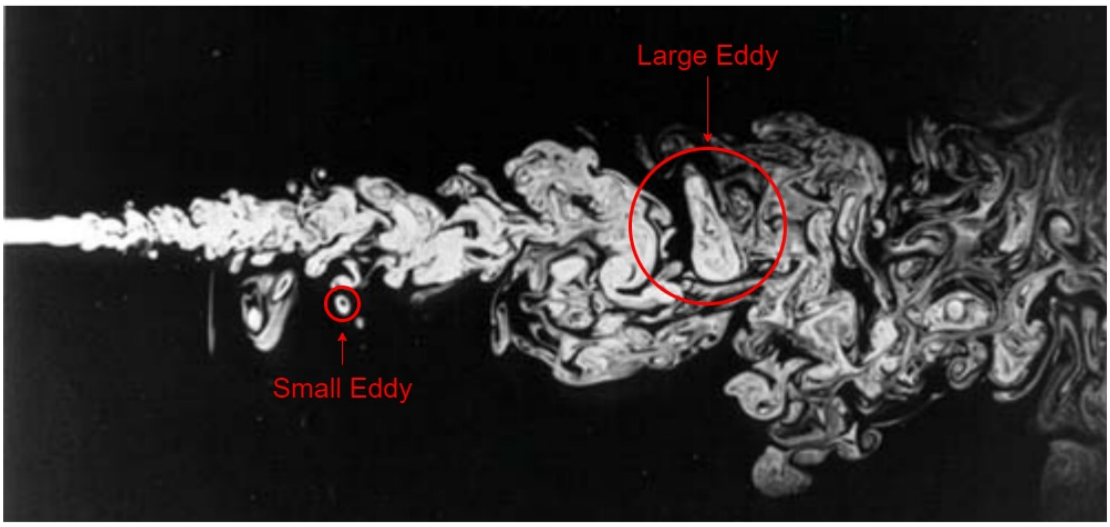


Figure 1.2: Jet fluid mixing at high *Re* presenting a turbulent flow structure; photo from Dimotakis et al. (1983).

(Pope, 2000). *Turbulent* flow is typically defined to have a high *Re* of $O(10^3)$, while low *Re* is called *laminar*. An example of turbulent mixing is shown in Figure 1.2.

In the photo, there are small and large whirls, i.e., eddies, that can be observed in the turbulent regime. Qualitatively such structure can be reasoned by the classic poem of Richardson (1922): "Big whirls have little whirls that feed on their velocity, and little whirls have lesser whirls and so on to viscosity". To better understand the turbulent behavior, it is essential to look at the energy spectrum in Figure 1.3. Energy is driven at the low-order modes, i.e., large-scale eddies, also known as the integral scale. Then the energy cascades down the spectrum following Kolmogorov's $-5/3$ slope through the inertial range (Kolmogorov, 1941). At the smallest, dissipative scale, turbulent energy transfers to heat. Ideally, the direct numerical simulations (DNS) would resolve

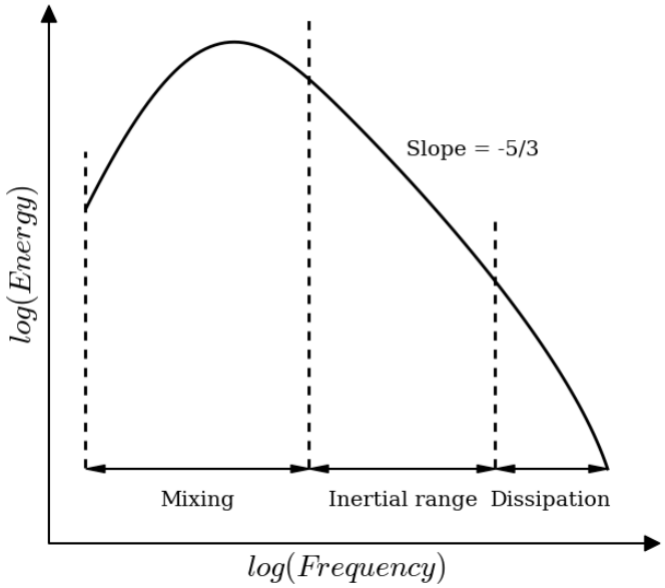


Figure 1.3: Illustration of the Kolmogorov energy spectrum. (not a final schematic; I might want to re-do a few labels)

the flow down to the smallest eddies for the turbulence to dissipate. However, this is incredibly cost prohibitive. For example, to model a hydrodynamic (HD) flow with $Re = 1000$ in 3D, the grid will have to be of $O(10^4)$ per axis (following the $N \sim Re^{\frac{3}{4}}$ requirement from Jiménez (2003)). This is already very challenging, while not getting anywhere close to $Re \sim O(10^{14})$ that can occur in SNe (Woosley et al., 2004; Müller et al., 2017).

Since global DNS is not feasible to model in most scenarios, the leading paradigms for HD simulations have been Large Eddy Simulation (LES) and Reynolds-Averaged-Navier-Stokes (RANS). In both methods, a mathematical decomposition is applied to fluid variables, where the spatial-averaged (for LES) or time-averaged (for RANS)

quantities are separated from the fluctuating ones, with their correlation modeled using the average quantities often through a hypothesis of the underlying turbulence properties. An analytical model can be used to *close* the system of NS equations, i.e., recover the subgrid-scale high-order modes that have been averaged. The common closure models used for LES in astrophysics are Dynamic Smagorinsky (Lilly, 1966) and gradient-type subgrid models (Schmidt, 2015; Miesch et al., 2015). Both of them extend the Kolmogorov energy cascade down to dissipation scales by filtering (i.e., averaging) the resolved grid and crudely extrapolating the effects of the filtering operation in a self-similar manner to the SGS structure. These models benefit from low computational overhead and straightforward implementation. The issues of such SGS models are the lack of flexibility, generalizability, and difficulty assessing the accuracy of the modeled unresolved turbulence. On a side note, it is also a common practice for global 3D simulations in astrophysics to use implicit LES (ILES) (Radice et al., 2015, 2018), which assumes numerical diffusion and viscosity to take care of turbulence, i.e., under-resolved simulation with no SGS model.

Throughout the last few decades, significant technological and algorithmic advancements have started a new era for the study of dynamically evolving turbulence. Following Moore's Law evolution of high-performance computing (HPC), non-global DNS turbulence models have emerged (Li et al., 2008; Mösta et al., 2015). These simulations model turbulence in a box, as if modeling a small portion of a global simulation.

While they do not reach the high Re found in SNe, it does get to sufficiently large magnitudes to study turbulent flow in these phenomena. It is noteworthy that previously, the role of turbulence in the CCSN engine could have been underestimated due to low *Re* given the low simulation resolution, which slowed the growth of turbulence (Mösta et al., 2015).

Today, DNS data can be used as ground truth, along with experimental data, for turbulence subgrid model optimization in large-scale 3D computational fluid dynamic (CFD) simulations. The relative abundance of DNS data presents opportunities for Machine Learning (ML) to augment turbulence modeling. The ML methods tend to be highly flexible, as demonstrated in many industrial and scientific applications (e.g., face recognition and self-driving cars). It has already shown the potential for turbulence modeling, both in the direct prediction of turbulent fluxes and analytical (e.g., RANS-based) model optimization through the means of ML (Ling et al., 2016; Wu et al., 2018; Wang et al., 2019; Wu et al., 2020; Rosofsky and Huerta, 2020; Beck and Kurz, 2021a). However, ML comes with its own range of challenges, especially when modeling physical systems. For example, turbulent regimes require physics-based constraints to the ML inference, which is still a raw field of study today. While typical ML methods can be considered universal approximators (Hornik et al., 1990), we should be careful to ensure the accuracy of ML-based turbulence models, embedding physics-based restrictions (Ling et al., 2016; Beucler et al., 2019; Portwood et al., 2019; Mohan et al., 2020;

Wu et al., 2020; Li et al., 2021; Rosofsky et al., 2023).

1.3 Machine Learning for Turbulence Closure Models

In the case of LES or RANS, after the appropriate (spatial, time, or both) averaging has been performed, the velocity field will have the following form:

$$u_i = \bar{u}_i + u_i' \tag{1.2}$$

where u is velocity, $\bar{u}$ is its average, and u' is the fluctuating velocity component. LES or RANS 3D simulations only account for the average $\bar{u}$, while u' fluctuations are left for the turbulence model to *close* the system. Upon averaging Navier-Stokes (NS) equations for incompressible (i.e., constant density ρ), hydrodynamic (HD) system, what is known as Reynolds Stress (R_{ij}) emerges:

$$\tau_{ij} = R_{ij} = \rho\overline{u_i' u_j'} \tag{1.3}$$

Modeling Reynolds Stress defines a turbulent closure model for the momentum NS equation. The simplest case is presented here to keep it concise, but physical systems can grow quite complex. For example, in compressible magneto-hydrodynamic (MHD) regimes, there are more stresses and terms that emerge through averaging, e.g., due to magnetic field influence on turbulent evolution and energy cascade.

ML methods to tackle turbulence closure differ significantly: from strictly statistical data training to embedding physical constraints to model stationary and dynamic

turbulence. In this section, a sample of ML algorithms that represent the field today will be discussed, briefly outlining the successes and struggles of each. Note that production ML models are not limited to using a single type of network, with researchers often mixing them up to reap the benefits and challenges of both, not to mention the added training difficulties.

1.3.1 Supervised Machine Learning

The most common type of ML is *supervised learning*, meaning the model is given a target feature to train against, e.g., the *Re* stress itself or the analytical closure coefficients. Such models are typically easier to design and implement, pointing at the exact feature relationships for the model to investigate.

Simplistic Regression models are the most commonly used variants of ML for turbulence modeling (e.g., Gaussian Process (Rasmussen et al., 2006), Kernel Ridge Regression (King et al., 2016)). One of the main benefits of using them is the transparency of these methods. They are relatively simple in form, easier to derive, and more interpretable. The methods have solidified their reliability for turbulent regimes that are stationary and isotropic, with numerous papers showing great performance in more idealized setups (Tracey et al., 2015; Zhang and Duraisamy, 2015; King et al., 2016). However, simplistic regression methods lack the flexibility to tackle more complex physical problems, such as dynamic 3D HD & MHD turbulent regimes.

Convolutional Neural Networks (CNN) are some of the more often used NN algorithms ranging from shallow (fewer layers) to deep (more layers) networks. In the industry, they are popular for image analysis due to their ability to capture spatial features across a wide range of scales. As a result, it is evident how CNNs can be a powerful tool to tackle LES-type turbulence modeling when spatial filtering is involved. However, those models have no inherent physics, making deep CNNs difficult to interpret. In addition, it is not trivial to generalize solutions to dynamic turbulence. (Wu et al., 2018; Zhang et al., 2018)

Recurrent Neural Networks (RNN) take into account the time-dependency of the training data, learning the patterns, hence giving an insight into dynamic evolution. A common variant of an RNN is a Long-Short Term Memory (LSTM) algorithm (Mohan et al., 2019). Similar to CNNs, deep RNN/LSTM-based models can serve as a good general-purpose estimator, tackling turbulent problems, among others. However, the lack of physical constraints preserves the black-box nature of such models, leaving generalizability to be used for a wide range of turbulent problems under question.

Generative Adversarial Networks (GAN) involves two networks that compete against each other, the generator and discriminator (Goodfellow et al., 2020). The former tries to generate a dataset similar to the target training, and the latter needs to distinguish between the fake and the ground truth. This concept forces the GAN model to learn the unknown probability distribution of the training dataset and generate the

data from it during inference. The model has shown to be particularly successful in super-resolution applications, i.e., upscaling the low-resolution images (Ledig et al., 2017; Wang et al., 2018). GAN recovers the unresolved subgrid features, effectively resembling a closure model. Moreover, considering that statistical properties of turbulence are more important than the individual spatial features, GANs have been gaining popularity among turbulence researchers (Lee et al., 2018; Bode et al., 2021). That said, GANs tend to be more challenging to train, further exacerbated by adding the physics-informed constraints. Despite the promising work in turbulence modeling, there are concerns related to the prediction of small-scale distribution and, among others, inference of dynamic turbulence with evolving distribution (King et al., 2018).

1.3.2 Unsupervised Machine Learning

In *unsupervised* ML, there is no target feature. A model is given complete flexibility to find relationships between the features in the training set. Even though this is a much more complex and unpredictable field of ML, there is potential to discover new physics and even define a universal formalism for turbulence. Some advancements for physical systems have already been made (Andreassen et al., 2019; Yao et al., 2022). While this is an exciting and promising branch of ML, convergence, training efficiency, and physical validation of those models remain challenging.

1.3.3 Physics Informed Machine Learning

Embedding physical constraints into ML models is a crucial step to ensure the physical consistency of their predictions. A few leading techniques utilized today to augment the aforementioned models are outlined next.

Non-Trainable Layers can enforce a physical condition within the network itself (Ling et al., 2016). For example, conservation of mass can be preserved through divergence of the vector potential being equal to zero ($\nabla \cdot A = 0$), in the case of incompressible turbulence (Mohan et al., 2020). Another example of preserving a realizability condition of the Re diagonal components prediction, i.e., enforcement of positive pressure prediction, will be discussed in Chapter 3. Designing such non-trainable layers for complex physical systems, i.e., compressible or MHD regimes, can be non-trivial.

Optimizing Analytical Models with ML can substantially improve their performance for a given problem while preserving the physics they were based on. There are examples of RANS, BHR, $k-\varepsilon$ optimization, and even combined RANS & LES methods (Wang et al., 2019). The caveat is that ML only learns the coefficients; hence the final SGS model relies upon the underlying physical assumptions of a given analytical model. The physics is not embedded into the ML model itself, and generalizability is not guaranteed. Thus, one has to be careful when choosing a specific analytical SGS model to optimize and apply to the turbulent regime at hand; the high flexibility of deep NNs is not utilized to its full potential.

Loss functions can be regularized to enforce physical constraints. Thus, the ML network will have a physics-based optimization at each epoch, i.e., iteration. Unlike the non-trainable layers, the enforcement is rather *soft*, rewarding the models for following certain trends or the expected physical behavior. An example would be using a loss derived from statistics rather than a point-to-point loss, e.g., L1 or L2 loss. Statistics-based regularization is especially relevant for turbulence, where individual spatial features of smaller eddies might not be as important as the overall statistical distribution, which applies to both statistically stationary and dynamic turbulence regimes. An example of utilizing a combination of point-to-point and statistical losses will be given in Chapter 3. The challenges arise due to the difficulty of tuning such regularizers (Beucler et al., 2019; Wu et al., 2020).

Ordinary and Partial Differential Equations can represent turbulent dynamics. ML is used to predict the coefficients, in turn calculating the evolution of the turbulent behavior. Thus far, there have been promising results in predicting decaying turbulence, while its growth remains challenging through neural ordinary differential equations (NODE) by Portwood et al. (2019). Examples of using partial differential equations include the physics-informed neural operators (PINO) method for modeling 1D and 2D physical systems (Li et al., 2021; Rosofsky et al., 2023).

1.3.4 Machine Learning in Astrophysics

In astrophysics, ML is primarily used in observational data processing, e.g., object detection, classification, and noise reduction (Kremer et al., 2017; Baron, 2019; Vojtekova et al., 2020). The methods are complimented by the popularity of image-processing research and development in the industry. However, modeling physical systems less reliant on spatial features, e.g., turbulence, is a niche application requiring tailored and often physics-informed solutions. Currently, there is an absence of ML-based methods in large astrophysical code-bases, e.g., SN and galaxy simulations. Adopting, constructing, and testing more intricate ML models is daunting, especially with the lack of tools available for computational astrophysics.

All the approaches for turbulence discussed above have their benefits, and more are being developed. With such an abundance of ML architectures to explore, it is necessary to simplify the design procedure and implementation of ML in astrophysical codes. The Sapsan framework was developed to aid these challenges, along with a PyTorch wrapper from COLLAPSO1D available for new and legacy codes, e.g., FLASH, FORNAX. These tools will allow for rapid iteration of the ML design process and will allow us to better evaluate the applicability and benefit of ML to the community.

Chapter 2

Sapsan Framework

Adapted from

Sapsan: Framework for Supernovae Turbulence Modeling with Machine Learning

P. I. Karpov, I. Sitdikov, C. Huang, C. L. Fryer

Journal of Open Source Software, Volume 6(67), pp.3199 (2021)

DOI: 10.21105/joss.03199, ©Authors

Summary

Sapsan is a framework designed to make Machine Learning (ML) more accessible in the study of turbulence, with a focus on astrophysical applications. Sapsan includes modules to load, filter, subsample, batch, and split the data from hydrodynamic (HD) simulations for training and validation. Next, the framework includes built-in conventional and physically-motivated estimators that have been used for turbulence modeling. This ties into Sapsan's custom estimator module, aimed at designing a custom ML model layer-by-layer, which is the core benefit of using the framework. To share your custom model, every new project created via Sapsan comes with pre-filled, ready-for-release Docker files. Furthermore, training and evaluation modules come with Sapsan as well. The latter, among other features, includes the construction of power spectra and comparison to established analytical turbulence closure models, such as a gradient model. Thus, Sapsan attempts to minimize the hard work required for data preparation and analysis, leaving one to focus on the ML model design itself.

2.1 Statement of Need

Domain sciences have been slow to adopt Machine Learning (ML) for a range of projects, but particularly for physical simulations modeling turbulence. It is challenging to prove that an ML model has learned the laws of physics in a particular problem, and

that it has the ability to extrapolate within the parameter-space of the simulation. The inability to directly infer the predictive capabilities of ML is one of the major causes behind the slow adoption rates; however, the community cannot ignore the effectiveness of ML.

Turbulence is ubiquitous in astrophysical environments, however, it involves physics at a vast range of temporal and spatial scales, making accurate fully-resolved modeling difficult. Various analytical turbulence models have been developed to be used in simulations using temporal or spatial averaged governing equations, such as RANS (Reynolds-averaged Navier-Stokes) and LES (Large Eddy Simulation), but the accuracy of these methods is sometimes inadequate. In search of better methods to model turbulence in core-collapse supernovae, it became apparent that ML has the potential to produce more accurate turbulence models on an un-averaged subgrid-scale than the current methods. Scientists from both industry and academia (King et al., 2016; Zhang et al., 2018) have already begun using ML for applied turbulent problems. Still, none of these efforts have yet reached the scales relevant for the physics and astronomy community on a practical level. For example, physics-based model evaluation and interpretability tools are not standardized, nor are they widely available. As a result, it is a common struggle to verify published results, with the setup not fully documented, the opaquely structured code lacking clear commenting, or even worse, not publicly available. This is a problem that the broader ML community can relate to as well (Hutson,

2018). Thus, it is not surprising that there is considerable skepticism against ML in physical sciences, with astrophysics being no exception (Carleo et al., 2019).

In pursuit of our supernova (SNe) study, the issues outlined above became painfully apparent. Thus, we have attempted to lower the barrier to entry for new researchers in domain science fields studying turbulence to employ ML, with the main focus on astrophysical applications. As a result, we developed an ML Python-based pipeline called Sapsan. The goals have been to make this library accessible and shared with the community through Jupyter Notebooks, a command-line-interface (CLI) and a graphical-user-interface (GUI)[1] available for end-users. Sapsan includes built-in optimized ML models for turbulence treatment, both conventional and physics-based. More importantly, at its core, the framework is meant to be flexible and modular; hence there is an intuitive interface for users to work on their own ML algorithms. Most of the mundane turbulence ML researcher needs, such as data preprocessing and prediction analysis, can be automated through Sapsan, with a streamlined process of custom estimator development. In addition, Sapsan brings best practices from the industry regarding ML development frameworks. For example, Sapsan includes docker containers for reproducible release, as well as MLflow for experiment tracking. Thus, Sapsan is a single, complete interface for ML-based turbulence research.

Sapsan is distributed through GitHub and pip. For further reference, visit Sapsan's wiki.

[1] A demo is available at sapsan.app

2.2 Framework

Sapsan organizes workflow via three respective stages: data preparation, machine learning, and analysis, as shown in Fig.2.1. The whole process can be further distributed using Docker for reproducibility. Let's break down each stage in the context of turbulence subgrid modeling, e.g., a model to predict turbulent behavior at the under-resolved simulation scales.

- **Data Preparation**
  - **Loading Data:** Sapsan is ready to process common 2D & 3D hydrodynamic (HD) and magnetohydrodynamic (MHD) turbulence data in simulation-code-specific data formats, such as HDF5 (with more to come per community need).
  - **Transformations:** A variety of tools are available for the user to prepare data for training:
    - **Filtering:** To build a subgrid model, one will have to filter the data to, for example, remove small-scale perturbations. Some possible choices include a box, spectral, or Gaussian filter. The data can be filtered on the fly within the framework.
    - **Sampling:** to run quick tests of your model, you might want to test on a sampled version of the data while retaining the full spatial domain.

For this application, equidistant sampling is available in Sapsan.

    * **Batching & Splitting:** The data are spatially batched and divided into testing and validation subsets.

- **Machine Learning**

  - **Model Setup:** Different ML models may be appropriate for different physical regimes, and Sapsan provides templates for a selection of both conventional and physics-based models with more to come. Only the most important options are left up to the user to edit, with most overhead kept in the backend. This stage also includes tools for defining ML layers, tracking parameters, and choosing and tuning optimization algorithms.

- **Analysis**

  - **Trained Model:** A turbulence subgrid model defines how small-scale structure affects the large scale quantities. In other words, it completes or "closes" the governing large-scale equations of motion with small-scale terms. The prediction from a trained ML model is used to provide the needed quantities.

  - **Analytical Tools:** There are also methods included for comparing the trained model with conventional analytic turbulence models [such as the Dynamic Smagorisnky (Lilly, 1966); or Gradient, (Liu et al., 1994b); models], or to

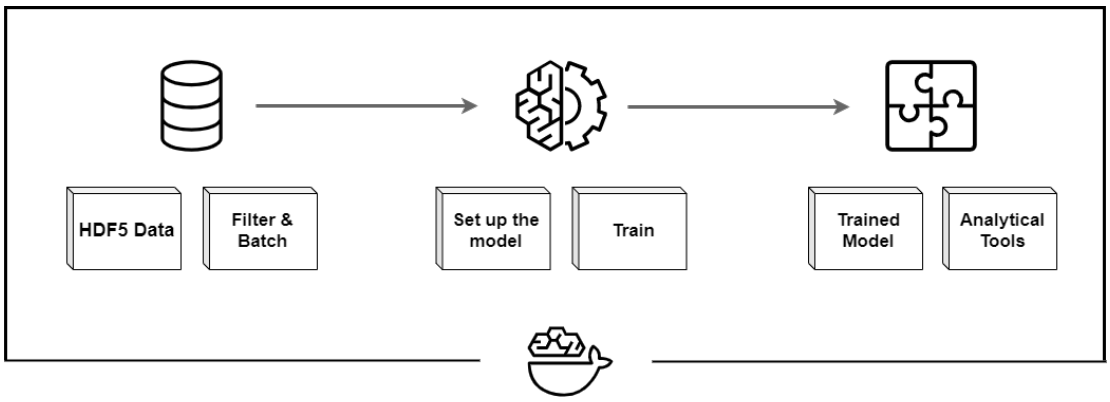


Figure 2.1: High-level overview of Sapsan's workflow.

conduct other tests of, for example, the power spectrum of the model prediction.

For further information on each stage, please refer to Sapsan's Wiki on Gihub.

Dependencies

The following is a list of the core functional dependencies[2] and a short description of how they are used within Sapsan:

- **PyTorch:** Sapsan, at large, relies on PyTorch to configure and train ML models. Thus, the parameters in the aforementioned **Model Setup** stage should be configured with PyTorch functions. Convolutional Neural Network (CNN) and Physics-Informed Convolutional Auto Encoder (PICAE) examples included with Sapsan are based on PyTorch. (Paszke et al., 2019)

[2]Please refer to GitHub for the complete list of dependencies.

- **Scikit-learn:** A alternative to PyTorch, as demonstrated in the Kernel Ridge Regression (KRR) example in Sapsan. Since scikit-learn is less flexible and scalable than PyTorch, the latter is the recommended interface. (Pedregosa et al., 2011)

- **Catalyst:** used as part of the backend to configure early-stopping of the model and logging. (Kolesnikov, 2018)

- **MLflow:** provides an intuitive web interface for tracking the results of large experiments and parameter studies. Beyond a few default parameters, a user can include custom parameters to be tracked. (Databricks, 2020)

- **Jupyter Notebook:** the most direct and versatile way to use Sapsan.

- **Streamlit (GUI):** a graphical user interface (GUI) for Sapsan. While not as flexible as the other interfaces, this can be useful for developing public-facing demonstrations. An example of this interface can be found online at sapsan.app. (Treuille, 2019)

- **Click (CLI):** a command-line interface (CLI) for Sapsan. It is used to get the user up and running with templates for a custom project. (Ronacher, 2021)

2.3 Applications

While Sapsan is designed to be highly customizable for a wide variety of projects in the physical sciences, it is optimized for the study of turbulence. In this section we will demostrate various capabilities of Sapsan working with 2D and 3D data, various machine learning libraries, and built-in analytical tools. The ML methods used are included in Sapsan's distribution as example Jupyter notebooks to get started with the framework.

2.3.1 Hydro simulations

Here is an examples of a turbulence closure model trained on the high-resolution Johns Hopkins Turbulence Database [JHTDB, Li et al. (2008)]. The training data is a 2D slice of a direct numerical simulation (DNS) of a statistically-stationary isotropic 3D MHD turbulence dataset, 1024^3 in spatial resolution and covering roughly one large eddy turnover time over 1024 checkpoints, i.e. the dynamical time of the system (Eyink et al., 2013). We compare it with a commonly used Dynamic Smagorinsky (DS) turbulence closure model (Lilly, 1966). On the Sapsan side, a Kernel Ridge Regression model (Murphy, 2012) by the means of scikit-learn is used to demonstrate the effectiveness of conventional ML approaches in tackling turbulence problems. In this test, we used the following setup:

- **Train features:** velocity (u), vector potential (A), magnetic field (B), and their

respective derivatives a checkpoint = 0. All quantities have been filtered down to 15 Fourier modes to remove small-scale perturbations, mimicking the lower fidelity of a non-DNS simulation. Next they were sampled down to 128^3, with the last step leaving a single slice of 128^2 ready for training.

- **Model Input:** low fidelity velocity (u), vector potential (A), magnetic field (B), and their respective derivatives at a set checkpoint = 10.

- **Model Output:** velocity stress tensor component (τ_{xy}) at the matching checkpoint in the future, which effectively represents the difference between large and small scale structures of the system.

In Fig.2.2, it can be seen that the ML-based approach significantly outperforms the DS subgrid model in reproducing the probability density function, i.e., a statistical distribution of the stress tensor. The results are consistent with (King et al., 2016).

