NEURAL MODELLING OF ASTROPHYSICAL GAS

NEURAL MODELLING OF ASTROPHYSICAL GAS

By EMMETT MCFARLANE, B.Sc.

A Thesis Submitted to the School of Graduate Studies in Partial Fulfillment of the Requirements for the Degree Master of Science

McMaster University

MASTER OF SCIENCE (2024)

Hamilton, Ontario, Canada (Physics and Astronomy)

TITLE:	Neural Modelling of Astrophysical Gas
AUTHOR:	Emmett McFarlane B.Sc. (Honours Mathematics and Physics), McMaster University, Hamilton, Canada
SUPERVISOR:	Dr. James Wadsley
NUMBER OF PAGES:	xvi, 95

Abstract

We investigate the abilities of neural models to model astrophysical gases, addressing limitations in traditional numerical solvers such as energy bottlenecks, resolution effects, grid anisotropy effects, and strict time-stepping constraints. In particular, we focus on simulations of turbulent flows and thermochemical networks in the interstellar medium (ISM).

We employ the chaotic Kuramoto-Sivashinsky (KS) equation as a one-dimensional testbed for neural network architectures commonly employed in simulations of three-dimensional hydrodynamical turbulence. We benchmark a wide range of state-of-the-art neural architectures. Our experiments demonstrate that hierarchical context aggregation, residual connections, and group equivariance play a critical role in capturing faithful dynamics and spectral properties of turbulent flows.

Additionally, we model the thermochemical evolution of astrophysical gas using a residual neural network (ResNet) trained on data generated by the `CHIMES` code. The network predicts the evolution of chemical abundances and thermal states across a range of densities, temperatures, and metallicities, effectively integrating over large timesteps to mitigate the stiffness issues of conventional solvers.

Our findings suggest that modern deep learning methods may provide a viable and efficient alternative to traditional numerical solvers for astrophysical simulations.

Acknowledgements

I would like to thank my advisor, Dr. James Wadsley, who shared my optimism for the future of machine learning, and provided invaluable guidance and support throughout every stage of this research. I am also grateful to my committee members: Dr. Ralph Pudritz, whom I first had the pleasure of meeting in Lyon, France, and later benefited from excellent teaching in his special topics course; and Dr. Erik Srensen, who laid the foundation of my machine learning knowledge during my undergraduate years – In the following years, I got to work alongside him in teaching machine learning to the next generation of undergraduates. Thanks to my labmates: Padraic and Hector, with whom I shared amazing experiences in Lyon, and Nick, who shared my enthusiasm for tinkering with large language models. I would like to thank my partner, Etain, for her support and encouragement, and my parents for always believing in me. I am indebted to my international collaborators, Dr. Tobias Buck, Dr. Wolfram Schmidt, and Dr. Christoph Federrath, for their open exchange of ideas that made this work possible.

Table of Contents

List of Figures

List of Tables

Notation, Definitions, and Abbreviations

Notation

ρ	Density
$\mathbf{v}$	Velocity vector
t_λ	Lyapunov Timescale
P	Pressure
E	Energy
T	Temperature
n_i	Number density of species i
k_{ij}	Rate coefficient for reaction between species i and j
Λ	Cooling function
Γ	Heating function

Z	Metallicity
$\mathcal{M}$	Mach number
σ	Standard deviation
θ	Neural network parameters
L	Loss function
η	Learning rate
$*$	Convolution operation
$\star$	Correlation operation (here, used interchangeably with convolution)
T_g	Group action of element g
L_u	Left-regular representation of group G
g, h	Elements of a group

Abbreviations

ML	Machine Learning
ANN	Artificial Neural Network
IGM	Intergalactic Medium
ISM	Interstellar Medium
GMC	Giant Molecular Cloud

SFR	Star Formation Rate
CFL	Courant-Friedrichs-Lewy Condition
CIE	Collisional Ionization Equilibrium
CNN	Convolutional Neural Network
DGM	Deep Galerkin Method
G-CNN	Group Equivariant Convolutional Neural Network
PINN	Physics-Informed Neural Network
ReLU	Rectified Linear Unit
MSE	Mean Squared Error
RMSE	Root Mean Square Error
GPU	Graphics Processing Unit
ODE	Ordinary Differential Equation
PDE	Partial Differential Equation
PDF	Probability Distribution Function
SPH	Smoothed Particle Hydrodynamics

Chapter 1

Introduction

The interstellar medium (ISM) was first discovered over 100 years ago, when Hartmann (1904) measured stationary CaII lines while observing the Delta Orionis star system. These stationary lines implied that the absorption did not originate from the star's atmosphere, but rather matter along the star's line of sight. Since then, our understanding and knowledge of astrophysical gas a grown enormously.

A large portion of the mass budget of baryons in galaxies is in gas (Tielens, 2005). The origin and evolution of observed galaxies and the associated gas are tightly coupled via cyclic process which exchange gas and dust through various heating, cooling and feedback processes. Gas clouds in the ISM collapse to form stars while the stars eject feedback gas back into the ISM (Tielens, 2005).

Observing these systems provides limited data on their origin and evolution, as the processes that drive them operate on a huge range of times scales and length scales. Due to the impossibility of performing direct experiments on such systems, astrophysicists often turn to simulations to validate theory on their detailed evolution.

Modern simulations use highly parallel numerical methods to solve equations governing the astrophysical gas (such as `Gasoline`, `AREPO`, `Athena`, `KROME` and other codes discussed in 1.2.2). These methods involve carefully crafted time-stepping schemes with strict stability and convergence criteria, and resolution requirements to capture all the detailed physical processes across all the relevant scales, which can create extreme computational requirements. For any practical simulation, the accuracy relative to the true physics is always limited by the resolution and the numerical method chosen.

Astrophysical equations (ODEs for chemistry and cooling, PDEs for hydrodynamics) can involve steep gradients, and are governed by stiff equations that create highly nonlinear trajectories. These properties mean highly restrictive time-stepping constraints such as the Courant (CFL) condition (for hydrodynamics) or very fast reactions (for chemistry), must be considered with numerical solvers. However, in the case of stiff equations particularly, the physical system often spends a lot of time at equilibrium and so the overall system can evolve slowly, suggesting new approaches could be more efficient.

Recent advancements in machine learning have shown promise to outperform traditional solver codes in both accuracy and speed (Branca and Pallottini, 2023; Rosofsky and Huerta, 2020; Stachenfeld et al., 2021; Sulzer and Buck, 2023). Neural networks have a time and memory complexity that depends only on the architecture an the dimensions and depth of the layers in the network. In particular, they can represent highly non-linear functions, offering promise to evolve physical dynamical systems without being limited by the same time step constraints that traditional solver codes require. In addition, neural networks (in particular, autoencoders) can

compress data and automatically discover efficient representations in latent space (Hinton and Salakhutdinov, 2006). This too could lead to savings in memory and time. These potential savings apply to a many areas in numerical physics. In astrophysics specifically, we expect that hydrodynamics solvers and chemical network solvers can benefit greatly from machine learning approaches.

In the remainder of the introduction, I describe astrophysical scenarios where machine learning methods could be helpful for the reasons presented above. I also describe current standard numerical treatments and their limitations. In Chapter 2, I give an overview of machine learning methods and model architectures, discussing their mathematical prescriptions and use cases. In Chapter 3, I showcase our experiments applying such methods to simulate the time-evolution of the astrophysical systems (and simple analogues) discussed in this introduction. The final chapter reviews the results we find, as well as future directions for training neural models to evolve astrophysical simulations.

1.1 Astrophysical Gas

Gas is ubiquitous throughout the universe, much of it in the form of plasma (Ferriere, 2001). In principle, we must then consider not only hydrodynamics but magneto-hydrodynamics, ionization sources, and detailed chemistry. In addition, there are complex and numerically difficult-to-resolve processes such as turbulence, accretion, cooling, and the formation of bound objects that then feed back into the resolved medium. The interstellar medium (ISM) is a key focus in astrophysics and all these important processes are represented there.

1.1.1 Modeling the Interstellar Medium

The ISM consists of gas and plasma between stars within galaxies. This environment is a source of material for giant molecular clouds (GMCs), which serve as formation sites for new stars and planets (Tielens, 2005). The ISM features a wide range of physical processes, including gas hydrodynamics, plasma magneto-hydrodynamics, radiative transfer, heating and cooling processes, and non-equilibrium chemistry.

The self-gravity of GMC clouds can be strong and the Mach numbers very high, leading to large dynamic ranges in the density of the ISM as a whole. Padoan et al. (1997), conducted simulations of hydrodynamic turbulence, studying the density PDF and its relationship to the Mach number of the gas. The mass distribution was dominated by only a small fraction of the volume. Kritsuk et al. (2007) fit the width of the log-normal density PDF with the mach number of the gas following the scaling law $\sigma^2 = ln(1 + b^2\mathcal{M}^2)$. At higher mach numbers (for example, ~ 6) the width of the density distribution can span 6 or more orders of magnitude (Kritsuk et al., 2007).

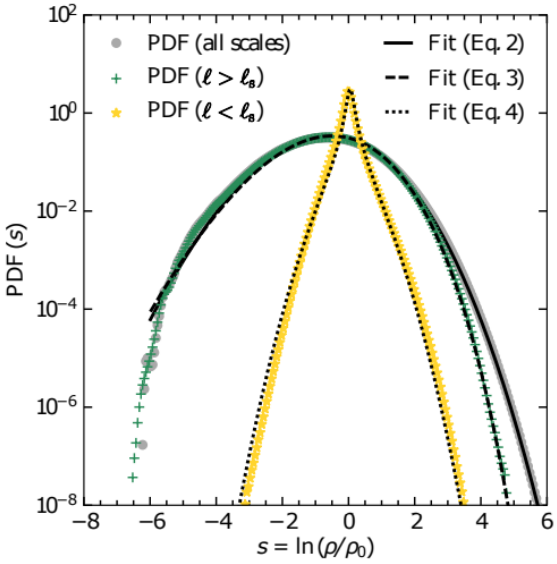


Figure 1.1: Density PDF of the Fourier-filtered density field, above and below the sonic scale l_s, the transition from supersonic to subsonic turbulence (Federrath et al., 2020).

To resolve both the rarified gas and the dense gas, one must use a large number of discretized cells (or particles, depending on the method) which comes at great computational cost. The low-viscosity and high-density setting of the ISM, often containing supersonic shocks, is challenging and expensive to model, and typically forces a limited resolution which will ultimately compromise accuracy. Thus, there is a constant need in the field for faster and more efficient ways to model ISM turbulence both at and below the grid scale.

1.1.2 Modeling Heating and Cooling

The thermal evolution of an astrophysical gas can be determined by solving heating and cooling functions, which depend on temperature, density, and metallicity (including the details of the chemical composition) (Sutherland and Dopita, 1993b). The

short timescales at play within the chemical reaction networks (including their thermal evolution) in the ISM cause them to often be the limiting factor in the simulation costgs. For example, the line cooling rate for metals scales as $\Lambda_{\rm metal} \propto n_H n_e$, meaning very dense regions require short timescales to accurately resolve their evolution. On the other hand, most of the gas sits close to an equilibrium temperature so that the actual time to see temperature changes is typically quite long. Cooling rates depend strongly on the ionization state of the gas, with ionized gas cooling rates orders of magnitude greater than that for neutral gases at $T < 10^4$ K (Gnedin and Hollon, 2012). In collisional ionization equilibrium (CIE), cooling and heating functions can be parameterized using temperature, density, and metallicity (Sutherland and Dopita, 1993b).

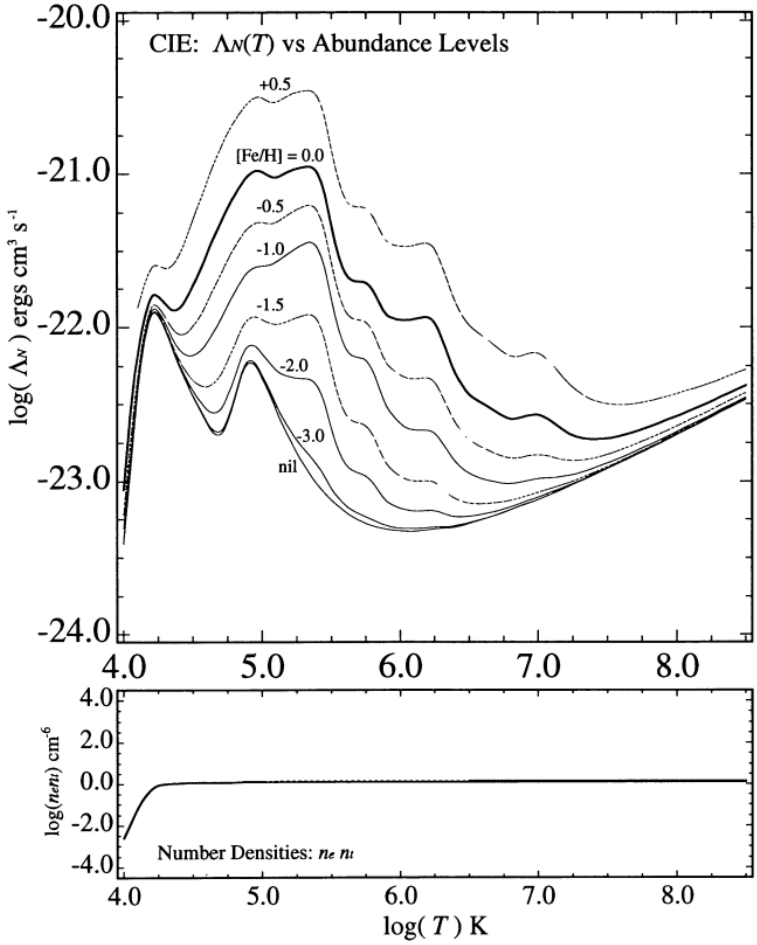


Figure 1.2: Normalized CIE cooling functions as a function of metallicity, with more metal-rich environments showing greater cooling rates (Sutherland and Dopita, 1993a).

1.2 Modeling Hydrodynamical Turbulence

Understanding and accurately modeling turbulence is essential for faithful modelling of star formation, galaxy evolution, and the dynamics of the ISM. The fundamental equations of hydrodynamics are the Euler equations, which describe the conservation

of mass, momentum, and energy in a fluid:

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{v}) = 0 \tag{1.2.1}$$

$$\frac{\partial (\rho \mathbf{v})}{\partial t} + \nabla \cdot (\rho \mathbf{v}\mathbf{v} + P\mathbf{I}) = 0 \tag{1.2.2}$$

$$\frac{\partial E}{\partial t} + \nabla \cdot [(E + P)\mathbf{v}] = 0 \tag{1.2.3}$$

where ρ is density, $\mathbf{v}$ is velocity, P is pressure, and E is total energy density.

Turbulence arises when these equations are solved in regimes of high Reynolds number, where inertial forces dominate over viscous forces. In astrophysical contexts, the Reynolds numbers are often extremely high, leading to fully developed turbulence across a wide range of scales.

Turbulence plays an important role in regulating the density distribution and the flow of gas throughout the ISM. In galaxy simulations, turbulence below the resolved grid scale can suppress star formation rate by combating gravitational collapse (Schmidt and Federrath, 2011).

1.2.1 Kolmogorov Theory

The classical theory of turbulence, developed by Kolmogorov (1941), predicts a cascade of energy from large scales to small scales. In the inertial range, where neither energy injection nor dissipation dominates, the energy spectrum follows a power law:

$$E(k) \propto k^{-5/3} \tag{1.2.4}$$

where k is the wavenumber. This -5/3 slope is a characteristic of fully developed, incompressible turbulence.

However, astrophysical turbulence often involves compressible fluids, magnetic fields, and self-gravity, complicating this picture. For instance, in supersonic turbulence, which is common in GMCs, the energy spectrum can steepen to $E(k) \propto k^{-2}$ (Burgers, 1948). A log-normal relationship between the mach number $\mathcal{M}$ and the density ρ has been demonstrated in simulations (Bauer and Springel, 2012; Lemaster and Stone, 2008; Squire and Hopkins, 2017), and accurately resolving both ends of this density distribution is of great importance in an astrophysical context.

However, accurately capturing turbulence in numerical simulations remains challenging. It requires resolving a wide range of scales, from the energy injection scale to the dissipation scale. Moreover, numerical dissipation can affect the inertial range, and different numerical schemes often produce widely different results, especially in the highly compressible regime relevant to many astrophysical problems (Schmidt and Federrath, 2011; Schmidt et al., 2006).

1.2.2 Numerical Methods and Limitations

Particle methods

Smoothed Particle Hydrodynamics (SPH) is a commonly used representative Lagrangian numerical method, where fluid elements move with the flow. We use SPH as a representative Lagragian method as it shares constraints with similar methods (e.g. AREPO). SPH codes have been widely used to study turbulence in astrophysical settings. SPH works by approximating fluid elements as particles, which interact with each other through a kernel function (Monaghan, 1992).

In SPH, the fluid is represented by a set of particles with positions $\mathbf{r}_i$, velocities $\mathbf{v}_i$, and masses m_i. The density at any point is estimated by:

$$\rho(\mathbf{r}) = \sum_j m_j W(\mathbf{r} - \mathbf{r}_j, h) \tag{1.2.5}$$

where $W(\mathbf{r}, h)$ is a smoothing kernel (typically a cubic spline function or similar) and h is the smoothing length.

The equations of motion for the particles are derived from the Lagrangian form of the fluid equations:

$$\frac{d\mathbf{v}_i}{dt} = -\sum_j m_j \left(\frac{P_i}{\rho_i^2} + \frac{P_j}{\rho_j^2} + \Pi_{ij} \right) \nabla_i W_{ij} \tag{1.2.6}$$

where P_i is the pressure of particle i, and Π_{ij} is an artificial viscosity term to handle shocks:

$$\Pi_{ij} = \begin{cases} \frac{-\alpha c_{ij} \mu_{ij} + \beta \mu_{ij}^2}{\rho_{ij}}, & \mathbf{v}_{ij} \cdot \mathbf{r}_{ij} < 0 \\ 0, & \mathbf{v}_{ij} \cdot \mathbf{r}_{ij} \geq 0 \end{cases} \tag{1.2.7}$$

The energy equation is:

$$\frac{du_i}{dt} = \frac{P_i}{\rho_i^2} \sum_j m_j \mathbf{v}_{ij} \cdot \nabla a_i W_{ij} \tag{1.2.8}$$

where u_i is the specific internal energy.

