UNIVERSITY OF CALIFORNIA
SANTA CRUZ

THE MOMENTUM YIELD OF CLUSTERED SNE; MACHINE LEARNING TO IDENTIFY DWARF GALAXIES

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

DOCTOR OF PHILOSOPHY

in

ASTRONOMY & ASTROPHYSICS
with an emphasis in STATISTICS

by

Eric S. Gentry

June 2019

The Dissertation of Eric S. Gentry
is approved:

Professor Piero Madau, Chair

Professor Mark Krumholz

Professor Alexie Leauthaud

Quentin Williams
Vice Provost and Dean of Graduate Studies

Table of Contents

List of Figures

List of Tables

Abstract

The Momentum Yield of Clustered SNe;

Machine Learning to Identify Dwarf Galaxies

by

Eric S. Gentry

Stars are born in clusters and massive stars die relatively quickly, so we expect core collapse supernovae (SNe) to be clustered in both space in time. Despite this, traditional SN feedback models assume SN blasts are isolated and do not interact. In my thesis I show that clustering *might* have a very large effect on the final momentum added into the SN's host galaxy's interstellar medium (*ISM*; as large as an order of magnitude increase), but also that the *actual* effect size is still unknown and requires further study.

Additionally, I perform experiments into machine learning-driven approaches to identifying nearby dwarf galaxies (a 300:1 class imbalanced problem) within a large (>100 TB) dataset of galaxy images. I show that our current training data and tools do a good start, but are ultimately too limited to get to satisfactory levels of completeness and purity for immediate use within a weak lensing analysis.

Acknowledgments

The text of this dissertation includes reprints of the following previously published material: Gentry et al. (2017, 2019). The co-authors listed in these publications directed and supervised research which forms the basis for the dissertation.

I thank the following people who have provided help, advice and comments on various stages of this work: Mark Krumholz, Piero Madau, Alexie Leauthaud, Alessandro Lupi, Marc Huertas-Company, Manfred Warmuth, Justin Brown, John Forbes, Anna Rosen, Brant Robertson, Norm Murray, Peter Mitchell.

Outside of the academic work, I would like to thank Zach Jennings for getting me back into running, so I can spend my energy on more than just work; after that, I would like the thank all of my friends from the Santa Cruz Track Club, who gave me such a great and varied community outside of the university.

Finally, I certainly cannot forget my parents and family.

Part I

Clustered SNe

Chapter 1

Introduction

Energy and momentum injection from supernovae (SNe) are thought to be one of the key ingredients regulating galaxy formation and the thermodynamics of the interstellar and circumgalactic media (ISM and CGM, respectively), but knowing exactly how *much* energy and momentum feeds back into the ISM is complicated.

At early times following a SN, it seems relatively simple. Detailed studies of massive star evolution, SN progenitors and their blasts can predict the mass, amount of metals and energy released by the blast (e.g., Sukhbold et al. 2016).

Unfortunately, the ISM does not actually retain all that mass, metals and energy. Instead most of the energy is typically radiated away (e.g., Thornton et al. 1998) largely at wavelengths at which galaxies are optically thin, allowing this radiated energy to escape. Similarly, if a SN is in a weak enough potential well, the SNR might "breakout" from the galaxy, launching a wind containing some of the SN ejecta materials (e.g., Mac Low et al. 1989). These types of losses, while complicating matters, are a key

component to our understanding of SN feedback: without it our models of SN feedback would be far too strong. In particular Dekel & Silk (1986) show that accounting for radiative losses of SN is necessary to construct populations of dwarfs with realistic properties, and that the amount of radiative losses could lead to different regimes of galaxy evolution at the low mass end of the spectrum.

Unfortunately, both radiative processes in the ISM and the development of shocks are non-linear processes, making it difficult to easily predict exactly how much energy will be lost and how that will impact the development and evolution of the SN's shockwave. To highlight the challenge, early galaxy simulators (e.g., Katz 1992) tried to directly inject 10^{51} erg per SN into hydrodynamic galaxy simulations and found that it had much less impact than expected; their galaxies were much too dense near the center and much too cold. While it was good that their numerical simulation could account for cooling, they were finding that there was an emergent *over-cooling* problem for which they had not accounted.

Numerical over-cooling, in the context of SN energy and the ISM, results from using a method that spreads a fixed amount of injected energy over too much gas mass, typically due to the limited numerical resolution of the method. While a SN should truly be able to heat its ejecta to 10^{51} erg$/(10 M_{\odot} k_{\rm B}/m_{\rm H}) \sim 10^9$ K shortly after the explosion, if the energy is spread across increasingly more mass, the average temperature within that mass will decrease. Initially (starting at high resolution) this is not a significant problem; the local gas mass is dominated by the ejecta mass. Furthermore, typical cooling functions within astrophysical plasma are less dependent of temperature at high

temperatures ($T \gtrsim 10^{6.5}$; see Draine (2011) figure 34.1). But once the temperature drops too much ($T \lesssim 10^{6.5}$), the efficiency of cooling starts to increases sharply with decreased temperature. This means that starting at very high resolution, the initial cooling rate will at first be insensitive to worsening resolutions, but then start becoming much too high once a critical resolution is passed.

This illustrates an important lesson that will come up in various forms: while the problem is complex, non-linear and resolution-dependent, there is still hope of a converged numerical answer with sufficiently high resolution (and good numerical methods). Alas, over-cooling is not just a problem at the moment of initial injection; it also problematic at later times when a SN remnant (SNR) has formed. The prototypical structure of a SNR is a hot, low-density bubble that contains most of the thermal energy, surrounded by a cold, high density shell that contains most of the mass. In theory the radiatively-efficient shell is separated from the thermal-energy-filled bubble by a contact discontinuity (e.g., Thornton et al. 1998), but as the resolution decreases, this line becomes blurred, the phases become mixed, and the calculated radiative cooling becomes too large.