2.3.2 Supernovae

Even though the conventional ML regression approach worked well in the 2D setup from the previous example, the complexity of our physical problem forced us to seek out a more sophisticated ML method. Supernovae host a different physical regime that is far from the idealistic MHD turbulence case from before. Here we are dealing with dynamically evolving turbulence that is not necessarily isotropic. Turbulence can

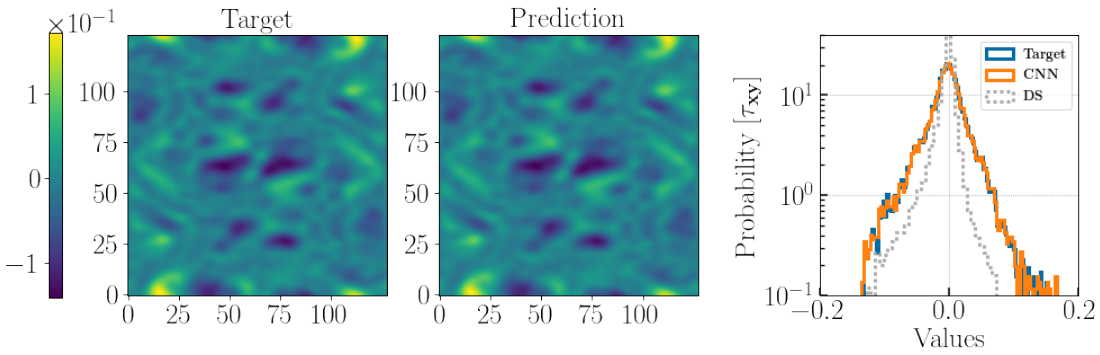


Figure 2.2: Predicting a 2D turbulent stress-tensor component (τ_{xy}) in statistically-stationary isotropic MHD turbulence setup. The left plot compares the original spatial map of the stress-tensor component to the predicted spatial map (middle). The plot on the right presents probability density functions (PDF), i.e., distributions, of the original stress-tensor component values, the ML predicted values, and the conventional Dynamic Smagorinsky (DS) subgrid model prediction.

behave drastically differently depending on the evolutionary stage. With Sapsan, we have tested a 3D CNN (Convolutional Neural Network) model built with PyTorch to predict a turbulent velocity stress tensor in a realistic Core-Collapse Supernova (CCSN) case. Fig.2.3 presents results of the following:

- **Train features:** velocity (u), magnetic field (B), and their respective derivatives at time steps before 5 ms (halfway of the total simulation). All quantities have been filtered down with a $\sigma = 9$ Gaussian filter to remove small-scale perturbations, mimicking the lower fidelity of a non-DNS simulation. Lastly they were sampled from the original 348^3 down to 116^3 in resolution.

- **Model Input:** low fidelity velocity (u), magnetic field (B), and their respective derivatives at a set time step in the future beyond 5 ms.

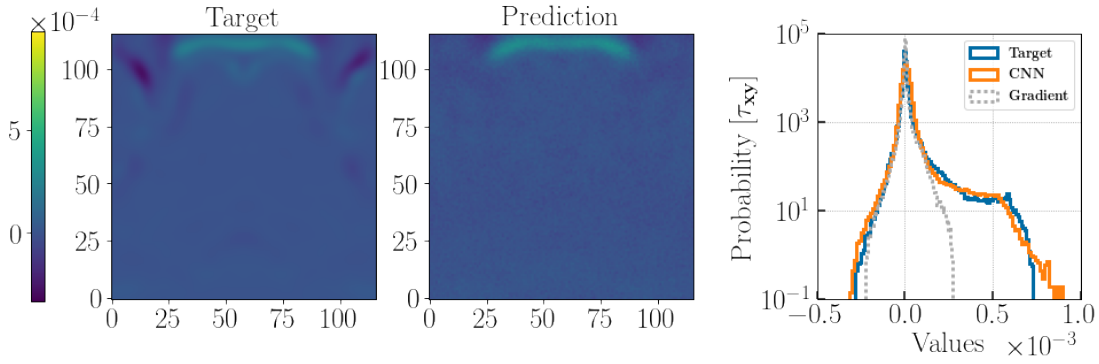


Figure 2.3: Predicting turbulent stress-tensor component in a core-collapse supernovae (CCSN). The model has been trained on a selection of dynamically evolving turbulence timesteps during the first 5 ms (out of the total $\sim$ 10 ms) of a 3D MHD direct numerical simulation (DNS) after the shockwave bounced off the core in a CCSN scenario. On the left, the two figures are the 2D slices of a 3D τ_{xy} prediction, with the right plot comparing PDFs of the original 3D data, 3D ML prediction, and a conventional Gradient subgrid model.

- **Model Output:** velocity stress tensor components (τ_{ij}) at the matching time step in the future, which effectively represents the difference between large and small scale structures of the system.

In this case, the probability density functions are overall consistent, with minor disagreement at the positive outliers, even though the prediction is performed far into the future (time = 9.48 ms, end of the simulation time). Predictive advantage is highlighted when compared with the analytical Gradient model that misses a large portion of positive data. The quantitative implications of the differences in predicting quality for turbulent pressure between these methods will be further discussed in Chapter 3.

Acknowledgements

The development of Sapsan was supported by the Laboratory Directed Research and Development program and the Center for Space and Earth Science at Los Alamos National Laboratory through the student fellow grant. We would like to thank DOE SciDAC for additional funding support.

Chapter 3

Modeling 3D MHD Turbulence

Adapted from

Physics-Informed Machine Learning for Modeling Turbulence in Supernovae

P. I. Karpov, C. Huang, I. Sitdikov, C. L. Fryer, S. Woosley

The Astrophysical Journal, num (2022)

Abstract

Turbulence plays an important role in astrophysical phenomena, including core-collapse supernovae (CCSN), but current simulations must rely on subgrid models since direct numerical simulation (DNS) 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 (CNN) to preserve the realizability condition of Reynolds stress that is necessary for accurate turbulent pressure prediction. The applicability of the ML subgrid model is tested here for magnetohydrodynamic (MHD) 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.

3.1 Introduction

Turbulence plays a key role in many astrophysical phenomena (Schekochihin and Cowley, 2007; Brandenburg and 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 8 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 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 free fall. 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 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 it 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, now taking 100's of milliseconds instead of milliseconds. At this stage, the core is a hot proto-neutron 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 neutrinos in depositing some fraction of their energy outside the proto-neutron star (outside the neutrinosphere and inside the accretion shock), and the distribution of pressure that energy deposition creates. A failed explosion will lead to a black hole and no supernova (Woosley and

Janka, 2005; Burrows and Vartanyan, 2021).

Over the past three decades, the community has focused on the importance of turbulence and convection in improving the efficiency with which energy released in the collapse of the core is converted into explosion energy (Herant et al., 1994a; Blondin et al., 2003; Fryer and Young, 2007a; Melson et al., 2015; Burrows et al., 2018). Most of these studies 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 outwards, ultimately achieving positive velocity and exploding the star. A recent study attributes up to $\sim 30\%$ of the gas pressure to turbulence to aid the CCSN explosion (Nagakura et al., 2019). Turbulence in this region has three origins: the primordial turbulence present because the star was convective in these zones before it collapsed; the turbulence 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 "magneto-rotational instability", MRI). Multi-dimensional 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 and White, 1966). A major difficulty is a physically correct description of the turbulence and its effective pressure in a multi-dimensional code that is unable to resolve the relevant length scales.

Here we focus on magnetically generated turbulence. This introduces additional

variables and uncertainties not contained in the non-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 3.3.4). 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 CCSN (Fryer and Young, 2007b). For DNS, the per-axis 3D resolution scales as $N \sim 2Re^{3/2}$ (Jiménez, 2003), leading to 3D DNS of CCSN requiring a grid size of 10^5 in each direction, which is extremely expensive (if possible) to achieve with today's HPC resources. Together with a vast spatial scale range needed to be resolved in CCSN (200 km inner convective region and out to 10^9 km outer shell) and the complexity of physical processes ongoing in CCSN, 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 1-dimensional simulations):

- **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 supernovae progenitors

(Arnett et al., 2015) and requires a turbulent closure model. An example of a popular technique based on RANS is Mixing Length Theory (MLT). It is used to model 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 applicable to stellar convection (including supernovae simulations (Couch et al., 2020)) in 1D simulations. MLT performs well for small mixing length scales, while turbulence in CCSN evolves over a wide range of scales, which is deemed problematic for MLT's accuracy (Joshi et al., 2019).

- **Large Eddy Simulation (LES)** - space averaged turbulence treatment. Similar to RANS, a subgrid model substitutes the turbulence effects absent from small spatial scales. For *closure*, it is common to use Dynamic Smagorinsky (Lilly, 1966) or gradient-type subgrid models (Schmidt, 2015; Miesch et al., 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 CCSN 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

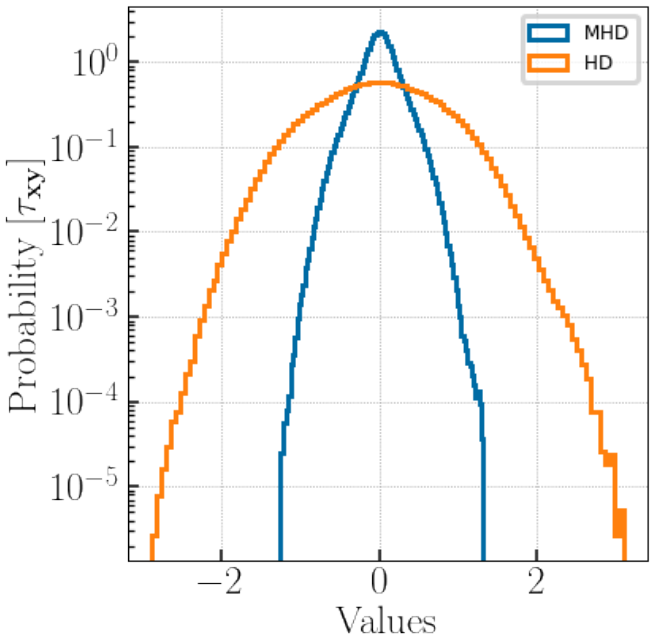


Figure 3.1: An example of the difference magnetic field can bring to the turbulence stress distribution. The data has been taken from JHTDB MHD and HD datasets (Li et al., 2008).

process (Beresnyak and Lazarian, 2014). The non-linear behavior of MHD further exacerbates simulation challenges leaving many open questions on the nature of turbulence despite decades of focused studies (Beresnyak, 2019). As an example of changes due to an introduction of a magnetic field within a simulation, we present a comparison of normalized *Re* stress tensor component distribution between HD and MHD simulations given comparable initial conditions from Johns Hopkins Turbulence Database (JHTDB) in Figure 3.1.

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. Throughout 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*. For our context, by *Big Data* we mean the abundance of turbulence DNS data that resulted from the Moore's Law evolution of high-performance computing (HPC). While *Re* of those simulations is not to the natural level of CCSN, 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 ground-truth, along with experimental data, for turbulence subgrid model optimization in large-scale 3D computational fluid dynamic (CFD) simulations. *Big Data* presents the opportunities for ML to help augment and improve turbulence modeling. As shown by the industry (i.e., face-recognition, self-driving cars), ML is highly flexible, and it has already shown its potential for turbulence modeling, both direct prediction of turbulent fluxes and analytical (e.g., RANS-based) model optimization (King et al., 2016; Wu et al., 2018; Zhang et al., 2018; Rosofsky and Huerta, 2020).

ML has been applied to quasi-stationary 2D ideal MHD turbulence in an astrophysical setting with promising results as compared to the conventional analytic gradient subgrid turbulence model (Rosofsky and Huerta, 2020). In this paper, we developed an ML model for 3D MHD turbulence for astrophysical simulations in reduced dimension, as well as performed a time-dependent prediction analysis. Considering the popular-

ity and success of Convolutional Neural Networks (CNNs) in the industry (Krizhevsky et al., 2012; He et al., 2016) for spatial pattern recognition, we sought to relate large-scale eddie structure to small-scale turbulence distribution. Thus, we based our approach on CNNs to develop a physics-informed machine learning (PIML) turbulence closure model, described in Section 3.3.3. It was applied for two regimes to test generalizability: statistically-stationary homogeneous isotropic MHD turbulence from a general-purpose dataset and dynamic MHD turbulence from a CCSN simulation. The former can be found in the ISM, studying the stellar formation, and the latter is directly applicable to high-energy events, such as CCSN. The model predicts Reynolds stress tensor (τ), with turbulent pressure (P_{turb}) defined as:

$$P_{turb} = tr(\tau) \tag{3.1}$$

where $tr(x)$ is a trace of x. Note that we neglect the $1/3$ coefficient in our definition of P_{turb}, which has no effect on the ML prediction results. We will primarily focus on analyzing statistical distribution, i.e., probability density function (PDFs), of time-dependent turbulence. In the case of stationary turbulence, we will be testing the stability of the ML model, ensuring prediction would remain 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., DNS data we assume to accurately represent the physical state of the system.

In Section 4.2, we will cover filtering, decomposition, and the result MHD formal-

ism. Section 3.3 introduces the analytical subgrid turbulence model we will be using for comparison, our ML pipeline, specifics of the ML & PIML models used to treat various τ_{ij} components, and the datasets used for training and testing them. In Section 4.4, we provide the analysis of stationary and dynamic results, with conclusion following in Section 4.6. Lastly, the Appendices include further details on the ML model developed and its training process.

3.2 Formalism

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

3.2.1 Filtering

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

$$u = \bar{u} + u' \tag{3.2}$$

where $\bar{u}$ (LES-simulated quantity) is the ensemble average of u (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:

$$\bar{u} = G * u \tag{3.3}$$

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 a 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:

$$G(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{x^2}{2\sigma^2}} \tag{3.4}$$

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.

3.2.2 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):

$$\partial_t \rho + \nabla \cdot (\rho \mathbf{u}) = 0 \tag{3.5}$$

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

$$\partial_t \mathbf{B} - \nabla \times (\mathbf{u} \times \mathbf{B}) = 0$$

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

3.2.3 MHD equations - Filtered

For the filtered LES equations, we need to apply Eq. 3.4 to Eqs. 3.5. As a result we get:

$$\partial_t \bar{\rho} + \nabla \cdot (\bar{\rho} \tilde{\mathbf{u}}) = 0 \tag{3.6}$$

$$\partial_t (\bar{\rho} \tilde{\mathbf{u}}) + \nabla \cdot (\bar{\rho} \tilde{\mathbf{u}} \otimes \tilde{\mathbf{u}} - \bar{\mathbf{B}} \otimes \bar{\mathbf{B}}) + \nabla \left(\bar{P} + \frac{\bar{B}^2}{2} \right) = -\nabla \cdot \tau^{mom}$$

$$\partial_t \bar{\mathbf{B}} - \nabla \times (\tilde{\mathbf{u}} \times \bar{\mathbf{B}}) = \nabla \times \mathcal{E}$$

where $\tilde{\mathbf{u}} = \overline{\rho \mathbf{u}} / \bar{\rho}$, 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:

$$\tau_{ij}^{mom} = \bar{\rho} \tau_{ij}^{kin} - \tau_{ij}^{mag} + \left(\overline{B^2} - \bar{B}^2 \right) \frac{\delta_{ij}}{2} \tag{3.7}$$

$$\mathcal{E} = \overline{\mathbf{u} \times \mathbf{B}} - \tilde{\mathbf{u}} \times \bar{\mathbf{B}} \tag{3.8}$$

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:

$$\tau_{ij}^{kin} = \widetilde{u_i u_j} - \tilde{u}_i \tilde{u}_j \tag{3.9}$$

$$\tau_{ij}^{mag} = \overline{B_i B_j} - \bar{B}_i \bar{B}_j \tag{3.10}$$

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

3.3 Subgrid Modeling

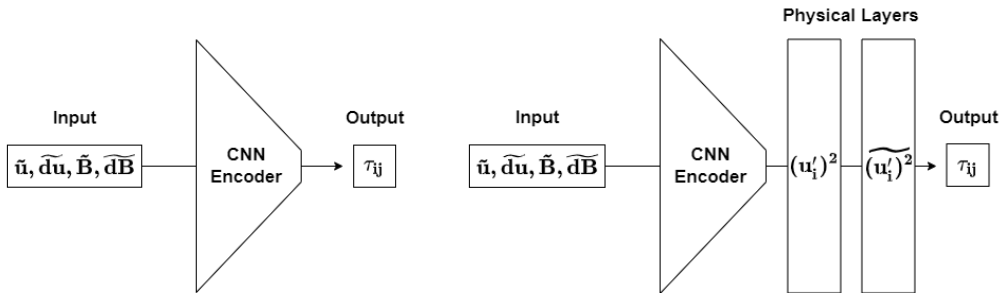
We will be comparing our ML results with a conventional turbulence subgrid model used widely in astronomy - the gradient model (Liu et al., 1994a).

3.3.1 Gradient model

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

$$\tau_{ij} = \frac{\tilde{\Delta}^2}{12} \partial_k \tilde{u}_i \partial_k \tilde{u}_j \tag{3.11}$$

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



(a) Off-diagonal terms (3D CNN)

(b) Diagonal terms (PIML)

Figure 3.2: Model schematics to calculate the Reynolds stress (τ_{ij}) components.

3.3.2 Machine Learning Pipeline

In its essence, machine learning (ML) is a sophisticated fitting routine of a multi-dimensional dataset against a target feature. However, unlike it, 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 model tuning based on a validation dataset (Bishop, 2006; LeCun et al., 2015). This opens up a possibility to learn new links between the input variable, 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 improving themselves as new data becomes available. That 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 convolutional neural networks. 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 (Karpov et al., 2021). Here is a high-level procedure overview:

1. **Data**: choose a relevant high-fidelity dataset. In our case, the data comes from the DNS simulations that we consider to be 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 under-resolved. Hence filtering applied to high-resolution DNS simulation data need to account for that adequately.

3. **Data Splitting**: split the data into training, validation, and testing portions.

4. **Optimization and Training**: optimizing 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 to be representative of the relevant physics, as well as an estimate of 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.

3.3.3 Machine Learning Models

We used two ML models to calculate all 9 components of the Reynolds stress τ_{ij}: a conventional CNN encoder for off-diagonal terms and a custom, physics-informed CNN encoder for diagonal terms. Schematics of the models are shown in Fig.3.2a and Fig.3.2b with discussion of each in Sections 3.3.3 & 3.3.3 respectively. Both models have been trained on a dual-GPU system, equipped with NVIDIA Quadro RTX 5000 cards.

Off-Diagonal Terms (3D CNN)

The idea behind a neural network is illustrated in Fig. 3.2a as a pipeline schematic. We first need to put in 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., *CNN Encoder*. At the end, there is an output layer predicting our target quantity.

We based our model for off-diagonal tensor components on the 3D Convolutional

Neural Network (3D CNN) with some modifications while keeping it conventional. In a CNN, a *convolution* layer is applied as a hidden layer. In that case, a given kernel is used to parse through the dataset to identify the spatial patterns needed for the given problem. The kernel has the form:

$$out(N_i, C_{out}) = bias(C_{out} + \sum_{k=0}^{C_{in}-1} weight(C_{out,k}) input(N_i, k)) \tag{3.12}$$

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

$$[N, C_{in}, D, H, W] \tag{3.13}$$

and the output is

$$[N, C_{out}, D_{out}, H_{out}, W_{out}] \tag{3.14}$$

where [D, H, W] are the [depth, height, width], i.e., [x,y,z]. The notation is in agreement with PyTorch documentation[1]. The reason behind choosing a CNN as our core ML algorithm was the goal to relate spatial structure of the turbulent eddies to the small scale structure.

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

- **Model**: a classical approach for CNN network architectures, where convolution and pooling layers are stacked up consequently followed by fully connected dense layers as it is shown in Fig. 3.11, based around 3D CNN[1].

[1] pytorch.org/docs/stable/generated/torch.nn.Conv3d.html

- **Optimizer**: Adam Optimizer[2] - extension of the stochastic gradient descent; it was picked due to the good performance on sparse gradients.

- **Activation function**: LogSigmoig[3] - a non-linear activation function to select values to pass from layer to layer. The function is defined by $log(\frac{1}{1+exp(-x)})$.

- **Loss function**: Custom SmoothL1Loss[4] - 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 (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 behind the choices of Optimizer, Activation Function, and the Loss Function were the broad applicability, availability, and success of these techniques. In addition, we performed cross-validation over available PyTorch functions to solidify our choices. These parameters were sufficient for off-diagonal terms of τ_{ij} of 3D MHD turbulence. However, physical conditions had to be enforced in order to model diagonal terms and ultimately predict P_{turb}.

[2] pytorch.org/docs/stable/generated/torch.optim.Adam.html
[3] pytorch.org/docs/stable/generated/torch.nn.LogSigmoid.html
[4] pytorch.org/docs/stable/generated/torch.nn.SmoothL1Loss.html

Diagonal Terms (PIML)

If calculated directly, Reynolds stress is defined by:

$$\tau_{ij} = \widetilde{u_i' u_j'} = \widetilde{u_i u_j} - \tilde{u}_i \tilde{u}_j \tag{3.15}$$

where u_i' is a velocity fluctuation component, and $\tilde{x}$ is the spatial average. Thus, for diagonal terms a realizibility condition is defined as $\tau_{ii} > 0$ (Schumann, 1977), making distribution of diagonal tensor components asymmetric. While the model in Section 3.3.3 excelled at quasi-symmetric distribution prediction, it struggled with asymmetric distributions. Further analysis of this will be covered in Section 3.4.2.

- **Model**: 3D CNN encoder as described in Section 3.3.3 with physics-informed (PI) layers. The encoder implicitly predicts velocity fluctuations (u_i'), then PI layers calculate $u_i'^2$ to enforce $\tau_{ii} > 0$ and filter to find the mean as per Eq. 3.15.

- **Optimizer**: Adam Optimizer - same as in Section 3.3.3.

- **Activation function**: LogSigmoid (stationary) and Tanhshrink[5] (dynamic) - the latter showed a faster model convergence rate for the dynamic turbulence. The function is defined by

$$f(x) = x - tanh(x)$$

- **Loss function**: Custom - a dynamic combination of SmoothL1Loss (point-to-point) and Kolmogorov-Smirnov (KS) Statistic (Massey, 1951) losses.

[5] pytorch.org/docs/stable/generated/torch.nn.Tanhshrink.html

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 Results Section 3.4.2.

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[6].

3.3.4 Datasets

In machine learning, 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 datasets. A quick parameter overview of both datasets can be found in Table 3.1.

- **Stationary**: high resolution statistically stationary, isotropic, forced, incompressible MHD turbulence dataset[7] from Johns Hopkins Turbulence Database (JHTDB) (Li et al., 2008). It has a low Reynolds number fluctuating around $Re \sim 186$.

- **Dynamic**: evolving highly-magnetized CCSN turbulence dataset by Mösta et al. (2015). It is a 3D DNS of an MHD CCSN. The dataset has been developed to study the effects of magneto-rotational instability (MRI) in growing turbulence in

[6] sapsan-wiki.github.io/details/estimators/#physics-informed-cnn-for-turbulence-modeling-pimlturb
[7] turbulence.pha.jhu.edu/Forced_MHD_turbulence.aspx

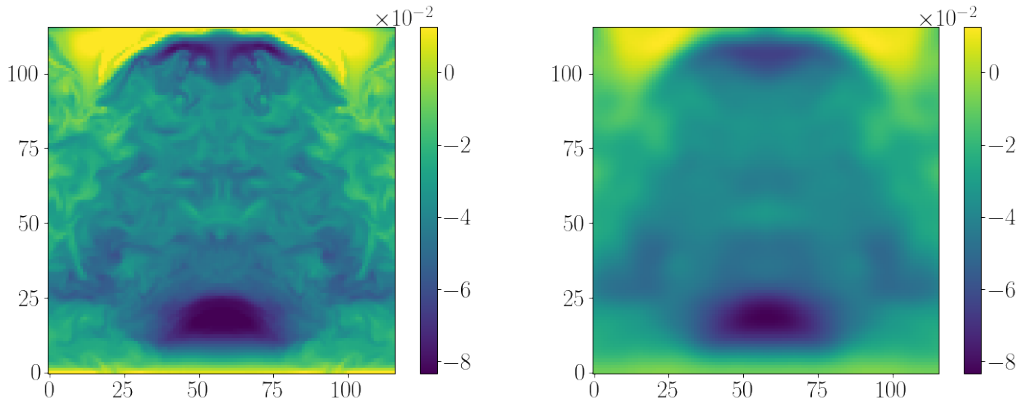


Figure 3.3: Box filter with a Gaussian kernel ($\sigma = 9$) was used to filter the data. Above is a slice of sampled u_x, down to (116,116,116); filter was applied prior to sampling. **Left:** Original, **Right:** Filtered

.

a CCSN scenario to aid the explosion, aiming to prove the plausibility of CCSN to be 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 $\sim 10\,ms$ 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 dataset with $\Delta x = 200\,m$ resolution that exhibits mild turbulence.

	Stationary	Dynamic
Resolution	1024^3	347^3
t_{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_{max}	482	348
$\overline{KE/E_{tot}}$	~ 0.5	~ 0.9
$\overline{E_B/E_{tot}}$	~ 0.5	~ 0.1
σ	16	9

Table 3.1: Parameters of the statistically *stationary* (JHTDB) and *dynamic* CCSN (Mösta et al., 2015) turbulence datasets. The time values for *stationary* dataset is in normalized numerical units amounting to 1 large-eddy turnover time. t_{tot} is the total simulation time, δt is a timestep, Δt is checkpoint time separation, k_{max} is the maximum Fourier mode, $\overline{KE/E_{tot}}$ and $\overline{E_B/E_{tot}}$ are the time-averaged fractions of kinetic and magnetic energy with respect to the total energy. σ is the Gaussian filter standard deviation we applied during data preparation. Lastly, the spatial resolution of the *Dynamic* dataset is $\Delta x = 200\,m$.

Data Preparation

In order to reflect a realistic low-fidelity environment of the CCSN simulations, we chose to apply Gaussian filter (Carati et al., 2001) as described in Section 3.2.1. Standard deviation σ of the filter (Eq.3.4) for each dataset was chosen to provide similar levels of filtering for both *stationary* and *dynamic* datasets, with exact σ specified in Table 3.1. An example of the filtering used is shown in Fig. 3.3. For both datasets, derivatives were calculated, and the filter was applied at the highest resolution available. Then, the data was equidistantly sampled down to 116^3 to fit the hardware memory constraints. The exact data preparation procedures are summarized:

1. Calculate derivatives of $[u, B]$

2. Calculate Reynolds stress tensor τ using Eq.4.7

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*

3.4 Results & Discussion

3.4.1 Stationary Turbulence

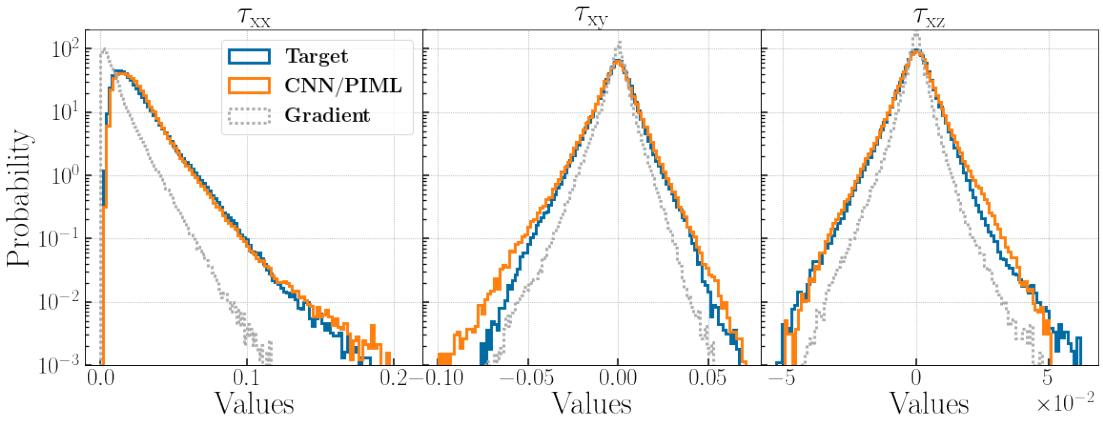


Figure 3.4: Prediction of x components of τ at $t = 1000$ of JHTDB MHD dataset in normalized numerical units. *Blue* is the original target data, *Orange* is prediction of the CNN or PIML models for off-diagonal and diagonal terms respectively, and *Gray* is the result of the gradient subgrid model as per Section 3.3.1.