SPH naturally handles large density contrasts and complex geometries by allowing the discretized units to move with the flow of gas, making it an enticing choice for astrophysical contexts. However, SPH can fail to represent the full inertial range of

the turbulent cascade, suppressing energy levels expected near the grid scale. (Price, 2012).

The accuracy of SPH codes can be limited by issues such as excessive artificial viscosity and gradient errors (Bauer and Springel, 2012). SPH, in its default implementation, introduces spurious pressure forces in regions with steep density gradients (Agertz et al., 2007). Agertz et al. (2007) showed that SPH has difficulties in modeling multiphase flows and fluid instabilities (i.e. Kelvin-Helmholtz and Reyleigh-Taylor). Despite promising fixes to SPH (Wadsley et al., 2008) through better modelling of turbulent diffusive processes, the need to constantly identify and resolve such artifacts is a recurring problem with existing numerical methods.

Common SPH codes used for astrophysical scenarios include `GADGET` (Springel, 2005), `Gasoline` (Wadsley et al., 2017), and `ChaNGa` (Jetley et al., 2008).

Grid Methods

Grid-based methods solve the fluid equations by discretizing space into a mesh of cells. The fluid variables are defined at fixed points in space, typically cell centers or faces. Grid codes evolve these variables in time by calculating fluxes between neighboring cells.

The discretized form of the Euler equations in a finite-volume scheme can be written as:

$$\frac{\partial \mathbf{U}_i}{\partial t} + \frac{1}{\Delta x}(\mathbf{F}_{i+1/2} - \mathbf{F}_{i-1/2}) = 0 \tag{1.2.9}$$

where $\mathbf{U}_i$ is the vector of conserved quantities in cell i, Δx is the cell width, and $\mathbf{F}_{i\pm1/2}$ are the fluxes at the cell interfaces.

A primary limitation of grid methods is that they explicitly advect material between cells. This process is diffusive at levels that can exceed physical viscosity and conduction by large margins and also depends explicitly on the velocity. In this sense they are not Galilean-invariant. This is a key physical property we might seek to recover with alternative methods. The practical impacts of these errors are velocity-dependent smearing effects close to the resolution scale which can couple badly to cooling and other key small scale processes.

Popular grid-based astrophysical fluid dynamics codes include: `RAMSES` (Teyssier, 2002), `Athena++` (Stone et al., 2020), `ENZO` (Bryan et al., 2014), `FLASH` (Fryxell et al., 2000), and `CHOLLA` (Schneider and Robertson, 2015).

Additionally, there are hybrid methods, such as `AREPO` (Weinberger et al., 2020), which uses a moving Voronoi mesh to accurately resolve the gas distribution. These methods have similar limitations.

1.2.3 Limitations of Hydrodynamics Codes

These state-of-the-art numerical methods suffer many long-standing drawbacks, including energy bottlenecks, the Gibbs phenomenon, grid anisotropy, CFL constraints. Additionally, traditional codes have expensive scaling laws, typically as $N_{1D}^{(}4/3)$ where N_{1D} is the number of cells per dimension. For example, a state-of-the-art hydrodynamical simulation on a $10,000^3$ grid by Federrath et al. (2020) required $\sim 65,000$ compute cores on the SuperMUC supercomputer, consuming ~ 50 million core hours to simulate 9 turbulent crossing times.

Despite ad hoc fixes to solver codes (for example, treatments for SPH with improved modelling of turbulent diffusive processes in SPH (Wadsley et al., 2008)), the

need to constantly identify and resolve such artifacts is a recurring problem with existing numerical methods.

Bottleneck Effects

Bottleneck effects refer to an artificial buildup of energy at small scales near the dissipation range in numerical turbulence simulations.

As shown in Figure 1.3, both grid-based and SPH codes exhibit bottleneck effects to varying degrees in subsonic turbulence simulations, where the energy spectra deviate from their expected Kolmogorov scaling at small scales.

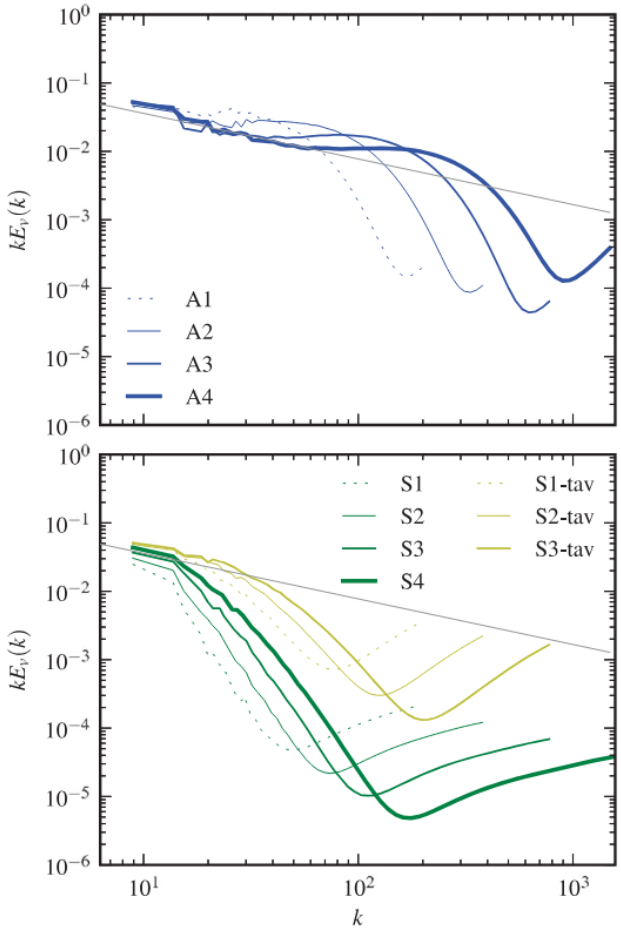


Figure 1.3: Velocity power spectra for subsonic turbulence simulations presented by Bauer and Springel (2012). Their results showcase the resolution dependence of the results of both the AREPO grid code (top) and the GADGET SPH code (bottom), with resolutions ranging from 64^3 to 512^3 cells.

Various sub-grid prescriptions exist to conrol the flow of turbulent energy at the scale of numerical dissipation (Schmidt and Federrath, 2011; Schmidt et al., 2006), often using spectral filtering smaller modes out of the flow field, and relying on the sub-grid model (of limited accuracy) to capture effects from smaller scales.

Machine learning shows great promise to relieve these bottleneck effects. Consider a well-resolved 128^3 turbulent grid. While running an identical simulation on a

smaller (32^3) grid, bottleneck effects will surely arise as energy dissipates at this new grid scale. On the other hand, training a neural network on a downsampled version of the well-resolved grid would offer a model that accurately predicts the exact dynamics and energy spectrum of the original well-resolved grid, but with these dynamics applied to the downsampled grid. Indeed, Stachenfeld et al. (2021) followed this exact methodology with a compressible turbulent box and found near-perfect faithfulness to the original well-resolved energy spectrum, while running the neural simulation on a very course grid (see figure 2.5)

Gibbs phenomenon

The Gibbs phenomenon refers to the occurrence of oscillatory artifacts near sharp discontinuities or steep gradients when approximating a discontinuous function using a finite Fourier series (Gibbs, 1899). In numerical simulations of fluid dynamics, they present as spurious oscillations ("ringing" artifacts) near shock fronts or other sharp transitions in the flow.

Various techniques can be introduced to mitigate this effect, such as slope limiters, spectral filtering, and Adaptive mesh refinement (AMR) near discontinuities. However, these approaches come at the cost of reduced accuracy or increased computational complexity, and may introduce additional numerical artifacts.

It has been demonstrated that, with the usage of neural networks, the Gibbs phenomenon does not occur (Mylavarapu et al., 2018).

Grid Anisotropy

Grid anisotropy is an artifact in grid-based fluid simulations where the discrete mesh introduces preferred directions aligned with grid axes. This can lead to non-physical results, especially for poorly resolved flows or those aligned diagonally to the grid. While higher-order schemes can reduce these effects, they require more compute and do not completely remove the effect.

Neural methods, although usually performed on an identical grid, do not evolve the flow according to strict flux-transfer schemes and thus do not suffer from similar anisotrophy issues. In particular, convolutional neural networks process the information from all neighbouring cells with the same treatment, regardless of if they lie along the cartesian axes (see 1.4.11), allowing fluxes to flow diagonally in the same manner as they do when aligned with the grid.

CFL condition

The Courant-Friedrichs-Lewy (CFL) condition is a necessary condition for the stability of explicit time integration schemes in numerical methods for solving partial differential equations (Courant et al., 1967). For fluid dynamics simulations, it can be expressed as:

$$C = \frac{u\Delta t}{\Delta x} \leq C_{max} \tag{1.2.10}$$

where C is the Courant number, u is the characteristic velocity, Δt is the time step, Δx is the spatial discretization, and C_{max} is the maximum allowed Courant number.

The CFL condition limits the timestep according to the spatial resolution and

the fastest signal speed in the system. This constraint ensures that information does not propagate across more than one cell per time step, which is a requirement for numerical stability.

Machine learning methods show promise to completely avoid such a constraint, as they are not bound by any particular numerical scheme and thus may learn a system's evolution operator out to arbitrarily large timesteps (Stachenfeld et al., 2021). It is important to note that one must still respect the speed of information propagation. A method that does not follow the standard CFL condition would still have to include the influence of all data out to a distance $\sim c_{sound}\Delta t$ - such as using a large receptive field in the convolution kernel

1.3 Thermochemistry

The chemical and thermal evolution of astrophysical gases plays an important role in determining structure formation and galaxy evolution. Here, we describe the main chemical and cooling processes that are important in the temperature range $10^2 \lesssim T \lesssim 10^8$ K relevant for the interstellar and intergalactic medium.

The evolution of the chemical abundances is governed by a set of rate equations of the form:

$$\frac{dn_i}{dt} = \sum_j k_{ij} n_j n_e - \sum_l k_{il} n_i n_e \tag{1.3.1}$$

where n_i is the number density of species i, n_e is the electron number density, and k_{ij} and k_{il} are the rate coefficients for formation and destruction processes respectively (Gnedin and Hollon, 2012). The dominant processes include collisional ionization,

radiative and dielectronic recombination, charge transfer reactions, and photoionization.

The cooling function Λ and heating function Γ determine the net cooling rate per unit volume:

$$\frac{dU}{dt} = n_b^2[\Gamma(T, ...) - \Lambda(T, ...)] \tag{1.3.2}$$

where U is the thermal energy density and n_b is the total baryon number density (Sutherland and Dopita, 1993b). The cooling function has contributions from various atomic and ionic species:

$$\Lambda = \sum_i \Lambda_i \tag{1.3.3}$$

For metal-enriched gas in the ISM and IGM, the key cooling processes are:

- Metal line transitions (e.g. C, O, Ne, Fe)
- Collisional excitation of H and He
- Recombination
- Free-free emission (bremsstrahlung)

These can be expressed in the general form:

$$\Lambda_i = f_i(T) n_e n_j \tag{1.3.4}$$

where $f_i(T)$ is a temperature-dependent cooling rate coefficient (Katz et al., 1996).

Metal line cooling dominates at temperatures $T \lesssim 10^7$ K and can be approximated as:

$$\Lambda_{\rm metal} \approx n_H n_e \Lambda_Z(T, Z) \tag{1.3.5}$$

where $\Lambda_Z(T, Z)$ is the metal cooling function dependent on temperature and metallicity (Wiersma et al., 2009).

The presence of an external radiation field introduces additional complexity through photoionization and photoheating processes. Gnedin and Hollon (2012) showed that the cooling and heating functions in this case can be approximated as

$$\{\Gamma, \Lambda\}(T, n_b, Z, J_\nu) \approx \sum_{i=0}^{2} \left(\frac{Z}{Z_\odot}\right)^i \{\Gamma, \Lambda\}_i(T, r_j, n_b) \tag{1.3.6}$$

where r_j are combinations of the normalized photoionization rates.

Solving this chemistry for metals out of equilibrium is computationally expensive and rises steeply as the number of species increases (Smith et al., 2017). This problem is further compounded by the fact that chemical rate equations are typically stiff (Richings et al., 2014a) and the chemical timescale is often orders of magnitude shorter than the hydrodynamical timescale. The dispersion and mixing of ejecta from stellar winds and supernovae across the ISM can cause galaxies to become very enriched with metals (Annibali and Tosi, 2022). For these reasons, researchers turn to either pre-computed tabulated datasets, or use highly optimized solver codes to accurately model the chemical and thermal evolution of an astrophysical gas (Robinson et al., 2024), which can become infeasible as the number of physical parameters sampled increases the dimension of the table.

Many such solver codes are available for astrophysical chemistry and cooling, including `CHIMES` (Richings et al., 2014a,b), `KROME` (Grassi et al., 2014), `Cloudy` (Ferland et al., 2013), and `Grackle`.

Equilibrium solvers, such as `Cloudy`, determine the steady-state composition. They are suitable when the chemical equilibrium timescale << dynamical timescales.

Non-equilibrium solvers, like `CHIMES` and `KROME` evolve the chemical composition over time, necessary when chemical timescales >> dynamical timescales, allowing for accurate resolution of this out-of-equilibrium behaviour. `Grackle` uses tabulated data from `Cloudy` with additional non-equilibrium chemistry for a Hydrogen and Helium chemical network.

Codes such as `Cloudy` allow simulators to pre-compute and tabulate cooling rates for various conditions. Of course, the accuracy of such a method is limited by its resolution, which is ultimately constrained by the machine's memory. Note that the dimensionality of this table can be impossibly high (> 150) to incorporate all chemical species, in which case a table is impossible.

1.4 Machine Learning

Most notable machine learning methods in recent years have been based on neural networks. Neural methods involving deep layers of trainable parameters have enjoyed success due to several factors, including the maturation of GPU-accelerated computing, breakthroughs in optimization techniques, architectural improvements, and the increasing scale of available data and computational resources.

Recent neural methods have shown particular promise due to several key developments in the field. The maturation of GPU-accelerated computing has enabled the training of much larger and more complex models. Breakthroughs[1] in optimization, such as the Adam algorithm (Kingma and Ba, 2014), have significantly improved the

[1]Breakthrough is no exaggeration; optimization algorithm `Adam` (Kingma and Ba, 2014) was the most cited paper of the decade

training dynamics of deep neural networks. Architectural innovations like convolutional neural networks, attention mechanisms, group-equivariant neural networks, and neural ODEs/SDEs have expanded the capabilities and applicability of these models (see 2.1). Furthermore, the increasing scale of available data and computational resources has allowed for the training of increasingly sophisticated models.

While deep neural networks have dominated recent machine learning research in physics, it is worth noting that some methods explored in recent literature do not use deep neural networks to approximate physical functions. Instead, these methods attempt to discover the underlying symbolic equations for a given physical dynamics dataset. Examples of such methods include `SINDy` (Brunton et al., 2016), `PySR` (Cranmer, 2023), and `AI Feynman` (Udrescu and Tegmark, 2020). These approaches aim to provide interpretable models that can offer insights into the fundamental equations governing physical systems.

1.4.1 Artificial Neural Networks

Artificial neural networks (ANNs) are computational models inspired by the biological structure of neural networks found in the brain (McCulloch and Pitts, 1943). The neurons within the cerebral cortex are organized in 6 distinct layers, an arrangement that increases its computational efficiency (Miyashita, 2022). Similarly, an ANN consists of layers of connected nodes acting as artificial neurons.

In an ANN, each neuron receives a weighted sum of signals from neurons in the previous layer, along with a bias term. Mathematically, for a neuron j in layer l, its

input z_j^l is computed as:

$$z_j^l = \sum_i w_{ij}^l a_i^{l-1} + b_j^l \tag{1.4.1}$$

where w_{ij}^l is the weight connecting neuron i in layer $l-1$ to neuron j in layer l, a_i^{l-1} is the activation of neuron i in layer $l-1$, and b_j^l is the bias term for neuron j in layer l.

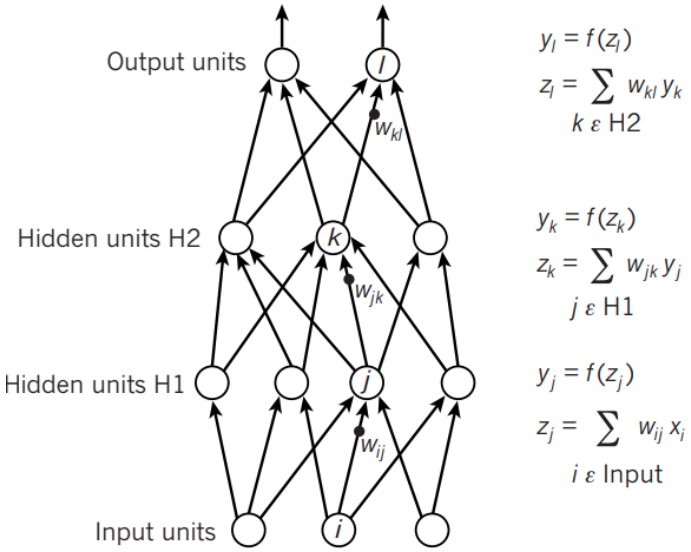


Figure 1.4: Diagram of an artificial neural network, showing operations at each layer. Arrows indicate the direction of information flow through the network (LeCun et al., 2015)

The output of each neuron, known as its activation, is then passed through a non-linear function f, typically a sigmoid or a rectified linear unit (ReLU) (Agarap, 2018):

$$a_j^l = f(z_j^l) \tag{1.4.2}$$

These activation functions at each layer allow the network to express non-linear relationships.