Once again, this over-cooling problem does not require infinitely high resolution to solve. In reality, *physical* processes exist which can transport mass or energy across a contact discontinuity. (We will loosely refer to these as "mixing" processes, but do not mean to imply that it *must* be mediated by moving mass across the boundary.) While the "true" level of physical mixing should remain the same regardless of resolution, artificial numerically-induced mixing should become weaker as resolution is improved;

at high enough resolution, artificial mixing should be negligible compared to the physical mixing that is present. Thermal conduction is a classic example of a physical mixing process active in SNRs, transporting energy from the hot bubble into the cold dense shell. This energy transport depends on the gradient of temperature though, and that gradient is limited by the scales a simulation can resolve for a fixed jump in temperatures between $T_{\rm hot}$ and $T_{\rm cold}$. If this temperature jump occurs over a length scale larger than the Field length (Begelman & McKee 1990):

$$\lambda_{\rm F} = \left(\frac{\kappa(T) T}{n^2 \Lambda} \right)^{1/2} \tag{1.1}$$

(for a typical gas density n within the transition region, temperature-dependent thermal conduction constant κ, and cooling function Λ), then any energy conducted from the hot bubble is expected to be radiated away before it can reach the cold shell or significantly affect the structure of the transition layer in between. It is only when the temperature jump can be resolved at scales smaller than this characteristic Field length that significant amounts of energy can be conducted between the two layers, in turn allowing a more physical (resolution-independent) structure to develop in this interface region. Therefore, we can loosely expect that at resolutions worse than the Field length, artificial mixing will be dominant and resolution-dependent, but at resolutions better than the Field length, physical mixing will be dominant and relatively resolution-independent. (In reality thermal conduction is not the only source of "other" physics which can dominate mixing rates at high resolution; it simply serves as an example. The

important thing to remember is that at sufficiently high resolution, we believe we can achieve results that are converged with respect to resolution, even if those simulations do not specifically include thermal conduction.)

Even though galactic and cosmological simulations will not be able to realistically reach these resolutions in the near future, we can still use a *multi-scale* approach: run a "representative" set of simulations at very high resolution which hopefully span the relevant parameter space, and use the results of these simulations to inform "subgrid" models that can be plugged into low resolution simulations. As we will discuss in chapter 4, there are a variety of philosophies behind these subgrid models. Some try to mimic the true behavior of SNRs by temporarily disabling different physics modules; for example, Stinson et al. (2006) use an approach which disables radiative cooling in young SNRs, while Springel & Hernquist (2003) disable hydrodynamic forces in "ejecta particles". Other groups take a different approach, and use high resolution simulations to determine what impact the SNR will have at "late" times, and just change the local hydrodynamic conservative variables in low resolution simulations accordingly (for example, Hopkins et al. (2018b) simply inject the asymptotic radial momentum, rather than trying to self-consistently develop a SNR with the proper radial structure).

One important output of high resolution simulations is the asymptotic radial momentum; as mentioned a moment ago it is used in some subgrid models, and it is also a key part of some more analytical studies of galaxies (e.g., Hayward & Hopkins 2017). One of the major reasons why we care about the momentum from SNRs is that in disk galaxies turbulence appears to be one of the dominant forms of support (Kim et al. 2011)

and SNRs appear to be one of the largest contributors to that momentum (Agertz et al. 2013). As a convenient, but less direct benefit for studying the momentum (as opposed to other quantities like added energy or the hot gas mass), the radial momentum only changes slowly at late time (Kim & Ostriker 2015). In theory we will talk about this value being "asymptotic" with respect to time, but in practice we will either measure the maximum momentum or the momentum at a specified moment in time.

Finally, in order to create meaningful subgrid models, we need to identify the parameter space that accounts for the most variation in behavior among SNRs. Simulations have been run in a number of subspaces, studying the effects of: density (Cioffi et al. 1988), metallicity (Thornton et al. 1998), location within a molecular cloud (Iffrig & Hennebelle 2015), pre-stellar feedback (Walch & Naab 2015), turbulent velocity dispersion (Martizzi et al. 2015) and bi-stable ISM (Kim et al. 2017) just to name a few. Unfortunately, to ensure we have fully explored this parameter space using a grid of simulations would require the number of simulations to scale exponentially with the dimensionality of the parameter space[1] Therefore, we typically restrict ourselves; we will choose some of the dimensions that appear to cause the largest amount of variation (density and metallicity), and will add another dimension—the number of core collapse SNe coming from a given star cluster—which had shown interesting preliminary results but which was lacking a well-designed, systematic study.

We will call these "clustered SNe", as we expect the blasts to be clustered in both space and time. Stars are typically formed in clusters, and massive stars die

[1] Although if feedback strength varied smoothly across this space, the number of simulations required to fit an interpolating function might only scale linearly with the dimensionality.

relatively quickly, so we expect most of the SN blasts to occur near each other (ignoring the fraction of massive stars that are strongly kicked out of their birth clusters by dynamical interactions). If the delay between SNe is short enough (i.e. the blasts are sufficiently clustered in time), the energy from each blast can merge into a common *superbubble*. For simplicity we will assume *all* core collapse SNe from a cluster are clustered and merge into a single superbubble. To first order, we can include the fraction of "kicked" stars, variations in the IMF or similar effects that might change the number of SNe producing a superbubble, by parameterizing our fitted model as a function of the number of SNe. At higher order, these changes in which stars explode presumably affect things like the SN delay time distribution, the distribution of SN yields, and other similar characteristics. This parameterizing also will not account for SNe which are so far spaced in time that each SNR is mixed into the ISM before the subsequent SN occurs. Still, parameterizing the effect by the number of SNe will give a good place to start.