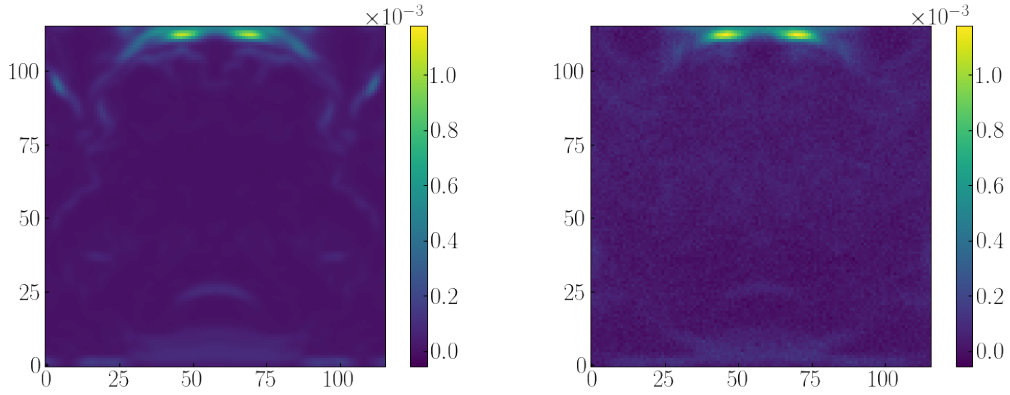


Figure 3.5: Slices presenting spatial distribution of 3D stress tensor component at $t = 7.55\,ms$, sampled down to 116^3. **Left:** Target τ_{xx}, **Right:** PIML Prediction of τ_{xx}. Statistical distribution of the above can be found in Fig. 3.6 (1^{st} row, 2^{nd} column)

Though the dataset is evolving spatially, statistics remain stationary in the JHTDB MHD dataset we used. Our CNN and PIML models for off-diagonal and diagonal terms respectively outperform the traditional gradient subgrid model. Fig.3.4 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. 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 4 checkpoints (Δt), separated by 10 time-steps (δt) each. Exact data preparation was performed as per Section 3.3.4. Our PIML method especially excelled at predicting τ_{xx} that is consequently important for P_{turb} calculation, while the gradient model completely misses the peak and the overall turbulent distribution. Next, τ_{xy} matches the bulk of the data but overpredicts the outliers. Note that $y - axis$ 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 applied to a changing spatial distribution. Since CNNs parse a kernel through the datacube, it is not trivial to assume statistical consistency in the predictions based on the evolving spatial distribution. Nonetheless, the statistically stationary dataset did not require significant tuning to achieve its results and served more as a consistency check of our algorithms before moving to a dynamic dataset.

3.4.2 Dynamic Turbulence

The ulimate goal of our study was to test the models on a more realistic astrophysical dataset. While DNS CCSN simulation from (Mösta et al., 2015) has its limitations, it is the best-resolved turbulence dataset investigating CCSN. Figure 3.6 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 statistically stationary JHTDB predictions. The gradient model continuously underpredicts the Reynolds stress distribution, performing especially poorly at capturing the outliers. Predictions of τ_y and τ_z components are comparable in accuracy across all timesteps, hence were omitted from the plot. Figure 3.5 presents an example of spatial distribution prediction, i.e. slice of the datacube at $t = 7.55\,ms$.

The key to capturing the dynamics was to train across a wide range of checkpoints

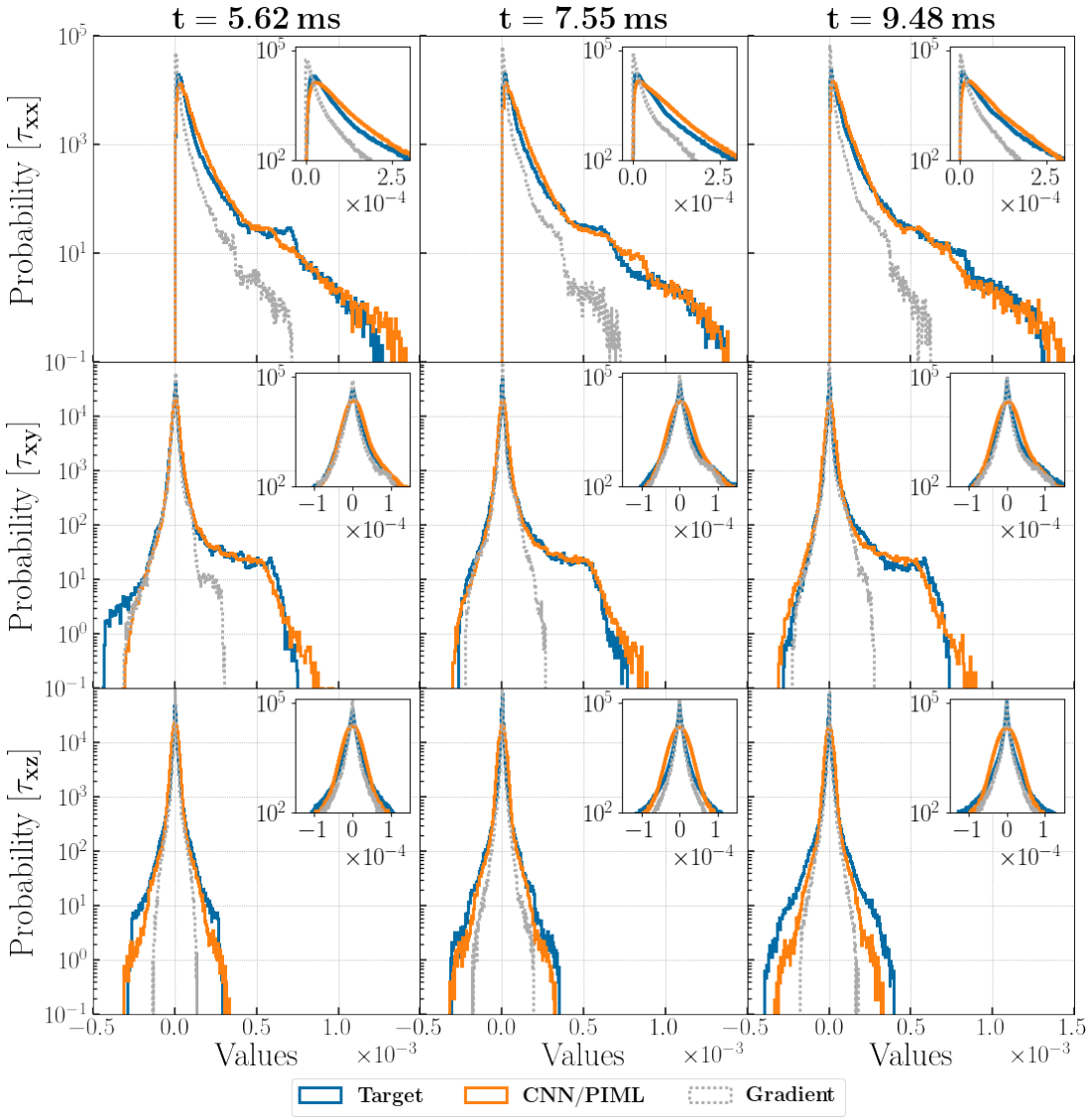


Figure 3.6: Statistical distributions of the stress tensor component, where the values are in units of u^2. **Rows**: $[\tau_{xx}, \tau_{xy}, \tau_{xz}]$ components; **Columns**: simulation time $[5.62, 7.55, 9.48]$; *ms*. **Blue**: Target τ distribution; **Orange**: CNN prediction; **Gray Dotted**: gradient turbulence subgrid model of the form $\tau_{ij} = 1/12\Delta^2 \delta u_{ik} \delta u_{jk}$ using Einstein notation, where Δ is filter width and u is velocity.

covering the first half of the simulation, $t < 5\,ms$. We were able to optimize the results using 9 equally distant checkpoints; the evolution of normalized PDFs of τ_{xy} from the exact checkpoints used in training can be seen in Figure 3.9. A diversified training dataset as such helped prevent overfitting while maintaining the flexibility of the model to predict the future timesteps ($t > 5\,ms$). To summarize, the total size of the training dataset was:

$$4\,[input\ variables] \times 9\,[checkpoints] \times 116^3\,[resolution] \quad \sim \quad 10^8\,[points]$$

Thus far, the CNN methods worked well for off-diagonal terms, and our PIML enforced realizability condition ($\tau_{ii} > 0$) for the diagonal terms. However, diagonal terms of the dynamic turbulence dataset presented another challenge - asymmetric statistical distributions. CNN with a point-to-point loss such as SmoothL1Loss has shown its robust performance at predicting quasi-symmetric distributions. This includes predictions of diagonal terms in JHTDB stationary data (τ_{xx} in Figure 3.4), where due to the shift of the peak, the distribution can be classified as quasi-symmetric. However, in the dynamic dataset, the peak is near the origin, making the distribution acutely asymmetric. As a result, while accurately capturing the outliers, the previous approach failed 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 Kolmogorov-Smirnov 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 distribution outliers while struggling to determine the peak position. On the other hand, 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 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 minimized KS loss to shift the peak into the right position. The results can be seen in the top row of Figure 3.6, with the detailed peaks presented in a separate box within each plot. Further details on the training loss behavior are provided in Appendix 3.5. While we primarily stressed accurate prediction of the statistical distribution, another benefit of not using an exclusively statistical loss is a sound spatial prediction, as shown in Figure 3.5.

The leading deliverable of τ_{ij} predictions is the ability to calculate turbulence pressure via Eq.4.8. Thus, any deviation in the peak of τ_{ii} is further exacerbated when computing P_{turb}. As an example of our PIML model performance, we present P_{turb} prediction calculation at $t = [5.62, 7.55, 9.48]\, ms$, as shown in Figure 3.7. There, the trace was taken of the *sorted* τ_{ii} components. During this operation, the *spatial* distribution of the P_{turb} is lost, though it is not required for our main goal: 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 under-resolved, while it is responsible

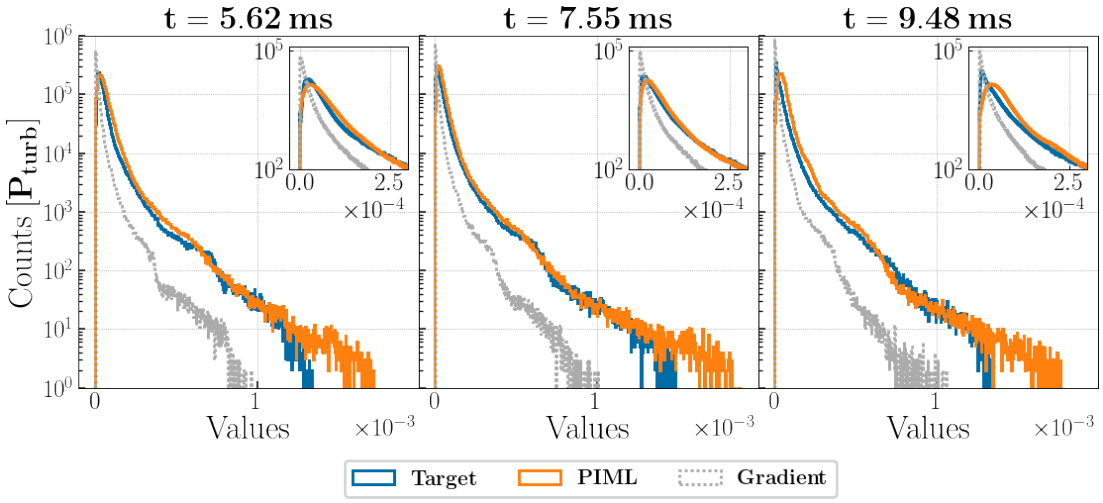


Figure 3.7: The plot presents an unnormalized distribution of P_{turb} as time evolves. P_{turb} is in units of u^2. We compare the performance of our PIML model with the gradient turbulence subgrid model.

for supplying P_{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 outwards (explosion) or inwards (black hole). Consequently, the accuracy of the total P_{turb} in the convection region becomes the most significant while alleviating the need for accurate prediction of the spatial distribution.

The performance of the PIML method shows significant advantages over the gradient model predicting the over distribution, the outliers, and the peak position. Its performance does deteriorate with time, as can be seen by the slight peak shift in Figure 3.7, right plot at $t = 9.48\,ms$. Quantitatively, performance metrics to compare PIML and the gradient model predictions are summarized in Figure 3.8. The *Top* panel shows that the total P_{turb} calculated from the PIML predictions over-predicts the target ground

truth (the dynamic 3D DNS CCSN data) by $\sim 5\% - 35\%$ depending on the future prediction time, resulting in a $\sim 19\%$ deviation on average. This is a significant advantage over $\sim 63\%$ under-prediction error of the gradient model that will fail to supply sufficient P_{turb} to re-energize the stalled shock to explode the star. This means that by using the PIML method, P_{turb} could reach on average $\sim 36\%$ of the gas pressure instead of the estimated $\sim 30\%$ in Nagakura et al. (2019), making it easier for the star to explode.

The *Middle* panel of Figure 3.8 shows KS_{stat} for the PIML method to degrade to gradient model level at the far-future checkpoints. This large discrepancy in the Target and PIML CDFs is due to the slight peak shift of the prior PIML prediction. While 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.

Lastly, the *Bottom* panel presents consistent variance between the Target and PIML results. The predicted distribution stays consistent in its dispersion, i.e., bulk and outlier distribution, which cannot be said about the gradient model results. Thus the PIML approach has an advantage in modeling the small-scale eddies that, in turn, can grow into large scales to provide the dominant fraction of the P_{turb} to re-energize the stalled shock as the simulation evolves.

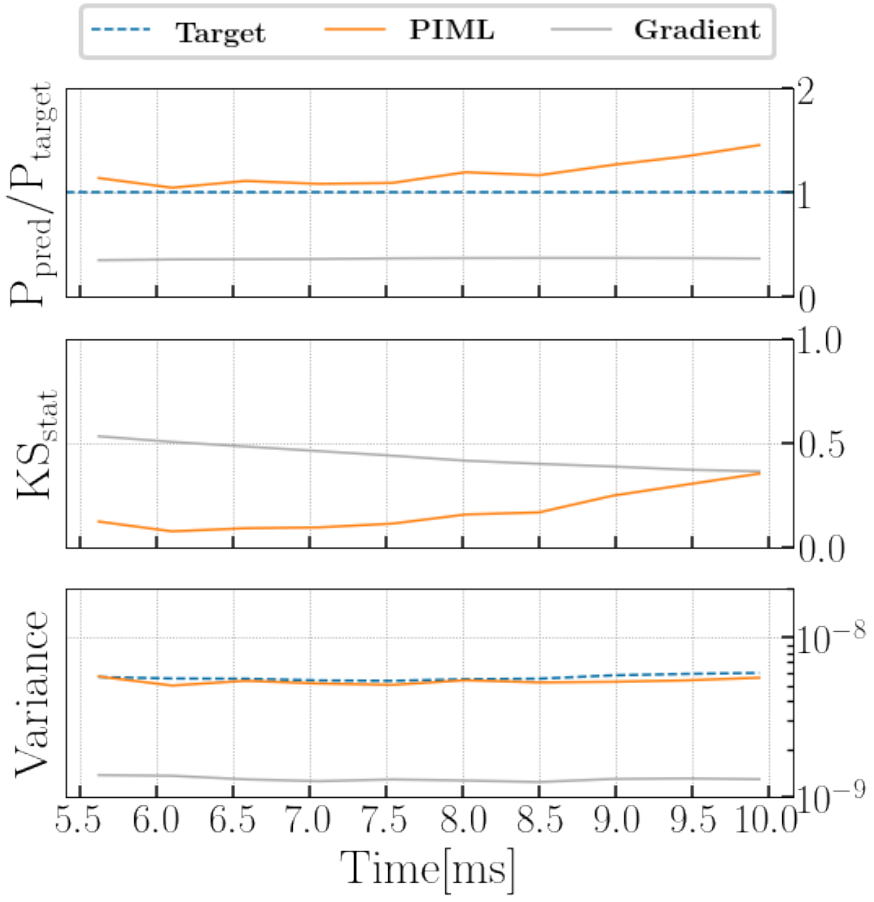


Figure 3.8: Performance metrics of the PIML vs. gradient subgrid model. The *Top* panel is the ratio of the total turbulent pressure calculated from the model prediction to the target dynamic 3D DNS CCSN data, *Middle* is the Kolmogorov-Smirnov statistic, and *Bottom* panel is variance. In total, metrics at 10 checkpoints equally separated in time are presented in the plots.

3.5 Conclusion

The study of CCSN requires an accurate treatment of turbulence, and yet conventional subgrid turbulence approaches are unreliable. A DNS treatment of turbulence in global 3D CCSN simulations is not achievable with the 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_{turb}) entirely, relying on numerical artifacts to represent its effect. Building upon prominent ML techniques used in the industry, we have developed PIML networks to increase the predictive accuracy of Reynolds stress (τ_{ij}), the trace of which is P_{turb}. P_{turb} is thus the main deliverable of this paper that can be used in a CCSN simulation in aid of re-energizing the stalled shock and exploding the star.

Our PIML approach consistently outperformed a conventional gradient subgrid model for both *stationary* and *dynamic* turbulence datasets. It resulted in a $\sim 19\%$ PIML average error of the total P_{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_{turb}, which are important for dynamic simulations to investigate the turbulent growth. Given the flexibility of ML algorithms used, these results should be reproducible across HD and MHD CCSN simulations, which we are currently investigating for the next publication. That being said, the performance of the ML models deteriorates further in time 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, temporal and overall performance can be further improved in our future work with the inclusion of recurrent neural network (RNN) layers in the models or by utilizing physics-informed neural operators (PINO) (Li et al., 2021; Rosofsky et al., 2023).

Furthermore, a broader application of the ML model can suffer from the data-model inconsistency when integrating a trained model within CCSN codes. The distribution discrepancies between the training dataset and the newly simulated grids, as well as the numerical errors, can lead to an exponential error growth in the predictions (Beck and Kurz, 2021b). Regularization of the model prediction can improve its stability. Our future work will be investigating the approaches to tackle 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 datasets. Specifically, here we used a DNS MHD CCSN dataset 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 $25\,M_\odot$) that exhibit great variation in their physical engines. In the 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 to understand its impact on the CCSN explosion rates.

Acknowledgment

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 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 dataset used throughout this paper, and Jonah Miller for insightful discussions that helped develop the ML methods.

Appendix

Training Features

The ML models were trained on u, du, B, dB as the input features and τ_{ij} as the target feature: a model per tensor component. Figure 3.9 presents an example of how τ_{xy} evolves at $t < 0.5\,ms$, following 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 dataset to prevent model overfitting and aid flexibility of the model.

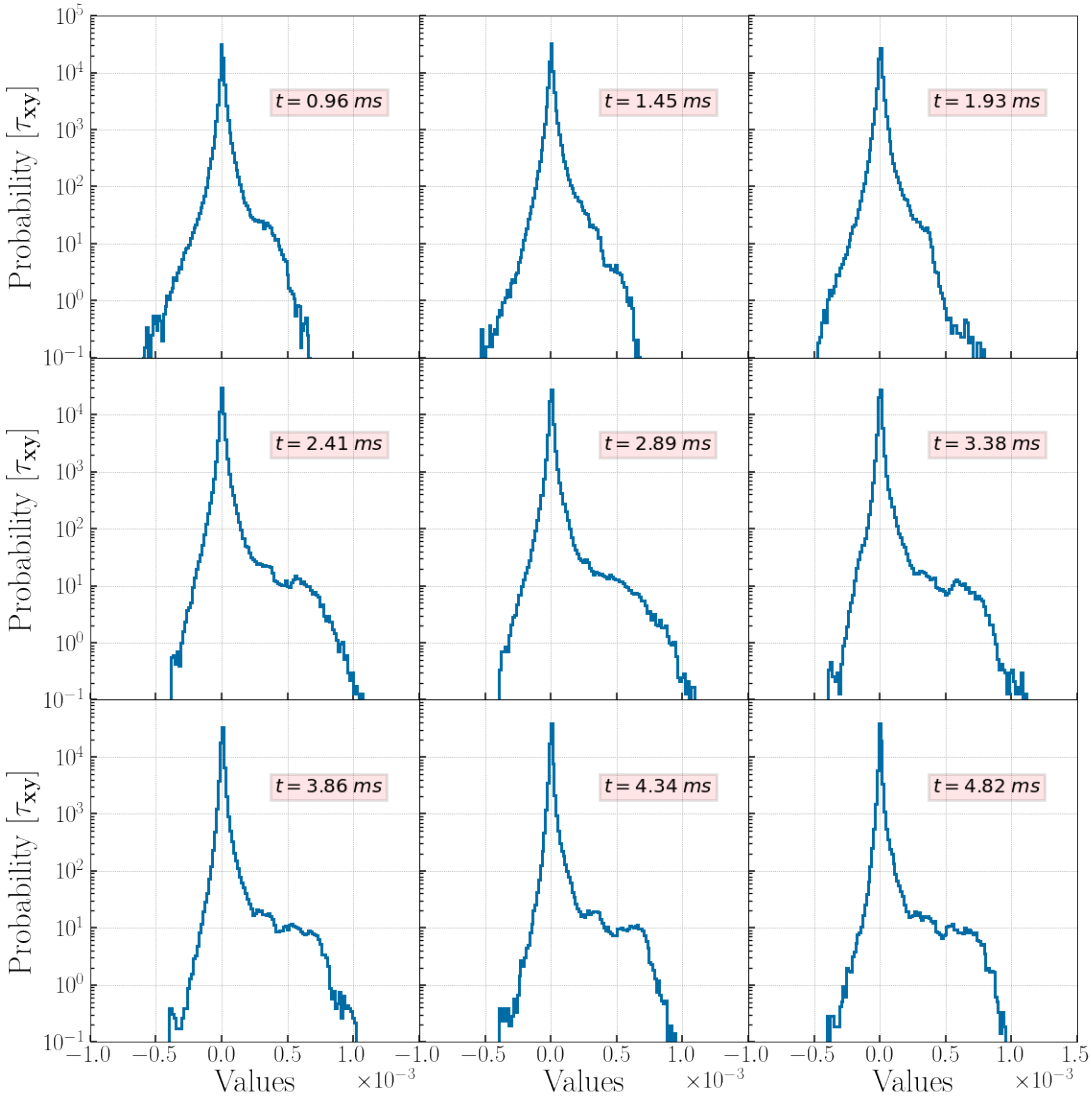


Figure 3.9: Evolution of the stress tensor component's (τ_{xy}) statistical distribution in the first half of the simulation ($t < 5\,ms$). These exact checkpoints, 9 in total, where used as a *target* to train the CNN network.

Training Loss

We developed a custom loss function (l) that combines a point-to-point (SmoothL1Loss, i.e. $L1$) with a statistical loss (Kolmogorov-Smirnov statistic, i.e., KS_{stat}) in a dynamic

fashion. The goal was to force the model to minimize $L1$ loss first to get the overall distribution shape. Then, the model prioritizes minimizing KS_{stat} to shift the peak into its correct position.

The two losses operate on different scales during training: $L1$ 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 $L1\ loss < 10^{-6}$ 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 to follow $L1$ loss's training dynamic as shown by Figure 3.10 *Top* and *Middle* plots. In summary, the two losses are combined as follows:

$$l = 0.5(\alpha\widetilde{L1}) + 0.5\widetilde{KS}_{stat} \tag{3.16}$$

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

Once KS_{stat} becomes important, the peak is being shifted, and we introduce an early stopping condition to prevent overfitting. Figure 3.10 *Bottom* plot presents the training evolution of the KS_{stat} loss component of l. After train loss (*blue*) and validation loss (*red*) cross at $\sim 4 \times 10^{-2}$, train loss decreases exponentially while 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 train and

validation losses.

More details on the application and reasoning behind the combined loss can be found in Section 3.4.2.

CNN Encoder

We present a graph of the CNN encoder we used in Figure 3.11. Data shape is noted at each arrow, akin to what was used to produce our results throughout the paper. For input and output, the shape is formatted as $[N,C,D,H,W]$ where N is the number of batches, C represents channels, i.e., features, and $[D,H,W]$ stand for depth, height, and width of the data. The notation is in agreement with PyTorch documentation. The graph was produced with Sapsan[8].

[8] sapsan-wiki.github.io/tutorials/model_graph

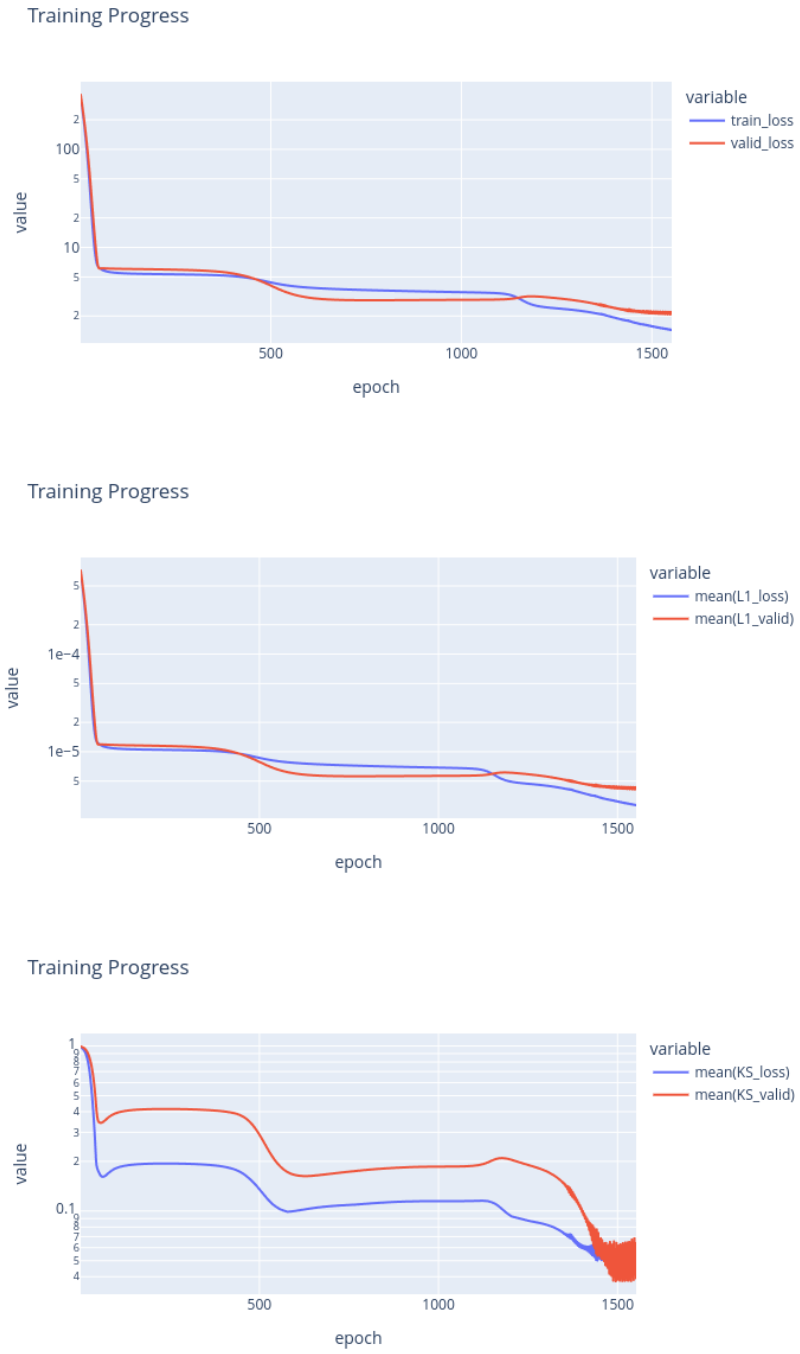


Figure 3.10: Training loss evolution: *Top* is the actual loss of the model that consists of a combined L_1 and KS loss components, *Middle* is the L_1 loss component, *Bottom* is the KS_{stat} loss component.