Artificial neural networks learn hierarchical representations of data through multiple layers of these non-linear transformations (LeCun et al., 2015). This architecture allows deep neural networks to capture complex, non-linear relationships in very high-dimensional datasets. Indeed, we find many such high dimensional non-linear problems across many domains of physics, making deep neural networks a promising candidate to improve physical simulations.

The universal approximation theorem (Cybenko, 1989; Hornik et al., 1989a) (see section 1.4.3) states that a feedforward network with a single hidden layer containing a finite number of neurons can approximate continuous functions on compact subsets of $\mathbb{R}^n$, under mild assumptions about the activation function. This implies a wide applicability of neural networks to various problem domains. Motivated by this fundamental theoretical result, many recent works have shown various successes applying models with deep-learning-based architectures to enhance physics simulations (Boral et al., 2023; Branca and Pallottini, 2023; Chattopadhyay et al., 2022; Guan et al., 2023; Li et al., 2021; Lu et al., 2021; Raissi et al., 2019; Rosofsky and Huerta, 2020; Sulzer and Buck, 2023; Wang et al., 2022, 2020).

1.4.2 Backpropagation

The loss function L defines how the model's output y is to be optimized using a ground truth dataset $\hat{y}$. Consider a loss function such as the mean squared error:

$$\mathrm{L} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{1.4.3}$$

where y_i represents the true value, $\hat{y}_i$ is the predicted value, and n is the total number of samples.

Backpropagation provides an efficient way to compute the gradient of the loss function with respect to the network parameters, which is then used to update these parameters through gradient descent or its variants.

The algorithm consists of two main phases: the *forward pass* and the *backward pass*. In the forward pass, the input data is propagated through the network to compute the output and the associated loss. For a regression task with mean squared error loss, L, this can be expressed as:

$$\hat{y} = f(x; \theta), \quad L = \frac{1}{n} \sum_{i=1}^{n} (\hat{y}_i - y_i)^2 \tag{1.4.4}$$

where f is the neural network function, x is the input, θ are the network parameters, $\hat{y}$ is the predicted output, y is the true output, and n is the number of samples.

In the backward pass, the gradient of the loss with respect to the network parameters is computed using the chain rule of calculus. This process starts at the output layer and moves backwards through the network:

For the output layer:

$$\frac{\partial L}{\partial \hat{y}} = \frac{2}{n}(\hat{y} - y) \tag{1.4.5}$$

For hidden layers:

$$\frac{\partial L}{\partial h_l} = \frac{\partial L}{\partial h_{l+1}} \frac{\partial h_{l+1}}{\partial h_l} \tag{1.4.6}$$

where h_l represents the pre-activation values of layer l.

For the network parameters:

$$\frac{\partial L}{\partial \theta} = \frac{\partial L}{\partial h}\frac{\partial h}{\partial \theta} \tag{1.4.7}$$

Finally, the parameters are updated using gradient descent

$$\theta_{t+1} = \theta_t - \eta \frac{\partial L}{\partial \theta} \tag{1.4.8}$$

or, in most recent literature, an adaptive momentum-based optimizer such as `Adam` (Kingma and Ba, 2014):

$$\theta_{t+1} = \theta_t - \eta \frac{m_t}{\sqrt{v_t} + \epsilon}$$

where m_t and v_t are the first and second moment estimates, and ϵ is a small constant to prevent division by zero.

1.4.3 Universal Approximation Theorem

The Universal Approximation Theorem, established by Cybenko (Cybenko, 1989) and Hornik (Hornik et al., 1989b), is a highly motivating theorem for the use of neural networks as novel solutions to many problems in physics.

Let $C(X, \mathbb{R}^m)$ denote the set of continuous functions from a subset X of a Euclidean $\mathbb{R}^n$ space to a Euclidean space $\mathbb{R}^m$. Let $\sigma \in C(\mathbb{R}, \mathbb{R})$, where $(\sigma \circ x)_i = \sigma(x_i)$.

Assume σ is not a polynomial function, then for every $n, m \in \mathbb{N}$, any compact set $K \subset \mathbb{R}^n$, any continuous function $f \in C(K, \mathbb{R}^m)$, and any $\varepsilon > 0$, there exist $k \in \mathbb{N}$, $W \in \mathbb{R}^{k \times n}$, $b \in \mathbb{R}^k$, and $C \in \mathbb{R}^{m \times k}$ such that

$\sup_{x \in K} \|f(x) - g(x)\| < \varepsilon$

where g represents a layer in a neural network, $g(x) = C \cdot (\sigma(Wx + b))$.

The theorem establishes that a feedforward network with a single hidden layer and a finite number of neurons can approximate any continuous function on $\mathbb{R}^n$, given certain conditions are met. The theorem is trivially generalized to networks with multiple layers by using the same construction for the first layer and approximating the identity function with subsequent layers. It should be noted that this is an existence proof, and does not specify how such a layer (or layers) can be constructed.

1.4.4 The Manifold Hypothesis

In most cases, training data is generated using a by sampling across a wide array of physical variables. The network's ability to learn an efficient latent space representation is essential to making accurate predictions. The dynamics of this latent space are highly constrained by underlying physics, and are presumed to reside on a manifold of significantly lower dimension than that of the input variables. This is the manifold known as hypothesis (Lee and Carlberg, 2020) and motivates the use of Autoencoder networks (a model trained to learn the identity function on the input variables, with a low-dimensional information bottleneck in the inner-most layer (Hinton and Salakhutdinov, 2006)).

1.4.5 Convolutional Neural Networks

Convolutional Neural Networks (CNNs), first introduced by Fukushima (1980) and later developed by Lecun et al. (1998) have been very successful in problems that exhibit translational equivariance, such as image recognition. CNNs use convolutional

layers, which apply small (typically 3x3) learnable kernel matrix (also known as a filter) that slides over the input data. In this text, we use the word convolution and correlation interchangeably, as they are equivalent for all provided explanations and proofs, with the indices being easier to manipulate.

The continuous convolution operation between two functions f and g is defined as:

$$(f * g)(t) = \int_{-\infty}^{\infty} f(\tau)g(t - \tau)d\tau \tag{1.4.9}$$

This operation measures the overlap between f and a reversed version of g as a function of the amount of translation between them. In the discrete case, which is more relevant to CNNs, the 1D convolution can be expressed as:

$$(f * g)[n] = \sum_{m=-\infty}^{\infty} f[m]g[n - m] \tag{1.4.10}$$

In a convolutional layer, each filter performs a convolution operation over the input. For a 2D input I and a kernel K, the convolution operation is defined as:

$$(I * K)ij = \sum_{m} \sum_{n} Ii + m, j + nK_{m,n} \tag{1.4.11}$$

This operation is performed across a "channel" dimension with multiple filters, each producing an output known as a feature map. The use of shared weights (filters) across the input space introduces translation equivariance, a key property that makes CNNs particularly effective for many tasks in computer vision (LeCun et al., 2015).

CNNs may also include pooling layers, which perform downsampling operations to reduce the spatial size of the feature maps. A common pooling operation is max

pooling, defined as:

$$y_{ij} = \max_{(m,n)\in R_{ij}} x_{mn} \tag{1.4.12}$$

where R_{ij} is a local region in the input centered at position (i, j).

The combination of convolutional and pooling layers allows CNNs to learn hierarchical representations of the input data, capturing both local and global features. This operation is lossy, as the max pooling operation removes data from the previous layer.

1.4.6 Train-test split

By isolating a training set and testing set from the initial dataset, the model can easily be trained on the training set and evaluated on the unseen test set, helping to assess how well your model generalizes to new data. Typically, a dataset split would look as follows:

- **Training set:** 70-90% of the data, used for model optimization.
- **Validation set:** 5-10% of the data, used for tuning hyperparameters of the network (such as learning rate or layer size) and early stopping.
- **Test set:** The remainder of the data, held out entirely during training and used only for final evaluation of the network's ability to generalize to new data.

The high-dimensional nature of astrophysical simulations often leads to prohibitively large datasets. For example, a full 3D hydrodynamical simulation with chemical evolution can easily exceed terabytes in size (Federrath et al., 2020). If the problem

admits it, selective sampling can be used, which considers regions of parameter space with significant variation, avoiding oversampling of quiescent regions. Additionally, data augmentation can be used to generate "new" rows of data by permuting the inputs in a manner that leaves the problem invariant.

Chapter 2

Review of Modern Machine Learning

2.1 Neural Architectures

2.1.1 Residual Connections

Residual connections are a structural modification in neural networks where the output of a layer is added to the input of that layer, allowing direct propagation of information across multiple layers. This addition operation enables the network to learn residual functions, which can be thought of as the difference between the desired mapping and the identity mapping

Residual connections (also known as skip connections), introduced by He et al. (2015), have become key in the design of deep neural networks. The output of a given

layer in a residual network y is computed as:

$$y = F(x, W_i) + x \tag{2.1.1}$$

where x is the input to the layer, $F(x, \{W_i\})$ represents the residual of the function to be learned, and $\{W_i\}$ are the weights of the layers.

Figure 2.1 shows three consecutive layers (n, $n+1$, and $n+2$), with a skip connection between layer n and the output of layer $n+2$. The output of layer $n+2$ is added to the output of layer n via the skip connection. This structure allows the network to learn residual functions with reference to the layer inputs, which can improve training dynamics and overall performance.

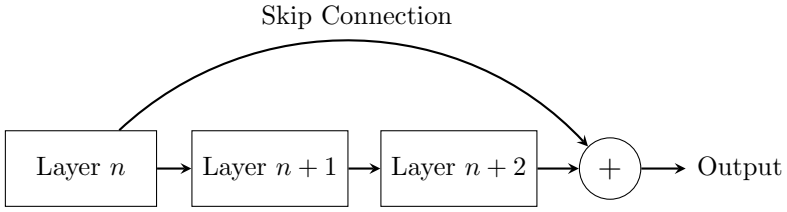
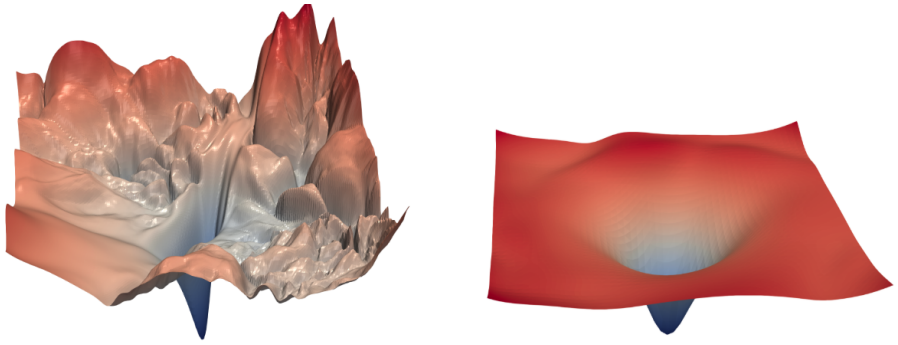


Figure 2.1: Illustration of the connections between layers within a residual block

He et al. (2015) showed that the residual of a function is empirically easier to learn than the function itself. They also offered a mechanistic explanation: skip connections allow the gradients to flow nearly unchanged, preventing gradients from approaching zero (the vanishing gradient problem) as they are backpropagated through deep networks.

Later work by Li et al. (2018) debated this reasoning and posited that skip connections improve convergence via a smoothing effect on the highly non-convex loss landscape (see 2.2. They demonstrated that the loss landscape for models with residual connections is smoother and more convex, which they proposed as a reason for

its superior training dynamics. Residual (skip) connections have become a standard component in many state-of-the-art neural network architectures.



(a) without skip connections (b) with skip connections

Figure 2.2: Comparison of loss landscapes for networks with and without residual connections (Li et al., 2018) across two representative directions in parameter space. Networks with residual connections have smoother and more convex loss landscapes, which can improve training dynamics.

2.1.2 U-Net

The U-Net architecture, introduced by Ronneberger et al. (2015), is a CNN originally designed for biomedical image segmentation. Its U-shaped structure consists of convolutional layers, contracting the input dimensions and expanding the channel dimension to capture context from across the input, and a symmetric expanding path that enables context aggregation across this downsampled spatial domain and learning of hierarchical spatial features. Skip connections are included between layers of equal shape.

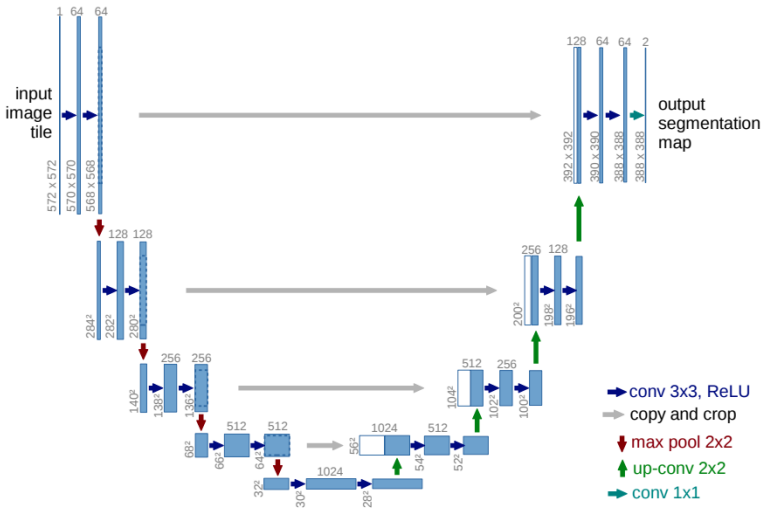


Figure 2.3: U-net architecture. Each blue box corresponds to a multi-channel feature map. White boxes represent copied feature maps (Ronneberger et al., 2015).

The contracting path follows the typical architecture of a convolutional network, comprising repeated application of convolutions, each followed by an activation and a max pooling operation for downsampling. At each downsampling step, the number of convolution kernels is multiplied. The expanding path consists of an upsampling of the feature map followed by a convolution that halves the number of feature channels, eventually arriving at the shape of the first convolutional layer.

The U-Net architecture (figure 2.3) has proven highly effective not only in image segmentation (Ronneberger et al., 2015) but has become popular in various other domains. Its success can be attributed to its ability to capture both local and global spatial features, and its efficient use of feature information through skip connections.

Building upon the U-Net architecture, Wang et al. (2019) introduced the Turbulent Flow Net (TF-Net) for turbulent flow prediction. TF-Net decomposes the turbulent flow with trainable spectral filters. In TF-Net, three identical encoders

process three scale components separately, while a shared decoder learns the interactions among these components. Each encoder-decoder pair can be viewed as a modified U-Net.

Wang et al. (2019) demonstrated that TF-Net outperforms standard U-Net and other baselines in turbulent flow prediction, achieving a modest 11.1% reduction in prediction RMSE and more significant improvements in preserving physical properties such as energy spectrum and turbulence kinetic energy.

2.1.3 Dilated Convolution

The receptive field of a neuron in a CNN refers to the region in the input space that can influence the neuron's activation.

Dilated convolutions (Yu and Koltun, 2016), also known as atrous convolutions, are a generalization of standard convolutions that allow for receptive fields to capture information across spatial domains larger than that of the filter size. A one dimensional dilated convolution operator $*_l$ with dilation factor l is defined as:

$$(F *_l k)(p) = \sum_{s+lt=p} F(s)k(t) \tag{2.1.2}$$

where F is the input feature map, k is the filter, and p is the spatial position. The standard convolution is simply the 1-dilated convolution.

Dilated convolutions allow for expansion of the receptive field without any loss of resolution or coverage (as would occur in a vanilla CNN). For example, consider applying 3x3 filters with exponentially increasing dilation:

$$F_{i+1} = F_i *_{2^i} k_i \quad \text{for } i = 0, 1, \ldots, n-2 \tag{2.1.3}$$

The size of the receptive field of each element in F_{i+1} is $(2^{i+2} - 1) \times (2^{i+2} - 1)$. Thus the receptive field can grow exponentially while the number of parameters only grows linearly. This property makes dilated convolutions well-suited for tasks that require multi-scale contextual information at high resolutions.

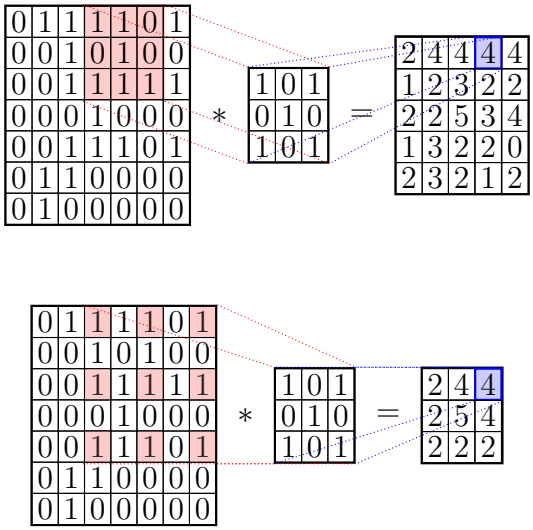


Figure 2.4: A standard matrix convolution, equivalent to dilation = 0 (top), vs. a matrix convolution with dilation = 1 (bottom). The receptive field of the dilated kernel is wider, without requiring a larger kernel matrix.

Dilated convolutions (Yu and Koltun, 2016), similarly to the U-Net, offer a way to incorporate long-range heirarchical relationships across the spatial domain, and naturally they have been employed for predicting the evolution of turbulent flow Stachenfeld et al. (2021).

2.1.4 Group Equivariant Convolutional Neural Networks

Group Equivariant Convolutional Neural Networks (G-CNNs), introduced by Cohen and Welling (2016), extend the concept of translation equivariance in standard CNNs (described in 1.4.5 to more general types of transformations. The motivation behind

these network is to design convolutional architectures that are equivariant to certain group actions. A function Φ is said to be equivariant to a group action T if:

$$\Phi(T_g x) = T'_g \Phi(x) \tag{2.1.4}$$

for all elements g of the group, where T_g and T'_g are the actions of T on the input and output spaces respectively.