At the time this thesis began, the number of SNe resulting from a star cluster was the parameter that led to the largest variation in momentum yield (i.e., the average momentum per SN) in high resolution simulations, which had not been methodically studied at a range of scales. For example, Kim & Ostriker (2015) found that a series of clustered SNe might *decrease* the momentum yield by a factor of 2, whereas Walch & Naab (2015) found as much as a 25% increase in the yield due to clustering. More extreme yet, Keller et al. (2014) found that the momentum yield for a cluster's worth of SNe could be at least 5 times greater than the yield of a single isolated SN. This range

in results, at least an order of magnitude depending on how clustering is included, remains one of the largest known variations of SN feedback strength. Unfortunately, these previous studies were rather limited: clustering of SNe was only meant to be a minor part of each work. As a result, the simulations were too limited to build a broader understanding. Some used extremely unlikely SNe delay time distributions (e.g., (Walch & Naab 2015; Kim & Ostriker 2015)); some only measured momentum at a single cluster mass (Keller et al. 2014); others were unconverged with respect to resolution (Fierlinger et al. 2016).

In this half of my thesis, we systematically study how the clustering of SNe affects the resulting momentum yield. In chapter 2, we run a suite of 1D, high resolution, converged simulations across a variety of cluster sizes. This paints a broad picture of how clustering can affect the momentum yield of SNe. Guided by those results, we identify literature results that appear to be conflicting. Since the approaches that led to these conflicting results differ in many ways (both in numerical methods as well as physical setup), in chapter 3 we then run a number of 3D simulations to help us start to understand how much of the conflict comes from different levels of *artificial* mixing and how much is due to *physical* mixing. In chapter 4, we put these results into a broader context, by evaluating what errors might exist within existing galactic simulations which use previously-existing subgrid models (which typically do not explicitly account for clustering). We do this by running 1D simulations at high resolution without those models, and then comparing the results to low resolution simulations with a variety of subgrid models. In chapter 10, we then identify key areas that remain uncertain with

regard to clustering's impact on SN momentum yields.

Chapter 2

1D simulations of clustered SNe: Enhanced momentum feedback from clustered SNe

(Much of this text is going to come directly from Gentry et al. (2017); no changes to the substance have been made.)

Young stars typically form in star clusters, so the supernovae (SNe) they produce are clustered in space and time. This clustering of SNe may alter the momentum per SN deposited in the interstellar medium (ISM) by affecting the local ISM density, which in turn affects the cooling rate. We study the effect of multiple SNe using idealized 1D hydrodynamic simulations which explore a large parameter space of the number of SNe, and the background gas density and metallicity. The results are provided as a table and an analytic fitting formula. We find that for clusters with up to ~ 100 SNe

the asymptotic momentum scales super-linearly with the number of SNe, resulting in a momentum per SN that can be an order of magnitude larger than for a single SN, with a maximum efficiency for clusters with $10-100$ SNe. We argue that additional physical processes not included in our simulations – self-gravity, breakout from a galactic disk, and galactic shear – can slightly reduce the momentum enhancement from clustering, but the average momentum per SN still remains a factor of 4 larger than the isolated SN value when averaged over a realistic cluster mass function for a star-forming galaxy. We conclude with a discussion of the possible role of mixing between hot and cold gas, induced by multi-dimensional instabilities or preexisting density variations, as a limiting factor in the buildup of momentum by clustered SNe, and suggest future numerical experiments to explore these effects.

2.1 Introduction

Supernovae (SNe) play a key role in regulating star formation at galactic scales. SN energy, if retained, can disrupt molecular clouds and small galaxies (Dekel & Silk 1986). Even if significant energy is lost to radiative cooling, SNe inject momentum that cannot be radiated away, which drives turbulence, the dominant form of dynamical pressure support in galactic discs (Kim et al. 2011; Jenkins & Tripp 2011). This turbulent support both prevents the collapse of star-forming regions (locally limiting star formation; Ostriker & Shetty 2011; Faucher-Giguère et al. 2013), and drives galactic winds (globally limiting star formation; Murray et al. 2005; Hopkins et al. 2012; Dekel

& Krumholz 2013; Creasey et al. 2013; Thompson & Krumholz 2016).

Unfortunately, the processes controlling supernova remnant (SNR) evolution operate at smaller scales than what can typically be resolved by galaxy or cosmological simulations. In particular, the dense shells of SNRs rapidly radiate away most of the SN energy, leaving a cold dense shell and a hot diffuse interior. If a simulation cannot resolve these two zones, then it cannot realistically evolve the SNR, resulting in problems such as over-cooling (Katz 1992). To counteract this, some authors have prescribed turning off cooling for young, unresolved SNRs (Gerritsen 1997; Stinson et al. 2006), while others have proposed models which mimic the otherwise unresolved multi-phase nature of the interstellar medium (ISM) (Keller et al. 2014). These methods have their strengths, but the most direct way to incorporate the relevant physics is multiscale modeling: evolve a number of SNRs in a representative set of environments using simulations with high enough resolution to resolve the relevant processes, and use those results in large, low resolution simulations.