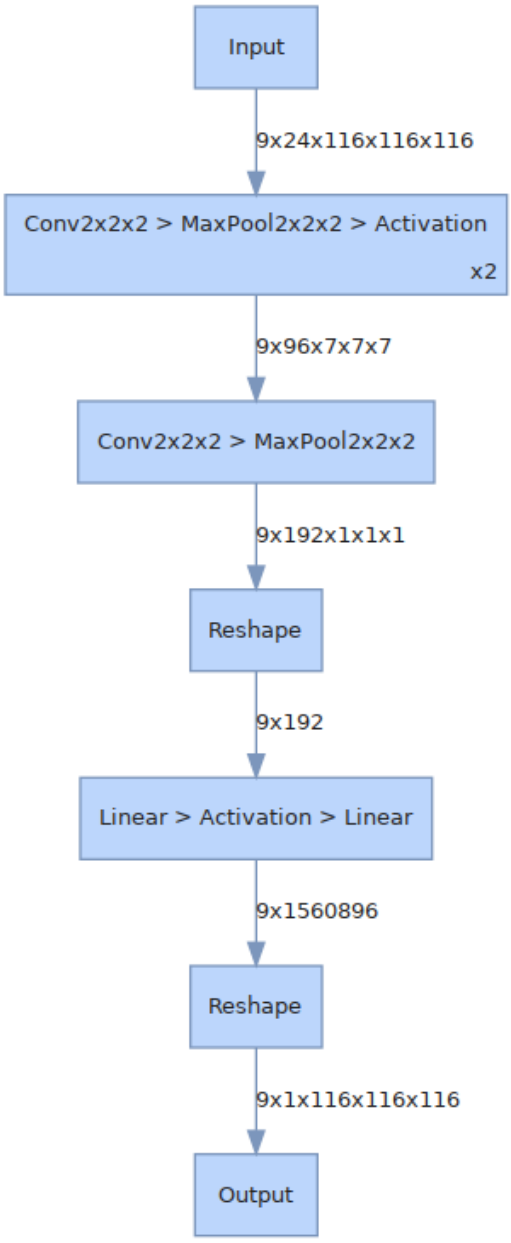


Figure 3.11: Graph of the CNN encoder used in all models for tensor component prediction. The activation function is either LogSigmoid or ShrinkTanh depending on the tensor component type. Further information can be found in Section 3.3.3

Chapter 4

Machine Learning for Core-Collapse Supernovae: 1D Models

Adapted from

Machine Learning for Core-Collapse Supernovae: 1D Models

P. I. Karpov, C. Huang, C. L. Fryer, S. Woosley

In prep for the Astrophysical Journal (2023)

Abstract

Turbulence, broadly defined, plays a key role in many astrophysical phenomena, often contributing a pressure that augments microscopic thermal motion. An example is the convection in the *hot bubble* surrounding the proto-neutron star (PNS) in a core-collapse supernova (CCSN). This has been speculated to play a significant role in the explosion mechanism behind CCSN. Unfortunately, testing this theory is challenging since accurate simulations require high-resolution 3D treatment, leading to prohibitive costs. While convection and turbulence are not the same phenomena, the Machine Learning (ML) techniques we have developed for a subgrid turbulence model in low-fidelity simulations can be applied to the chaotic circulation that develops there. Here we present a similar physics-informed convolutional-neural-network (CNN) method preserving the realizability condition of Reynolds-like stress to capture a convective pressure term from global 3D CCSN simulations and introduce it into the cost-efficient 1D CCSN simulations. We ran [12,13,16,17,18,19] $M_{\odot}$ simulations, observing either an appreciable change in the shock radius or an explosion for models that included ML. However, an ML model trained on a successful explosion in 3D will not necessarily yield an explosion in 1D. After all, the model has to remain flexible to accommodate the differences in the underlying physics of the 1D code and its evolution. Possible improvements and future directions are discussed.

4.1 Introduction

Few problems in theoretical astrophysics have been as enduring, complex, and computationally intensive as the energizing mechanism for core-collapse supernovae (CCSNe). The generally favored paradigm was proposed by Colgate and White (1966), though see also Baade and Zwicky (1934). The iron core of a highly evolved massive star collapses to a neutron star (NS), producing a flood of neutrinos. A portion of these neutrinos interacts with the accreting matter just outside the proto-neutron star (PNS), energizing and inflating a *bubble* of radiation and pairs whose expansion, in the successful case, powers the explosion. Many changes have occurred, and substantial progress has been made in the intervening decades. We now understand that the explosion does not occur promptly as Colgate and many since hypothesized (Bethe and Wilson, 1985). The accretion must slow, and the bubble takes time to inflate. There are new species of neutrinos to consider and new physics for their interactions, especially by neutral weak current interactions. The equation of state (EOS) for nuclear matter is now better understood, and the pre-supernova stellar models are much more realistic. A key advancement relevant to this paper has been the necessity of including multi-dimensional effects, especially the circulation of matter in the bubble, i.e., convection, and how that affects both the transport and absorption of neutrino-deposited energy. Many excellent reviews exist. Some recent ones are Fryer et al. (2007); Janka et al. (2016); Janka (2017); Radice et al. (2018); O'Connor et al. (2018); Müller (2020);

Burrows and Vartanyan (2021).

While 1D models are simpler and cheaper to run, spherically symmetric CCSN models lack the necessary physics to explode. Simple piston (Woosley et al., 1995) or thermal-bomb (Nomoto et al., 2006) injections are often used. These methods can produce very different results, especially when following the effects of fallback (Young and Fryer, 2007). A number of increasingly sophisticated models have been developed to improve the fidelity of artificially driven explosions, including enhanced neutrino luminosities (Perego et al., 2015) and continuous energy or entropy injection within the convective region (Fryer et al., 2018; Imasheva et al., 2023). As understanding of the importance of convection grew, some 1-dimensional models attempted more self-consistent approaches to include the convection (Mabanta et al., 2019; Couch et al., 2020)

Many of these 1-dimensional prescriptions have been influenced by the results of multi-dimensional models. Over the last three decades, there has been a growing realization that multi-dimensional effects play an important role in producing a successful explosion (Herant et al., 1994b; Burrows et al., 1995; Janka and Mueller, 1996; Fryer and Heger, 2000; Fryer and Warren, 2002; Fryer and Young, 2007a; Murphy et al., 2013; Couch and O'Connor, 2014; Couch and Ott, 2015; Lentz et al., 2015; Janka et al., 2016; Radice et al., 2018; O'Connor and Couch, 2018; Nagakura et al., 2019; Müller, 2020; Burrows et al., 2020). The term *multi-dimensional* here covers a host of

phenomena, even without rotation or magnetic fields, including:

- The standing accretion shock instability, or SASI (Blondin et al., 2003; Foglizzo et al., 2007)
- Asymmetric explosions induced by large-scale asymmetries in the collapse (Burrows and Hayes, 1996; Fryer, 2004; Wongwathanarat et al., 2013)
- Non-spherical flows in the post-accretion shock zone, which increases the mass in the gain region (Couch and Ott, 2013; Couch and Ott, 2015; Müller, 2020)
- Convection powered by a negative entropy gradient set up by the initial bounce shock and driven by neutrino absorption (Herant et al., 1994b; Fryer and Heger, 2000; Fryer and Warren, 2002) or solely by neutrino absorption (Burrows et al., 1995; Janka and Mueller, 1996)
- Pre-existing progenitor turbulence seeding that convection (Couch and Ott, 2013; Müller et al., 2017)

See the review by (Radice et al., 2018) for a recent discussion and additional references. We shall be primarily interested here in approximating the effects of neutrino-powered convection, an inherently 3D phenomenon, on the explosion.

Convection assists in re-energizing the shock in a variety of ways. Some of them involve providing extra pressure and energy at the shock (Murphy et al., 2013; Couch

and Ott, 2015; Radice et al., 2016; Fryer et al., 2018), and dissipation of kinetic energy to heat (Mabanta and Murphy, 2018). Others involve redistributing entropy in the region beneath the shock (Murphy and Meakin, 2011) and changing the efficiency of energy deposition by cooling the base of the convection zone (Janka, 2012). Convection can also alter the geometry beneath the shock and increase the accretion rate in a star that is in the process of exploding (Burrows et al., 2020). While all these can have a significant contribution to a successful explosion (Mabanta et al., 2019; Couch et al., 2020), in this paper, we focused solely on modeling the pressure term, commonly referred to as turbulent pressure, P_{turb}. It has been estimated to provide up to 50% of the gas pressure P_{gas} (Couch and Ott, 2015; Nagakura et al., 2019). In our analysis of the Burrows et al. (2020), we confirmed similar values for the P_{turb} contribution.

The importance of multi-dimensional simulations, particularly in 3D, to study CCSN cannot be overstated. Unfortunately, they are also very expensive to run, making it difficult to run extensive parameter studies (Couch et al., 2020). On the other hand, statistically significant simulation ranges could be done in reduced dimensionality if multi-D flow effects could be recovered. Several attempts have been made by using analytical models to augment the 1D CCSN simulations by Mabanta et al. (2019) and Couch et al. (2020). While the results are promising, the analytical models rely on the underlying physical assumptions and closure models. Here we have tried to develop a more generalizable approach to be used in any CCSN code, inferring the effects of

convection from fundamental runtime observables, such as velocity, temperature, and others (more in Section 4.3.2).

We have used a machine-learning (ML) method to approximate P_{turb} due to convection trained on the existing 3D models of Burrows et al. (2020); produced by FORNAX. The datasets were angle-averaged, which both filtered and mapped them to 1D. In addition, only the convective region between the PNS and the shock was used for training and inferencing. The time-independent ML model is based on Karpov et al. (2022), adapted to 1D. It is a physics-informed convolutional neural network (CNN) designed to preserve the realizability condition of the Reynolds stress that is necessary for accurate turbulent pressure predictions.

To test the ML approach, we used a 1D Lagrangian code called COLLAPSO1D, first described by Herant et al. (1994b). As part of this project, the code has been open-sourced (available on GitHub[1]). It was modernized in a few key ways to match better with FORNAX that generated the training dataset. The ML integration is a PyTorch wrapper that is decoupled from COLLAPSO1D and can be used in any FORTRAN code for either training or inferencing from a pre-trained model; the wrapper is also open-sourced and available. We performed the latter: training a PyTorch model in python, and then using it at runtime only for inferencing in COLLAPSO1D. Using a set of FORNAX-consistent progenitors from Sukhbold et al. (2016) from 12 to 19 $M_{\odot}$, we ran high-resolution 1D CCSN models to study the effects of P_{turb} on the shock position

[1] github.com/pikarpov-LANL/COLLAPSO1D

and test applicability of ML to a CCSN problem as a whole.

We describe the 1D CCSN code we used, i.e., COLLAPSO1D, and define our P_{turb} extraction from first principles in Section 4.2. Next, Section 4.3 discusses all of the ML model details, including its architecture, training dataset, and its preparation. Then, we present the ML training and its application results in Section 4.4. In section 4.5 we analyze our results and discuss them in the context of the models from other groups, and the future of ML for simulating astrophysical flows, augmenting 1D and 3D models. Lastly, we conclude the paper in Section 4.6 and provide further technical details and a resolution study in the Appendix.

4.2 Formalism

4.2.1 Convection or Turbulence?

Often, any and all multi-D velocity fluctuations are amalgamated into the term *turbulence*, including convection (Müller, 2020). Convection in a medium with a large Reynolds number (Re) produces turbulence on its smallest scales, eventually dissipating as heat. However, convection also exhibits large-scale correlated motions that are not truly turbulent in nature, but they are relevant for low-Re flows such as here.

The large-scale motions carry momentum flux that resembles an anisotropic pressure; from the integral scale of $\sim 100\,km$ down to $1.5\,km$ of the "dissipative" range.

This large-scale motion is chiefly responsible for augmenting the pressure near the accretion, and it is this major effect that our ML models attempt to capture in 3D and transcribe to 1D.

4.2.2 1D Code

COLLAPSO1D[2] (now open-sourced) is a 1D Lagrangian hydrodynamics (Benz, 1991) code with artificial viscosity for modeling CCSN, introduced by Herant et al. (1994b) and Fryer (1999). It contains the basic physics for core-collapse simulations and should be a suitable platform for differential studies, e.g., of accretion shock radius, with and without ML additions. COLLAPSO1D includes a post-Newtonian, spherically-symmetric implementation of general relativity (van Riper, 1979).

The continuity equations with the physics included in our code are:

$$\frac{1}{\rho} = 4\pi r^2 \frac{\partial r}{\partial m} \tag{4.1}$$

$$\frac{dv}{dt} = -\frac{4\pi r^2}{\partial m}\left(\frac{\partial P_{gas}}{\partial m} + \frac{\partial P_{turb}}{\partial m} + \frac{\partial q}{\partial m}\right) - G\frac{M(r)}{r^2} \tag{4.2}$$

$$\frac{ds}{dt} = \frac{4\pi r^2}{T}\left(\frac{1}{2}\frac{\partial q}{\partial m} - \frac{\partial u_\nu}{\partial m} + \frac{\partial Y_e}{\partial m}\mu_e\right) \tag{4.3}$$

using density ρ, radius r, cell mass m, enclosed mass M, velocity v, pressure P, artificial viscosity q, gravitational constant G, entropy s, temperature T, neutrino energy u_ν, electron fraction Y_e, and electron chemical potential μ_e. The main addition of this work is the pressure term $\frac{\partial P_{turb}}{\partial m}$ responsible for modeling fluctuations in multi-D flow, i.e.,

[2] github.com/pikarpov-LANL/COLLAPSO1D

convection, specifically in the θ and ϕ directions of the spherical coordinates, in the momentum equation (Eq. 4.2). Throughout this paper, we will call this term turbulent pressure to stay consistent with the literature, even though small-scale turbulence is not contained in the original training data[3]. Further details on modeling that term are given in Section 4.3.

The neutrino transport utilizes a gray, 3-species ($\nu_e, \bar{\nu}_e$, and $\nu_{\mu,\tau}$) flux-limited diffusion scheme below the "neutrinosphere" and a light-bulb/free-streaming approximation assuming an $e^{-\tau}$ energy loss where τ is the optical depth of the neutrinos. The dividing line need not be at the $\tau = 1$ surface (standard placement of a photosphere), and determining the dividing line between flux-limited diffusion and the light-bulb approximation can lead to variations in the final result.

Since the neutrino trapping scheme in COLLAPSO1D is based on examining diffusion times and optical depths in individual zones, the time step and results are zoning-dependent, which proved problematic. The abrupt change in zoning beneath the neutrinospheres occasionally caused issues in neutrino transport. As a result, the luminosity, especially of $\mu-$ and $\tau-$ neutrinos, was unstable but often exhibited large excursions. Usually, these variations did not cause significant changes in the accretion shock radius, but occasional glitches were sometimes noted and excluded from the analysis. As in Herant et al. (1994b), the $\mu-$ and $\tau-$ neutrino luminosities were generally sev-

[3]Current state-of-the-art global CCSN simulations are extremely under-resolved, running at low Reynolds number Re of several hundred at most, far from the high-Re usually associated with a general definition of turbulence.

eral times larger than in modern calculations, including those of Burrows et al. (2020). Their mean energy was also larger, $\sim 40 MeV$. Otherwise, though, the shock radius history for the various models and the ν_e- and $\bar{\nu}_e$- luminosities and temperatures were similar to modern studies, with the former hovering around $150 km$ and showing substantial increases of order 20 - 30% when the high entropy material associated with the base of the oxygen burning shell in the pre-supernova star was accreted. Entropies per baryon in the hot bubble were around 10, and the infall velocity at the accretion shock was 3 - 4 $\times 10^9 cm/s$. As judged by the location of a density of $10^{12} g/cm^3$, PNS radii were near $30 km$.

The original coupled EOS combined the Lattimer-Swesty EOS at high densities (Lattimer and Douglas Swesty, 1991) and the Blinnikov EOS at low densities (Blinnikov et al., 1996), and an 18-isotope nuclear network (Fryer, 1999). For this project, we have implemented SFHo[4] by Steiner et al. (2013), consistent with the source of our training dataset (Burrows et al., 2020).

Further details of the code setup for our models are given below:

- **Resolution:** a non-uniform, static Lagrangian- (i.e., mass-) grid of 9000 cells. It is meant to capture the hot convective region below the shock ($100 - 150 km$), which was typically $\sim 0.01\ M_\odot$, resulting in the fine-zoning requirement of $\sim 0.0001\ M_\odot$ to capture the shock. Due to the accreting material, the fine zoning

[4]We adopted an EOS wrapper by Evan O'Connor: github.com/evanoconnor/EOSdriver with further implementation details in COLLAPSO1D's documentation: pikarpov-lanl.github.io/COLLAPSO1D/eosdriver/

has to cover 1.2-1.7 $M_\odot$, raising the total cell count to 9000. The grid includes three regions: medium proto-NS, high shifting-convective, and low-resolution outer zones (more in Appendix 4.6).

- **Progenitors:** [12.0,13.0,16.0,17.0,18.0,19.0] $M_\odot$ from Sukhbold et al. (2016).
- **EOS:** SFHo EOS tables Steiner et al. (2013) by integrating the EOSdriver by O'Connor and Ott (2010) into the code.
- **Machine Learning:** a C wrapper for ML with PyTorch is based on the pytorch-fortran[5] wrapper (Alexeev and Hrywniak, 2022) that was expanded for gfortran & ifort compatibility. It is set up to perform real-time inferencing and can be adjusted to utilize single-core, multi-core CPU, or GPU via CUDA (see Section 4.4 for overhead and its mitigation). The wrapper is not exclusive to COLLAPSO1D, with the interface made for easy integration with any code written in FORTRAN 90 & above (follow COLLAPSO1D's documentation[6]).

The progenitors and the EOS are consistent with the training data from 3D CCSN FORNAX simulations by Burrows et al. (2020) used for our ML model. While the neutrino physics and Lagrangian vs. Eulerian architecture of the codes differ, we tried to stay consistent in other details of the code, including the spatial resolution of the convective bubble.

[5] github.com/alexeedm/pytorch-fortran
[6] pikarpov-lanl.github.io/COLLAPSO1D/

4.2.3 Turbulence Extraction

When bridging the gap between 3D and 1D simulations, we have to treat the 3D simulations as the ground-truth Direct Numerical Simulation (DNS), even if they are under-resolved. The latter would act as a Large Eddy Simulation (LES) simulation needing a closure model for turbulence. In this section, we will present DNS formalism with an LES-like procedure of extracting turbulence for ML model training purposes, similar to procedures described in Karpov et al. (2022); Nagakura et al. (2019).

By applying a spatial filter, each DNS variable is decomposed into a mean ($\bar{u}$) and fluctuating (u') parts:

$$u = \bar{u} + u' \tag{4.4}$$

To extract the fluctuating (turbulence) part, we define solid angle averaging for the mean, i.e., averaging over θ and ϕ shells as a function of radius, as the filtering operation. This procedure allows us to both filter and collapse the 3D data to 1D at the same time, and the filtering operation satisfies Reynolds rules of the mean Germano (1992). Applying said filter to the NS equations results in the following hydrodynamic equations for continuity and momentum:

$$\partial_t \bar{\rho} + \nabla \cdot (\bar{\rho}\tilde{\mathbf{u}}) = 0 \tag{4.5}$$

$$\partial_t (\bar{\rho}\tilde{\mathbf{u}}) + \nabla \cdot (\bar{\rho}\tilde{\mathbf{u}} \otimes \tilde{\mathbf{u}} + \bar{P}I) - \bar{\rho} g = -\nabla \tau \tag{4.6}$$

where $\tilde{\mathbf{u}} = \overline{\rho \mathbf{u}}/\bar{\rho}$, which represents the mass-weighted filtering, i.e., Favre filtering; τ is

the stress due to the turbulent motion, i.e., Reynolds stress, defined as:

$$\tau_{ij} = (\widetilde{u_i u_j} - \tilde{u}_i \tilde{u}_j) \tag{4.7}$$

In turn, total pressure due to turbulence (P_{turb}) is defined as the product of the trace of **R** and density:

$$P_{turb} \quad = \quad \rho \, tr(\mathbf{R}) \tag{4.8}$$

since in the training dataset, the radial term was dominant in comparison to angular components, $R_{rr} \gg R_{\theta\theta} + R_{\phi\phi}$, we set

$$P_{turb} \approx \rho R_{rr} \tag{4.9}$$

4.3 Turbulence Model

4.3.1 Machine Learning Model

The ML model we employ here is based on our previous Physics-Informed (PI) ML method from Karpov et al. (2022) adapted to a 1D setting with a few key changes. We used the PI layer to preserve the realizability of the τ, ensuring predicted P_{turb} stays positive. Also, a filtering layer with a Gaussian kernel is used for smoothing the prediction, preventing sharp intermittent gradients of P_{turb}. The overarching model structure is summarized in the schematic of Figure 4.1. Furthermore, the custom loss function

from the aforementioned paper that combines Smooth L1 (spatial) and Kolmogorov-Smirnov (statistical) losses improved the training time and accuracy of the model.

Considering the integration with COLLAPSO1D, both input and output for the model training must be in 1D. In total, 5 features (density ρ, sound speed u_s, gas pressure P_{gas}, temperature T, and entropy S) were used for training, each of size 200, across ~ 20 checkpoints. Even though not all training features are independent, ML benefits from data transformations to accelerate the training convergence. In ML, this is generally known as feature engineering (Zheng and Casari, 2018). In our case, it was best to use physical intuition for guidance, i.e., using inter-dependent features for training such as P_{gas} and u_s.

This amounted to ~ 4000 training data points per ML model. Due to the 1D nature and the smaller size of the training dataset, a shallow 1D convolutional neural network (CNN) proved sufficient to capture turbulence's dynamic states presented in the training data, although no turbulence model of its dynamic growth is used here. Next, we did not need to predict the full stress tensor τ_{ij} but instead wanted to directly predict P_{turb}. Thus, Eq.4.8 is folded into the model itself. Lastly, the output was a normalized quantity of P_{turb}/P_{gas} for stability reasons.

- **Model**: 1D CNN encoder with physics-informed layers. The encoder implicitly predicts velocity fluctuations (u_i'), then PI layers calculate $u_i'^2$ to enforce $\tau_{ii} > 0$ and filter to find the mean.

- **Optimizer**: Adam Optimizer[7] - extension of the stochastic gradient descent; it was picked due to the good performance on sparse gradients.

- **Activation function**: Tanhshrink[8], for which the function is defined by

$$f(x) = x - tanh(x)$$

- **Loss function**: Custom SmoothL1Loss[9] - 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 (behaves as L_1 if the absolute value is high or as L_2 if the absolute value is low).

4.3.2 Basis 3D Dataset

We trained our models using non-rotating HD CCSN simulations performed by Burrows et al. (2020). Most of the included corresponding [12,13,16,17,18,19] $M_\odot$ stars exploded, except the 13 $M_\odot$. The 14 & 15 $M_\odot$ were not included despite being available since they also did not explode and exhibited lower turbulence levels. Thus they made unlikely candidates to push our 1D models to explode, leaving the test bed for non-exploding training datasets to 13 $M_\odot$. The simulations were performed using

[7] pytorch.org/docs/stable/generated/torch.optim.Adam.html
[8] pytorch.org/docs/stable/generated/torch.nn.Tanhshrink.html
[9] pytorch.org/docs/stable/generated/torch.nn.SmoothL1Loss.html

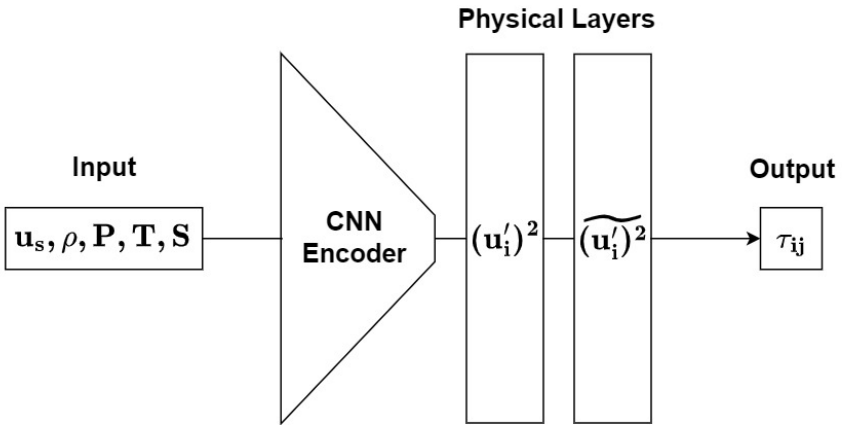


Figure 4.1: ML model schematic to predict P_{turb}/P_{gas} (output). Besides a conventional CNN Encoder, it includes a physical layer to enforce a realizability condition to ensure P_{turb} stays positive, and a smoothing filter layer.

an Eulerian spherical 3D code called FORNAX (Skinner et al., 2019) with a dendritic grid of $678 \times 256 \times 128$ ($r \times \theta \times \phi$). For progenitors, Sukhbold et al. (2016) models were used for the subset of models we use in this study. It is important to note that the first 10 *ms* of the simulations after the bounce were performed in 1D, at which point mild velocity perturbations were introduced, and the models were mapped to 3D. As such, we are not utilizing the early checkpoints for training. For the complete setup details, please refer to the aforementioned manuscript.

Given our interest in the convective region to revive the shock, we focus on the subset of the data between the PNS and the shock position. To extract the P_{turb}, we followed a procedure described in Nagakura et al. (2019), which studied the effects of turbulence through Reynolds-like stress components.

Data Preparation

The raw 3D data was unsuitable for training and making the ML model usable in a 1D simulation. In addition, the subset used for training was limited by both space and time. Below is the complete procedure to transform the original data into the final training dataset, in order:

1. **Calculate $\mathbf{P_{turb}}$**, i.e. τ_{rr}, radial Reynolds-like stress tensor component, using Eq.4.7 from the original 3D data

2. **Average** over solid angle, which simultaneously filters and collapses the data from 3D to 1D for all relevant training variables $[u, u_s, \rho, P_{gas}, T, S]$

3. **Select checkpoints** until the shock either reaches $300\,km$ (at which point it is considered exploding) or collapses below $100\,km$. Using only 10% of the total number of $1\,ms$-checkpoints, amounting to ~ 20 equidistant checkpoints in total per $M_\odot$, proved sufficient for training. Then, 3% of in-between checkpoints were used for validation.

4. **Interpolate** linearly the grid between PNS and the shockfront to a constant number (i.e., 200) of cells to equate the size of the convective region through all checkpoints used for training

5. **Scale** the variables to unity to equalize their weight during model training

6. **Target** feature is set as P_{turb}/P_{gas} instead of P_{turb} to relate the predicted turbulence to present in COLLAPSO1D P_{gas}.

The collapsed 3D-to-1D datasets, after the above step 2, are summarized by the shock position vs. time after the bounce plot in Figure 4.2. It is directly comparable to Figure 2 in Burrows et al. (2020).

Note that a time-diverse set of checkpoints is used for training to capture the evolution of turbulence and prevent overfitting. That said, validation checkpoints are equally important, and despite being only 3% of the total dataset, they are responsible for the early-stopping condition in model training. Without them, the model could overfit to the average of all training snapshots instead of remaining flexible enough to predict the dynamic nature of turbulence.

A sample of the result target feature used in training, P_{turb}/P_{gas}, is shown in Figure 4.3. The snapshots are taken at $[100, 150, 200]$ ms, to make it easier to compare 19 $M_{\odot}$ with the medium resolution 19 $M_{\odot}$ (green curve) in Figure 11, Left from Nagakura et al. (2019). Our training targets, i.e., P_{turb}/P_{gas}, are consistent with their paper, given the similar feature extraction procedure.