G-CNNs achieve this equivariance by defining convolution operations over the group rather than just over the input space. Herein, as done by Cohen and Welling (2016), we use the words correlation and convolution interchangeably. For a group G, the G-correlation is defined as:

$$[f \star \psi](g) = \sum_{h \in G} \sum_{k} f_k(h) \psi_k(g^{-1}h) \tag{2.1.5}$$

where f is the input feature map, ψ is the filter, and k indexes the channels. This operation is equivariant to the action of G, as can be shown through the following derivation:

$$[L_u f \star \psi](g) = \sum_{h \in G} \sum_k [L_u f]_k(h) \psi_k(g^{-1}h) \tag{2.1.6}$$

$$= \sum_{h \in G} \sum_k f_k(u^{-1}h) \psi_k(g^{-1}h) \tag{2.1.7}$$

$$= \sum_{h \in G} \sum_k f_k(h) \psi_k(g^{-1}uh) \tag{2.1.8}$$

$$= \sum_{h \in G} \sum_k f_k(h) \psi_k((u^{-1}g)^{-1}h) \tag{2.1.9}$$

$$= [f \star \psi](u^{-1}g) \tag{2.1.10}$$

$$= [L_u(f \star \psi)](g) \tag{2.1.11}$$

where L_u is the left-regular representation of G.

The G-CNN architecture typically consists of multiple G-convolution layers, each followed by a nonlinearity which also commutes with the group action. G-CNNs have shown superior performance on tasks with symmetries inherent in the data, such as rotation-invariant image classification (Cohen and Welling, 2016). By explicitly encoding symmetries into the network architecture, G-CNNs can often achieve better sample efficiency during training and better generalization during testing compared to standard CNNs

Alternatively, an ad hoc regularization term can be added to the loss function when training a standard CNN to encourage the learning of equivariant latent structures:

$$\mathcal{L} = \mathcal{L}_{MSE} + \beta \left\| T(f(x)) - f(T(x)) \right\|^2 \tag{2.1.12}$$

where L_{MSE} is the mean squared error loss, $f(x)$ is the output of the neural

network for input x, T is the transformation from the symmetry group, and β is a hyperparameter that controls the strength of the regularization.

2.1.5 Recurrent Neural Networks (RNNs)

Recurrent Neural Networks (RNNs) (Rumelhart et al., 1986) are a type of sequence-to-sequence (Seq2seq) model, which unlike previously discussed architectures, accept states at multiple timesteps (the input sequence) and produce a new sequence of states.

RNNs can be unrolled in time, effectively creating a deep feedforward network with shared weights across layers:

$$h_t = f_h(W_{hh}h_{t-1} + W_{xh}x_t + b_h)$$

$$y_t = f_y(W_{hy}h_t + b_y)$$

where x_t is the input at time step t, h_t is the hidden state at time step t,y_t is the output at time step t, W represents weight matrices, b represents a bias term, and f is the activation function.

However, vanilla RNNs struggle to learn long-term dependencies due to the vanishing gradient problem. As the error gradients are backpropagated through many time steps, their floating point values tend to either vanish or explode, making it difficult for the network to learn long-range dependencies (Bengio et al., 1994).

To address this issue, more advanced RNN architectures have been introduced, such as Long Short-Term Memory (LSTM) networks (Hochreiter and Schmidhuber,

1997). These architectures introduce gating mechanisms that allow the network to selectively remember or forget information over long sequences, mitigating the vanishing gradient problem. Note that we will not return to RNNs in this work, as they have been ubiquitously replaced by LSTMs in recent literature.

2.1.6 Long Short-Term Memory (LSTM)

Long Short-Term Memory (LSTM) networks, introduced by Hochreiter and Schmidhuber (1997), they are a type of recurrent neural network (RNN) that avoid the vanishing gradient by introducing gates which regulate the flow of gradients during a backward pass and regulate the flow of information during a forward pass. The LSTM architecture consists of a memory cell c_t and three gating mechanisms: input gate i_t, forget gate f_t, and output gate o_t:

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$$

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$$

$$\tilde{c}_t = \tanh(W_c \cdot [h_{t-1}, x_t] + b_c)$$

$$c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}t$$

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o)$$

$$h_t = o_t \odot \tanh(c_t)$$

where x_t is the input vector at time t, h_t is the hidden state, W and b are learnable parameters, σ is the sigmoid function, and $\odot$ denotes element-wise multiplication. The cell state c_t allows for long-term memory preservation, while the gating mechanisms control information flow, enabling the network to learn long-range dependencies. In essense, this method's route of improving the loss landscape is somewhat similar to that of residual networks (described in 2.1.1), allowing information to flow directly across many layers as needed.

2.1.7 Transformers

Transformers are sequence-to-sequence models that have revolutionized natural language processing and other sequence-based tasks. They are built on the attention mechanism introduced by Vaswani et al. (2017). The core of this mechanism is the scaled dot-product attention. The attention mechanism uses a set of three learned matrices: the queries (Q), keys (K), and values V.

The queries, keys, and values each play a distinct role in the attention mechanism. Queries represent the current context or position for which we want to compute attention. Keys are used to match against queries to determine relevance. Values contain the actual information that we want to aggregate based on the attention weights.

The attention mechanism works by a dot product between the queries and keys, which are then used to weight the values. This process allows the model to choose which relevant parts of the input sequence to aggregate (or, in the terminology of transformers, which inputs to "attend" to) when producing each output element. By doing so, attention allows with global context aggregation, enabling the model to

learn which long-range dependencies it should capture.

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{2.1.13}$$

where Q is the query matrix, K is the key matrix, V is the value matrix, d_k is the dimension of the keys, and n is the sequence length. The $\frac{1}{\sqrt{d_k}}$ factor prevents the dot products from growing too large in magnitude for large values of d_k.

Transformers have overtaken RNNs and LSTMs in nearly all applications due to their superior performance. The attention mechanism allows the model to "attend" to different parts of the input sequence when producing each element of the output sequence, enabling the capture of long-range dependencies within a sequence without the need for recurrent connections, and without the vanishing gradient problem.

2.1.8 Neural ODEs

Some of works discussed in further sections use neural ordinary differential equations in their architecture. Consider a neural network h with learnable parameters θ predicts the next timestep:

$$h_{t+1} = h_t + f(h_t, \theta) \tag{2.1.14}$$

These iterations are equivalent to the Euler discretization of a continuous transformation. As the number of layers increase and timestep size decreases, the dynamics of hidden layers can be parameterized with an ordinary differential equation (ODE)

specified by a neural network (Chen et al., 2018):

$$\frac{dh(t)}{dt} = \bar{f}(h(t), t, \theta) \tag{2.1.15}$$

This idea has been extended to stochastic differential equations (Neural SDEs) specifically for use in a turbulence prediction model (Boral et al., 2023).

2.2 Neural Methods for Turbulent Flows

We now discuss existing applications of neutral networks to physical problems. Hydrodynamics codes are computationally expensive and central to astrophysical simulation. There are many aspects of these codes that can potentially benefit from machine learning techniques.

Most literature focuses on solving the Euler equations at the global grid scale, using network architectures that share some equivariance properties with the hydrodynamical equations to manage this increased computational challenge. However, specific components of the numerical solution can be addressed in isolation.

There are several considerations when evaluating these new ideas. A primary one is accuracy and the potential for an improved treatment compared to a traditional solver. However, computational expense is always an overwhelming concern in numerical work. Given the considerable expense of evaluating a neural network even once, it is typically going to be much more expensive (by orders of magnitude) than a straightforward explicit solution of the hydrodynamics equation for a single timestep on a per cell basis.

2.2.1 Neural Riemann Solvers

Ruggeri et al. (2022) trained a fully connected neural network to act as a Riemann solver at each cell interface. This methodology is not easily extended to large-scale 3D simulation grids, as it requires one forward pass of the network at each cell interface, at each timestep. This is more expensive than an explicit Riemann solver in most cases.

2.2.2 Incompressible Turbulence

SPARTA (Sparse Regression of Turbulent Stress Anisotropy) (Schmelzer et al., 2020) is a deterministic symbolic regression method for discovering algebraic Reynolds stress models directly from high-fidelity simulation data. The method constructs models as tensor polynomials built from a library of candidate functions. SPARTA regularizes the loss to encourage sparsity in the inferred models. SPARTA demonstrated improved predictions over a commonly used traditional eddy-viscosity (k-ω SST) (Menter, 1994) model model, even at higher Reynolds numbers.

Portwood et al. (2019) apply Neural ODEs to model the time evolution of turbulent kinetic energy dissipation. learn the dynamics of kinetic energy k and dissipation rate ϵ extracted from DNS datasets. Their model first runs a forward pass through an RNN to encode the spatial and temporal data, then a Neural ODE model to predict trajectories in latent space h_t governed by $\frac{dh_t}{dt} = f(h_t, \theta)$ where f specifies the dynamics in latent space and θ are neural network parameters. A decoder then maps back to data space to complete the evolution. Their model outperforms a state-of-the-art analytic model for turbulent dissipation (Perot and de Bruyn Kops, 2008). When predicting dissipation rate evolution, errors remained within 1-2% of direct numerical

simulations compared to over 10% for existing models (Portwood et al., 2019).

Boral et al. (2023) introduce a data-driven learning approach called neural ideal Large Eddy Simulation (niLES) that uses neural stochastic differential equations (NSDEs), learning the stochastic equation to evolve the state in latent space. As is a common theme in recent literature, their method uses encoder-decoder architecture. In this case, the encoder maps from the filtered flow field to the latent space. The authors demonstrate improved accuracy in long-term flow statistics and enhanced stability for extended rollouts compared to deterministic closure models (Boral et al., 2023).

2.2.3 Compressible Turbulence

Stachenfeld et al. (2021) present a neural simulator for 3D compressible turbulence which outperforms the `Athena++` code (Stone et al., 2020) on a 32^3 grid across a variety of metrics when compared to a "ground truth" 128^3 grid. The learned neural model offered reduced error in the energy field, reduced error in the energy spectra, and improved faithfulness to the shape of the true energy spectrum. Their network, running on a single GPU, improves wall-clock simulation time by 1000x compared to `Athena++` running on an 8-core CPU.

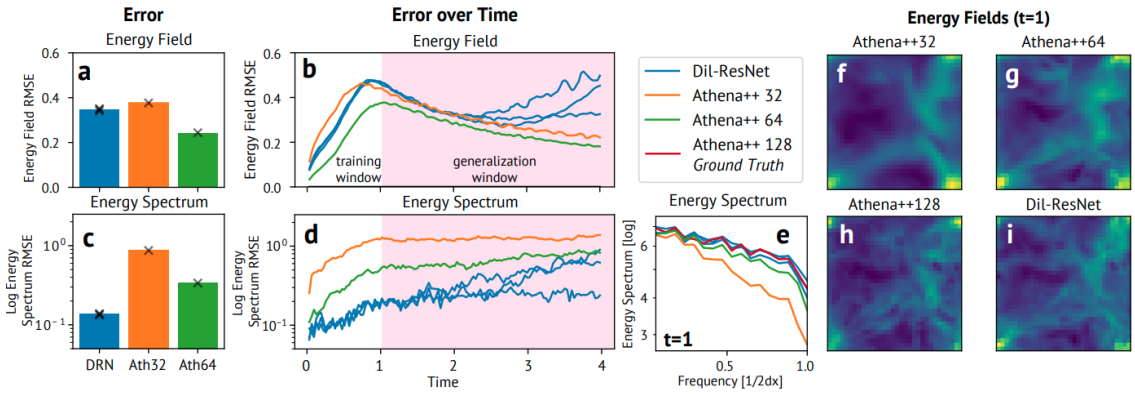


Figure 2.5: Energy errors and energy spectra for Athena++ compared to the dilated residual convolutional network trained in Stachenfeld et al. (2021). The neural model, having been trained on dynamics from a higher resolution grid, is able to resolve a significantly more faithful energy spectrum than the traditional Athena++ code on an equivalent course grid.

Dang et al. (2022) introduce a Transformer architecture dubbed `TNT` to learn turbulent dynamics using the now-ubiquitous attention mechanism introduced by Vaswani et al. (2017). `TNT` uses a novel Temporal Mutual Self Attention (TMSA) mechanism (Liang et al., 2022), which combines temporal and spatial data into a single unified vector for downstream use in the transformer network. They show significant significantly lower RMSE and a higher R^2 than predictions from a U-net (Dang et al., 2022).

2.3 Neural Methods for Astrochemistry

Previous works have used various neural methods to approximate the chemical and thermal evolution of astrophysical gas: Sulzer and Buck (2023) proposed a Neural ODE (Chen et al., 2018) solver to accelerate an astrochemical reaction network for a prescribed evolution scenario at a given temperature and density. They used an

autoencoder to reduce the dimensionality of the chemical species before evolving the state in latent space. They found a speed-up of 55x compared to a standard ODE solver, while maintaining a median errors ~ 0.015. It is worth keeping mind that this was a very constrained scenario with just two test cases to be learned. It is unclear whether such a small, fast network could handle more general problems.

Galligan et al. (2019) developed a neural emulator called `deepCool` to follow total cooling rates, total heating rates, and metal-line only cooling rates of irradiated gas. They trained on a high-resolution cosmological simulation, using `CLOUDY` to generate the data for ∼850,000 cells. Their architecture consists of an input layer with 9 dimensions (temperature, density, metallicity, and 6 radiation parameters), two hidden layers of 20 neurons each, and one output neuron. The networks achieved median fractional errors of 6.3%, 6.6%, and 3.8% for total cooling, heating, and metal-line cooling rates respectively. However, there were extreme outliers with (multiple) order of magnitude errors and it was not clear when they occurred (i.e. if they were problematic in practice).

They found little to no increase in simulation runtime when deployed in an isolated disk galaxy test with a simpler, standard cooling function without local variations due to the radiation field. Galligan et al. (2019) emphasized the importance of local radiation fields, as neglecting this can lead to cooling rates that are an order of magnitude too strong for a significant fraction of cells.

Cifuentes et al. (2021) used a neural approach to predict isothermal chemical evolution, however as Branca and Pallottini (2023) point out, its generalization to a scenario with thermal evolution is non-trivial.

2.3.1 Chemulator

Holdship et al. (2021) introduced `Chemulator`, a neural network-based emulator for emulating thermochemistry. They employed an autoencoder to reduce the dimensionality of the chemical abundances, temperature, an density variables and then trained an ensemble of "emulator" models to predict the thermochemical evolution in latent space. To generate their dataset, `Chemulator` used Latin Hypercube Sampling (LHS) to efficiently sample the physical parameters within specified ranges. Specifically, they sampled 10,000 initial conditions in log-space for a set of 9 physical parameters, leading to a dataset containing 2×10^7 rows of abundances at various stages of chemical evolution. `Chemulator` presents with a mean squared error of 1.7×10^{-4} for a single time step, exhibiting stabile predictions out to 1000 time steps. The authors demonstrated a speed-up $\sim$ 50,000x compared to the `ULCHEM` code.

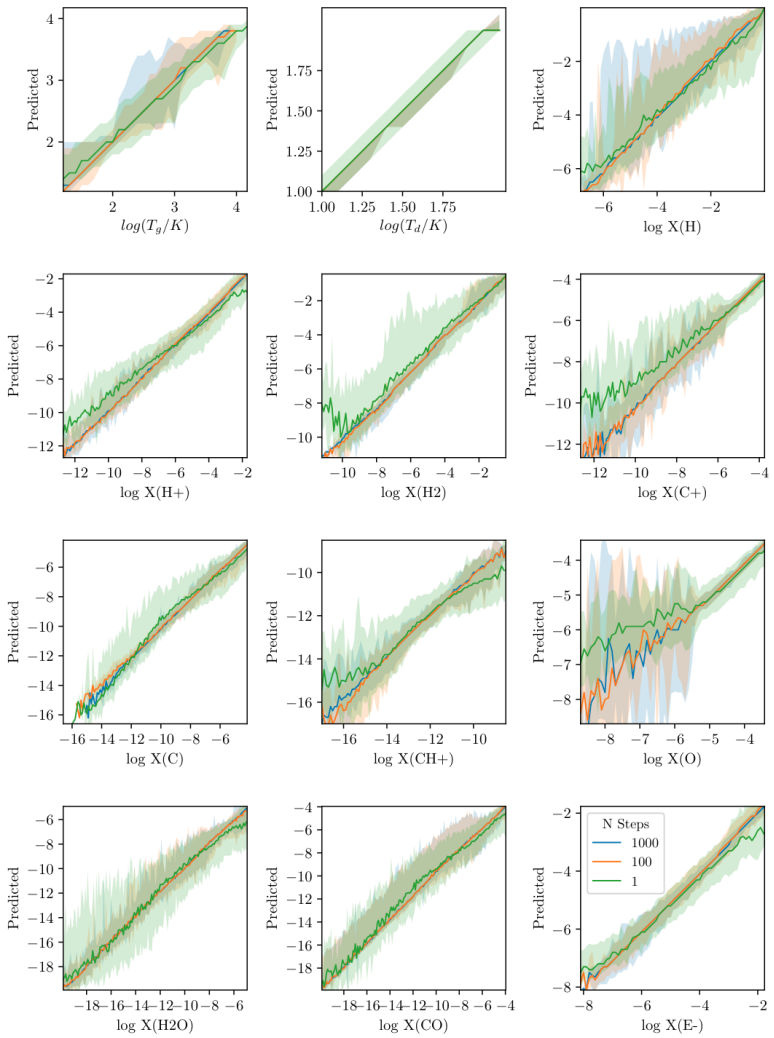


Figure 2.6: Comparison of predicted and actual values for time steps in the test set, for `chemulator` (Holdship et al., 2021). The median prediction is denoted by a line, while the shaded region represents the range predicted by 95% of the models. The colors correspond to the number of time steps emulated, illustrating the predicted chemical evolution over varying timescales. Despite their network showing great promise towards the feasibility of learning such a generalized evolution operator, the errors may span multiple orders of magnitude in some cases.

2.3.2 Branca and Pallotini models

Branca and Pallottini (2023) trained a physics-inspired neural network (PINN) (Raissi et al., 2019) to evolve an astrophysical chemical reaction network, giving speed-ups up to a factor of ~ 200 with respect to traditional ODE solvers.