Early attempts at using high resolution simulations to create subgrid SN feedback models focused on producing energy-driven models (Thornton et al. 1998), but recently there has been increased interest in momentum-driven models, both by those using high resolution simulations to study SNRs directly (Martizzi et al. 2015; Walch & Naab 2015; Kim & Ostriker 2015; Iffrig & Hennebelle 2015) and by those who use such models in lower resolution simulations (Hopkins et al. 2011; Shetty & Ostriker 2012; Hennebelle & Iffrig 2014; Goldbaum et al. 2016). This change in emphasis has been driven by the realization that, while the energy content of SNRs is important for

producing hot galactic winds that are observable in X-rays, the momentum budget is more important when it comes to SNRs regulating star formation and possibly ejecting cool gas from galaxies (Dekel & Krumholz 2013).

At early times, before radiative losses are important, a SNR is in the *Sedov* stage, during which the energy is approximately conserved and the radial momentum is increasing. Once radiative losses become significant, it enters a *pressure-driven snowplow* phase, during which the energy is decreasing and the momentum is still increasing. As the bubble expands and cools (adiabatically and radiatively), its pressure will eventually decrease to the ISM pressure, at which point it becomes a *momentum-driven snowplow*. Asymptotically, in the idealized case of a spherical SNR expanding into a uniform, cold medium, this results in zero energy being added to the ISM, but a non-zero and finite amount of momentum being added. The goal of high resolution simulations is to follow all of these phases, and identify the asymptotic momentum as a function of the properties of the driving stars and the large-scale environment, making this value available for use in larger-scale models.

A number of authors have performed systematic parameter studies of the expansion of a SNR from a single SN in spherical symmetry (Chevalier 1974; Cioffi et al. 1988; Thornton et al. 1998). The most complete of these studies, that of Thornton et al. (1998), spanned metallicities from $10^{-3} - 10^{0.5}$ times Solar and ambient densities from $0.1 - 10^{3}$ H atoms cm^{-3}. More recently, there have been a number of 3D simulations which allowed the study of SNR evolution within a more realistic, non-spherically symmetric background. Martizzi et al. (2015), Kim & Ostriker (2015), and Walch & Naab

(2015) all found that inhomogeneities present prior to the first SN explosion – such as those expected due to a multi-phase structure of the ISM or ionized bubbles created by pre-SN radiation – did not change the final momentum by more than 60%. A more interesting effect was found by considering the inhomogeneities that result from bubbles of previous SNRs. Kim & Ostriker (2015) found that a series of clustered SNe can *decrease* the momentum per SN, in some cases by almost a factor of two. On the other hand, Walch & Naab (2015) found that multiple SNe might *increase* the momentum per SNe, by at least 25%, depending on the delay time between SNe. The dependence on delay time further complicates this discrepancy between authors since neither set of authors used realistic delay time distributions for the number of SNe considered. Yadav et al. (2017) used 3D simulations to study how clustered SNe can merge into a superbubble (using a realistic SN delay time distribution), but did not study the momentum produced and did not test the effect of gas metallicity. So we are left with a series of questions: for a realistic delay time distribution of clustered SNe, does the momentum per SN increase or decrease relative to single SN models, and by how much? Does the result depend on the density or metallicity of the environment in which the SNe explode? Does it vary systematically with the number of SNe that are clustered together?

In this paper we seek to measure directly the impact that clustering has on the momentum budget of SNe. In order to sample a wide range of densities, metallicities and cluster sizes, we create a suite of several hundred 1D, spherically-symmetric simulations. Using a 1D geometry means we lose the ability to simulate non-spherically symmetric

inhomogeneities but in doing so we gain the ability to probe a far wider parameter space than any previous studies of multiple SNe, and to achieve far higher spatial resolutions than previous works studying momentum feedback. As we will show, both are necessary for understanding how clustering impacts momentum feedback.

The remainder of this paper is as follows. In section 2.2 we discuss the numerical methods used in this study. The numerical results of our simulations are presented in section 2.3. In section 2.4 we use these results to build a model that can predict the momentum injection per SN as a function of density, metallicity and number of SNe, in a form suitable for inclusion in subgrid and analytic models. We discuss the significance of our results and model in section 2.5, comparing to previous works. Finally, we summarize our conclusions in section 2.6.

2.2 Numerical Methods

Our simulations make use of a custom-built 1D spherically-symmetric moving-mesh code that solves the finite volume equations of compressible hydrodynamics. Our code includes radiative cooling and injection of mass and energy by both SNe and pre-SN winds. The star cluster is assumed to lie at the centre of our simulation, with our computational domain beginning just outside the cluster and extending outwards. SN ejecta and pre-SN winds are added to the innermost zone of this domain. We run these simulations until all SNe have occurred and the momentum reaches a maximum.

In the rest of this section we go into greater depth on the numerical methods

used in our simulations and the limitations of our assumptions. In Appendix A.1 we test our code against both the analytic Sedov solution and the earlier numerical results of Thornton et al. (1998) for isolated SNe, and verify that it reproduces them well. For the interested reader, our code has been publicly released[1].