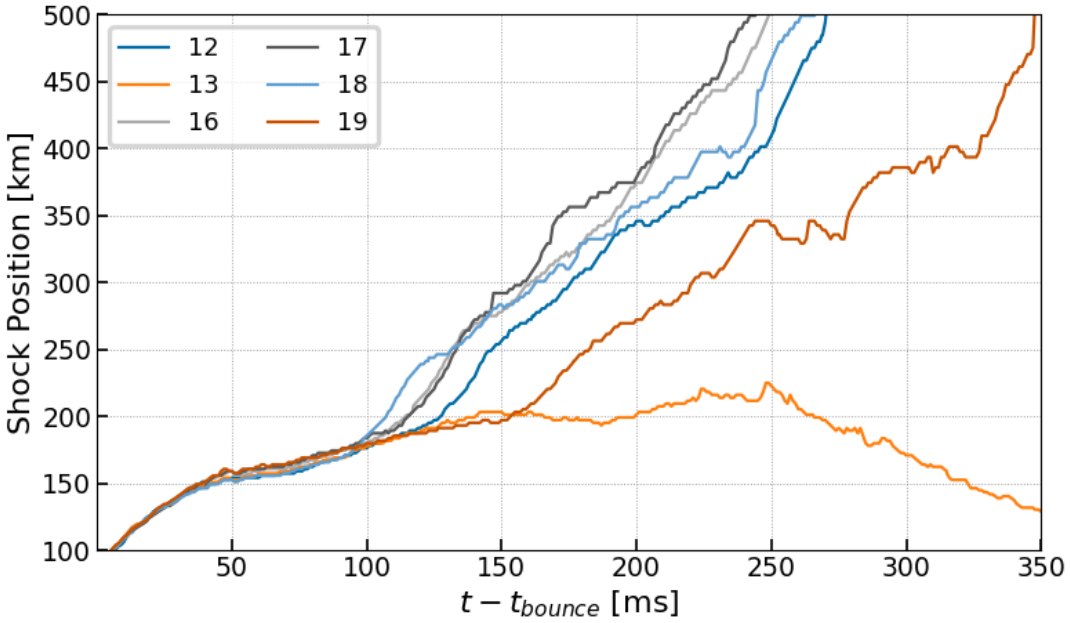


Figure 4.2: Shock evolution after the bounce of the 3D-to-1D mapped datasets from Burrows et al. (2020) selected for ML model training. Please note that $13M_{\odot}$ fails to explode.

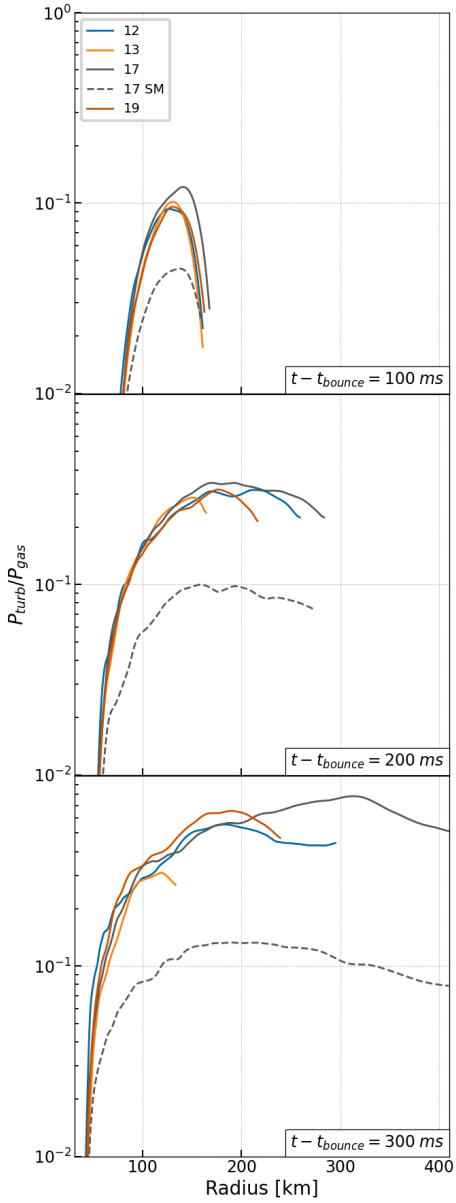


Figure 4.3: Evolution of the training P_{turb}/P_{gas} for [12,13,17,19] $M_{\odot}$ in the convective region between the PNS and the shockfront, where all explode except the 13 $M_{\odot}$. At 200 *ms* (the last plot), 12 and 17 $M_{\odot}$ have already passed 300 *km* per Figure 4.2. The dashed grey curve for 17 $M_{\odot}$ labeled "SM" stands for "small-scale" turbulence, i.e., extracted on the integral scale (a few *km*) of the simulation using a Gaussian filter with $\sigma = 2\ [cells]$. Depending on the definition of turbulence, large vs. smallest-resolved scale, its role in CCSN can be affected by a factor of ~ 4. The base 3D dataset was produced using FORNAX.

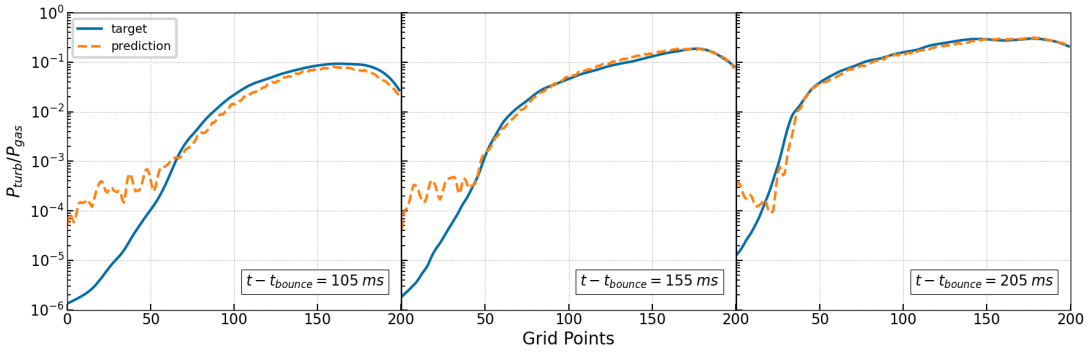


Figure 4.4: Predicting P_{turb}/P_{gas} profiles of the unseen timesteps from the training dataset of the 12 $M_{\odot}$. Prediction is not oscillatory and stays consistent to the shape across 2-3 orders of magnitude above 10^{-3}, i.e., the model predicts a consistent trend above 0.1% of P_{turb}/P_{gas}.

4.4 Results

4.4.1 ML Training

For each progenitor, a corresponding ML model was trained following the procedure described in Section 4.3.1. Considering the small size of the 1D dataset, training on a CPU (12-core, 24-thread AMD Ryzen 9 3900X) proved to be sufficient. With our custom loss-function, each model would converge within 1000 epochs. The model was constructed and trained using the Sapsan framework (Karpov et al., 2021), which includes production-ready example of it, called PIMLTurb1D[10]. Refer to Appendix 4.6 for more details on the training evolution.

Testing on the remaining 90% of the unseen data from Burrows et al. (2020), the

[10] sapsan-wiki.github.io/details/estimators/#physics-informed-cnn-for-1d-turbulence-modeling-pimlturb1d

models have performed well predicting the growth of P_{turb}/P_{gas}. Figure 4.4 presents a prediction of select, unseen checkpoints at far spread-out simulation times, [105, 155, 205] ms, for the 12 $M_{\odot}$ simulation. It is important to note that the y-scale is logarithmic; hence the tail below 10^{-3} is expected to misbehave. In addition, such precision is not necessary, as it is expected for $P_{turb}/P_{gas} > 10^{-2}$, i.e., 1%, to potentially affect the shock radius to cause an explosion. The most important was for the model to capture the plateau at maximum P_{turb}/P_{gas} and its evolution. The level of performance is comparable across all ML models and progenitors in this paper.

4.4.2 Baseline 1D CCSN

At first, we conducted baseline simulations for $[12, 13, 16, 17, 18, 19]$ $M_{\odot}$ without any additional P_{turb}. As shown in Figure 4.5, none of the models exploded (as expected). They have primarily settled at $\sim 130\,km$ and will eventually collapse. After the bounce, the shock radius has a minor oscillatory behavior due to the region behind the shock not being in the high-resolution zone as per Figure 4.14. It takes $15\,ms$ after the bounce for the models to stabilize. This behavior will dictate the delay of turning on ML to apply P_{turb} in the next batch of simulations. We compiled analytic profile evolution of velocity, pressure, temperature, and many others, accessible on COLLAPSO1D's documentation[11]; full setup details can be found there as well.

[11] pikarpov-lanl.github.io/COLLAPSO1D/research/

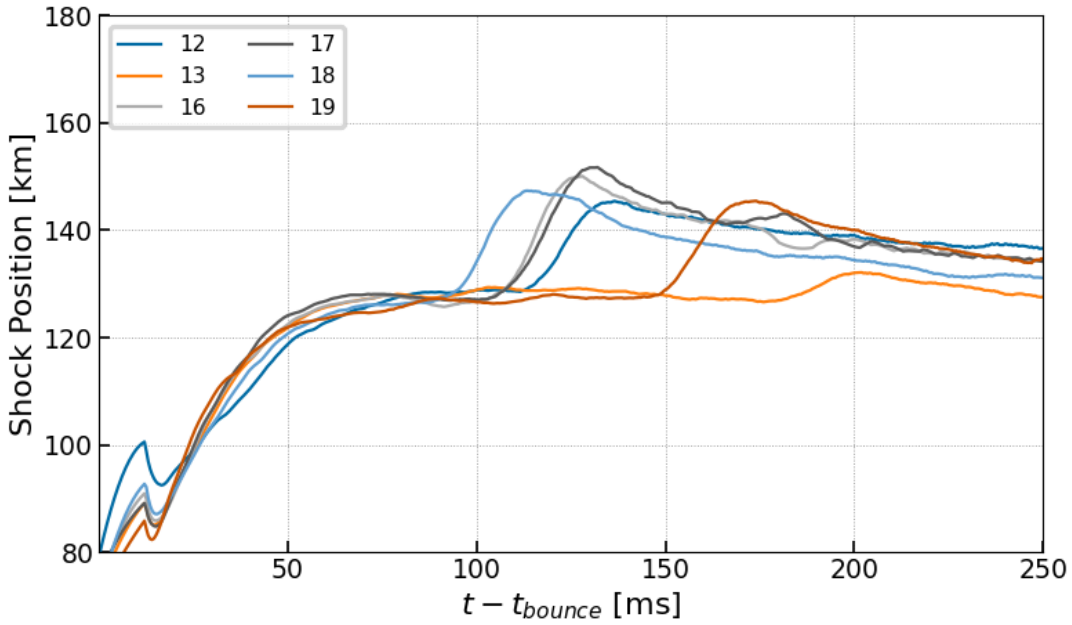


Figure 4.5: Shock evolution of the baseline simulations *without* ML using [12,13,16,17,18,19] $M_{\odot}$ progenitors. None of the 1D COLLAPSO1D simulations have exploded, and no positive velocities have been generated. Numerical noise has been filtered out of the curves by applying a Savitzky–Golay filter.

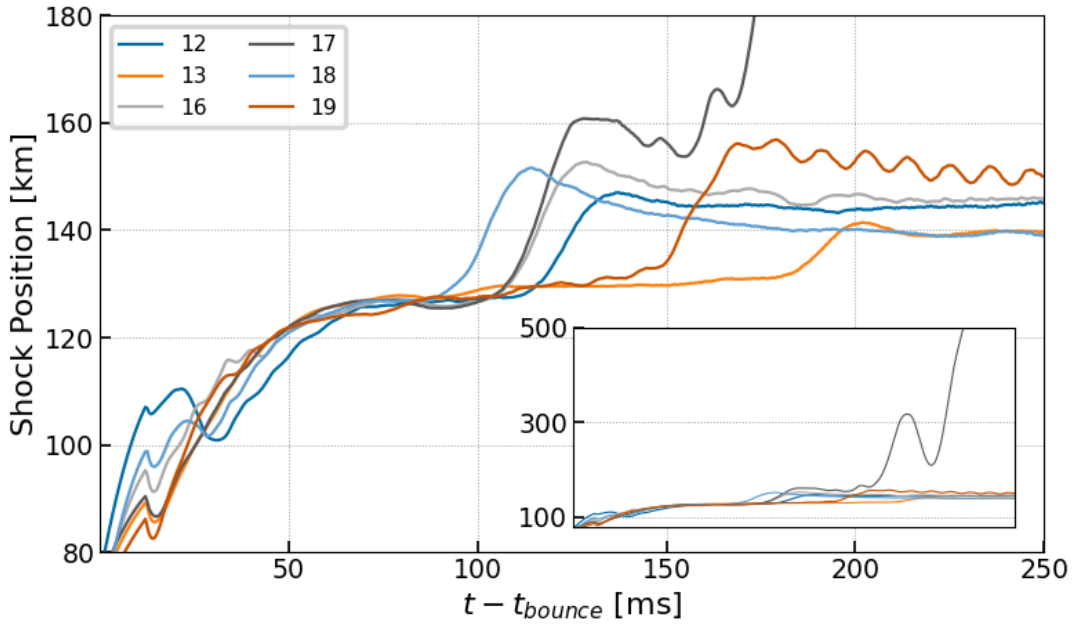


Figure 4.6: Shock evolution of the ML-augmented [12,13,16,17,18,19] $M_\odot$ simulations with the added P_{turb}. The smaller subplot has the same x-axis range from 0 to 250 ms, but expands the y-axis out to 500 km to show the exploding 17 $M_\odot$. Numerical noise has been filtered out of the curves by applying a Savitzky–Golay filter.

4.4.3 1D CCSN with ML subgrid model

Keeping the setup exactly the same, P_{turb} was introduced via the ML subgrid model. To make it work, we employed the following procedure:

1. **Identify convective region** but calculating the current PNS and shock positions, excluding the width of the diffused shock
2. **Scale** the features identically to the training dataset
3. **Interpolate** the features in the region to array size 200
4. **Pass to** PyTorch **wrapper** that converts the features into torch tensors, loads the trained model, and performs inferencing
5. **Recover** P_{turb} by multiplying the prediction by P_{gas}
6. **Augment Pressure** by adding P_{turb} in the momentum equation

Even though ML inferencing has a relatively low cost, it still causes a $\sim 200\%$ overhead per timestep (per $\sim 10^{-7}$ s). However, given the relatively slow evolution of P_{turb} on such timescale, updating it through inferencing every 5 timesteps proved sufficient. As a result, the total overhead of running COLLAPSO1D with the ML model was $40-50\%$ in comparison to baseline runs.

To give a perspective into simulation evolution, i.e., the dynamics of the runtime features feeding into the ML model to predict P_{turb}, plots of $[u, \rho, T]$ leading to an

explosion of 17 $M_\odot$ are shown in Figure 4.7. The convective bubble there used for inferencing lies between the red lines identifying the PNS edge and the shock. Note that a few cells right below the shock are also excluded from inferencing to account for shock diffusion. At 175 ms, large positive velocities are developed (top right plot in Figure 4.7), resulting in an explosion of that simulation.

The shock position has moved out for all progenitors and caused the 17 $M_\odot$ model to explode. Figure 4.6 summarizes the shock evolution in all models, with the smaller inside figure showcasing the runaway of the exploding models beyond 500 km. Comparing with the baseline simulations in Figure 4.5, 12 & 17 $M_\odot$ comparison is plotted in Figure 4.8: baseline runs in solid lines, and dashed for models with the ML subgrid model. The 12 $M_\odot$ with ML slowly diverged from the baseline, moving the shock further out by $\sim 5\%$ in comparison to the baseline without developing positive velocities. On the other hand, the 17 $M_\odot$ diverged rapidly to a similar 5% shock position increase at the early time and then proceeded to explode shortly after reaching the “plateau".

4.5 Discussion

Adding P_{turb} through ML directly affects the 1D CCSN simulations by increasing the shock radius. The outwards adjustment of the shock was generally in the range suggested by other studies (Radice et al., 2018), about 10-30%. However, the correlation between the outcomes in COLLAPSO1D and FORNAX was not directly consistent, i.e.,

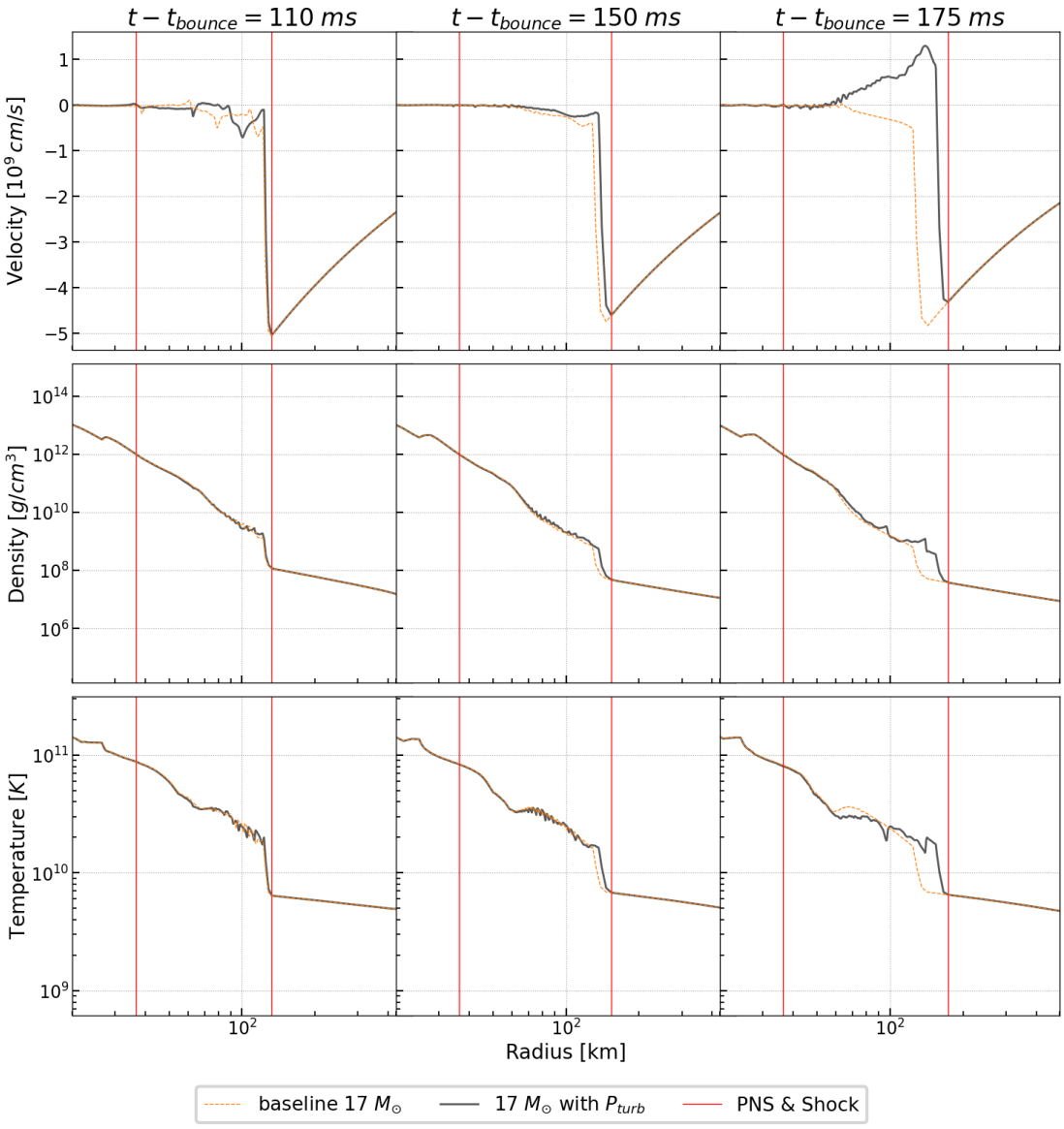


Figure 4.7: Velocity, density and temperature evolution for the baseline (dashed orange) and the ML-augmented with P_{turb} (solid gray) 17 $M_\odot$ simulations. In the 3^{rd} column at 175 ms, the latter has developed strong positive velocities for the first push towards an eventual explosion. The red lines signify the PNS edge and the shock position in the ML-augmented case. The region in-between is fed into the ML model to predict P_{turb}. The checkpoints are consistent with the predictions in Figure 4.9.

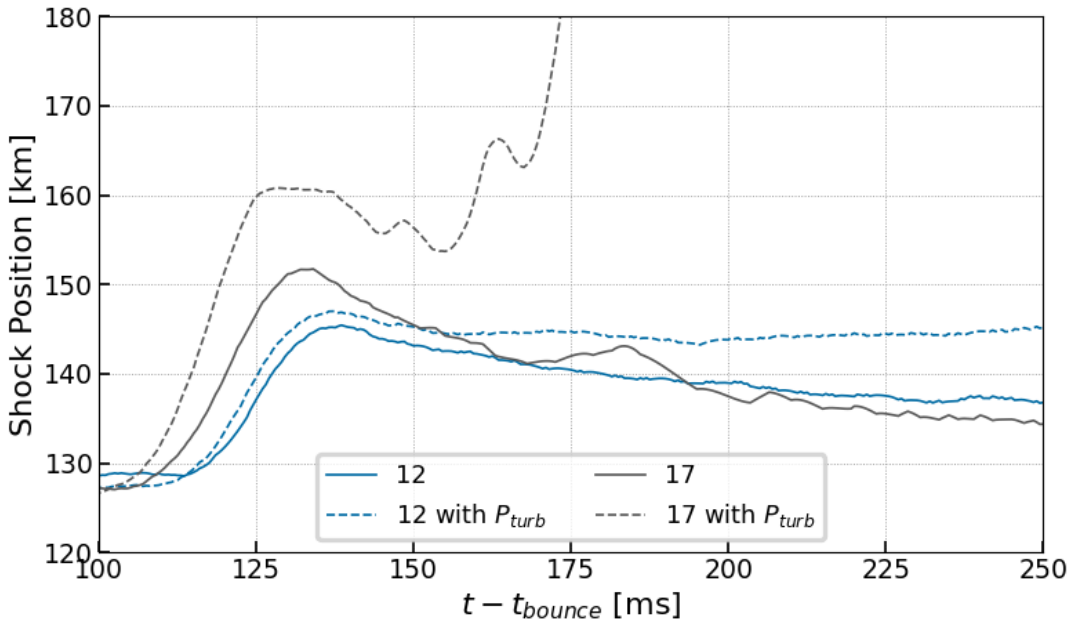


Figure 4.8: Comparison of the shock evolution between the baseline and ML-augmented models with P_{turb} for 12 & 17 $M_{\odot}$, a failed and a successful explosion respectively.

training the ML model on an exploded simulation did not necessarily produce an explosion for the same progenitor mass in 1D. To some extent, that is expected. Adding a simple scalar pressure does not capture the full effects of a 3D simulation and the basic physics in COLLAPSO1D and FORNAX are different. In addition, we need to evaluate the ML performance to validate its accuracy further and make suggestions for the future of ML in HD & MHD simulations.

4.5.1 Effect of turbulence in 1D models

It is important to understand the ML model behavior at runtime and what caused 17 $M_\odot$ to explode. P_{turb}/P_{gas} has been plotted at different simulation times for [12,17,19] $M_\odot$ in Figure 4.9, similar to Figure 4.3 for the training dataset. Let us break down this set of three plots in more detail:

- **Magnitude:** P_{turb} predictions stayed within the training dataset and physical bounds, not exceeding P_{gas}, even for explosive cases.

- **Shape:** the curves are primarily affected by the Lagrangian grid used: a large number of points at the smaller radii given the higher density, while there are fewer cells near the shock. In an Eulerian case of the original training dataset, the convective region is more uniformly resolved, giving a smoother overall shape. An adaptive grid would improve the shock resolution.

- **Evolution Timescale:** given the 15 ms delay and the underlying physical differences between COLLAPSO1D and FORNAX, P_{turb} evolution timescales are different between the codes, which is to be expected. As such, it is more useful to look at the relative growth and peak magnitudes of turbulence within COLLAPSO1D simulations.

- **Shock Position:** note the increase of P_{turb}/P_{gas} relative to the larger shock radius for the corresponding models (check Figure 4.6 for shock positions). While there is no linear relationship between the shock radius and P_{turb}, the larger outer radius indicates favorable conditions for turbulence growth.

It is important to remember that the training dataset dictates that prediction. In our case, how closely the conditions are matched will affect the turbulence evolution. In the non-exploding cases, P_{turb}/P_{gas} reached $\sim 10\%$, unlike $\sim 70\%$ for the exploding 17 $M_{\odot}$ due to the rest of the observables not passing the conditions for the explosion. Such differences can be due to several reasons, including the shock position that signifies more significant underlying differences. For example, the discrepancies in the neutrino engine, i.e., trapping/free streaming conditions, can dictate the fate of the star.

There is an apparent dichotomy between exploding and non-exploding models, as Figure 4.10 illustrates. The successful 17 $M_{\odot}$ quickly expands to 7% larger than the baseline and then undergoes large shock radius oscillation while staying $> 30\%$ with the baseline leading to explosion by $\sim 220\,ms$. On the other hand, the failed explo-

sions stay relatively close to one another, slowly growing up to 5-10% by late times ($\geq 250 ms$). Examining the P_{turb}/P_{gas} in Figure 4.11, it shoots up to 70% in the successful 17 $M_{\odot}$, consistent with training dataset (Burrows et al., 2020). The rapid drop at $\sim 225 ms$ is due to the model being trained only on the initial shock moving out to $300 km$, i.e., following it through the explosion. The back-propagating, post-explosion shocks are then detected, under which our model continues to inject P_{turb}, but that cannot be deemed accurate and goes beyond the scope of this paper.

As for the non-exploding models, P_{turb}/P_{gas} stays just below the 20% mark, where it converges, unable to grow further. This is fairly consistent with the non-exploding 13 $M_{\odot}$ training dataset from Burrows et al. (2020), where P_{turb}/P_{gas} reaches a maximum of $\sim 30\%$ before dying down. Considering the training dataset has a larger shock radius before P_{turb} becomes significant, the smaller maximum P_{turb}/P_{gas} in our simulations is to be expected. In fact, the shock in our non-exploding low-resolution (grid=2000 cells) runs does expand to a larger shock radius, and reach $\sim 30\%$ of P_{turb}/P_{gas} (Figure 4.19).

A similar dichotomy between successful and failed explosions was also observed in low-resolution runs. While they support our hypothesis of the larger baseline shock radius leading to its greater increase with the addition of P_{turb}, there are resolution effects that start playing a dominant role in model dynamics. The inability to resolve the shock well, leading to large shock oscillations over 10s of km per time step, results in untrustworthy models to draw definitive conclusions. Nonetheless, it serves as a

good training exercise, explored further in Appendix 4.6.

To summarize, even though adding P_{turb} does help to re-energize the shock and move it outwards, its ability to do so effectively depends on the underlying physics and likelihood of an explosion, creating a positive feedback loop. This observation speaks to the idea of a successful CCSN being on a proverbial "cliff", where a small extra contribution can push it over the edge to produce an explosion. In this context, adding P_{turb} can be an important factor in dictating the fate of a star.

4.5.2 Comparison & Limitations

Our high-resolution findings are comparable with the 3D CCSN models by Couch and Ott (2015), which concluded that P_{turb} can reach up to 50% of P_{gas}. Inspired by those conclusions, work has been done by Mabanta et al. (2019) and Couch et al. (2020), developing parameterized models for turbulence in 1D. They showed the importance of turbulence in the neutrino-driven CCSNs through convection below the shock. It is important to remind that not only pressure was considered as the dominant effect of turbulence, but also energy flux F_e and dissipation of kinetic energy to heat ε_{heat} through the means of turbulence. This work focuses solely on the pressure term P_{turb} to test the applicability of ML for supernovae problems. That said, all of these turbulent terms are related to velocity v_{turb}; hence our ML model can be used to include those terms without re-training.