Further, they found the traditional solver code had simulation times that vary by roughly 30% for different initial density and temperature, while their neural method gives no variation. Their method additionally used a deep Galerkin method (DGM) (Sirignano and Spiliopoulos, 2018) to model chemical evolution. It is noteworthy that, unlike the works referenced above, Branca and Pallottini (2023) do not use an autoencoder-emulator architecture, likely a by-product of the PINN approach being unable to work in learned latent spaces, requiring physically interpretable state variables to compute the loss function.

Very recent works by Branca and Pallottini (2024) showcase a neural emulator for non-equilibrium ISM photo-chemistry using a `DeepONet` architecture (Lu et al., 2021) consisting of two feed-forward neural networks (dubbed branch and trunk). Their model emulates a chemical network with 9 species and 52 reactions, including H2 formation. They trained separate emulators for temperature and each chemical species. The training set, generated using `KROME` (Grassi et al., 2014), sampled initial conditions in ranges $-2 \leq \log(n/\mathrm{cm}^{-3}) \leq 3.5$ for density, $\log(20) \leq \log(T/\mathrm{K}) \leq 5.5$ for temperature, and $-6 \leq \log(n_i/n) < 0$ for species fractions. A key innovation was incorporating arbitrary radiation fields with 10 energy bins. The emulator achieved relative errors below 3% for most species, with a 128x speedup compared to the traditional ODE solver.

A key difference between these works is the range of T, n_H, Z, and the number

of species. `Chemulator` used a network of 33 species, with gas densities from 10^1 to $10^6\,\mathrm{cm}^{-3}$, temperatures from 10^1 to $10^4\,\mathrm{K}$, and metallicity Z from 10^{-2} to $10^{0.5}$. A larger network of 215 species was tested but proved too difficult to emulate (Holdship et al., 2021). Branca and Pallottini (2023) focused on a simpler network of 9 species and 46 reactions, exploring densities between 10^{-2} and $10^3\,\mathrm{cm}^{-3}$ and temperatures from 10^1 to $10^5\,\mathrm{K}$, achieving speed-ups of up to 200 times using PINNs, though with significantly less complexity in parameter space than `Chemulator`.

Sulzer and Buck's work used a chemical network with 29 species and 224 reactions, focusing on a constant temperature of 50 K, cosmic ray ionization rate of $\zeta = 10^{-16}s^{-1}$, and total density of $n_{tot} = 10^4 cm^{-3}$ (Sulzer and Buck, 2023). This setup is significantly less complex than the larger network explored by Holdship et al. (2021); it covers a narrower range of physical parameters, and accordingly they find near-perfect fits in chemical abundance measurements on this reduced setup.

Chapter 3

Learning Dynamical Systems

3.1 Kuramoto-Sivashinsky (KS) Equation

The equations of hydrodynamics are complex and non-linear, exhibiting chaotic sensitivity to initial conditions, but also recurring features that are recognizable as correct fluid flow. These include transient shocks and filamentary structures, and also persistent properties such as density distributions and turbulent spectra.

3D Hydrodynamics is computationally expensive and thus hard to use to efficiently study neural PDE solvers. We expect to get key insights from smaller systems that have the same complex behaviours.

The Kuramoto-Sivashinsky (KS) equation can be used as a 1D "toy model" for 3D hydrodynamical turbulence. The symmetries inherent in the KS equation make the equation of interest for understanding how models with different equivariance properties are able to evolve turbulent flows. Namely, translational and reflectional symmetry allow for the use of convolutional and group-equivariant architectures respectively. Translational symmetry is equivalent to Galilean invariance.

The KS equation is a nonlinear partial differential equation (PDE) that exhibits spatiotemporal chaos (Kuramoto and Yamada, 1976; Sivashinsky, 1977). It was originally developed to study instabilities in flame front propagation. The PDE has a diffusive second-order term, an "anti-diffusive" fourth-order term, and a nonlinear advection term.

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} + \frac{\partial^2 u}{\partial x^2} + \frac{\partial^4 u}{\partial x^4} = 0$$

3.1.1 Managing Chaos

While neural networks show promise to predict turbulent flows, the authors often fail to present their model performance in the context of chaos. For example, Stachenfeld et al. (2021) present a convincing turbulence model by some metrics but don't discuss their time-evolution in the critical context of the chaotic timescale. One may see predictions diverge from ground truth and incorrectly assume that this means their model architecture performs poorly. In reality, the system is chaotic, we expect predictions to diverge on a finite timescale.

In chaotic systems, nearby trajectories diverge exponentially in time, at a rate characterized by the Lyapunov exponents λ_i. The largest Lyapunov exponent λ_1 determines the timescale over which predictions remain valid, known as the Lyapunov time $\tau_\lambda = 1/\lambda_1$. For a system with state vector $\mathbf{x}(t)$, the divergence of nearby trajectories is given by:

$$|\delta\mathbf{x}(t)| \approx e^{\lambda_1 t}|\delta\mathbf{x}(0)| \tag{3.1.1}$$

where $\delta\mathbf{x}(t)$ represents the separation between trajectories.

In many cases, computing Lyapunov exponents (and thus Lyapunov times) is non-trivial and often done with numerical methods (Sandri, 1996). For this reason, we build on the work of Edson et al. (2019), which suggests a maximum Lyapunov exponent ~ 0.1 for the KS equation on a spatial domain of $N = 64$. To best understand model performance in the context of this chaotic system, we study how trajectories diverge in neural models compared to the traditional solver across Lyapunov times, and in the traditional solver itself when perturbed at various scales (see Figure 3.1)

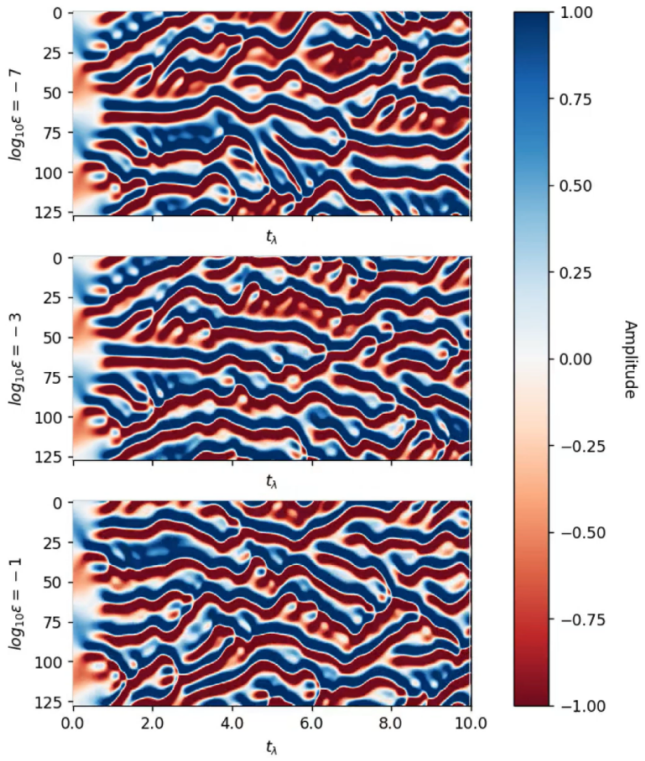


Figure 3.1: Rolled out time evolutions from the numerical solver, perturbed from the floating point error scale to 10% of the maximum amplitude. Transitions to chaos can be observed on the order of the Lyapunov timescale, as visualized in this video https://www.emcf.xyz/kschaos

3.1.2 Symmetries of the KS equation

The Kuramoto-Sivashinsky equation exhibits several symmetries (Cvitanovi et al., 2010). All of which are shared with 3D hydrodynamical turbulence.

1. **Translational Symmetry:** Represented by the group $\mathbb{Z}^1$,

$$T_\delta u(x,t) = u(x+\delta,t)$$

   leaves the equation unchanged for any constant δ.

2. **Reflectional Symmetry:** Represented by the cyclic group C_2,

$$Ru(x) = -u(-x)$$

   leaves the equation unchanged. Note that this requires negation of the field itself, unlike the reflection transformation for which the hydrodynamics equations are equivariant, $Ru(x) = u(-x)$

3. **Galilean Invariance:** Represented by the group $SGal(3)$,

$$u(x-ct,t) = u(x,t) - c$$

   leaves the equation unchanged for any constant c. Since u is a velocity, describes a transformation moving reference frame with constant relative velocity c

3.1.3 Data generation

To generate evolution data for the KS equation, we use a spectral scheme that combines the Crank-Nicolson method for the linear terms and the second-order Adams-Bashforth method for the nonlinear terms. To further test generalization of neural networks in our own study, we apply group operations to the dataset corresponding to the symmetries of the system at hand as a form of data augmentation. We generate a reshaped dataset by sampling the entire flow field at each timestep and pairing it with the following timestep.

3.1.4 Training and validation

We introduce `KSBench`, a benchmarking code to formally evaluate any model architecture for solving the time-evolution of turbulent flows. `KSBench` measures the training dynamics, rolled out prediction errors, and the spectrum of the $u(x)$ field in the KS equation. It accepts a wide variety of model architectures, and accomodates input structures for both sequential (Seq2seq) and non-sequential models. To promote fair comparison, we initialize each model between 80,000 to 90,000 trainable parameters, with model depth and layer width varying to accommodate each model's architectural constraints. We train for 5 epochs with a learning rate of 10^{-4} with 17000 timesteps generated across 100 different initial conditions.

Models are validated in three ways: (one) mean squared error in the $u(x)$ field during training, (two) mean squared error as they evolve a flow over time, and (three) the spectrum of the $u(x)$ field, time-averaged after the transition to chaos. The MSE is given by:

$$\text{MSE} = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$$

where y_i is the true value, $\hat{y}_i$ is the predicted value, and N is the number of samples.

The spectrum is computed using the discrete Fourier transform (DFT) of the spatial field $u(x)$ at each time step:

$$|\hat{u}(k)|^2 = \left|\sum_{x=0}^{N-1} u(x)e^{-2\pi ikx/N}\right|^2$$

where k is the wavenumber, N is the number of spatial points, and $\hat{u}(k)$ is the Fourier coefficient at wavenumber k. The time-averaged spectrum is then calculated by averaging $|\hat{u}(k)|^2$ over multiple time steps after one Lyapunov time:

$$\langle|\hat{u}(k)|^2\rangle_t = \int_{t_\lambda}^{\infty} |\hat{u}(k,t)|^2 dt$$

3.1.5 The NN models

We evaluate seven different neural network architectures for solving the time-evolution of the Kuramoto-Sivashinsky equation. each model was chosen to have between $80,000$ and $90,000$ parameters. We aimed to use a consistent number of layers and neurons per layer, however these numbers vary widely due to architectural constraints inherent within the network (for example, doubling the dimension of the transformer embedding layer quickly explodes its total parameter count by over an order of magnitude. The U-Net suffers a similar issue with forming a fair architecture: channel counts halving/doubling in each en- coder/decoder step mean that adding one layer

to the encoder necessitates an additional layer in the decoder, and an additional doubling of channel count in the hidden layers. The effects of these architectural constraints on the depth and width of the networks is explored in 4.1.

- Vanilla CNN: A standard convolutional neural network with 9 1D convolutional layers (1 encoder, 7 hidden, 1 decoder), each with 64 channels and kernel size 3.

- GCNN: A group-equivariant convolutional neural network with 1 layer to lift into the reflection group R, 4 group convolutional layers, and 1 final convolutional layer. Hidden layers have 42 channels and kernel size 3.

- Dilated ResNet (DilatedResidualCNN): A CNN with 1 input convolutional layer, 1 dilated residual block (containing 7 dilated convolutions with rates [1, 2, 4, 8, 4, 2, 1]), and 1 output convolutional layer. All layers have 64 channels and kernel size 3.

- Dilated ResNet GCNN (DilatedResidualGCNN): A group-equivariant version of the dilated CNN, with 1 layer to lift into the reflection group R, 1 dilated residual block (7 group dilated convolutions), and 1 final convolutional layer. Hidden layers have 32 channels and kernel size 3.

- ConvLSTM: A convolutional long short-term memory network with 1 ConvLSTM layer (84 hidden channels, kernel size 3) followed by a 1x1 convolutional layer.

- U-net: A U-shaped convolutional network with 3 encoder blocks, 1 bottleneck block, and 3 decoder blocks. Each block contains 2 convolutional layers. The

bottleneck has 96 channels, with channel counts halving/doubling in each encoder/decoder step.

- Transformer (TNT1D): An attention-based model with patch embedding (patch size 16), positional and temporal embeddings, 1 transformer encoder layer with 1 attention head, and an output projection. The embedding dimension is 20.

3.1.6 Results

Figure 3.2 shows the rolled out test set predictions versus Lyapunov time for each model architecture, along with the ground truth data. This visual representation allows us to directly observe how well each model captures the spatiotemporal evolution of the KS equation.

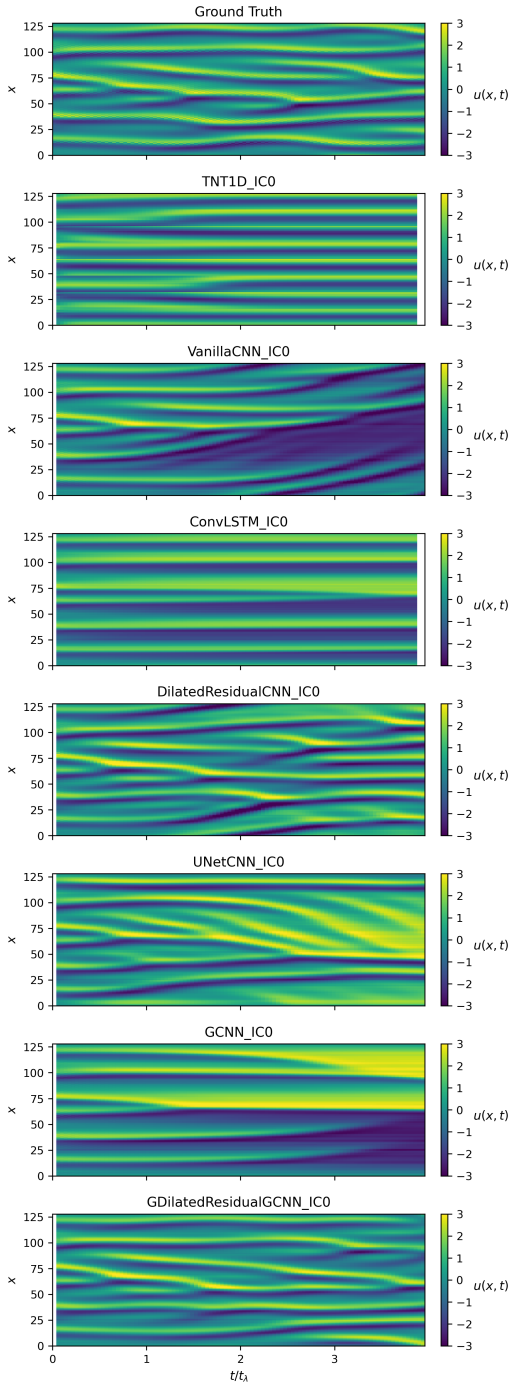


Figure 3.2: Rolled out test set predictions vs Lyapunov time for each model architecture (lower), along with ground truth data (top).

Figure 3.3 shows the training dynamics and rollout performance of different architectures. Despite showing similar loss during training, the dilated GCNN performs slightly better when generalized to our test data than the dilated CNN. Models with no ability to aggregate wide-range context along the spatial domain (the Vanilla CNN, GCNN, and ConvLSTM) are unable to predict a stable and accurate time-evolution. The U-net and transformer are also unable to make convincing rolled out predictions (see figure 3.2) in this prescribed testbed scenario, despite their ability to learn spatial dependencies on any scale.

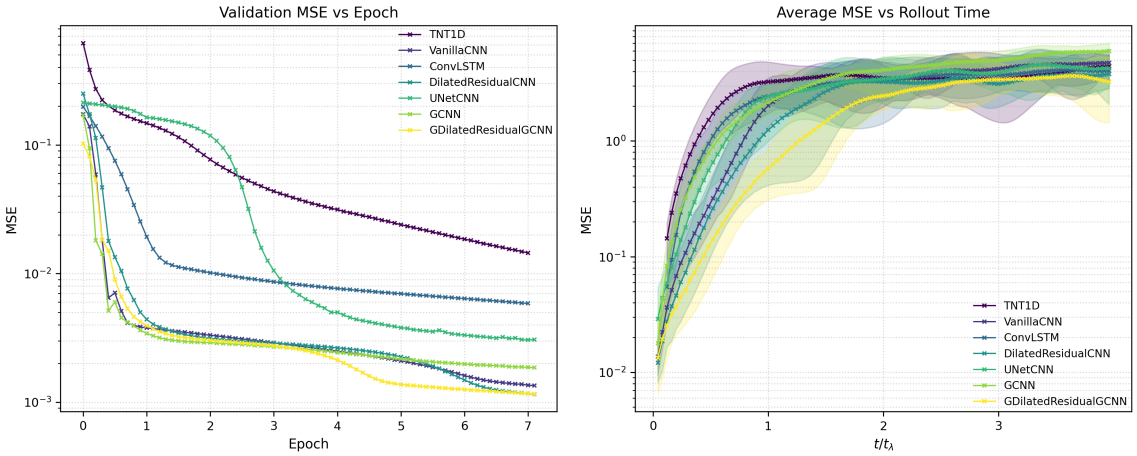


Figure 3.3: MSE vs training batch (left) and average MSE of rolled out time evolutions (right) for a variety of model architectures.

The time-averaged spectrum in figure 3.4 shows that dilated networks, particularly the dilated residual GCNN, excel in capturing both large-scale modes and the peak of the spectrum. This ability to accurately represent the flow across a wide range of length scales is critical for turbulent flows, thus we expect this model would perform exceptionally well for 3D hydrodynamical turbulence. The dilated architectures performing better, and the equivariant model outperforming other models, aligns with findings in the literature (Cohen and Welling, 2016; Stachenfeld et al., 2021).