2.2.1 Initial Conditions

All our simulations begin with a star cluster of mass $M_{\rm cluster}$ placed at the origin surrounded by an initially-uniform, stationary ideal gas with adiabatic index $\gamma = 5/3$. We vary $M_{\rm cluster}$ from $10^2 - 10^5$ $M_\odot$ (using steps of 1 dex, with an additional step at $10^{2.5}$ $M_\odot$ to better resolve a key region of parameter space). We explore gas mass densities in steps of 1 dex, ranging from $\rho_0 = 1.33 \times 10^{-3} - 10^2$ $m_{\rm H}$ cm^{-3}, where $m_{\rm H} = 1.67 \times 10^{-24}$ g is the mass of the hydrogen atom, corresponding to gas number densities of $n_0 = \rho_0/(1.33 m_{\rm H})$ H nuclei cm^{-3} for a helium mass fraction $Y = 0.23$ and a metal mass fraction $Z = .02$. We consider gas metallicities in steps of half a dex, ranging from $Z_0 = 10^{-3} - 10^{0.5} Z_\odot$, excluding $10^{-2.5} Z_\odot$, where we have taken solar metallicity to be $Z_\odot = 0.02$. The density and metallicity grids are chosen to closely match those explored by Thornton et al. (1998). Consistent with Thornton et al., we compute mean molecular weights assuming a fixed helium mass fraction, $Y = 0.23$, taking the remainder to be hydrogen ($X = 1 - Z - Y$). The gas has an initial temperature of 10^4 K, but is allowed to cool to lower temperatures via radiation – see subsection 2.2.3.

Our simulations start with 1024 zones, linearly spaced from radii $R_{\rm in}$ to $R_{\rm out}$,

[1]Source code available at: github.com/egentry/clustered_SNe

which follow the scaling

$$R_{\rm out} = 300 \left(\frac{\rho_0}{1.33 m_{\rm H}\ {\rm cm}^{-3}} \right)^{-1/3} \left(\frac{M_{\rm cluster}}{100 M_\odot} \right)^{1/3} {\rm pc} \tag{2.1}$$

$$R_{\rm in} = 10^{-4} R_{\rm out} \tag{2.2}$$

This scaling is somewhat arbitrary and was set by initial tests; it was chosen to approximately reflect the final size of each simulation. If the outer boundary is too small, the domain will automatically extend when a shock nears the outer boundary.

2.2.2 Hydrodynamics

Our code solves the equations of compressible hydrodynamics in spherical symmetry using a moving-mesh finite volume method, including source terms for radiative cooling. Our method is an extension of the one implemented by Duffell (2016). The equations we solve are

$$\frac{\mathrm{d}}{\mathrm{d}t} \int \boldsymbol{U} \mathrm{d}V - \int \boldsymbol{F} \mathrm{d}A = \boldsymbol{S} \tag{2.3}$$

where $\boldsymbol{U}\mathrm{d}V$ is the vector of conserved quantities

$$\boldsymbol{U} = \begin{pmatrix} \rho \\ \rho u_r \\ \rho e \\ \rho Z \end{pmatrix} \tag{2.4}$$

for a density ρ, a bulk fluid velocity u, a specific total energy e and a local metallicity Z. The quantity $\boldsymbol{F}$ is the conservative flux, given by

$$\boldsymbol{F} = \begin{pmatrix} (u_r - w_r)\rho \\ (u_r - w_r)\rho u_r & +P \\ (u_r - w_r)\rho e & +Pu_r + H \\ (u_r - w_r)\rho Z \end{pmatrix} \tag{2.5}$$

where w is the computational mesh velocity, which is set to be the average velocity of the two zones adjacent to the boundary

$$w_r^{(i+1/2)} = \frac{u_r^{(i)} + u_r^{(i+1)}}{2} \tag{2.6}$$

approximating Lagrangian hydrodynamics. Here P is the pressure given by

$$P = (\gamma - 1)\rho e_{\rm int} \tag{2.7}$$

and $e_{\rm int}$ is the specific internal energy:

$$e_{\rm int} = e - \frac{1}{2}v_r^2. \tag{2.8}$$

At the inner boundary, we enforce a zero flux boundary condition; the outer boundary condition does not matter, as we automatically add zones before the shock reaches the outer boundary (and we have assumed the background is homogeneous).

This formulation implicitly introduces artificial mesh viscosity, particular at the inner boundary (i.e. wall heating), which leads to unphysically high temperatures. We counteract this by explicitly including an artificial conduction term, H. We use the artificial conduction prescription of Noh (1987) (Eq. 2.3):

$$H = \begin{cases} h_0 \rho |\Delta u| \Delta e_{\rm int} + h_1 \rho c_{\rm s} \Delta e_{\rm int} & \Delta u < 0 \\ 0 & \Delta u > 0 \end{cases} \tag{2.9}$$

where c_s is the adiabatic sound speed, h_0 and h_1 are tunable constants, typically of order unity, and Δ represents the differential of a variable across adjacent zones. We chose these constants to be $h_0 = 0$ and $h_1 = 0.1$, which experimentation showed were the smallest values that were still sufficient to remove most unphysical wall heating. This parameterization is similar to physical conduction in the strong-shock regime with a saturated conduction coefficient which has been lowered by a factor of a few by turbulence and magnetic fields (Cowie & McKee 1977).

In addition to these conservative fluxes, we also include non-conservative source terms

$$\boldsymbol{S} = \boldsymbol{S}_{\rm hydro} + \boldsymbol{S}_{\rm cooling} + \boldsymbol{S}_{\rm winds} \tag{2.10}$$

$$\boldsymbol{S}_{\text{hydro}} = \begin{pmatrix} 0 \\ \int 2(P/r)dV \\ 0 \\ 0 \end{pmatrix} \tag{2.11}$$

$$\boldsymbol{S}_{\text{cooling}} = \begin{pmatrix} 0 \\ 0 \\ \dot{E}_{\text{cooling}} \\ 0 \end{pmatrix} \tag{2.12}$$

$$\boldsymbol{S}_{\text{winds}} = \dot{M}_{\text{winds}}\Delta t \begin{pmatrix} 1 \\ u_{\text{winds}} \\ (1/2)u_{\text{winds}}^2 + e_{\text{int, winds}} \\ Z_{\text{winds}} \end{pmatrix}. \tag{2.13}$$

We defer a discussion of the cooling rate $\dot{E}_{\text{cooling}}$ to subsection 2.2.3, and a discussion of the wind source term (which is only added to the innermost zone) to subsubsection 2.2.5.1.