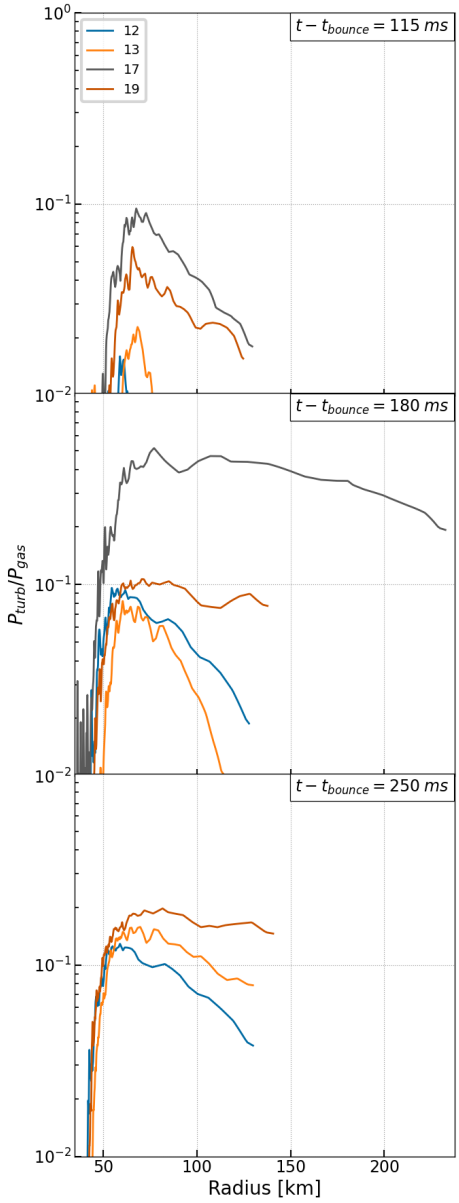


Figure 4.9: Evolution of the ML-predicted P_{turb}/P_{gas} for the [12,13,17,19] $M_{\odot}$ in COLLAPSO1D. The plots show only the region where ML was applied, i.e., the region between the PNS and the shock front. Besides, P_{turb} fraction continuing to grow beyond 0.2 for the exploding 17 $M_{\odot}$, note that the shock radius for that progenitor mass was also the farthest out, indicating favorable underlying physical conditions. The last plot at 250 *ms* does not have the 17 $M_{\odot}$ since it already exploded. At later times, 12 & 13 $M_{\odot}$ reach the same 0.2 P_{turb}/P_{gas} as the 19 $M_{\odot}$ at 250 *ms*, hence non-exploding models are consistent in their evolution.

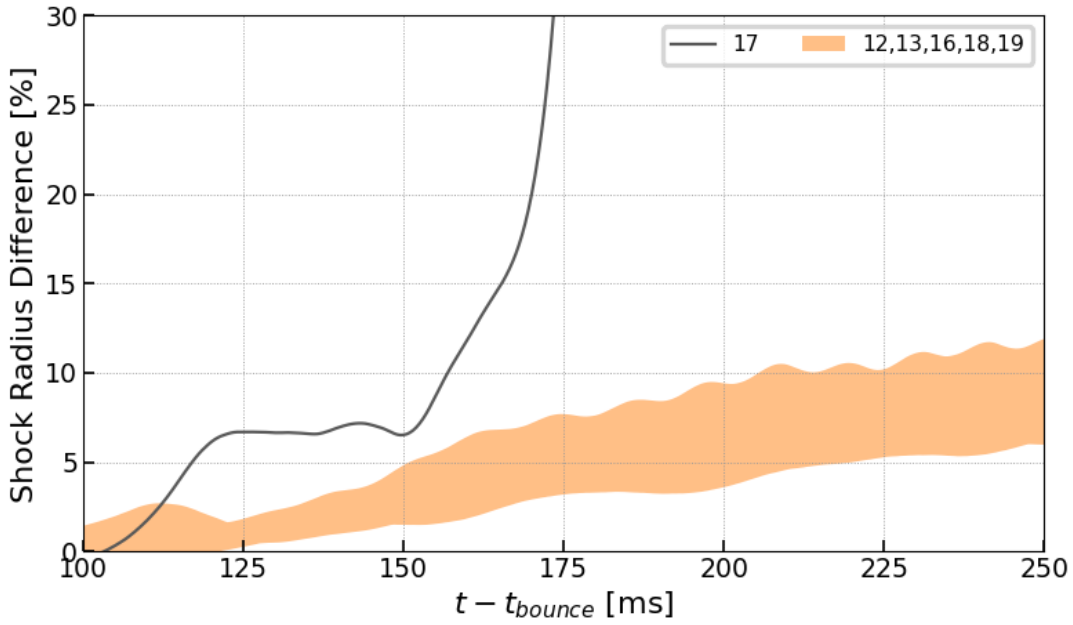


Figure 4.10: Cumulative comparison of the shock position increase for exploding 17 $M_\odot$ and non-exploding [12,13,16,18,19] $M_\odot$. The differences between the latter models are minor while exhibiting a similar growth behavior with oscillatory dynamic; hence they are combined. This presents the dichotomy of the failed vs. successful explosions, and how a small early push by P_{turb} can result in the explosion.

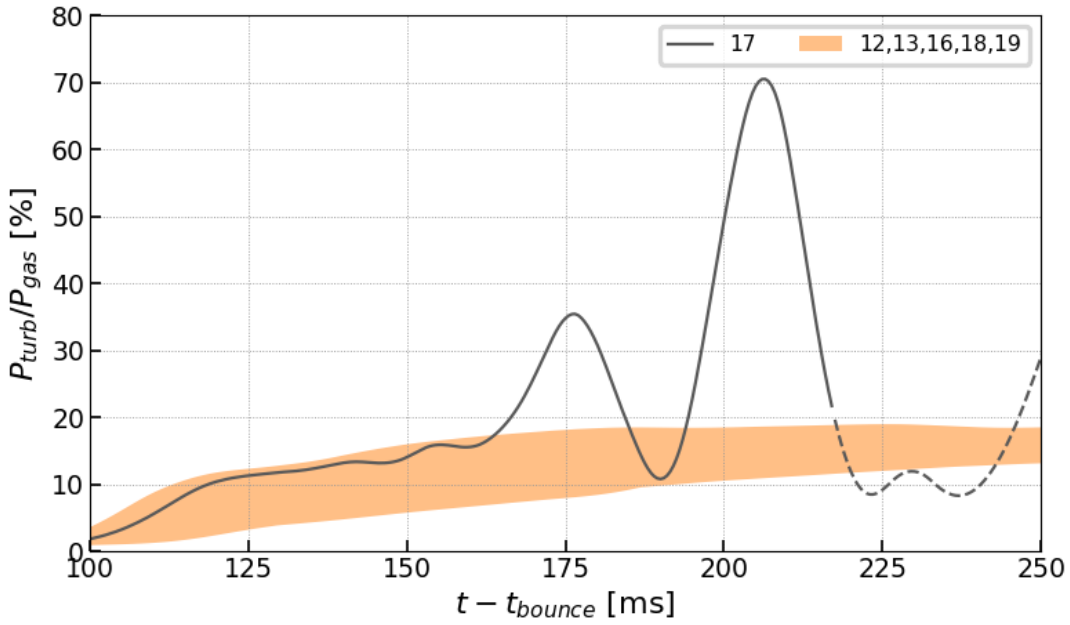


Figure 4.11: Evolution of the maximum predicted P_{turb}/P_{gas} at each checkpoint for exploding 17 $M_{\odot}$ and non-exploding [12,13,16,18,19] $M_{\odot}$. The former corresponds well with the training dataset (Burrows et al., 2020) and the shock-position dynamic from Figure 4.6. As the shock moves out, P_{turb}/P_{gas} also increases. After the shock has moved far past $300\,km$ ($\sim 220\,ms$), the ratio drops since the ML model was not trained passed that regime, and it suppresses those outliers. Hence, those predictions are not expected to be accurate and are marked with a dashed line.

With that in mind, there are fundamental differences between our turbulence model and the previous work done in 1D CCSN codes, which can relate to the differences between analytical and ML approaches to turbulence modeling. The former is limited by the approximations used for their derivation. For example, in the case of Mabanta et al. (2019), a constant luminosity tuned by hand is used where turbulent dynamics is derived directly from neutrino luminosity. Hence other means of turbulence generation are ignored. Couch et al. (2020) proposed a different approach based on MLT called Turbulence in Reduced dimensionality (STIR). With a more elaborate neutrino-transport scheme which complemented a more generalizable turbulence modeling approach, it came with the downsides of MLT. In particular, the dependence on the mixing length scale factor λ_{mix} itself, which works well for small scales but is problematic at capturing the full range of scales turbulence evolves on (Joshi et al., 2019).

On the other hand, the ML approach is limited solely by the training dataset, which should improve as better-resolved datasets with higher physical fidelity become available. It can capture a wide range of scales, i.e., the growth of turbulence, as long as the training dataset exhibits that behavior. While an analytical model typically strives to reduce the number of variables in its formalism, it is much easier to include and investigate a broader range of physical dependencies on a target observable with ML. Thus, while previous physical assumptions can guide the architecture, the models are not limited by them, potentially giving the ability to uncover new physical dependen-

cies. Outside of some physical limitations built into the model's architecture, it remains flexible and physically scalable, improving as better, preferably DNS, training datasets come to fruition.

Our approach also does not capture the increase in accretion rate that might occur beneath an unstable accretion shock. Mapping the 3D model into a 1D scalar field results in lost information. Such an effect is due to a macroscopically anisotropic velocity field resulting from discrete convective plumes, which follows the non-conventional definition of *turbulence* used in this paper. Nevertheless, it is thought to be important (Couch and Ott, 2013; Müller and Janka, 2015), and its neglect diminishes the fidelity of our ML model, at least as applied to 1D models.

The approach also does not realistically include the effect of a *cooled* gain radius that might affect the neutrino absorption efficiency at the base of the convective region. That is, the entropy profile is not smoothed, and it is rendered flat as true convection would do. This may be a limit to the 1D approximation, but further exploration is needed.

4.6 Conclusion & Future Directions

The ML framework we developed shows great promise for including the effects of turbulence and other chaotic motions in simplified calculations of many phenomena, core-collapse included. We started with baseline [12,13,16,17,18,19] $M_\odot$ simulations,

of which none exploded, as was expected. The addition of the ML-predicted P_{turb} term resulted in the P_{turb}/P_{gas} to reach $\sim 20\%$ for the non-exploding models, and up to $\sim 70\%$ for the exploding 17 $M_{\odot}$. These numbers, albeit with some discrepancies, are consistent with the training set from Burrows et al. (2020). As for the shock position increase, in the case of non-exploding models, it gradually grew to a $\sim 10\%$ larger radius than the baseline. The effect is in the bounds from Radice et al. (2018) ($\sim 25\%$), despite the differences in dimensionality and neutrino physics.

It is worth noting the observed correlation between the shock radius, the maximum predicted P_{turb}, and ultimately explodability. For example, the exploded 17 $M_{\odot}$ moved out the farthest in comparison to all the other models in both the baseline ($\sim 150\,km$) and in the early times of ML-augmented runs ($\sim 160\,km$). We also conducted extensive code-optimization studies not presented in this paper, which explored the sensitivity of both *baseline* models (without the ML augmentation) and those with additive pressure to assumptions regarding zoning, EOS, and, especially, neutrino trapping parameters. Generally speaking, *failed* (non-exploding) models where the shock stalled at a larger radius were more likely to explode with an ML augmentation. We suspect that the larger shock radius indicates the underlying physical conditions that resemble more the exploding FORNAX models on which it was trained.

As for the non-exploding models presented here, despite their smaller shock radius, it is unreasonable to expect an augmentation from convectively generated pres-

sure alone to make the difference between failure and explosion in all cases. Instead, a more representative metric becomes the increase in shock radius due to P_{turb}. Indeed Murphy et al. (2013) and Radice et al. (2018) find the effective pressure support generated by the turbulence only accounts for an increase in the shock radius up to $\sim 25\%$. Janka et al. (2016) estimate that the effect of turbulence with Mach number (Ma) is the increase in the shock radius by a factor of about $(1+4/3Ma^2)^{2/3}$ or about 20% for Ma = 1/2. This increase is consistent with what we see for the 17 $M_\odot$ model (and in the low-resolution models in Appendix 4.6).

Overall, important lessons were learned, and progress was made that points the way for future improvements. The ML model itself does an excellent job of describing the fluid motions in the 3D models of Burrows et al. (2020) (see Figure 4.4). It inherently includes all the 3D effects present in the training models. The six training variables used here could be changed or expanded to include additional dependencies, but even the present tables could be used in other, more modern, 1D codes. An obvious next step would be to apply the same tables to 1D models using FORNAX, which has the same physics, except for dimensionality, as the 3D training set. We are optimistic that better correlations in outcome would be observed.

However, there are inherent deficiencies in 1D. By limiting the output of the ML model to a single scalar-pressure augmentation - other important multi-dimensional effects are neglected. However, it can already be used to include energy flux and heat

dissipation due to their dependence on turbulent velocity, which can be easily extracted from the predicted pressure term (Couch et al., 2020). Other examples of the missing multi-D features include the augmentation of accretion by instabilities at the accretion shock is ignored, and the redistribution of entropy and matter near the gain radius. Both of these are probably important effects (Murphy and Meakin, 2011; Burrows et al., 2020).

Lastly, we would like to list several potential improvements/approaches to utilizing ML for the modeling of astrophysical flows:

A Generalized Model - a single model to predict for all progenitors. In this paper, we trained a unique ML model per each progenitor while using the same architecture. However, the next step would be to train a single multichannel-output model to include progenitor mass dependency. For example, a classification-type NN can be used to achieve that, where the continuous mass range can be used as the *class*. As a result, it would be reasonable to utilize the model for a continuous range of progenitor masses in-between the training dataset masses.

Time Dependence - the inclusion of the time variable in the training dataset. The model architecture could include Recurrent NN (RNN) layers to capture the timescales. However, a concern is the naturally different simulation evolution timescales due to the underlying physics and dimensionality discrepancies. As such, while a worthy parameter to investigate, time dependency must be included with caution.

Analytical Model Tuning - recover subgrid turbulence through established analytical models in conjunction with ML. While there are no DNS CCSN simulations that resolve the dissipation scale of turbulence available for training, the approach presented here can be extended via a simple analytical model, e.g., Dynamic Smagorinsky (Lilly, 1966), to subgrid scales. A step further would be to use ML for tuning a more sophisticated analytical model, such as RANS or BHR (Besnard et al., 1992; Denissen et al., 2012, 2014).

3D Subgrid Model - one could also easily modify the ML model to train on differences among 3D simulations of varying resolution (Nagakura et al., 2019), essentially acting as a subgrid model in cheaper low-resolution 3D models. Then the hydro code could capture important macroscopic effects, and ML compensate for information lost with coarse resolution. As shown by Karpov et al. (2022), CNN ML models can successfully capture P_{turb} term from 3D MHD DNS data to perform 3D inferencing. While 3D ML models take longer to train, given the larger training dataset, there is an opportunity to use deeper networks to allow more complex nonlinear models to be trained. Increased training time is a worthy price to pay, especially when it does not significantly affect the overhead for inferencing at runtime. Such ML subgrid models will drive the computational cost down and can pave the way to more extensive parameter studies of 3D CCSN. Our open-sourced ML implementation is not limited to COLLAPSO1D; in fact, it is agnostic to the base FORTRAN code, its dimensionality, or even the ML

model architecture itself, which can be tailored further depending on the closure model in question, i.e., prediction target.

Acknowledgement

This research was partially supported by LANL and the DOE ASCR SciDAC program. We would like to thank Adam Burrows and David Vartanyan for the discussions and for providing us with the training data from FORNAX, as well as Jonah Miller for insights into building the ML pipeline and code optimizations.

Appendix

Training Details

1D PIML Model from Figure 4.1 is detailed as a layer-by-layer schematic in Figure 4.12. The changes in the data shape can be tracked as noted at each arrow. For input and output, the shape is formatted as $[N, C, L]$ where N is the number of batches, C represents channels, i.e., features, and L stands for the length or size of the data. The notation is in agreement with PyTorch documentation. Thus the graph presents an input size of [20,5,200] with 20 checkpoints and 5 features of interpolated size 200. Note that even though the number of checkpoints differed for every progenitor depending on the training dataset, it was close to 20 for all models. The graph was produced with

Sapsan[12].

Custom Loss Function from Karpov et al. (2022) that combines a spatial point-to-point (SmoothL1Loss, i.e., $L1$) with a statistical loss (Kolmogorov-Smirnov statistic, i.e., KS_{stat}) was used for this project. The latter is used to define an early stopping condition to prevent overfitting. The total loss (*Top*) and its $L1$ (*Middle*) & KS_{stat} (*Bottom*) are presented in Figure 4.13 as a function of the training iterations, i.e., epochs. For the first few hundred epochs, $L1$ loss dominates. However, once it gets below 1, becoming comparable to KS_{stat} in magnitude, the prediction adjusts its shape to match the training dataset statistically. No significant moves in the total loss and its L1 component are seen at that point. However, the KS_{stat} component keeps adjusting until it drops below the cutoff of 0.11, which was tested to be sufficient for the model to remain flexible and accurate without overfitting across all progenitors.

Resolution Study

Besides the high-resolution (HR) simulations with a grid of 9000 cells, we performed the same study for a low-resolution grid with 2000 cells[13]. The ML models stayed the same as presented in the paper, with the only difference being the underlying grid. A comparison of the grids is shown in Figure 4.14. Thus, the main difference between the grids was the "high-resolution" region, through which the shock moves

[12] sapsan-wiki.github.io/tutorials/model_graph
[13] pikarpov-lanl.github.io/COLLAPSO1D/research/

out as the mass accretes onto the PNS. Between the HR and LR grids, the zone was resolved 6 times worse in the case of the latter. To better understand its effect on the shock position, check Figure 4.15. Recall that the region between the PNS and the shock (between the red lines) is used as an input to the ML model to predict P_{turb}. The LR (orange curve) shows only a few cells in the span of a $\sim 100 km$, which causes significant oscillations of the shock radius, affecting ML inferencing. The Eulerian grid would have been superior to a Lagrangian counterpart used here. If the grid would be adjustable, it would reduce the zoning requirement.

The average shock position moved out significantly farther than in the HR runs (from $\sim$ 150 to $\sim 200 km$) for most models, as per Figure 4.16. The 12 $M_{\odot}$ stood out here, extending to $\sim 280 km$, yet not developing any positive velocities and exploding. Once ML was applied, the shock moved out further in all models, relative to the HR runs, with 12 $M_{\odot}$ exploding (Figure 4.17). It is better illustrated in Figure 4.18, where is shock radius difference reaches 18%, in comparison to 12% in the HR runs (Figure 4.10). Furthermore, Figure 4.19 shows the P_{turb}/P_{gas} converging to 30%, as per the training dataset (Burrows et al., 2020) for non-exploding models. However, notice the large spread of the orange-filled region representing the non-exploding [13,16,17,18,19] $M_{\odot}$ models. This spread is due to the large shock oscillations mentioned above, with the shock jumping up to $100 km$ per timestep.

In the paper, we proposed that the larger shock radius would result in higher P_{turb}/P_{gas}

and, in turn, increase the likelihood of an explosion. However, the LR simulations have significant flaws to support that undoubtedly. While P_{turb} did have a larger effect here, the previously exploded 17 $M_{\odot}$ has failed here. Instead, the 12 $M_{\odot}$ exploded. The expectation was for the 17 to explode, along with others either moving out to a larger radius (which did happen) or exploding. This discrepancy could be attributed to the poorly-resolved shock causing the models to grow unstable.

In conclusion, the LR results are consistent with the dichotomy observed in HR runs and agree with our assumption of a large shock radius leading to higher P_{turb}/P_{gas}. However, the large oscillatory behavior prevents us from solidifying those claims. It is best to treat this LR study as a cautionary tale for grid setup when applying ML in a shock-dependent zone and modeling CCSN turbulence in general.

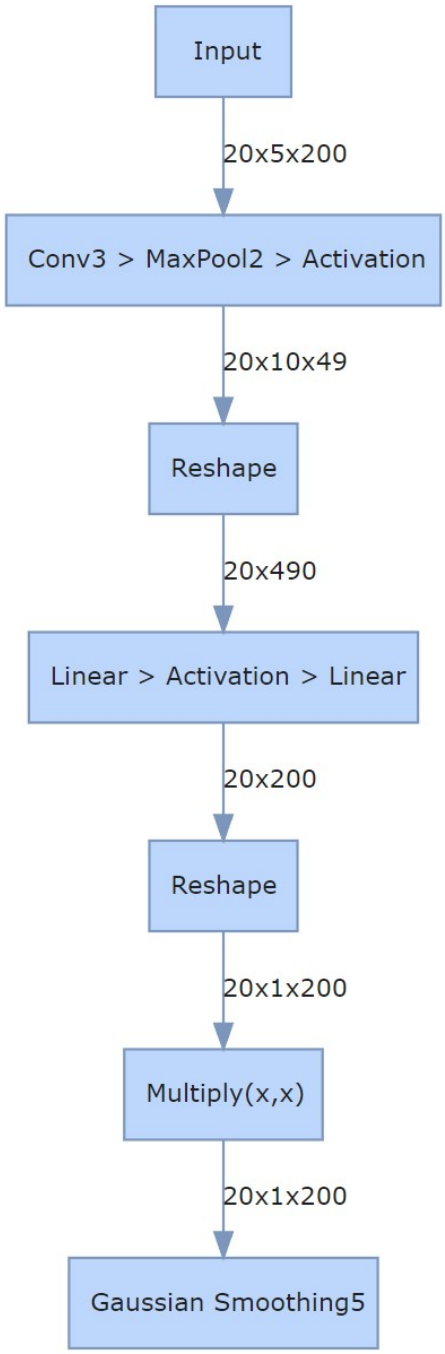


Figure 4.12: Graph of the 1D PIML model with 20 checkpoints and 5 training features as an input. Please note that the number of checkpoints was slightly different for each $M_{\odot}$ based on the available training dataset.

Figure 4.13: ML model training loss evolution for the 19 $M_{\odot}$: *Top* is the actual loss of the model that consists of a combined L_1 and KS loss components, *Middle* is the L_1 loss component, *Bottom* is the $\mathrm{KS}_{\mathrm{stat}}$ loss component. The training early-stop condition is based on the KS_{stat} of the validation dataset. Thus, the training halted based on the KS_{stat} dropping to ~ 0.1.

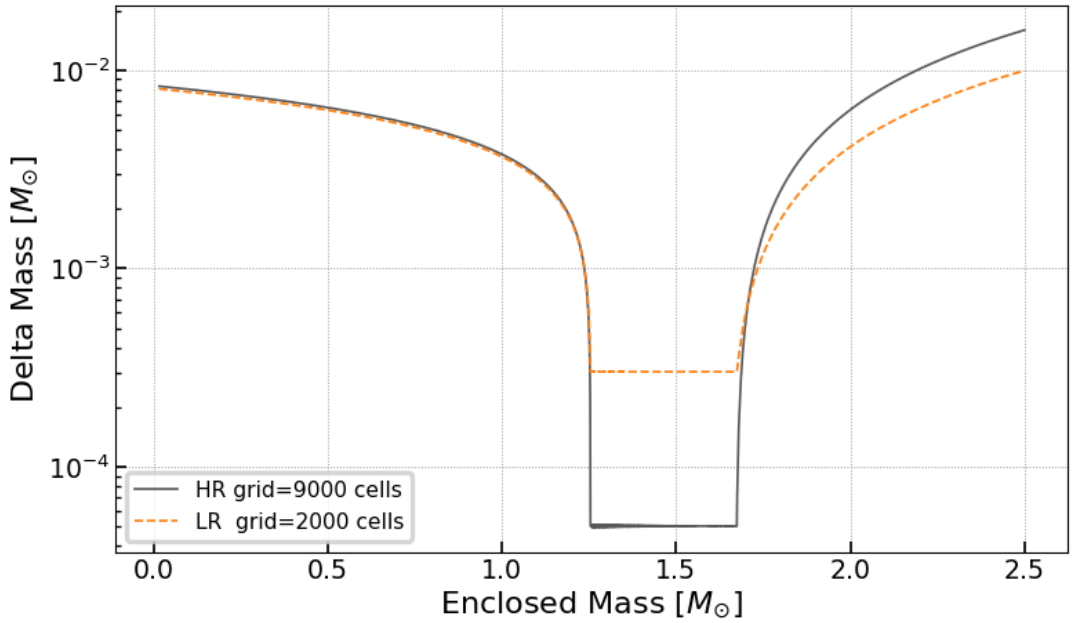


Figure 4.14: Comparison of the high-resolution (HR) and low-resolution (LR) Lagrangian grids used for all of our simulations. They are static, and non-uniform to include 3 regions: medium PNS, high shifting-convection (between the red lines), and the low outer. The fine-zoning extended between approximately 1.2 and 1.7 $M_\odot$ due to matter accreting onto the PNS, and the region below the shock staying relatively thin ($\sim 0.01\,M_\odot$). That region was 8700 cells for HR, and 1700 cells for LR. This amounted to the LR fine-zoning being $3 \times 10^{-4}\ M_\odot$, which was 6 times worse than our HR runs.

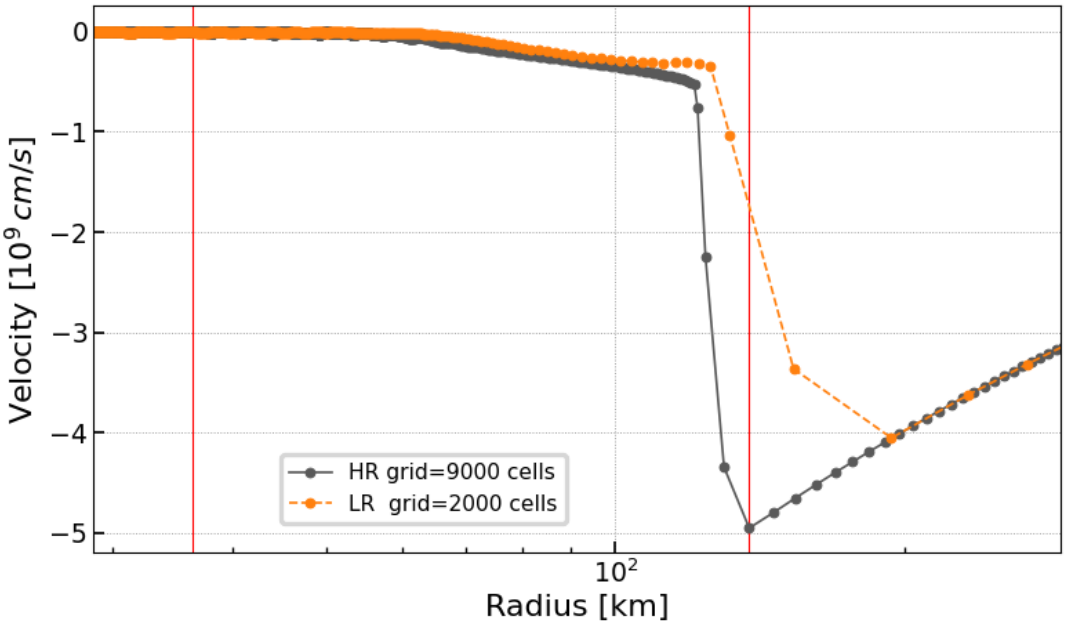


Figure 4.15: Velocity profiles presenting the shock resolution for the baseline (no ML) 17 $M_\odot$ at 200 ms for high-resolution grid with 9000 cells (solid gray) and low-resolution grid with 2000 cells (dashed orange) simulations. The red lines show the *convective* region between the PNS (left line) and the shock (right line). Such low resolution in the grid=2000 cells case results in high oscillations of the shock radius, as the jump over a single cell results in 10s of *km* shifts.