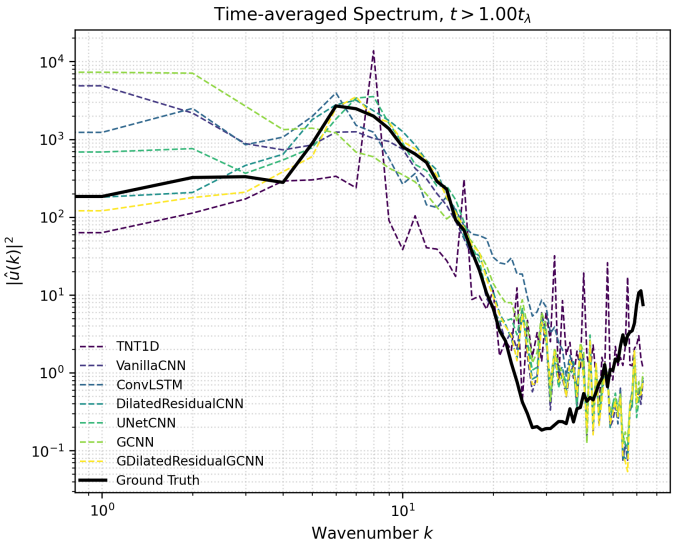


Figure 3.4: Spectra, time-averaged throughout the chaotic regime $t > t_\lambda$. Dilated networks are better able to capture large modes corresponding to long-range spatial dependencies. They also better match the solver at the peak of the spectrum, corresponding to the filament width.

The vanilla CNN and GCNN, while showing reasonable performance during training, struggle with long-term predictions. This is likely due to their limited receptive field, which prevents them from capturing the full range of spatial interactions in the system.

The ConvLSTM, despite its ability to model temporal dependencies, does not perform as well as the dilated architectures. This suggests that for the KS equation, capturing long-range spatial dependencies is more critical than explicit modeling of temporal dependencies.

The U-net and transformer, despite their theoretical ability to capture dependencies at any scale, do not perform as well as expected in this context. In the case of the transformer, the poor performance is likely an outlier due to the parameter count limit constraining the architecture required for the transformer to learn a sufficient

evolution operator. This is further discussed in 4.1.

The dilated residual CNN and dilated residual GCNN consistently outperform other architectures across all metrics. These models effectively capture spatial dependencies within an appropriate range, unlike models with with a limited receptive field (Vanilla CNN, GCNN, and ConvLSTM) The group-equivariant nature of the GCNN models better respect the symmetries inherent in the KS equation, leading to improved generalization.

3.2 Thermochemical Evolution

Chemistry is increasingly important to interpret astrophysical simulations. We observe the universe in specific emission and absorption lines, so we need to know what state elements are in to make predictions that can be compared to observed data. We also have advancements in simulations methods that can follow feedback in detail (creating the elements and evolving their abundances) and predict the radiation fields that photo-ionize and excite ions and molecules (For example, `RAMSES-RTZ` (Katz, 2022)). This leads to complex chemical reaction networks of over one hundred species with many hundreds of reactions that need to be solved. In addition, these reactions are typically fast so that the solvers may need to take many steps over a typical hydrodynamical timestep. This has created a situation where traditional numerical solvers for chemical networks can be the most expensive part of some modern simulations. Faster neural network solvers are thus an active area for study.

Prior work described in section 2 shows progress in using neural networks to simulate astrochemical systems. However, there are gaps in these studies such as

selecting relatively simple networks, studying limited cases, and learning a fixed-timestep solver.

In this work, we seek to address the limitations of existing astrochemical models. Firstly, future solvers will need to handle significantly larger ranges of densities, temperatures, and metallicities and with significantly more chemical species than previous studies. Secondly, we could benefit from the ability to efficiently and accurately integrate abundances and temperature over arbitrary timescales significantly larger than the reaction timescale.

3.2.1 Functions of many parameters

The cooling rate can be considered to be a function of many parameters. The basic ones are temperature, density and metallicity. Radiation fields require 6 or more parameters to describe. Modern metal yields typically require at least 3 parameters covering alpha element, iron and carbon pathways. However, these approaches assume equilibrium. If we wish to treat non-equilibrium conditions, the count rapidly exceeds 100 as every ion or molecular abundance must be considered as an input parameter.

Many prior approaches have relied on tabulated cooling rate data, accessed at runtime. A brute force approach would require an N dimensional table for N input parameters. Even just 10 points per parameter would require 4 terabytes of data for $N = 12$ parameters (4×10^N bytes per float, accessible on every core). Even a bare bones 2 points per parameter hits this limit with $N = 40$ parameters.

We find favorable memory, accuracy, and time trade-offs with neural networks compared to tables for approximating functions with a large dynamic range across a large number of dimensions, such as the physical parameter space of an astrophysical

gas with many parameters (e.g. abundances and state information such as density, temperature and radiation fields).

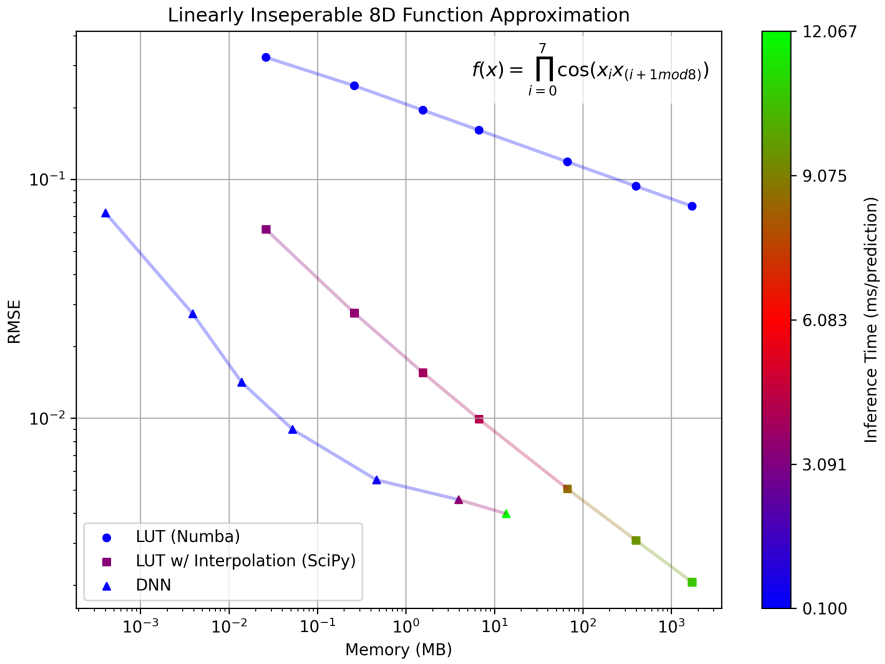


Figure 3.5: We compared memory requirement, inference time, and root mean squared error for a lookup table, an interpolated lookup table, and a deep neural network. Neural networks show favourable errors and inference times when tested with comparable memory requirements, with a diminishing return on model size. Inference times are wall-clock measurements on an 8 core CPU and an NVidia RTX 3060 GPU

Further motivated by the result from this test, we begin constructing a feed-forward network to predict the nonequilibrium evolution operator for the (very high dimensional) thermochemical state of an astrophysical gas.

3.2.2 Data generation

For our best performing network, We create a training set representing the thermochemical evolution of a gas at a wide range of physical parameters (density, temperature, and metallicity), and we train a feed-forward neural network to predict the chemical abundances and thermal state of the gas every 10 timesteps of the numerical solver. We use `CHIMES` (Richings et al., 2014a,b) to generate our training dataset. The physical parameter space was sampled as follows:

Variable	**Range**	**Sampling**
$\log T$ (Temperature)	$[2, 6]$ K	$\Delta \log T = 0.2$
$\log n_{\rm H}$ (Density)	$[-2, 2]$ cm^{-3}	$\Delta \log n_H = 0.2$
$\log Z$ (Metallicity)	$[0, 0.5]$	$\Delta \log Z = 0.1$
Δt (Timestep)	1 Myr	Constant
$\log \frac{n_i}{n_H}$ (Abundances)	[-10, 0]	N/A*
ζ (Cosmic Ray Rate)	1.8×10^{-16} s^{-1}	Constant
UV Field	Haardt and Madau (2001)	Constant

Table 3.1: Physical variables and ranges for our CHIMES Simulations, excluding chemical abundances

*Abundances were chosen by letting runs with no thermal evolution reach equilibrium.

To efficiently sample across a large number of physical variables, we run a number of simulations without thermal evolution and run them to equilibrium, creating plausible initial conditions for any given density and temperature. It should be noted that individual runs which took prohibitively long (> 5 minutes) to reach equilibrium

were automatically ended and used as-is.

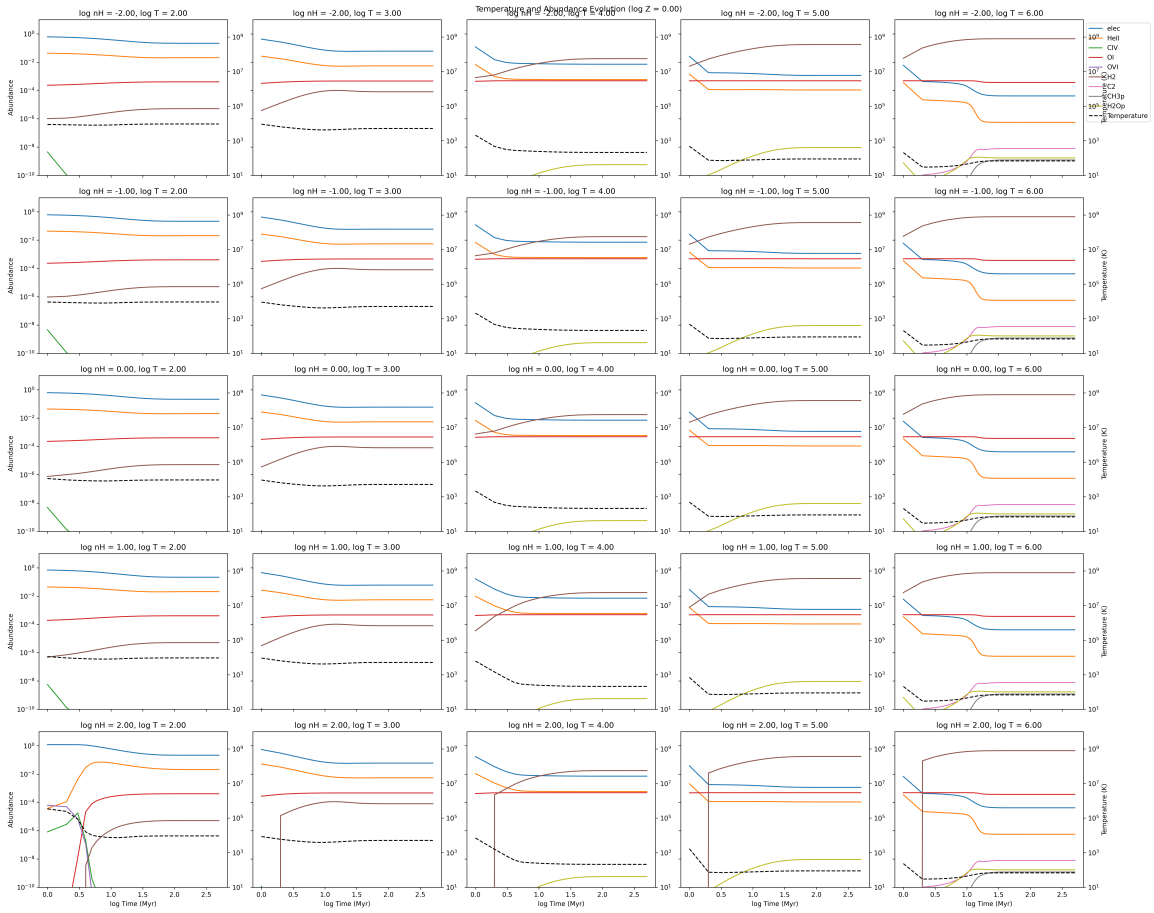


Figure 3.6: Samples of time-evolution of chemical abundances from the `CHIMES` solver, at various n_H and T, plotted at a constant Z

3.2.3 Time Evolution Data

We wish to use this data to train a network to evolve between two states at two different times. The two states will not necessarily be at equilibrium so all the abundances are both input and output parameters. For time evolution of our initial attempt, we also need to input Δt as a parameter. For our final model we use constant timestep and thus do not need to include this parametre.

Our data sample preparation pipeline consists of the following steps:

1. A sliding window method, moving two pointers t_1 and t_2 along the time dimension to sample chemical and thermal states.

2. Log-scaling and clipping of abundance ratios to the range $[-10, 1]$ and the temperature to the range $[0, 10]$.

3. Skipping of "stagnant" data where $\Delta T < 0.1$ K

4. Normalization of all data to the range $[0, 1]$ for stable training.

3.2.4 Network architecture

We test a ResNet (He et al., 2015) with encoder-decoder layers to allow the efficient encoding of relationships in the high-dimensional chemical network. The network consists of:

- 2 Encoder layers to for initial state vector (temperature, density, abundances)

- 8 Residual blocks, each containing 2 fully connected layers with 512 neurons, with activation functions of learnable steepness

- 2 decoder layers to output the evolved state vector

When trained with a activation function $\sigma(x) = \tanh(x/s)$ as introduced by (Sulzer and Buck, 2023), the steepness s can offer better fits at sharper gradients (see figure 3.7). We implement these activations throughout the network.

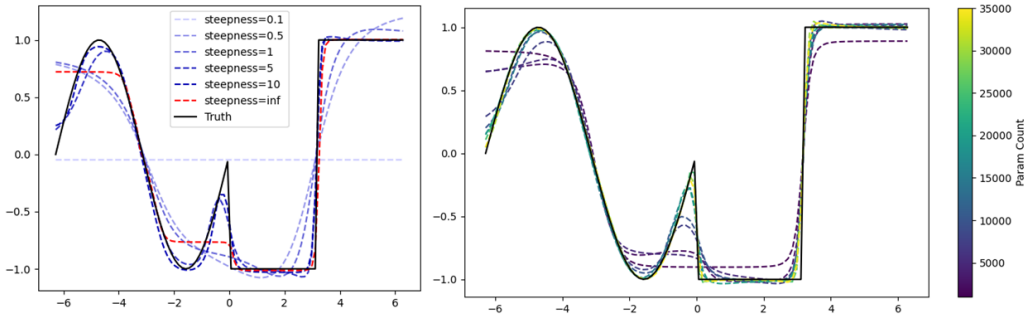


Figure 3.7: Our test of various activation functions and model scales in capturing a sharp discontinuity. Networks that are larger in scale are better able to approximate sharper gradients. Adding a trainable steepness parameter in the activation also improves performance on this task.

3.2.5 Training and validation

We are working with a larger array of species and a wider dynamic range of temperatures and density than previously demonstrated networks. Thus, we want to learn a very efficient latent space representation to reduce the dimensionality of the problem. Real astrochemical systems are highly constrained by underlying physics and how they evolved to their current state. For example we do not expect arbitrary combinations of high and low ionization states. For this reason, we equip the network with an encoder, ResNet processor, and decoder layers, trained to learn the identity function on the given data.

3.2.6 Results

Many issues arose during our initial training runs, such as failure for the loss function to converge to acceptable levels of error when trained on a dataset containing variable heating rates Γ. Initially, we also attempted to allow an arbitrary timestep as an

input, to allow the model to learn the entire time-integration of the ODEs, rather than predicting for a fixed timestep. Our interpretation is that the initial target was too general and thus difficult for the network to learn, possibly due to insufficient flexibility (free parameters) available to the network or a lack of sufficient training data.

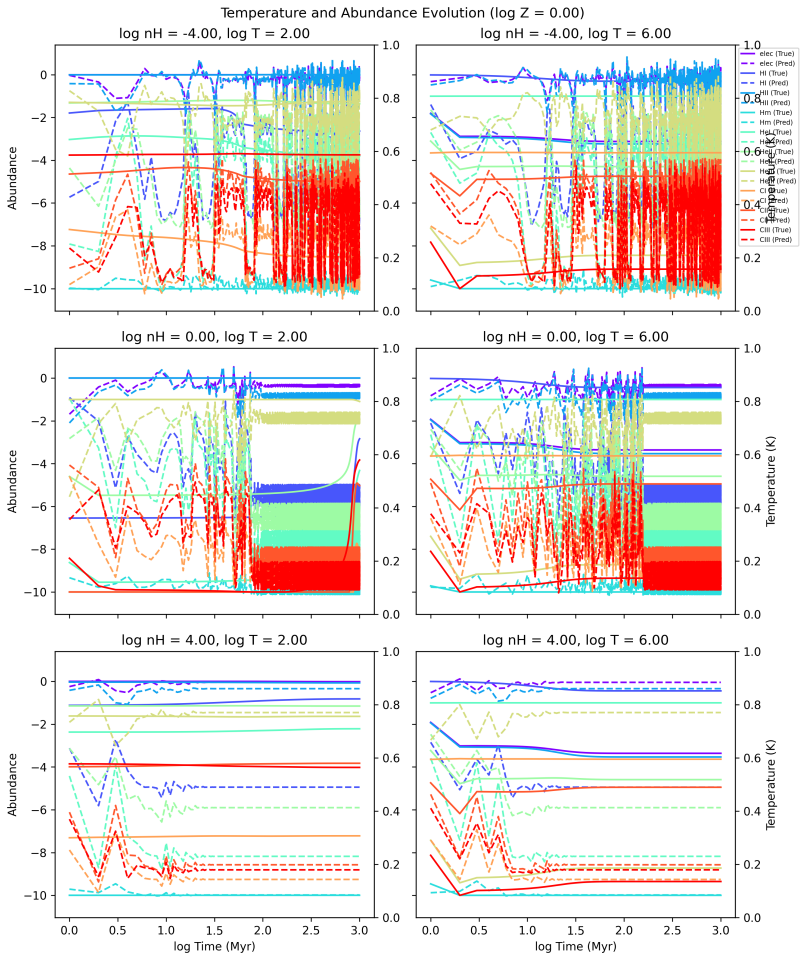


Figure 3.8: Predicted chemical abundance evolution compared to the ground truth data, shown for select species. This highly unstable model was trained on a very large number (157) of species, and an adaptable timestep input Δt. The problem domain was also very large compared to previous works discussed in 2.3, spanning across $-4 < n_H < 4$, $2 < T < 8$, and $-1 < Z < 0.5$

Through many iterations of model sizes, dataset sizes, and problem simplification, we discovered that learning to make stable predictions with an adaptable timestep was exceptionally difficult, and this was compounded by the number of species. Only after constraining the problem domain to only variable temperatures, densities, and metallicities, did we arrive at a model with acceptable performance.