The conservative fluxes (excepting conduction) were solved using an HLLC Riemann solver (Toro et al. 1994) taken from the implementation of Duffell (2016). Artificial conduction and the non-conservative source terms were handled by operator splitting, solving each term individually.

By using a moving mesh with $w_r \approx u_r$, we approximate Lagrangian hydrody-

namics. This reduced numerical errors in the advective flux terms, as well as automatically adjusting to give higher resolution at locations with higher densities (assuming we start with a grid of uniform density). This improves our accuracy at shocks, where high densities lead to rapid cooling, which drives the subsequent evolution of the SNR. By using an approximately Lagrangian scheme, we can better resolve the dynamically important regions, without wasting computational time on the less important diffuse bubble.

For strong shocks we need to set a limit on how much zones can expand or compress. The innermost zone (where SN energy is injected) will significantly expand, so we need to split it in order to retain accuracy where our blasts are being injected. For computational efficiency we also need to allow zones to merge, because otherwise the zones near the shock become so thin that the computational cost of evolving them is prohibitive. We handle zone splitting and merging using the adaptive mesh algorithm implemented by Duffell (2016): zones thicker (thinner) than 10 (0.1) times the average zone thickness are split (merged).

To improve numerical stability in regions of highly-supersonic flow, we also implement a dual energy formalism. This approach counteracts the common problem in conservative, total energy codes such as ours that, at Mach numbers greater than unity, the internal energy $e_{\rm int}$ can be much smaller than the total energy e, so that small truncation errors in e can correspond to an order unity or larger error in $e_{\rm int}$, and thus in the temperature and radiative cooling rate. Our dual energy approach is as follows. For most zones and time steps, we follow the update procedure described above and

derive $e_{\rm int}$ from the mass, momentum and total energy (Equation 2.8). However, in any zone and time step where this procedure yields a value of $e_{\rm int} < 0$, we instead compute the internal energy via

$$e_{\rm int}(t + \Delta t) = e_{\rm int}(t) \left(\frac{dV(t + \Delta t)}{dV(t)} \right)^{1-\gamma} + \Delta e_{\rm cool}. \tag{2.14}$$

This includes adiabatic heating/cooling and radiative cooling; this ignores advective fluxes, which should be minimal for a pseudo-Lagrangian code, and conductive fluxes. The errors introduced by this dual energy formalism have negligible effects on the overall dynamics and numerical conservation of energy.

2.2.3 Cooling

Cooling plays a significant role in SNR evolution, with most of the energy from the SN being radiated from a thin, dense shell. To include this cooling, we use the `Grackle` chemistry and cooling library (Bryan et al. 2014; Kim et al. 2014), using operator splitting to evolve the thermal energy over each time step. `Grackle` sub-steps the thermal evolution using cooling rates pre-computed using `Cloudy` (Ferland et al. 1998), assuming ionization equilibrium but not thermal equilibrium between metallicity-dependent optically thin cooling and a cosmological UV background at redshift $z = 0$ providing photo-heating and photo-ionization (Haardt & Madau 2012). For simplicity we only include heating from a cosmological background, rather than including galactic heating sources. We leave testing more realistic heating backgrounds and non-ionization

equilibrium cooling models for a later work.

2.2.4 Cluster Model

In order to test the SN momentum produced by a cluster, we need to determine the number of SNe from a cluster and when those SNe will occur. For each simulation with a given cluster mass, we use the `SLUG2` code (da Silva et al. 2012; da Silva et al. 2014; Krumholz et al. 2015) to draw the desired mass in stars from a Kroupa (2002) IMF, using the default "Stop-nearest" policy. All stars above an initial mass of 8 $M_\odot$ are assumed to result in core-collapse SNe, after stellar lifetimes determined by the Geneva stellar evolution tracks assuming solar metallicity ($Z = 0.014$) (Ekström et al. 2012). Generally, we find 1 SN per roughly 100 $M_\odot$ of stars, and those SNe occur roughly $3-40$ Myr after the birth of the cluster. Given the power-law tail of the IMF, we expect most SN to come from relatively low mass stars, $M_\star \approx 8\ M_\odot$. We do not include Type Ia SNe for most of our simulations, but we do test the impact of short-delay Type Ia SNe in subsubsection 2.5.3.2.

Since we are stochastically drawing an IMF, our results for low mass clusters can depend significantly on the random seed. To minimize the uncertainty in our results induced by this stochasticity, we ran multiple realizations of the lowest cluster masses. Specifically, we ran 9 realizations of each $10^2\ M_\odot$ cluster and 4 realizations of each $10^{2.5}$ $M_\odot$ cluster.

This cluster model is not perfect. The stellar evolution tracks assume a single stellar metallicity, while ideally we would like the tracks to depend on the background

metallicity for each simulation. Our cluster model also ignores the effects of stellar rotation and binarity. All of these can affect stellar lifetimes.

2.2.5 SN Injection Model

When a SN occurs, we add energy, mass and metals to the innermost computational zone. For the energy, we adopt a fixed injection of 10^{51} ergs per SN. For the mass and metallicity, we use the data of Woosley & Heger (2007), who provide a grid of SN mass and metal yields as a function of initial stellar mass over a range of initial masses from 12 $M_\odot$ to 120 $M_\odot$. Within this range we linearly interpolate as a function of initial mass; outside this range, we use the nearest neighbor (i.e. stars with masses $8 - 12$ $M_\odot$ are assumed to produce the same yield as 12 $M_\odot$ stars).