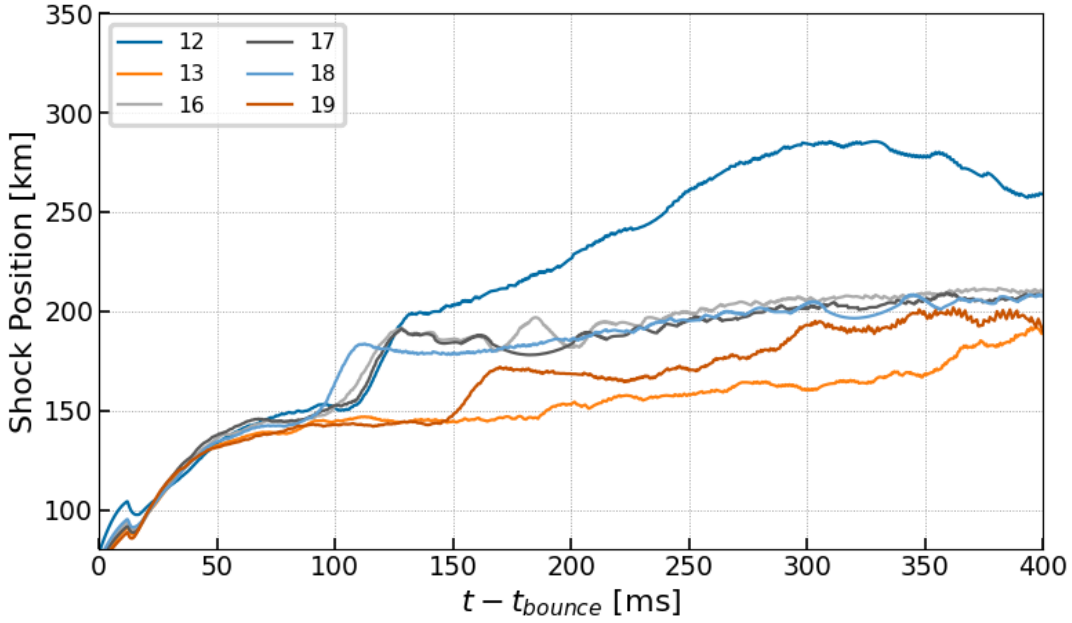


Figure 4.16: Shock evolution of the low-resolution baseline simulations *without* ML using [12,13,16,17,18,19] $M_{\odot}$ progenitors with a grid of 2000 cells. The shock has moved out further in comparison to the high-resolution runs from Figure 4.5, converging near 200 *km* for most models. That said, none of the 1D COLLAPSO1D simulations have exploded, and no positive velocities have been generated. Numerical noise has been filtered out of the curves by applying a Savitzky–Golay filter.

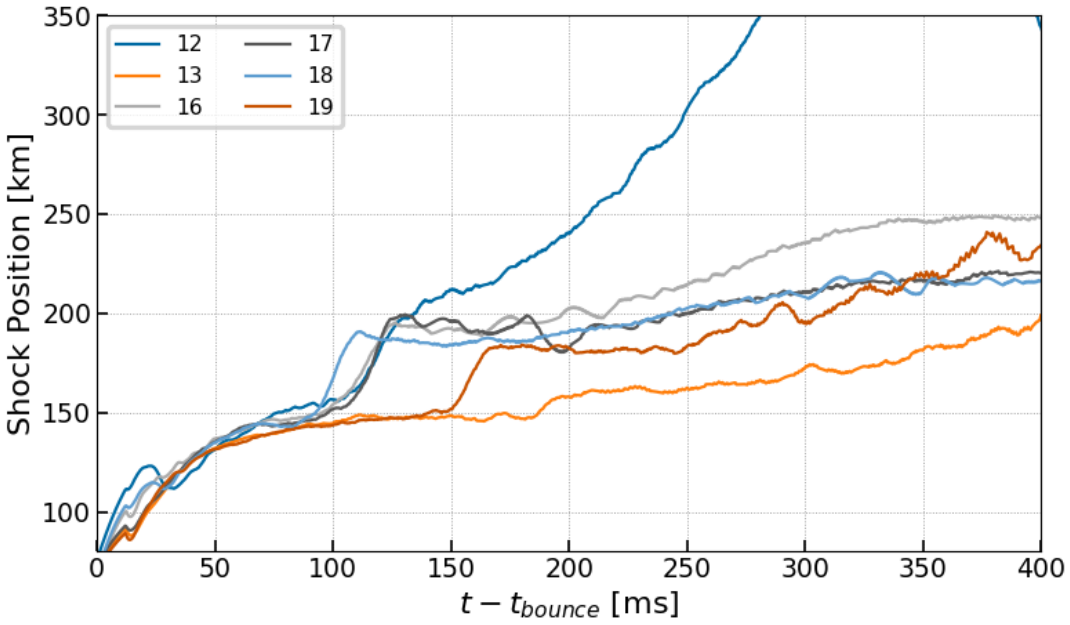


Figure 4.17: Shock evolution of the low-resolution simulations *with* ML using [12,13,16,17,18,19] $M_\odot$ progenitors with a grid of 2000 cells. The shock has moved out further in comparison to the high-resolution (HR) runs from Figure 4.6. In this case, only the 12 $M_\odot$ has exploded. Numerical noise has been filtered out of the curves by applying a Savitzky–Golay filter.

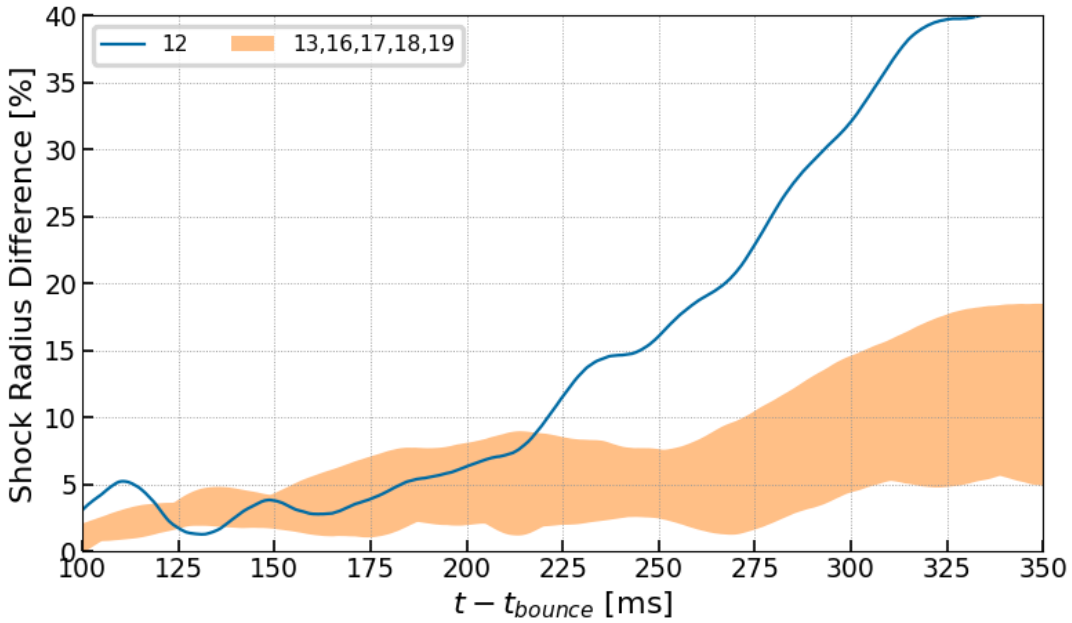


Figure 4.18: Cumulative comparison of the shock position increase for exploding 12 $M_\odot$ and non-exploding [13,16,17,18,19] $M_\odot$. The differences between the latter models are minor, while exhibiting a similar growth behavior with large oscillatory dynamic due to low-resolution of the shock; hence they are combined. The dashed line for the 12 $M_\odot$ is the back-propagating shock.

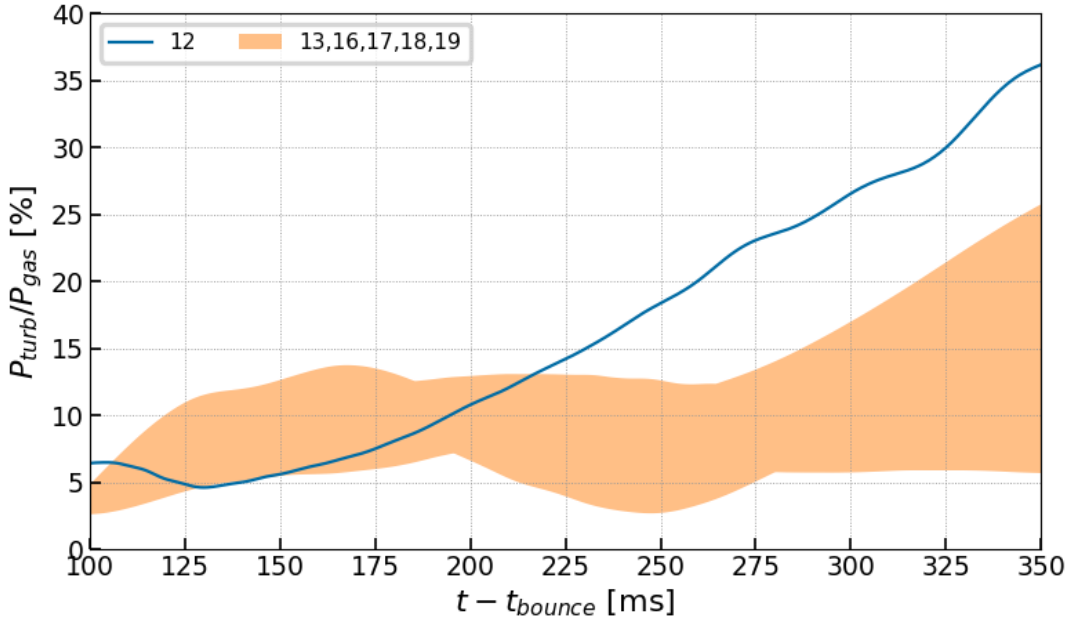


Figure 4.19: Evolution of the maximum predicted P_{turb}/P_{gas} at each checkpoint for exploding 12 $M_{\odot}$ and non-exploding [13,16,17,18,19] $M_{\odot}$. The former corresponds well with the training dataset (Burrows et al., 2020) and the shock-position dynamic from Figure 4.17. For the 12 $M_{\odot}$ the ratio does drop similar to Figure 4.11 at a later time ($> 400\,ms$). However, the low-resolution simulations become very unstable at that point, with very large shock oscillations; hence the later times are not plotted.

Chapter 5

Conclusion

5.1 Paper Summary

The work for this thesis was structured as follows: develop a pipeline to design ML models for turbulence in astrophysics, test learning of 3D turbulent behavior with ML, and test the applicability of using ML to retrieve 3D turbulent-convective dynamics in 1D CCSN simulations, along with developing a wrapper to use ML inferencing in legacy and modern Fortran codes.

ML framework for Astrophysics, Sapsan, was developed to introduce industry-leading tools. It includes compatibility with both scikit-learn and PyTorch for model design, MLflow for experiment tracking, Docker to ease the development and distribution of the ML models, and Streamlit-based GUI web interface for ML models demon-

strations. Sapsan aims to reduce the boilerplate code and streamline ML development, giving more opportunities to focus on physics-informed methods. The pipeline also includes analytical and plotting modules, as well as turbulence SGS models. These tools help to gauge model accuracy and performance and to aid model interpretability. A thorough documentation was written and distributed along with Sapsan via GitHub[1], open-sourced for the community.

Modeling 3D MHD turbulence was the next step for this project to test the capabilities of ML. The training dataset from Mösta et al. (2015) was the highest-resolution DNS simulation of the dynamically evolving CCSN turbulence available to us, which was the best setup for ML model design iteration. As part of this step, we developed our own CNN-based PIML method with a custom loss function to capture 3D MHD turbulence. Typically, magnetic fields pose severe complications when studying turbulence. However, for the ML model, strong magnetic fields proved to be a defining feature of turbulent growth through MRI. As a result, ML proved to have relatively fast convergence on GPUs and consistent prediction accuracy with the unseen checkpoints from the dataset. Furthermore, the method was generalizable to other applications, such as 2D and HD regimes.

Introduction of 3D dynamics into 1D CCSN simulations was done to test the ML approach of reproducing the effects of turbulence/convection in reduced dimensionality. The ML model was adopted from the 3D MHD turbulence case, with minor

[1] github.com/pikarpov-LANL/Sapsan

adjustments to adapt for 1D turbulence prediction. It was trained on the global 3D CCSN models by Burrows et al. (2020). Despite the comparatively low fidelity within the hot bubble between the PNS and the shock, the training dataset exhibited larger-scale convective motion, often referred to as *turbulence* in such global simulations. It is important to note that only the additional pressure term was modeled. The ML model showed a notable increase in the shock radius and an explosion in the 1D models, comparable to other publications. Since this project required performing ML inferencing at simulation runtime, a PyTorch wrapper was developed to integrate ML into our test Fortran code COLLAPSO1D. That said, the wrapper was built to be generalized for easy integration within any Fortran code, independent of dimensionality. It is open-sourced, documented, and available to the community via GitHub[2].

5.2 Challenges in Machine Learning

As promising ML-based turbulence models are, there are challenges to be aware of. These are not unique to astrophysics and are actively being worked on in other fields and the industry. These problems can be generally consolidated into three active areas of research.

Generalization remains an open question in ML, which has been referenced heavily throughout this thesis. A trained model heavily depends on the training dataset;

[2] github.com/pikarpov-LANL/COLLAPSO1D

hence, more data should yield better results. Then, we can be sure that great interpolation performance can be achieved. On the other hand, predicting on new unseen data can present a challenge for extrapolation (Arjovsky, 2020), e.g., the late-stage of a physical system evolved beyond the training dataset. In the context of turbulence, embedded physical constraints can help enforce the model to stay within physical boundaries. However, if there is no question about the need to enforce conservation laws and invariants, what about the unknown physical conditions that affect turbulent behavior? In CCSN, an ML model can be trained on idealistic DNS setups, e.g., isotropic and stationary turbulence, hoping to adapt to a significantly more complex simulation. It becomes a non-trivial problem on what physics to introduce as a part of the ML algorithms and how to test the extrapolation of the trained turbulence models, leading us to the next point.

Uncertainty quantification is a significantly understudied area in ML. Typically, ML model performance is assessed solely based on the *loss-function* applied during training, i.e., a measurement of model accuracy based on the target feature (at least in supervised ML). The underlying uncertainties of the training data, whether experimentally or computationally obtained, are often ignored (except for Bayesian-based algorithms). A trustworthy model uncertainty would aid in prediction robustness and evaluation. For a more detailed review of various levels of uncertainty and its quantification, see Duraisamy et al. (2019).

Interpretability of ML models is an active area of research in the industry and physics community. Indeed, ML models are often treated as a black-box. It is not trivial to represent the model-learned relationships between specific features. While we can learn new physics from ML, it is challenging to define due to its convoluted nature through many layers. The problem is further exacerbated by the popularity and flexibility advantages of deep learning (DL) neural networks, consisting of many network layers (e.g., Krizhevsky et al., 2012; He et al., 2016). Training on Big Data is beneficial to cover a larger physical domain, but it further complicates interpretability. Extracting the learned physical relationships to assess predictive validity becomes incredibly challenging. To remedy this, one might want to rely solely on embedding hard-constraints physical constraints and use shallow neural networks. However, in return, that will inhibit the flexibility of the models and will not aid the opportunity to learn new turbulence physics. The balance between ML model interpretability, flexibility, and generalizability can be challenging.

5.3 Overlook

Accurately predicting turbulence with ML requires care, especially concerning its long-term temporal dependencies. It is important to account for intermittency and the general chaotic nature of turbulence. That said, there have been significant advancements in physics-informed ML methods, as presented in this work and by the commu-

nity, specifically looking at the problems of turbulent growth and decay.

In CCSN, ML opens up many exciting opportunities for the future of turbulence and convection modeling. These processes can impact the explodability to produce successful CCSNe, as we observed in our results, consistent with the literature. Besides the pressure term, our work can be applied to capture other contributions of turbulence with ML, e.g., energy flux and its dissipation to heat, and introduce them into either 1D or low-fidelity 3D CCSN simulations. The next step for this project would be to test the contribution of those terms and evaluate the role of turbulence in CCSNe comprehensively. In addition, our ML paradigm would be tested independently and in conjunction with analytical models within 3D global CCSN simulations, inferenced at runtime. Such an approach will further extend the recovery of the missing SGS structure in the more cost-effective global 3D simulations.

Bibliography

Dmitry Alexeev and Markus Hrywniak. alexeedm/pytorch-fortran: Version v0.3, November 2022. URL https://doi.org/10.5281/zenodo.7304774.

Anders Andreassen, Ilya Feige, Christopher Frye, and Matthew D Schwartz. Junipr: a framework for unsupervised machine learning in particle physics. *The European Physical Journal C*, 79:1–24, 2019.

Martin Arjovsky. *Out of distribution generalization in machine learning*. PhD thesis, New York University, 2020.

W. David Arnett, Casey Meakin, Maxime Viallet, Simon W. Campbell, John C. Lattanzio, and Miroslav Mocák. BEYOND MIXING-LENGTH THEORY: A STEP TOWARD 321d. *The Astrophysical Journal*, 809(1):30, aug 2015. doi: 10.1088/0004-637x/809/1/30. URL https://doi.org/10.1088/0004-637x/809/1/30.

W. Baade and F. Zwicky. On Super-novae. *Proceedings of the National Academy of Science*, 20(5):254–259, May 1934. doi: 10.1073/pnas.20.5.254.

Dalya Baron. Machine learning in astronomy: a practical overview, 2019.

E. Baron, H. A. Bethe, G. E. Brown, J. Cooperstein, and S. Kahana. Type ii supernovae from prompt explosions. *Phys. Rev. Lett.*, 59:736–739, Aug 1987. doi: 10.1103/PhysRevLett.59.736. URL https://link.aps.org/doi/10.1103/PhysRevLett.59.736.

Andrea Beck and Marius Kurz. A perspective on machine learning methods in turbulence modeling. *GAMM-Mitteilungen*, 44(1):e202100002, 2021a. doi: https://doi.org/10.1002/gamm.202100002. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/gamm.202100002.

Andrea Beck and Marius Kurz. A perspective on machine learning methods in turbulence modeling. *GAMM-Mitteilungen*, 44(1):e202100002, 2021b. doi: https://doi.org/10.1002/gamm.202100002. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/gamm.202100002.

Willy Benz. An introduction to computational methods in hydrodynamics. In C. B. de Loore, editor, *Late Stages of Stellar Evolution Computational Methods in Astrophysical Hydrodynamics*, pages 258–312, Berlin, Heidelberg, 1991. Springer Berlin Heidelberg. ISBN 978-3-540-46921-6.

Andrey Beresnyak. Mhd turbulence. *Living Reviews in Computational Astrophysics*, 5 (1), Sep 2019. ISSN 2365-0524. doi: 10.1007/s41115-019-0005-8. URL http://dx.doi.org/10.1007/s41115-019-0005-8.

Andrey Beresnyak and Alex Lazarian. Mhd turbulence, turbulent dynamo and applications. *Magnetic Fields in Diffuse Media*, page 163–226, Oct 2014. ISSN 2214-7985. doi: 10.1007/978-3-662-44625-6_8. URL http://dx.doi.org/10.1007/978-3-662-44625-6_8.

D. Besnard, F. H. Harlow, R. M. Rauenzahn, and C. Zemach. Turbulence transport equations for variable-density turbulence and their relationship to two-field models. *U.S. Department of Energy: Office of Scientific and Technical Information*, 6 1992. doi: 10.2172/7271399. URL https://www.osti.gov/biblio/7271399.

H. A. Bethe and J. R. Wilson. Revival of a stalled supernova shock by neutrino heating. *ApJ*, 295:14–23, August 1985. doi: 10.1086/163343.

Tom Beucler, Michael Pritchard, Stephan Rasp, Jordan Ott, Pierre Baldi, and Pierre Gentine. Enforcing Analytic Constraints in Neural-Networks Emulating Physical Systems. *arXiv e-prints*, art. arXiv:1909.00912, September 2019.

Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag, Berlin, Heidelberg, 2006. ISBN 0387310738.

S. I. Blinnikov, N. V. Dunina-Barkovskaya, and D. K. Nadyozhin. Equation of State of a Fermi Gas: Approximations for Various Degrees of Relativism and Degeneracy. *Astrophysical Journal Supplement Series*, 106:171, September 1996. doi: 10.1086/192334.

John M. Blondin, Anthony Mezzacappa, and Christine DeMarino. Stability of Standing Accretion Shocks, with an Eye toward Core-Collapse Supernovae. *The Astrophysical Journal*, 584(2):971–980, February 2003. doi: 10.1086/345812.

John M. Blondin, Anthony Mezzacappa, and Christine DeMarino. Stability of standing accretion shocks, with an eye toward core-collapse supernovae. *The Astrophysical Journal*, 584(2):971, feb 2003. doi: 10.1086/345812. URL https://dx.doi.org/10.1086/345812.

Mathis Bode, Michael Gauding, Zeyu Lian, Dominik Denker, Marco Davidovic, Konstantin Kleinheinz, Jenia Jitsev, and Heinz Pitsch. Using physics-informed enhanced super-resolution generative adversarial networks for subfilter modeling in turbulent reactive flows. *Proceedings of the Combustion Institute*, 38(2):2617–2625, 2021. ISSN 1540-7489. doi: https://doi.org/10.1016/j.proci.2020.06.022. URL https://www.sciencedirect.com/science/article/pii/S1540748920300481.

Axel Brandenburg and A Lazarian. Astrophysical hydromagnetic turbulence. *Space Science Reviews*, 178(2):163–200, 2013.

A. Burrows and D. Vartanyan. Core-collapse supernova explosion theory. *Nature*, 589(7840):29–39, jan 2021. doi: 10.1038/s41586-020-03059-w. URL https://doi.org/10.1038%2Fs41586-020-03059-w.

A. Burrows, D. Vartanyan, J. C. Dolence, M. A. Skinner, and D. Radice. Crucial Physical Dependencies of the Core-Collapse Supernova Mechanism. *Space Science Reviews*, 214(1):33, February 2018. doi: 10.1007/s11214-017-0450-9.

Adam Burrows and John Hayes. Pulsar recoil and gravitational radiation due to asymmetrical stellar collapse and explosion. *Phys. Rev. Lett.*, 76:352–355, Jan 1996. doi: 10.1103/PhysRevLett.76.352. URL https://link.aps.org/doi/10.1103/PhysRevLett.76.352.

Adam Burrows, John Hayes, and Bruce A. Fryxell. On the Nature of Core-Collapse Supernova Explosions. *ApJ*, 450:830, September 1995. doi: 10.1086/176188.

Adam Burrows, David Radice, David Vartanyan, Hiroki Nagakura, M. Aaron Skinner, and Joshua C. Dolence. The overarching framework of core-collapse supernova explosions as revealed by 3D FORNAX simulations. *Monthly Notices of the Royal Astronomical Society*, 491(2):2715–2735, January 2020. doi: 10.1093/mnras/stz3223.

Daniele Carati, Grégoire S Winckelmans, and Hervé Jeanmart. On the modelling of the subgrid-scale and filtered-scale stress tensors in large-eddy simulation. *Journal of Fluid Mechanics*, 441:119–138, 2001.

Giuseppe Carleo, Ignacio Cirac, Kyle Cranmer, Laurent Daudet, Maria Schuld, Naftali Tishby, Leslie Vogt-Maranto, and Lenka Zdeborová. Machine learning and the physical sciences. *Reviews of Modern Physics*, 91(4):045002, October 2019. doi: 10.1103/RevModPhys.91.045002.

Stirling A. Colgate and Richard H. White. The hydrodynamic behavior of supernovae explosions. *The Astrophysical Journal*, 143:626, 1966.

Sean M. Couch and Evan P. O'Connor. High-resolution three-dimensional simulations of core-collapse supernovae in multiple progenitors. *The Astrophysical Journal*, 785 (2):123, apr 2014. doi: 10.1088/0004-637X/785/2/123. URL https://dx.doi.org/10.1088/0004-637X/785/2/123.

Sean M. Couch and Christian D. Ott. Revival of the stalled core-collapse supernova shock triggered by precollapse asphericity in the progenitor star. *The Astrophysical Journal Letters*, 778(1):L7, oct 2013. doi: 10.1088/2041-8205/778/1/L7. URL https://dx.doi.org/10.1088/2041-8205/778/1/L7.

Sean M. Couch and Christian D. Ott. The Role of Turbulence in Neutrino-driven Core-collapse Supernova Explosions. *ApJ*, 799(1):5, January 2015. doi: 10.1088/0004-637X/799/1/5.

Sean M. Couch, MacKenzie L. Warren, and Evan P. O'Connor. Simulating turbulence-aided neutrino-driven core-collapse supernova explosions in one dimension. *The Astrophysical Journal*, 890(2):127, feb 2020. doi: 10.3847/1538-4357/ab609e. URL https://doi.org/10.3847/1538-4357/ab609e.

Inc Databricks. Mlflow. https://github.com/mlflow/mlflow, 2020.

Nicholas A. Denissen, Jimmy Fung, Jon M. Reisner, and Malcolm J. Andrews. Implementation and validation of the bhr turbulence model in the flag hydrocode. *U.S. Department of Energy: Office of Scientific and Technical Information*, 8 2012. doi: 10.2172/1050005. URL https://www.osti.gov/biblio/1050005.

Nicholas A. Denissen, Bertrand Rollin, Jon M. Reisner, and Malcolm J. Andrews. The Tilted Rocket Rig: A Rayleigh–Taylor Test Case for RANS Models1. *Journal of Fluids Engineering*, 136(9), 07 2014. ISSN 0098-2202. doi: 10.1115/1.4027776. URL https://doi.org/10.1115/1.4027776. 091301.

Paul E. Dimotakis, Richard C. Miake-Lye, and Dimitris A. Papantoniou. Structure and dynamics of round turbulent jets. *The Physics of Fluids*, 26(11):3185–3192, 11 1983. ISSN 0031-9171. doi: 10.1063/1.864090. URL https://doi.org/10.1063/1.864090.

Karthik Duraisamy, Gianluca Iaccarino, and Heng Xiao. Turbulence modeling in the age of data. *Annual Review of Fluid Mechanics*, 51(1):357–377, 2019. doi: 10.1146/annurev-fluid-010518-040547. URL https://doi.org/10.1146/annurev-fluid-010518-040547.

Gregory Eyink, Ethan Vishniac, Cristian Lalescu, Hussein Aluie, Kalin Kanov, Kai Bürger, Randal Burns, Charles Meneveau, and Alexander Szalay. Flux-freezing

breakdown in high-conductivity magnetohydrodynamic turbulence. *Nature*, 497 (7450):466–469, May 2013. ISSN 1476-4687. doi: 10.1038/nature12128. URL https://doi.org/10.1038/nature12128.

Claude-André Faucher-Giguère and Victoria M. Kaspi. Birth and evolution of isolated radio pulsars. *The Astrophysical Journal*, 643(1):332, may 2006. doi: 10.1086/501516. URL https://dx.doi.org/10.1086/501516.

T. Foglizzo, P. Galletti, L. Scheck, and H.-Th. Janka. Instability of a stalled accretion shock: Evidence for the advective-acoustic cycle. *The Astrophysical Journal*, 654 (2):1006, jan 2007. doi: 10.1086/509612. URL https://dx.doi.org/10.1086/509612.

Chris L. Fryer. Mass Limits For Black Hole Formation. *ApJ*, 522(1):413–418, September 1999. doi: 10.1086/307647.

Chris L. Fryer. Neutron Star Kicks from Asymmetric Collapse. *Astrophysical Journal Letters*, 601(2):L175–L178, February 2004. doi: 10.1086/382044.