We evaluated the performance of our neural network model in predicting the thermochemical evolution of astrophysical gases across a range of initial conditions. Figure 3.9 presents the predicted trajectories alongside the ground truth for various species and temperature, under different initial conditions of n_H and T at a fixed metallicity $log(Z) = 0$ within the test set.

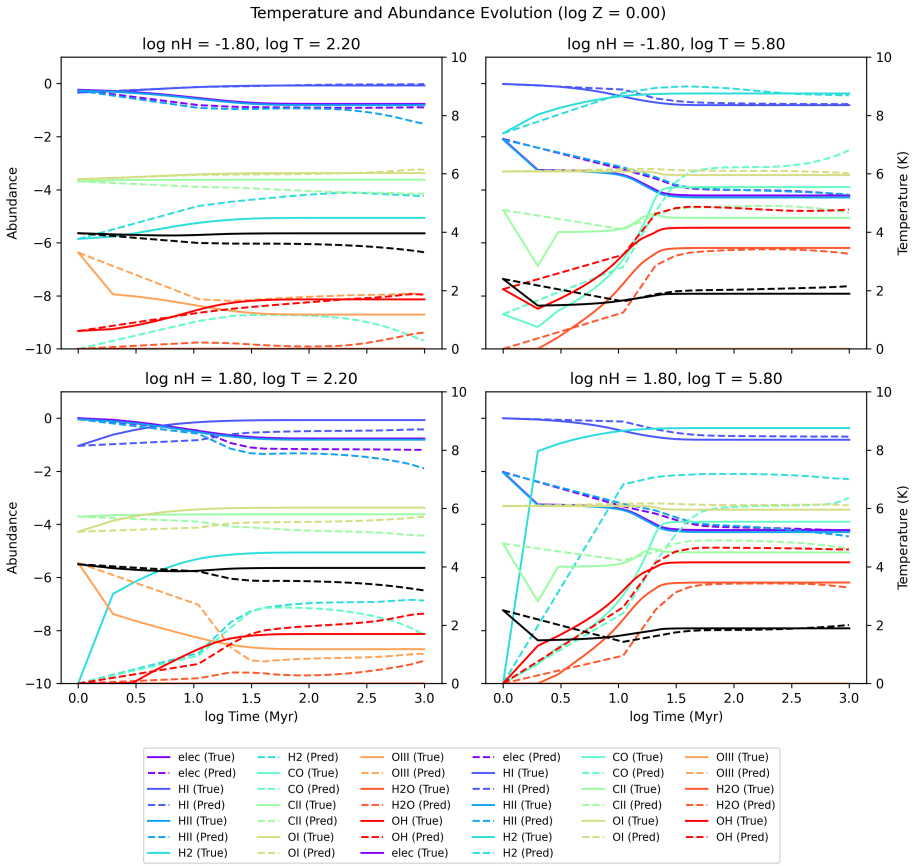


Figure 3.9: Predicted thermochemical trajectories for the test set. Predicted trajectories do not precisely follow the ground truth, however it is notably able to predict, within the correct order of magnitude, the final equilibrium state for many species. Errors are shown for select species of interest.

To quantify the model's performance, we calculated error statistics for the final timestep predictions on the test set. Table 3.2 shows the mean, maximum, and minimum absolute errors in log space for temperature and selected species.

Table 3.2: Mean absolute error (MAE), maximum absolute error, and minumum absolute error in log space for final timestep ($t = 1000$ Myr) predictions, after 100 rolled out inferences of the network

Output	MAE	Max	Min
T (K)	0.5119	0.8550	0.0122
$n_{\mathrm{HI}}/n_{\mathrm{H}}$	0.4881	0.5871	0.4012
$n_{\mathrm{HII}}/n_{\mathrm{H}}$	0.8113	1.735	0.3258
$n_{\mathrm{H_2}}/n_{\mathrm{H}}$	1.217	1.678	0.9501
$n_{\mathrm{CO}}/n_{\mathrm{H}}$	1.337	1.859	0.5309
$n_{\mathrm{CII}}/n_{\mathrm{H}}$	0.5711	1.056	0.0798
$n_{\mathrm{OI}}/n_{\mathrm{H}}$	0.2478	0.4555	0.1010
$n_{\mathrm{OIII}}/n_{\mathrm{H}}$	0.7556	2.277	0.1372
$n_{\mathrm{H_2O}}/n_{\mathrm{H}}$	0.7701	1.398	0.2930
$n_{\mathrm{OH}}/n_{\mathrm{H}}$	0.7380	1.685	0.0983

Chapter 4

Conclusion

4.1 Discussion

Our review in 2 revealed several trends in the application of neural networks to fluid dynamics and astrochemistry. In fluid dynamics, we observed a shift towards architectures that can capture long-range spatial dependencies, such as the dilated convolutions used by Stachenfeld et al. (2021). For astrochemistry, recent works like Holdship et al. (2021) and Branca and Pallottini (2023) have demonstrated the potential of emulating complex chemical networks across a given range of physical states, albeit with limitations in the range of physical parameters and number of species considered.

Our experiments with `KSBench` offer a much clearer direction towards improving state-of-the-art neural turbulence models. They suggests that future turbulence models can likely benefit from aggregating context hierarchically throughout the spatial domains, and incorporating relevant group symmetries into the network's architecture. Given that the orders of the symmetry groups involved in the 3D hydrodynamics

equations are greater than that of the 1D KS equation, one may expect equivariance to symmetry transformations to become more important. The U-Net is unable to make convincing rollout predictions; we suspect this is due to the downsampling process at each layer causing significant information loss through each forward pass. Indeed, Stachenfeld et al. (2021) preferred a dilated architecture to a U-Net architecture in the same setting, although by a less significant margin. The Transformer and the Convolutional LSTM are also unable to effectively learn dynamics within the given training epochs. For the transformer, this is likely a due to the architectural constraints required for this scenario: to keep the transformer network below 90,000 parameters, the latent dimension must be prohibitively small (about 4x smaller than models without attention mechanisms). To ensure a fairer comparison between models, an obvious solution would be to increase the parameter count of all tested models to allow the transformer to effectively encode the dynamical state. This, given the small spatial domain of the dataset, would invite the overfitting problem – which would likely require us to artificially increase the spatial domain to increase the complexity of the problem.

Our thermochemistry model demonstrates that neural models can feasibly learn an evolution operator for thermochemical network across a large number of species, by reducing the training space to near-equilibrium states. As shown in Figure 3.9, the model captures the overall trends for many species and temperature profiles, even when using a timestep 10x larger than that of the `CHIMES` solver code. Errors in table 3.2 show we can often predict abundances and temperatures within an order of magnitude. An interesting observation is the ability to find a roughly correct correct equilibrium state despite deviations in the transient behavior. Notably, our

neural network model is able to produce predictions on a significantly larger (10x) timestep than the original solver. This demonstrates the potential for neural network approaches to overcome traditional time-stepping constraints in stiff thermochemical systems. Furthermore, our network demonstrated a 8.74x faster wall-clock time to simulate 500 Myr on two NVidia V100 GPUs compared to `CHIMES` on a 16 core CPU.

The errors presented in table 3.2 are notably lower compared to the abundance errors produced by `Chemulator`, however it should be noted that `Chemulator` covers a much larger range in abundance and metallicity space than the model we present, albeit with fewer species. Models from Branca and Pallottini (2023) and Branca and Pallottini (2024) emulate 9 species exhibit significantly lower errors with only 9 species. These works along with ours demonstrate that neural models have the ability to achieve: low errors with many species in a slim range of physical conditions, low errors with few species in a wide range of conditions, and moderate errors when simulating many species across a wide range of conditions. This suggests that future models will be able to exploit an increased data and model scale to achieve low errors among many species and conditions.

4.2 Future Directions

4.2.1 Arbitrary timestepping

Many recent works (Branca and Pallottini, 2023; Dang et al., 2022; Holdship et al., 2021; Wang et al., 2019) focus on predicting the next timestep of a simulation. Our attempts were unable to successfully learn an evolution operator at arbitrary timescales, however we were able to take timesteps 10x larger than the solver code uses. Given

that neural models are not bound by the same timestepping constraints as finite difference methods, further work should be done to examine how to produce stable evolution predictions far above the solver code's timestep and at any timestep input by the user.

4.2.2 Self gravity

Early results from neural self-gravitating simulations take one further step towards fully neural astrophysical simulation. Auddy et al. (2024) introduced `GRINN` (Gravity-Informed Neural Network), a PINN for hydrodynamic systems with self-gravity. `GRINN` approximates the solutions ρ, $\mathbf{v}$, and ϕ as outputs of a neural network $\mathcal{N}(\mathbf{X};\theta)$, with $\mathbf{X}$ as space-time coordinates and θ as network parameters. The network is trained by minimizing a loss function incorporating PDE residuals and boundary/initial conditions. For 3D simulations, `GRINN` was over an order of magnitude faster than its finite difference method counterpart. As the model is a traditional MLP, the question of how to improve the efficiency of the architecture is a clear next step to explore. In particular, future work should explore what network architectures can learn to efficiently evolve a flow within a gravitational potential while maintaining the benefits of equivariance properties and dilation.

4.2.3 Radiative transfer

To extend our thermochemical model, a clear next step would be to incorporate radiative transfer effects by generating a new dataset using CHIMES that includes varying radiation fields, and retrain our model on this expanded parameter space. Robinson et al. (2024) explored using a gradient boosting (that is, learning an optimal ensemble

of weak prediction models) method, `XGBoost`, to predict gas cooling and heating functions in the presence of an incident radiation field, where their models outperformed traditional interpolation tables at fixed metallicity, reducing mean squared errors by over an order of magnitude.

4.2.4 Heating rates

To integrate our thermochemical model into a full hydrodynamic simulation, one should include PdV work to account for on the specific internal energy evolution due to adiabatic compression or expansion. This term may be incorporated into our neural network model as an additional input feature, however during our experiments we found no success in learning the complexity added by this seemingly simple additional energy parameter.

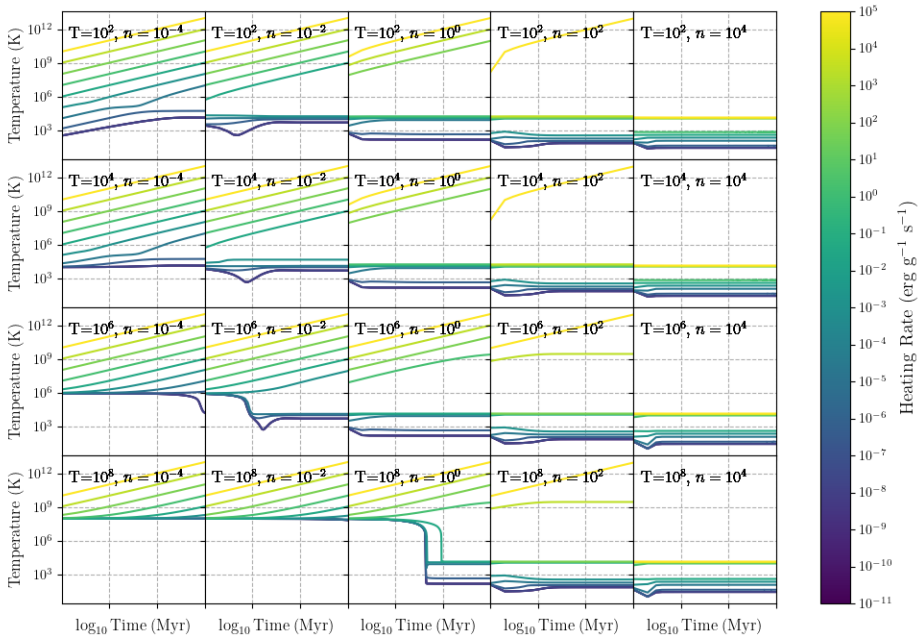


Figure 4.1: Thermal evolution history for thermochemical simulations with CHIMES at various constant heating rates. Although we prepared this larger dataset, it was not used due to failure to converge during training.

4.2.5 Kolmogorov-Arnold networks

Neural networks offer diminishing returns in error as the parameter count increases (as can be seen in figure 3.5), which Kolmogorov-Arnold Networks (KANs) (Liu et al., 2024) offer to remedy. Rather than learning weights connecting the nodes and passing results through fixed activation functions, KANs learn activation functions at the nodes and have no weights at all. Liu et al. (2024) ran benchmark results at various model scales which suggest that KAN performance does not taper off with model size with the same scaling law as traditional feed-forward neural networks.

With more favourable scaling laws, a large network should, in theory, be able to learn to make more accurate predictions in larger dynamical spaces.

4.2.6 Transformers

There is a growing trend in recent literature of Transformer-based methods replacing many MLP-based methods. Examples include ViT and MViT for general computer vision tasks, Turbulence Neural Transformer (TNT) (Dang et al., 2022) for turbulent hydrodynamics, and Chemformer for predicting the synthesis of chemical reactions (Irwin et al., 2022).

Building on our initial experiments with transformers in KSBench, it is evident that a modification must be made for a more comprehensive and fair evaluation of transformer-based architectures. Our previous transformer implementation was constrained by the parameter count limit, resulting in a prohibitively small latent dimension. To address this, we plan to increase the overall parameter budget for all models in our comparison, allowing for a more capable transformer architecture.

The importance of capturing long-range spatial information seems clear from our work and that of (Stachenfeld et al., 2021; Yu and Koltun, 2016). Building on our initial experiments with transformers in KSBench, a more comprehensive and fair evaluation of transformer-based architectures for turbulent flow prediction should be conducted. Our previous transformer implementation was constrained by the parameter count limit, resulting in a prohibitively small latent dimension. To address this, the overall parameter budget for all models could then be increased, allowing for a more capable transformer architecture.

Future studies will likely benefit from the improvement that the attention mechanism (Vaswani et al., 2017) offers in this direction, which can operate on information from arbitrarily long-range dependencies in its input sequence. Indeed, the recent advent of vision transformers (ViT) (**?**) and hierarchical techniques such as Multiscale ViT (MViT) (Fan et al., 2021) show promise to aggregate information from multiple length scales more efficiently than traditional ViT models, which have been successfully applied to computer vision problems.

There is ongoing discussion on the superiority of transformers to convlutional methods (Bai et al., 2021; Liu et al., 2022), including evidence that its better performance is not a direct result of its inherent architecture differences (Smith et al., 2023). If we increase the resolution of our solver and allow for models with larger parameter counts, These performance questions can be formally evaluated for physics problems using our benchmarking framework.

4.2.7 PINNs

The integration of physical constraints and domain knowledge into neural networks, as demonstrated by Physics-Informed Neural Networks (PINNs) (Raissi et al., 2019), offers a promising direction for improving accuracy and physical consistency. Physics-inspired neural ODE methods (Grassi et al., 2014; Sulzer and Buck, 2023) are promising candidates to be generalized to give robust predictions in a wide range physical conditions, and our own works could likely benefit from using physics-inspired terms in the loss function.

Bibliography

Agarap, A. F. (2018). Deep learning using rectified linear units (relu). *arXiv preprint arXiv:1803.08375*. Version 2, last revised 7 Feb 2019.

Agertz, O., Moore, B., Stadel, J., Potter, D., Miniati, F., Read, J., Mayer, L., Gawryszczak, A., Kravtsov, A., ke Nordlund, Pearce, F., Quilis, V., Rudd, D., Springel, V., Stone, J., Tasker, E., Teyssier, R., Wadsley, J., and Walder, R. (2007). Fundamental differences between sph and grid methods. *Monthly Notices of the Royal Astronomical Society*, 380:963–978.

Annibali, F. and Tosi, M. (2022). Chemical and stellar properties of star-forming dwarf galaxies. *Nature Astronomy*. arXiv:2201.06600 [astro-ph.GA].

Auddy, S. et al. (2024). Grinn: A physics-informed neural network for solving hydrodynamic systems in the presence of self-gravity. *arXiv preprint arXiv:2308.08010*.

Bai, Y., Mei, J., Yuille, A. L., and Xie, C. (2021). Are transformers more robust than cnns? In *Advances in Neural Information Processing Systems 34 (NeurIPS 2021)*. Curran Associates, Inc.

Bauer, A. and Springel, V. (2012). Subsonic turbulence in smoothed particle hydrodynamics and moving-mesh simulations. *Monthly Notices of the Royal Astronomical Society*, 423(3):2558–2578.

Bengio, Y., Simard, P., and Frasconi, P. (1994). Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, 5(2):157–166.

Boral, A., Wan, Z. Y., nez, L. Z.-N., Lottes, J., Wang, Q., Chen, Y.-F., Anderson, J., and Sha, F. (2023). Neural ideal large eddy simulation: Modeling turbulence with neural stochastic differential equations. In *Advances in Neural Information Processing Systems*, pages 69270–69283. NeurIPS.

Branca, L. and Pallottini, A. (2023). Neural networks: solving the chemistry of the interstellar medium. *Monthly Notices of the Royal Astronomical Society*, 518(4):5718–5733.

Branca, L. and Pallottini, A. (2024). Emulating the interstellar medium chemistry with neural operators. *Astrophysics of Galaxies (astro-ph.GA)*. Accepted for publication in A&A.

Brunton, S. L., Proctor, J. L., and Kutz, J. N. (2016). Discovering governing equations from data by sparse identification of nonlinear dynamical systems. *Proceedings of the National Academy of Sciences*, 113(15):3932–3937.