As with our cluster model, this SN model is imperfect. First, this model over-predicts the ejecta mass for stars with initial masses of $8 - 12$ $M_\odot$ (the most common progenitors). For example this model predicts that a 9 $M_\odot$ star will eject 9.4 $M_\odot$ of material. This is clearly unphysical, but the true ejected mass is comparable; Sukhbold et al. (2016) show that the true ejected mass (≈ 7.4 $M_\odot$) differs by less than 50% from our simplified model. Overall, this will tend to over-predict the ejecta mass, biasing our results towards slightly more efficient cooling and slightly lower momenta.

Biasing our results in the opposite direction (for fixed cluster mass), we assume all of our massive stars explode, even though some low mass progenitors ($8 - 9$ $M_\odot$) may not explode (Woosley & Heger 2015) and some high mass progenitors will collapse directly into black holes (Ertl et al. 2016; Sukhbold et al. 2016), a pathway which can

depend significantly on the stellar metallicity (Pejcha & Thompson 2015). However, while models differ as to whether more massive stars explode, almost all models agree that all $9-12\ M_{\odot}$ stars should explode, so the total number of SNe should not change drastically.

More significantly, SN energies can vary with initial progenitor mass. In particular, Sukhbold et al. (2016) find that stars with initial masses of $9-12\ M_{\odot}$ explode with $< 0.7 \times 10^{51}$ ergs of energy. As these are the most common progenitors, this leads to an IMF-averaged explosion energy of $\approx 0.6 - 0.8 \times 10^{51}$ ergs, depending on the explosion model. While it is common to assume an explosion energy of 10^{51} ergs (e.g. (Thornton et al. 1998; Kim et al. 2014; Kim & Ostriker 2015; Iffrig & Hennebelle 2015; Martizzi et al. 2015; Walch & Naab 2015), in doing so we over-predict the average SN energy by a factor of $1.2-1.7$.

As with our stellar evolution tracks, the data of Woosley & Heger (2007) are only for stars of solar metallicity, so we are unable to vary our SN model with the background metallicity. Moreover, we are combining the ejecta computed by Woosley & Heger (2007) with the lifetimes computed by Ekström et al. (2012); these make different assumptions about stellar evolution, and are not fully consistent. Theoretical uncertainties in stellar lifetimes are not too worrisome though; for low mass clusters, we are dominated by stochastic scatter in the IMF; for high mass clusters, the SNe are effectively a continuous wind. The models of Woosley & Heger (2007) and Ekström et al. (2012) also differ in the details of pre-SN mass loss, but these discrepancies primarily exist for the most massive stars, which are the least common in our simulations.

2.2.5.1 Winds

If SN ejecta mass is important because more mass leads to faster cooling, then we cannot simply ignore pre-SN mass loss; that mass has to go somewhere. The pre-SNe mass loss can be determined from the data of Woosley & Heger (2007), but that does not tell us when that mass was lost, or its physical properties when it was lost (e.g. metallicity, wind velocity, wind energy). For simplicity, we assume that pre-SN mass loss occurs uniformly through a star's lifetime, as a wind with metallicity equal to the background metallicity, at a velocity of 10^3 km s^{-1}, and a temperature of 10^4 K. The total mass, metal mass, momentum and energy of this wind are added to the innermost zone.

2.3 Numerical Results

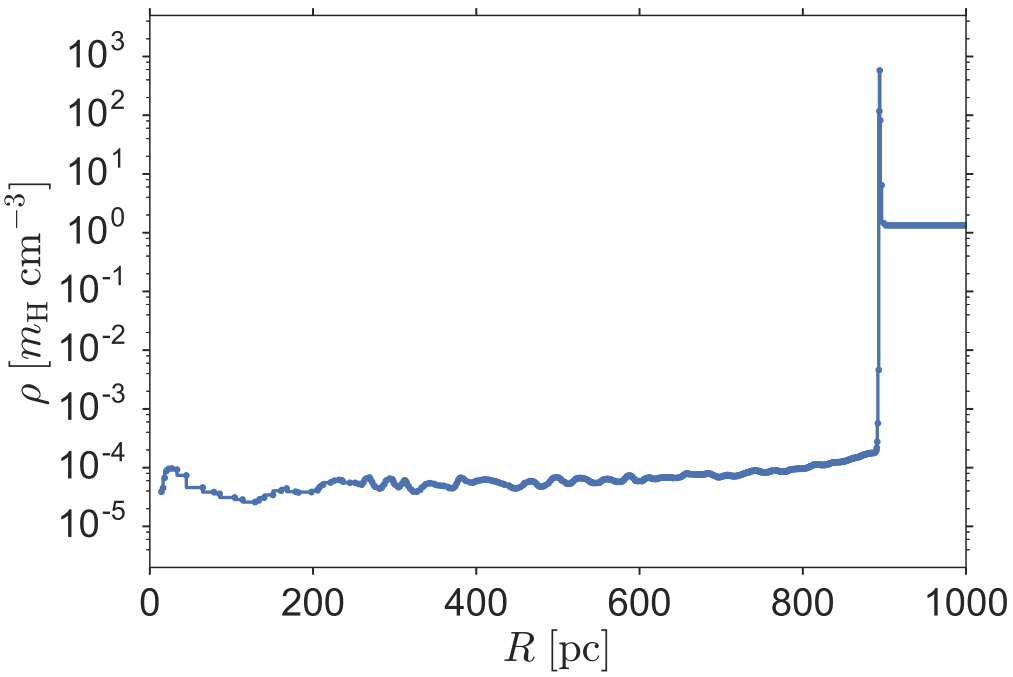

Figure 2.1: Example density profile of a simulation with $Z = Z_\odot$, $\rho = 1.33\ m_{\rm H}\ {\rm cm}^{-3}$ and $M_{\rm cluster} = 10^5\ M_\odot$ ($N_{\rm SNe} = 1008$), shortly after the last SN ($t = 38$ Myr).