Chris L. Fryer and Alexander Heger. Core-Collapse Simulations of Rotating Stars. *ApJ*, 541(2):1033–1050, October 2000. doi: 10.1086/309446.

Chris L. Fryer and Michael S. Warren. Modeling Core-Collapse Supernovae in Three Dimensions. *Astrophysical Journal Letters*, 574(1):L65–L68, July 2002. doi: 10.1086/342258.

Chris L. Fryer, Aimee L. Hungerford, and Gabriel Rockefeller. Supernova Explosions:. Understanding Mixing. *International Journal of Modern Physics D*, 16(6):941–981, January 2007. doi: 10.1142/S0218271807010523.

Chris L. Fryer, Sydney Andrews, Wesley Even, Alex Heger, and Samar Safi-Harb. Parameterizing the Supernova Engine and Its Effect on Remnants and Basic Yields. *ApJ*, 856(1):63, March 2018. doi: 10.3847/1538-4357/aaaf6f.

Christopher L. Fryer and Patrick A. Young. Late-Time Convection in the Collapse of a 23 M_{solar} Star. *The Astrophysical Journal*, 659(2):1438–1448, April 2007a. doi: 10.1086/513003.

Christopher L. Fryer and Patrick A. Young. Late-Time Convection in the Collapse of a 23 M_{solar} Star. *The Astrophysical Journal*, 659(2):1438–1448, April 2007b. doi: 10.1086/513003.

M. Germano. Turbulence: the filtering approach. *Journal of Fluid Mechanics*, 238: 325–336, may 1992. ISSN 0022-1120. doi: 10.1017/S0022112092001733. URL https://www.cambridge.org/core/product/identifier/S0022112092001733/type/journal_article.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *Communications of the ACM*, 63(11):139–144, 2020.

Philipp Grete. *Large eddy simulations of compressible magnetohydrodynamic turbulence*. PhD thesis, Max-Planck-Institute for Solar System Research, Lindau, February 2017.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

Alexander Heger and S. E. Woosley. Nucleosynthesis and Evolution of Massive Metal-free Stars. *ApJ*, 724(1):341–373, November 2010. doi: 10.1088/0004-637X/724/1/341.

Marc Herant, Willy Benz, W. Raphael Hix, Chris L. Fryer, and Stirling A. Colgate. Inside the Supernova: A Powerful Convective Engine. *The Astrophysical Journal*, 435:339, November 1994a. doi: 10.1086/174817.

Marc Herant, Willy Benz, W. Raphael Hix, Chris L. Fryer, and Stirling A. Colgate. Inside the Supernova: A Powerful Convective Engine. *ApJ*, 435:339, November 1994b. doi: 10.1086/174817.

G. Hobbs, D. R. Lorimer, A. G. Lyne, and M. Kramer. A statistical study of 233 pulsar proper motions. *Monthly Notices of the Royal Astronomical Society*, 360(3):974–992, 07 2005. ISSN 0035-8711. doi: 10.1111/j.1365-2966.2005.09087.x. URL https://doi.org/10.1111/j.1365-2966.2005.09087.x.

Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Universal approximation of an unknown mapping and its derivatives using multilayer feedforward networks. *Neural Networks*, 3(5):551–560, 1990. ISSN 0893-6080. doi: https://doi.org/10.1016/0893-6080(90)90005-6. URL https://www.sciencedirect.com/science/article/pii/0893608090900056.

Matthew Hutson. Artificial intelligence faces reproducibility crisis. *Science*, 359 (6377):725–726, 2018. ISSN 0036-8075. doi: 10.1126/science.359.6377.725. URL https://science.sciencemag.org/content/359/6377/725.

Liliya Imasheva, Hans-Thomas Janka, and Achim Weiss. Parametrizations of thermal bomb explosions for core-collapse supernovae and ^{56}Ni production. *Monthly Notices of the Royal Astronomical Society*, 518(2):1818–1839, January 2023. doi: 10.1093/mnras/stac3239.

H. T. Janka and E. Mueller. Neutrino heating, convection, and the mechanism of Type-II supernova explosions. *Physics Reports*, 306:167, February 1996.

Hans-Thomas Janka. Explosion mechanisms of core-collapse supernovae. *Annual Review of Nuclear and Particle Science*, 62(1):407–451, 2012. doi: 10.1146/annurev-nucl-102711-094901. URL https://doi.org/10.1146/annurev-nucl-102711-094901.

Hans-Thomas Janka. *Neutrino-Driven Explosions*, pages 1095–1150. Springer International Publishing, Cham, 2017. ISBN 978-3-319-21846-5. doi: 10.1007/978-3-319-21846-5_109. URL https://doi.org/10.1007/978-3-319-21846-5_109.

Hans-Thomas Janka, Tobias Melson, and Alexander Summa. Physics of core-collapse supernovae in three dimensions: A sneak preview. *Annual Review of Nuclear and Particle Science*, 66(1):341–375, 2016. doi: 10.1146/annurev-nucl-102115-044747. URL https://doi.org/10.1146/annurev-nucl-102115-044747.

Javier Jiménez. Computing high-Reynolds-number turbulence: will simulations ever replace experiments?*. *Journal of Turbulence*, 4(1):22, June 2003. doi: 10.1088/1468-5248/4/1/022.

Jyeshtharaj B. Joshi, Krishnaswamy Nandakumar, Ashwin W. Patwardhan, Arun K. Nayak, Vishnu Pareek, Monica Gumulya, Chunliang Wu, Nitin Minocha, Eshita Pal, Mukesh Kumar, Vishal Bhusare, Shashank Tiwari, Dhiraj Lote, Chaitanya Mali, Ameya Kulkarni, and Sarang Tamhankar. 2 - computational fluid dynamics. In Jyeshtharaj B. Joshi and Arun K. Nayak, editors, *Advances of Computational Fluid Dynamics in Nuclear Reactor Design and Safety Assessment*, Woodhead Publishing Series in Energy, pages 21–238. Woodhead Publishing, 2019. ISBN 978-0-08-102337-2. doi: https://doi.org/10.1016/B978-0-08-102337-2.00002-X. URL https://www.sciencedirect.com/science/article/pii/B978008102337200002X.

Platon I. Karpov, Iskandar Sitdikov, Chengkun Huang, and Chris L. Fryer. Sapsan: Framework for supernovae turbulence modeling with machine learning. *Journal of Open Source Software*, 6(67):3199, 2021. doi: 10.21105/joss.03199. URL https://doi.org/10.21105/joss.03199.

Platon I. Karpov, Chengkun Huang, Iskandar Sitdikov, Chris L. Fryer, Stan Woosley, and Ghanshyam Pilania. Physics-informed machine learning for modeling turbulence in supernovae. *The Astrophysical Journal*, 940(1):26, nov 2022. doi: 10.3847/1538-4357/ac88cc. URL https://dx.doi.org/10.3847/1538-4357/ac88cc.

Ryan King, Oliver Hennigh, Arvind Mohan, and Michael Chertkov. From deep to physics-informed learning of turbulence: Diagnostics, 2018.

Ryan N. King, Peter E. Hamlington, and Werner J. A. Dahm. Autonomic closure for turbulence simulations. *Phys. Rev. E*, 93:031301, Mar 2016. doi: 10.1103/PhysRevE.93.031301. URL https://link.aps.org/doi/10.1103/PhysRevE.93.031301.

Sergey Kolesnikov. Accelerated dl r&d. https://github.com/catalyst-team/catalyst, 2018.

A. Kolmogorov. The Local Structure of Turbulence in Incompressible Viscous Fluid for Very Large Reynolds' Numbers. *Akademiia Nauk SSSR Doklady*, 30:301–305, January 1941.

Jan Kremer, Kristoffer Stensbo-Smidt, Fabian Gieseke, Kim Steenstrup Pedersen, and Christian Igel. Big universe, big data: machine learning and image analysis for astronomy. *IEEE Intelligent Systems*, 32(2):16–22, 2017.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012. URL https://proceedings.neurips.cc/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf.

J. M. Lattimer and F. Douglas Swesty. A generalized equation of state for hot, dense matter. *Nuclear Physics A*, 535:331–376, December 1991. doi: 10.1016/0375-9474(91)90452-C.

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

Christian Ledig, Lucas Theis, Ferenc Huszár, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, et al. Photo-realistic single image super-resolution using a generative adversarial network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4681–4690, 2017.

Jinu Lee, Sangseung Lee, and Donghyun You. Deep learning approach in multi-scale prediction of turbulent mixing-layer. *arXiv e-prints*, art. arXiv:1809.07021, September 2018.

Eric J. Lentz, Stephen W. Bruenn, W. Raphael Hix, Anthony Mezzacappa, O. E. Bronson Messer, Eirik Endeve, John M. Blondin, J. Austin Harris, Pedro Marronetti, and Konstantin N. Yakunin. Three-dimensional core-collapse supernova simulated using a 15 $m_{\odot}$ progenitor. *The Astrophysical Journal Letters*, 807(2):L31, jul 2015. doi: 10.1088/2041-8205/807/2/L31. URL https://dx.doi.org/10.1088/2041-8205/807/2/L31.

Yi Li, Eric Perlman, Minping Wan, Yunke Yang, Charles Meneveau, Randal Burns, Shiyi Chen, Alexander Szalay, and Gregory Eyink. A public turbulence database cluster and applications to study Lagrangian evolution of velocity increments in turbulence. *Journal of Turbulence*, 9:N31, January 2008. doi: 10.1080/14685240802376389.

Zongyi Li, Hongkai Zheng, Nikola Kovachki, David Jin, Haoxuan Chen, Burigede Liu, Kamyar Azizzadenesheli, and Anima Anandkumar. Physics-informed neural operator for learning partial differential equations. *ICLR 2022 Conference*, 2021. doi: 10.48550/ARXIV.2111.03794. URL https://arxiv.org/abs/2111.03794.

D. K. Lilly. On the application of the eddy viscosity concept in the Inertial sub-range of turbulence. *NCAR Manuscript 123*, January 1966.

Julia Ling, Andrew Kurzawski, and Jeremy Templeton. Reynolds averaged turbulence modelling using deep neural networks with embedded invariance. *Journal of Fluid Mechanics*, 807:155–166, 2016. doi: 10.1017/jfm.2016.615.

Shewen Liu, Charles Meneveau, and Joseph Katz. On the properties of similarity subgrid-scale models as deduced from measurements in a turbulent jet. *Journal of Fluid Mechanics*, 275:83–119, 1994a. doi: 10.1017/S0022112094002296.

Shewen Liu, Charles Meneveau, and Joseph Katz. On the properties of similarity subgrid-scale models as deduced from measurements in a turbulent jet. *Journal of Fluid Mechanics*, 275:83–119, 1994b. doi: 10.1017/S0022112094002296.

Quintin A. Mabanta and Jeremiah W. Murphy. How turbulence enables core-collapse supernova explosions. *The Astrophysical Journal*, 856(1):22, mar 2018. doi: 10.3847/1538-4357/aaaec7. URL https://dx.doi.org/10.3847/1538-4357/aaaec7.

Quintin A. Mabanta, Jeremiah W. Murphy, and Joshua C. Dolence. Convection-aided explosions in one-dimensional core-collapse supernova simulations. i. technique and

validation. *The Astrophysical Journal*, 887(1):43, dec 2019. doi: 10.3847/1538-4357/ab4bcc. URL https://doi.org/10.3847%2F1538-4357%2Fab4bcc.

Frank J. Massey. The kolmogorov-smirnov test for goodness of fit. *Journal of the American Statistical Association*, 46(253):68–78, 1951. ISSN 01621459. URL http://www.jstor.org/stable/2280095.

Tobias Melson, Hans-Thomas Janka, Robert Bollig, Florian Hanke, Andreas Marek, and Bernhard Müller. Neutrino-driven Explosion of a 20 Solar-mass Star in Three Dimensions Enabled by Strange-quark Contributions to Neutrino-Nucleon Scattering. *The Astrophysical Journal Letters*, 808(2):L42, August 2015. doi: 10.1088/2041-8205/808/2/L42.

Mark Miesch, William Matthaeus, Axel Brandenburg, Arakel Petrosyan, Annick Pouquet, Claude Cambon, Frank Jenko, Dmitri Uzdensky, James Stone, Steve Tobias, Juri Toomre, and Marco Velli. Large-eddy simulations of magnetohydrodynamic turbulence in heliophysics and astrophysics. *Space Science Reviews*, 194(1–4):97–137, Jul 2015. ISSN 1572-9672. doi: 10.1007/s11214-015-0190-7. URL http://dx.doi.org/10.1007/s11214-015-0190-7.

Arvind Mohan, Don Daniel, Michael Chertkov, and Daniel Livescu. Compressed Convolutional LSTM: An Efficient Deep Learning framework to Model High Fidelity 3D Turbulence. *arXiv e-prints*, art. arXiv:1903.00033, February 2019.

Arvind T. Mohan, Nicholas Lubbers, Daniel Livescu, and Michael Chertkov. Embedding Hard Physical Constraints in Neural Network Coarse-Graining of 3D Turbulence. *arXiv e-prints*, art. arXiv:2002.00021, January 2020.

Philipp Mösta, Christian D. Ott, David Radice, Luke F. Roberts, Erik Schnetter, and Roland Haas. A large-scale dynamo and magnetoturbulence in rapidly rotating core-collapse supernovae. *Nature*, 528(7582):376–379, December 2015. doi: 10.1038/nature15755.

B. Müller and H. Th. Janka. Non-radial instabilities and progenitor asphericities in core-collapse supernovae. *Monthly Notices of the Royal Astronomical Society*, 448(3):2141–2174, April 2015. doi: 10.1093/mnras/stv101.

Jeremiah W. Murphy and Casey Meakin. A GLOBAL TURBULENCE MODEL FOR NEUTRINO-DRIVEN CONVECTION IN CORE-COLLAPSE SUPERNOVAE. *The Astrophysical Journal*, 742(2):74, nov 2011. doi: 10.1088/0004-637x/742/2/74. URL https://doi.org/10.1088%2F0004-637x%2F742%2F2%2F74.

Jeremiah W. Murphy, Joshua C. Dolence, and Adam Burrows. The dominance of neutrino-driven convection in core-collapse supernovae. *The Astrophysical Journal*, 771(1):52, jun 2013. doi: 10.1088/0004-637X/771/1/52. URL https://dx.doi.org/10.1088/0004-637X/771/1/52.

Kevin P. Murphy. Machine learning: A probabilistic perspective. In *Machine Learning: A Probabilistic Perspective*, chapter 14.4.3, pages 492–493. The MIT Press, 2012.

Bernhard Müller. Hydrodynamics of core-collapse supernovae and their progenitors. *Living Reviews in Computational Astrophysics*, 6(1), jun 2020. doi: 10.1007/s41115-020-0008-5. URL https://doi.org/10.1007%2Fs41115-020-0008-5.

Bernhard Müller, Tobias Melson, Alexander Heger, and Hans-Thomas Janka. Supernova simulations from a 3D progenitor model – Impact of perturbations and evolution of explosion properties. *Monthly Notices of the Royal Astronomical Society*, 472(1):491–513, 08 2017. ISSN 0035-8711. doi: 10.1093/mnras/stx1962. URL https://doi.org/10.1093/mnras/stx1962.

Hiroki Nagakura, Adam Burrows, David Radice, and David Vartanyan. Towards an understanding of the resolution dependence of Core-Collapse Supernova simulations. *Monthly Notices of the Royal Astronomical Society*, 490(4):4622–4637, December 2019. doi: 10.1093/mnras/stz2730.

C.-Y. Ng and Roger W. Romani. Birth kick distributions and the spin-kick correlation of young pulsars. *The Astrophysical Journal*, 660(2):1357, may 2007. doi: 10.1086/513597. URL https://dx.doi.org/10.1086/513597.

Nobuya Nishimura, Tomoya Takiwaki, and Friedrich-Karl Thielemann. The r-process nucleosynthesis in the various jet-like explosions of magnetorotational core-collapse supernovae. *The Astrophysical Journal*, 810(2):109, sep 2015. doi: 10.1088/0004-637x/810/2/109. URL https://doi.org/10.1088%2F0004-637x%2F810%2F2%2F109.

Ken'ichi Nomoto, Nozomu Tominaga, Hideyuki Umeda, Chiaki Kobayashi, and Keiichi Maeda. Nucleosynthesis yields of core-collapse supernovae and hypernovae, and galactic chemical evolution. *Nuclear Physics A*, 777:424–458, October 2006. doi: 10.1016/j.nuclphysa.2006.05.008.

Martin Obergaulinger, Pablo Cerdá-Durán, Ewald Müller, and Miguel Angel Aloy. Semi-global simulations of the magneto-rotational instability in core collapse supernovae. *Astronomy & Astrophysics*, 498(1):241–271, 2009.

Evan O'Connor and Christian D Ott. A new open-source code for spherically symmetric stellar collapse to neutron stars and black holes. *Classical and Quantum Gravity*, 27(11):114103, may 2010. doi: 10.1088/0264-9381/27/11/114103. URL https://dx.doi.org/10.1088/0264-9381/27/11/114103.

Evan O'Connor, Robert Bollig, Adam Burrows, Sean Couch, Tobias Fischer, Hans-Thomas Janka, Kei Kotake, Eric J Lentz, Matthias Liebendörfer, O E Bronson Messer, Anthony Mezzacappa, Tomoya Takiwaki, and David Vartanyan. Global comparison of core-collapse supernova simulations in spherical symmetry. *Journal of Physics G: Nuclear and Particle Physics*, 45(10):104001, sep 2018. doi: 10.1088/1361-6471/aadeae. URL https://dx.doi.org/10.1088/1361-6471/aadeae.

Evan P. O'Connor and Sean M. Couch. Exploring fundamentally three-dimensional phenomena in high-fidelity simulations of core-collapse supernovae. *The Astrophysical Journal*, 865(2):81, sep 2018. doi: 10.3847/1538-4357/aadcf7. URL https://dx.doi.org/10.3847/1538-4357/aadcf7.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019. URL http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

A. Perego, M. Hempel, C. Fröhlich, K. Ebinger, M. Eichler, J. Casanova, M. Liebendörfer, and F. K. Thielemann. PUSHing Core-collapse Supernovae to Explosions in Spherical Symmetry I: the Model and the Case of SN 1987A. *ApJ*, 806(2):275, June 2015. doi: 10.1088/0004-637X/806/2/275.

Stephen B. Pope. *Turbulent Flows*. Cambridge University Press, 2000. doi: 10.1017/CBO9780511840531.

Gavin D. Portwood, Peetak P. Mitra, Mateus Dias Ribeiro, Tan Minh Nguyen, Balasubramanya T. Nadiga, Juan A. Saenz, Michael Chertkov, Animesh Garg, Anima

Anandkumar, Andreas Dengel, Richard Baraniuk, and David P. Schmidt. Turbulence forecasting via Neural ODE. *arXiv e-prints*, art. arXiv:1911.05180, November 2019.

Y. Z. Qian and S. E. Woosley. Nucleosynthesis in Neutrino-driven Winds. I. The Physical Conditions. *ApJ*, 471:331, November 1996. doi: 10.1086/177973.

David Radice, Sean M Couch, and Christian D Ott. Implicit large eddy simulations of anisotropic weakly compressible turbulence with application to core-collapse supernovae. *Computational Astrophysics and Cosmology*, 2(1), Aug 2015. ISSN 2197-7909. doi: 10.1186/s40668-015-0011-0. URL http://dx.doi.org/10.1186/s40668-015-0011-0.

David Radice, Christian D. Ott, Ernazar Abdikamalov, Sean M. Couch, Roland Haas, and Erik Schnetter. Neutrino-driven convection in core-collapse supernovae: High-resolution simulations. *The Astrophysical Journal*, 820(1):76, mar 2016. doi: 10.3847/0004-637X/820/1/76. URL https://dx.doi.org/10.3847/0004-637X/820/1/76.

David Radice, Ernazar Abdikamalov, Christian D Ott, Philipp Mösta, Sean M Couch, and Luke F Roberts. Turbulence in core-collapse supernovae. *Journal of Physics G: Nuclear and Particle Physics*, 45(5):053003, Apr 2018. ISSN 1361-6471. doi: 10.1088/1361-6471/aab872. URL http://dx.doi.org/10.1088/1361-6471/aab872.

Carl Edward Rasmussen, Christopher KI Williams, et al. *Gaussian processes for machine learning*, volume 1. Springer, 2006.

Serena Repetto, Melvyn B. Davies, and Steinn Sigurdsson. Investigating stellar-mass black hole kicks. *Monthly Notices of the Royal Astronomical Society*, 425(4):2799–2809, 10 2012. ISSN 0035-8711. doi: 10.1111/j.1365-2966.2012.21549.x. URL https://doi.org/10.1111/j.1365-2966.2012.21549.x.

Lewis Fry Richardson. *Weather prediction by numerical process*. University Press, 1922.

Armin Ronacher. Click, 2021. URL https://click.palletsprojects.com/.

Shawn G. Rosofsky and E. A. Huerta. Artificial neural network subgrid models of 2d compressible magnetohydrodynamic turbulence. *Physical Review D*, 101(8), Apr 2020. ISSN 2470-0029. doi: 10.1103/physrevd.101.084024. URL http://dx.doi.org/10.1103/PhysRevD.101.084024.

Shawn G Rosofsky, Hani Al Majed, and E A Huerta. Applications of physics informed neural operators. *Machine Learning: Science and Technology*, 4(2):025022, may 2023. doi: 10.1088/2632-2153/acd168. URL https://dx.doi.org/10.1088/2632-2153/acd168.

Alexander A. Schekochihin and Steven C. Cowley. *Turbulence and Magnetic Fields in Astrophysical Plasmas*, pages 85–115. Springer Netherlands, Dordrecht, 2007. ISBN 978-1-4020-4833-3. doi: 10.1007/978-1-4020-4833-3_6. URL https://doi.org/10.1007/978-1-4020-4833-3_6.

Wolfram Schmidt. Large eddy simulations in astrophysics. *Living Reviews in Computational Astrophysics*, 1(1):1–69, 2015.

U. Schumann. Realizability of Reynolds-stress turbulence models. *Physics of Fluids*, 20(5):721–725, May 1977. doi: 10.1063/1.861942.

M. Aaron Skinner, Joshua C. Dolence, Adam Burrows, David Radice, and David Vartanyan. Fornax: A flexible code for multiphysics astrophysical simulations. *The Astrophysical Journal Supplement Series*, 241(1):7, feb 2019. doi: 10.3847/1538-4365/ab007f. URL https://dx.doi.org/10.3847/1538-4365/ab007f.

Edward A Spiegel. A generalization of the mixing-length theory of turbulent convection. *The Astrophysical Journal*, 138:216, 1963.

A. W. Steiner, M. Hempel, and T. Fischer. Core-collapse Supernova Equations of State Based on Neutron Star Observations. *ApJ*, 774(1):17, September 2013. doi: 10.1088/0004-637X/774/1/17.

Tuguldur Sukhbold, T. Ertl, S. E. Woosley, Justin M. Brown, and H. T. Janka. Core-collapse Supernovae from 9 to 120 Solar Masses Based on Neutrino-powered Explosions. *ApJ*, 821(1):38, April 2016. doi: 10.3847/0004-637X/821/1/38.

Brendan D. Tracey, Karthikeyan Duraisamy, and Juan J. Alonso. *A Machine Learning Strategy to Assist Turbulence Model Development*. 53rd AIAA Aerospace Sciences Meeting, 2015. doi: 10.2514/6.2015-1287. URL https://arc.aiaa.org/doi/abs/10.2514/6.2015-1287.

Adrien Treuille. Turn python scripts into beautiful ml tools. *Towards Data Science*, 8, October 2019.

K. A. van Riper. General relativistic hydrodynamics and the adiabatic collapse of stellar cores. *ApJ*, 232:558–571, September 1979. doi: 10.1086/157314.

Antonia Vojtekova, Maggie Lieu, Ivan Valtchanov, Bruno Altieri, Lyndsay Old, Qifeng Chen, and Filip Hroch. Learning to denoise astronomical images with u-nets. *Monthly Notices of the Royal Astronomical Society*, 503(3):3204–3215, nov 2020. doi: 10.1093/mnras/staa3567. URL https://doi.org/10.1093%2Fmnras%2Fstaa3567.

Rui Wang, Karthik Kashinath, Mustafa Mustafa, Adrian Albert, and Rose Yu. Towards Physics-informed Deep Learning for Turbulent Flow Prediction. *arXiv e-prints*, art. arXiv:1911.08655, November 2019.

Xintao Wang, Ke Yu, Shixiang Wu, Jinjin Gu, Yihao Liu, Chao Dong, Chen Change Loy, Yu Qiao, and Xiaoou Tang. Esrgan: Enhanced super-resolution generative adversarial networks, 2018.

A. Wongwathanarat, H. Th. Janka, and E. Müller. Three-dimensional neutrino-driven supernovae: Neutron star kicks, spins, and asymmetric ejection of nucleosynthesis products. *Physics Reports*, 552:A126, April 2013. doi: 10.1051/0004-6361/201220636.

S. E. Woosley, J. R. Wilson, G. J. Mathews, R. D. Hoffman, and B. S. Meyer. The r-Process and Neutrino-heated Supernova Ejecta. *ApJ*, 433:229, September 1994. doi: 10.1086/174638.

S. E. Woosley, Norbert Langer, and Thomas A. Weaver. The Presupernova Evolution and Explosion of Helium Stars That Experience Mass Loss. *ApJ*, 448:315, July 1995. doi: 10.1086/175963.

S. E. Woosley, S. Wunsch, and M. Kuhlen. Carbon ignition in type ia supernovae: An analytic model. *The Astrophysical Journal*, 607(2):921, jun 2004. doi: 10.1086/383530. URL https://dx.doi.org/10.1086/383530.

Stan Woosley and Thomas Janka. The physics of core-collapse supernovae. *Nature Physics*, 1(3):147–154, 2005.

Jin-Long Wu, Heng Xiao, and Eric Paterson. Physics-informed machine learning approach for augmenting turbulence models: A comprehensive framework. *Phys. Rev. Fluids*, 3:074602, Jul 2018. doi: 10.1103/PhysRevFluids.3.074602. URL https://link.aps.org/doi/10.1103/PhysRevFluids.3.074602.

Jin-Long Wu, Karthik Kashinath, Adrian Albert, Dragos Chirila, Prabhat, and Heng Xiao. Enforcing statistical constraints in generative adversarial networks for modeling chaotic dynamical systems. *Journal of Computational Physics*, 406:109209, April 2020. doi: 10.1016/j.jcp.2019.109209.

Yudong Yao, Henry Chan, Subramanian Sankaranarayanan, Prasanna Balaprakash, Ross J Harder, and Mathew J Cherukara. Autophasenn: unsupervised physics-aware deep learning of 3d nanoscale bragg coherent diffraction imaging. *npj Computational Materials*, 8(1):124, 2022.

Patrick A. Young and Chris L. Fryer. Uncertainties in Supernova Yields. I. One-dimensional Explosions. *ApJ*, 664(2):1033–1044, August 2007. doi: 10.1086/518081.

Weiwei Zhang, Linyang Zhu, Yilang Liu, and Jiaqing Kou. Machine learning methods for turbulence modeling in subsonic flows over airfoils. *arXiv e-prints*, art. arXiv:1806.05904, June 2018.

Ze Jia Zhang and Karthikeyan Duraisamy. *Machine Learning Methods for Data-Driven Turbulence Modeling*. 22nd AIAA Computational Fluid Dynamics Conference, 2015. doi: 10.2514/6.2015-2460. URL https://arc.aiaa.org/doi/abs/10.2514/6.2015-2460.

Alice Zheng and Amanda Casari. *Feature engineering for machine learning: principles and techniques for data scientists*. " O'Reilly Media, Inc.", 2018.