Bryan, G. L., Norman, M. L., O'Shea, B. W., Abel, T., Wise, J. H., Turk, M. J., Reynolds, D. R., Collins, D. C., and Wang, P. (2014). Enzo: An adaptive mesh refinement code for astrophysics. *The Astrophysical Journal Supplement Series*, 211:19.

Burgers, J. M. (1948). A mathematical model illustrating the theory of turbulence. *Advances in Applied Mechanics*, 1:171–199.

Chattopadhyay, A., Pathak, J., Subramanian, S., Harrington, P., Raja, S., Mardani, M., Kurth, T., Hall, D., Li, Z., Azizzadenesheli, K., Hassanzadeh, P., Kashinath, K., and Anandkumar, A. (2022). Fourcastnet: A global data-driven high-resolution weather model using adaptive fourier neural operators. *Proceedings of the International Conference on Learning Representations (ICLR)*.

Chen, R. T. Q., Rubanova, Y., Bettencourt, J., and Duvenaud, D. K. (2018). Neural ordinary differential equations. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc.

Cifuentes, L., Dopazo, C., Sandeep, A., Chakraborty, N., and Kempf, A. (2021). Filtering of combustion dns data using convolutional autoencoders. *Physics of Fluids*, 33(8):085119.

Cohen, T. and Welling, M. (2016). Group equivariant convolutional networks. In Balcan, M. F. and Weinberger, K. Q., editors, *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 2990–2999, New York, New York, USA. PMLR.

Courant, R., Friedrichs, K., and Lewy, H. (1967). On the partial difference equations of mathematical physics. *IBM Journal of Research and Development*, 11(2):215–234.

Cranmer, M. (2023). Interpretable machine learning for science with pysr and

symbolicregression.jl. *arXiv preprint arXiv:2305.01582*. `https://github.com/MilesCranmer/PySR`.

Cvitanovi, P., Davidchack, R. L., and Siminos, E. (2010). On the state space geometry of the kuramotosivashinsky flow in a periodic domain. *SIAM Journal on Applied Dynamical Systems*, 9(1):1–33.

Cybenko, G. (1989). Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals and Systems*, 2(4):303–314.

Dang, Y., Hu, Z., Cranmer, M., Eickenberg, M., and Ho, S. (2022). Tnt: Vision transformer for turbulence simulations. *arXiv preprint arXiv:2207.04616*.

Edson, R. A., Bunder, J. E., Mattner, T. W., and Roberts, A. J. (2019). Lyapunov exponents of the kuramotosivashinsky pde. *The ANZIAM Journal*, 61(3):270–285.

Fan, H., Xiong, B., Mangalam, K., Li, Y., Yan, Z., Malik, J., and Feichtenhofer, C. (2021). Multiscale vision transformers. *arXiv preprint arXiv:2104.11227*. Technical report.

Federrath, C., Klessen, R. S., Iapichino, L., and Beattie, J. R. (2020). The sonic scale revealed by the worlds largest supersonic turbulence simulation. *arXiv preprint arXiv:2011.06238*. arXiv:2011.06238v1 [astro-ph.GA].

Ferland, G. J., Porter, R. L., van Hoof, P. A. M., Williams, R. J. R., Abel, N. P., Lykins, M. L., Shaw, G., Henney, W. J., and Stancil, P. C. (2013). The 2013 release of cloudy. *Revista Mexicana de Astronoma y Astrofsica*, 49:137–163. Major review article to be published in RevMexAA. 27 pages, 18 figures.

Ferriere, K. M. (2001). The interstellar environment of our galaxy. *Reviews of Modern Physics*, 73:1031–1066.

Fryxell, B., Olson, K., Ricker, P., Timmes, F. X., Zingale, M., Lamb, D. Q., MacNeice, P., Rosner, R., Truran, J. W., and Tufo, H. (2000). Flash: An adaptive mesh hydrodynamics code for modeling astrophysical thermonuclear flashes. *The Astrophysical Journal Supplement Series*, 131:273–334.

Fukushima, K. (1980). Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological Cybernetics*, 36(4):93–202.

Galligan, T. P., Katz, H., Kimm, T., Rosdahl, J., Blaizot, J., Devriendt, J., and Slyz, A. (2019). deepcool: Fast and accurate estimation of cooling rates in irradiated gas with artificial neural networks. *arXiv preprint arXiv:1901.01264*.

Gibbs, J. W. (1899). Fourier's series. *Nature*, 59:606.

Gnedin, N. Y. and Hollon, N. (2012). Cooling and heating functions of photoionized gas. *The Astrophysical Journal Supplement Series*, 202(2):13. C 2012. All rights reserved. Printed in the U.S.A.

Grassi, T., Bovino, S., Schleicher, D. R. G., Prieto, J., Seifried, D., Simoncini, E., and Gianturco, F. A. (2014). Krome - a package to embed chemistry in astrophysical simulations. *Monthly Notices of the Royal Astronomical Society*, 439(3):2386–2419. accepted for publication in MNRAS.

Guan, P., Li, W., Brenner, M., and Meneveau, C. (2023). Learning physics-constrained subgrid-scale closures in the small-data regime for stable and accurate les. *arXiv preprint arXiv:2301.07347*.

Haardt, F. and Madau, P. (2001). Modelling the uv/x-ray cosmic background with cuba. In Neumann, D. and Van, J., editors, *Clusters of galaxies and the high redshift universe observed in X-rays, Recent results of XMM-Newton and Chandra, XXXVIth Rencontres de Moriond , XXIst Moriond Astrophysics Meeting*, Savoie, France.

Hartmann, J. (1904). Investigations on the spectrum and orbit of delta orionis. *Astrophysical Journal*, 19:268–286.

He, K., Zhang, X., Ren, S., and Sun, J. (2015). Deep residual learning for image recognition. *arXiv preprint arXiv:1512.03385*.

Hinton, G. E. and Salakhutdinov, R. R. (2006). Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, 9(8):1735–1780.

Holdship, J., Viti, S., Haworth, T. J., and Ilee, J. D. (2021). Chemulator: Fast, accurate thermochemistry for dynamical models through emulation. *Astronomy & Astrophysics*, 653:A76.

Hornik, K., Stinchcombe, M., and White, H. (1989a). Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366.

Hornik, K., Stinchcombe, M., and White, H. (1989b). Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366.

Irwin, R., Dimitriadis, S., He, J., and Bjerrum, E. J. (2022). Chemformer: a pre-trained transformer for computational chemistry. *Machine Learning: Science and Technology*, 3(1):015022.

Jetley, P., Gioachin, F., Kale, L. V., and Quinn, T. R. (2008). Massively parallel cosmological simulations with changa. In *Proceedings of IEEE International Parallel and Distributed Processing Symposium*.

Katz, H. (2022). Ramses-rtz: non-equilibrium metal chemistry and cooling coupled to on-the-fly radiation hydrodynamics. *Monthly Notices of the Royal Astronomical Society*, 512(1):348–365. Published: 18 February 2022.

Katz, N., Weinberg, D. H., and Hernquist, L. (1996). Photoionization and galaxy formation. *The Astrophysical Journal Supplement Series*, 105(1):19–50.

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Kolmogorov, A. (1941). The local structure of turbulence in incompressible viscous fluid for very large reynolds numbers. *Doklady Akademiia Nauk SSSR*, 30:301–305.

Kritsuk, A. G., Norman, M. L., Padoan, P., and Wagner, R. (2007). The statistics of supersonic isothermal turbulence. *The Astrophysical Journal*, 665(1):416–431. 2007. The American Astronomical Society. All rights reserved.

Kuramoto, Y. and Yamada, T. (1976). A reduced model showing chemical turbulence. *Progress of Theoretical Physics*, 56(2):681–684.

LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *Nature*, 521(7553):436–444.

Lecun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

Lee, K. and Carlberg, K. T. (2020). Model reduction of dynamical systems on nonlinear manifolds using deep convolutional autoencoders. *Journal of Computational Physics*, 404:108973.

Lemaster, M. N. and Stone, J. M. (2008). Density probability distribution functions in supersonic hydrodynamic and mhd turbulence. *The Astrophysical Journal*, 682(2):L97–L100.

Li, H., Xu, Z., Taylor, G., Studer, C., and Goldstein, T. (2018). Visualizing the loss landscape of neural nets. In *Advances in Neural Information Processing Systems 31 (NeurIPS 2018)*.

Li, Z., Kovachki, N., Azizzadenesheli, K., Liu, B., Bhattacharya, K., Stuart, A., and Anandkumar, A. (2021). Fourier neural operator for parametric partial differential equations. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Liang, J., Cao, J., Fan, Y., Zhang, K., Ranjan, R., Gool, L. V., and Timofte, R. (2022). Vrt: A video restoration transformer. *arXiv preprint arXiv:2201.12288*.

Liu, Z., Mao, H., Wu, C.-Y., Feichtenhofer, C., Darrell, T., and Xie, S. (2022). A convnet for the 2020s. In *Proceedings of the 2022 IEEE/CVF Conference on*

Computer Vision and Pattern Recognition (CVPR), pages 11–67, New Orleans, LA, USA. IEEE.

Liu, Z., Wang, Y., Vaidya, S., Ruehle, F., Halverson, J., Soljačić, M., Hou, T. Y., and Tegmark, M. (2024). Kan: Kolmogorov-arnold networks. *arXiv preprint arXiv:2404.19756*. 48 pages, 20 figures. Codes are available at this https URL.

Lu, Y., Chen, H., Lu, J., Ying, L., and Blanchet, J. H. (2021). Machine learning for elliptic pdes: Fast rate generalization bound, neural scaling law and minimax optimality. *arXiv preprint arXiv:2110.06897*.

McCulloch, W. S. and Pitts, W. H. (1943). A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, 5:115–133.

Menter, F. R. (1994). Two-equation eddy-viscosity turbulence models for engineering applications. *AIAA Journal*, 32(8):1598–1605.

Miyashita, Y. (2022). Operating principles of the cerebral cortex as a six-layered network in primates: beyond the classic canonical circuit model. *Proceedings of the Japan Academy, Series B*, 98(3):93–111.

Monaghan, J. J. (1992). Smoothed particle hydrodynamics. *Annual Review of Astronomy and Astrophysics*, 30:543–574.

Mylavarapu, R. T., Mylavarapu, B. K., and Sekhar, U. S. (2018). Interpolation of generalized functions using artificial neural networks. *Journal of Computer and Communications*, 6(7):41–52.

Padoan, P., Nordlund, A., and Jones, B. J. T. (1997). The universality of the stellar

initial mass function. *Monthly Notices of the Royal Astronomical Society*, 288:145–152. Accepted 1997 January 23, Received 1996 December 2, In original form 1996 July 5.

Perot, J. and de Bruyn Kops, S. (2008). Modeling turbulent dissipation at low and moderate reynolds numbers. *Journal of Fluid Mechanics*, 611:173–200.

Portwood, G. D. et al. (2019). Turbulence forecasting via neural ode. *arXiv preprint arXiv:1911.05180*.

Price, D. J. (2012). Resolving high Reynolds numbers in smoothed particle hydrodynamics simulations of subsonic turbulence. *Monthly Notices of the Royal Astronomical Society: Letters*, 420:L33–L37.

Raissi, M., Perdikaris, P., and Karniadakis, G. (2019). Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707.

Richings, A. J., Schaye, J., and Oppenheimer, B. D. (2014a). Non-equilibrium chemistry and cooling in the diffuse interstellar medium - i. optically thin regime. *Monthly Notices of the Royal Astronomical Society*, 440(4):3349–3369.

Richings, A. J., Schaye, J., and Oppenheimer, B. D. (2014b). Non-equilibrium chemistry and cooling in the diffuse interstellar medium - ii. shielded gas. *Monthly Notices of the Royal Astronomical Society*, 442(3):2780–2796.

Robinson, D., Avestruz, C., and Gnedin, N. Y. (2024). Exploring the dependence of gas cooling and heating functions on the incident radiation field with machine

learning. *Monthly Notices of the Royal Astronomical Society*, 528(1):255–269. Published: 18 December 2023.

Ronneberger, O., Fischer, P., and Brox, T. (2015). U-net: Convolutional networks for biomedical image segmentation. In *Proceedings of the International Conference on Medical Image Computing and Computer-Assisted Intervention (MICCAI)*.

Rosofsky, S. G. and Huerta, E. A. (2020). Artificial neural network subgrid models of 2-d compressible magnetohydrodynamic turbulence. *Phys. Rev. D*, 101:084024.

Ruggeri, M., Roy, I., Mueterthies, M. J., Gruenwald, T., and Scalo, C. (2022). Neural-network-based Riemann solver for real fluids and high explosives; application to computational fluid dynamics. *Physics of Fluids*, 34(11):116121.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature*, 323:533–536.

Sandri, M. (1996). Numerical calculation of lyapunov exponents. *Mathematica Journal*, 6(3):78–84.

Schmelzer, M., Dwight, R. P., and Cinnella, P. (2020). Discovery of algebraic reynolds-stress models using sparse symbolic regression. *Flow, Turbulence and Combustion*, 104:579–603.

Schmidt, W. and Federrath, C. (2011). A fluid-dynamical subgrid scale model for highly compressible astrophysical turbulence. *Astronomy & Astrophysics*, 528:A106.

Schmidt, W., Hillebrandt, W., and Niemeyer, M. J. (2006). A localised subgrid scale

model for fluid dynamical simulations in astrophysics. *Astronomy & Astrophysics*, 450:265–281.

Schneider, E. E. and Robertson, B. E. (2015). Cholla: A new massively parallel hydrodynamics code for astrophysical simulation. *The Astrophysical Journal Supplement Series*, 217(2):24.

Sirignano, J. and Spiliopoulos, K. (2018). Dgm: A deep learning algorithm for solving partial differential equations. *Journal of Computational Physics*, 375:1339–1364.

Sivashinsky, G. I. (1977). Nonlinear analysis of hydrodynamic instability in laminar flames. *Acta Astronautica*, 4:1177–1206.

Smith, B. D., Bryan, G. L., Glover, S. C. O., Goldbaum, N. J., Turk, M. J., Regan, J., Wise, J. H., Schive, H.-Y., Abel, T., Emerick, A., O'Shea, B. W., Anninos, P., Hummels, C. B., and Khochfar, S. (2017). Grackle: a chemistry and cooling library for astrophysics. *Monthly Notices of the Royal Astronomical Society*, 466:2217–2234. Advance Access publication 2016 December 17.

Smith, S. L., Brock, A., Berrada, L., and De, S. (2023). Convnets match vision transformers at scale. *arXiv preprint arXiv:2310.16764*.

Springel, V. (2005). The cosmological simulation code gadget-2. *Monthly Notices of the Royal Astronomical Society*, 364:1105–1134.

Squire, J. and Hopkins, P. F. (2017). The distribution of density in supersonic turbulence. *Monthly Notices of the Royal Astronomical Society*, 471(3):3753–3767.

Stachenfeld, K., Fielding, D. B., Kochkov, D., Cranmer, M., Pfaff, T., Godwin, J., Cui, C., Ho, S., Battaglia, P., and Sanchez-Gonzalez, A. (2021). Learned coarse

models for efficient turbulence simulation. *arXiv preprint arXiv:2112.15275*. (2022) International Conference on Learning Representations.

Stone, J. M., Tomida, K., White, C. J., and Felker, K. G. (2020). The athena++ adaptive mesh refinement framework: Design and magnetohydrodynamic solvers. *The Astrophysical Journal Supplement Series*, 249(1):4.

Sulzer, I. and Buck, T. (2023). Speeding up astrochemical reaction networks with autoencoders and neural odes. *arXiv preprint arXiv:2312.06015*. Accepted at the "Machine Learning and the Physical Sciences" Workshop at NeurIPS, 2023.

Sutherland, R. S. and Dopita, M. A. (1993a). Cooling functions for low-density astrophysical plasmas. *The Astrophysical Journal Supplement Series*, 88:253–327. Received 1991 May 6; accepted 1993 February 2.

Sutherland, R. S. and Dopita, M. A. (1993b). Cooling functions for low-density astrophysical plasmas. *Astrophysical Journal Supplement*, 88:253.

Teyssier, R. (2002). Cosmological hydrodynamics with adaptive mesh refinement: a new high resolution code called ramses. *Astronomy and Astrophysics*, 385:337–364.

Tielens, A. G. G. M. (2005). *The Physics and Chemistry of the Interstellar Medium*. Cambridge University Press, Cambridge, UK.

Udrescu, S.-M. and Tegmark, M. (2020). AI Feynman: A physics-inspired method for symbolic regression. *Science Advances*, 6(16).

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, ., and Polosukhin, I. (2017). Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008.

Wadsley, J. W., Keller, B. W., and Quinn, T. R. (2017). Gasoline2: A modern sph code. *Monthly Notices of the Royal Astronomical Society*, 471(2):2357–2369.

Wadsley, J. W., Veeravalli, G., and Couchman, H. M. P. (2008). On the treatment of entropy mixing in numerical cosmology. *Monthly Notices of the Royal Astronomical Society*, 387(1):427–438.

Wang, R., Cao, Z., Zhou, X., Wen, Y., and Tan, R. (2022). Physics-guided machine learning for wind-farm power prediction: Toward interpretability and generalizability. *ACM Transactions on Cyber-Physical Systems*.

Wang, R., Kashinath, K., Mustafa, M., Albert, A., and Yu, R. (2019). Towards physics-informed deep learning for turbulent flow prediction. *arXiv preprint arXiv:1911.08655*.

Wang, R., Kashinath, K., Mustafa, M., Albert, A., and Yu, R. (2020). Towards physics-informed deep learning for turbulent flow prediction. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*.

Weinberger, R., Springel, V., and Pakmor, R. (2020). The arepo public code release. *The Astrophysical Journal Supplement Series*, 248(2):32.

Wiersma, R. P. C., Schaye, J., and Smith, B. D. (2009). The effect of photoionization on the cooling rates of enriched, astrophysical plasmas. *Monthly Notices of the Royal Astronomical Society*, 393(1):99–107.

Yu, F. and Koltun, V. (2016). Multi-scale context aggregation by dilated convolutions. In *Proceedings of the International Conference on Learning Representations (ICLR)*.