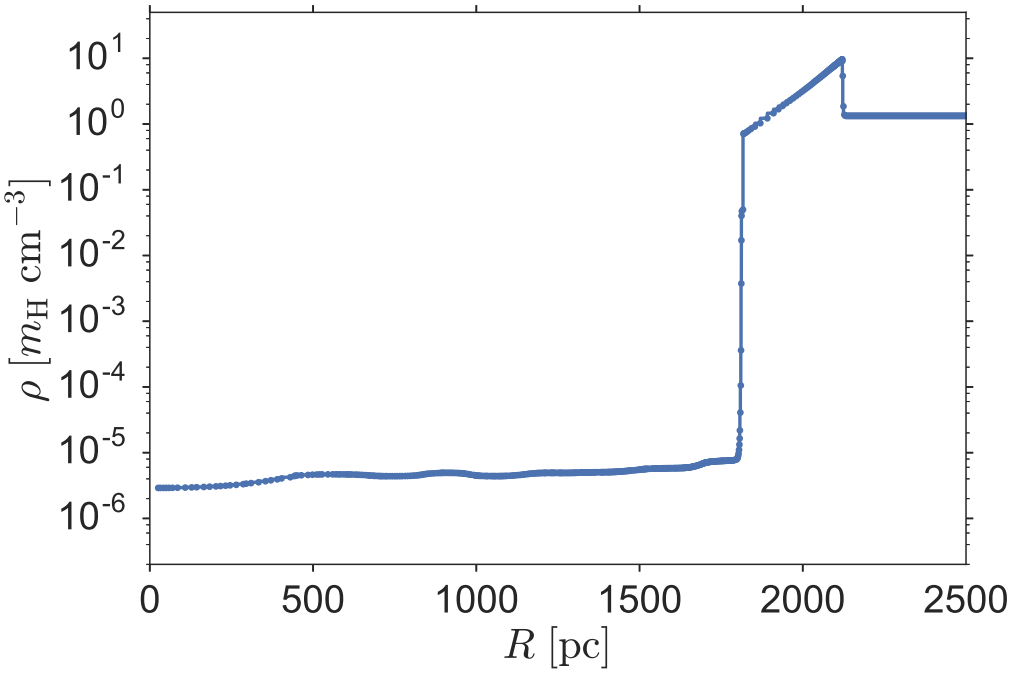

Figure 2.2: The same simulation as the one shown in Figure 2.1, except now at the moment of peak momentum ($t = 285$ Myr). The shock has weakened, causing it to thicken considerably.

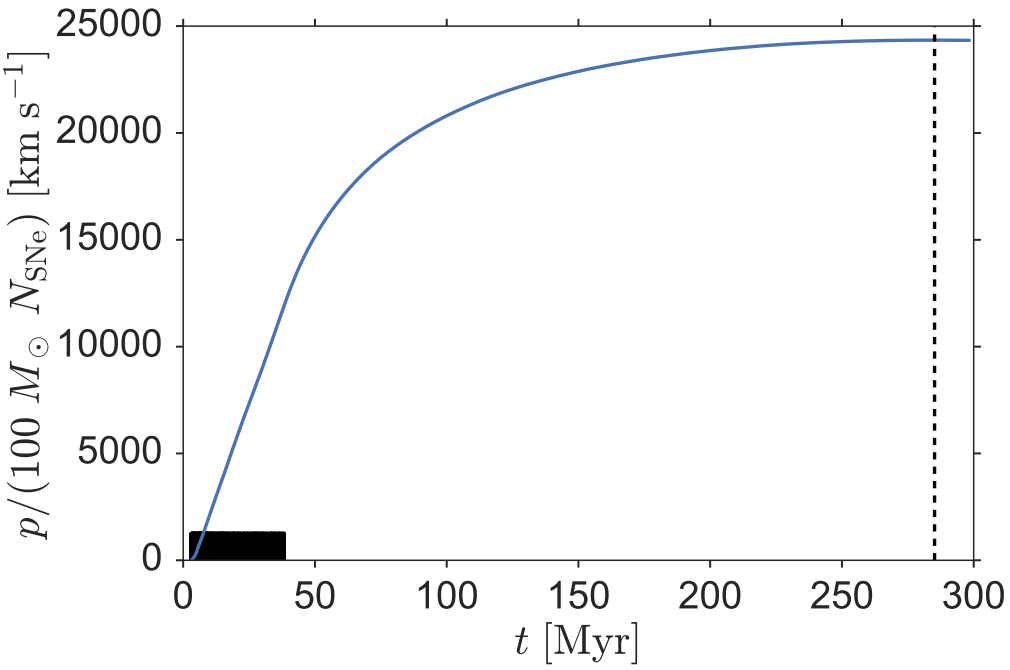

Figure 2.3: The evolution of the momentum per SN of the cluster shown in Figure 2.1 and Figure 2.2. The time of maximal momentum is marked by a vertical black dashed line; the duration of SN events is denoted by solid black ticks. For many SNe, the energy injection behaves more like a continuous wind rather than discrete explosions.

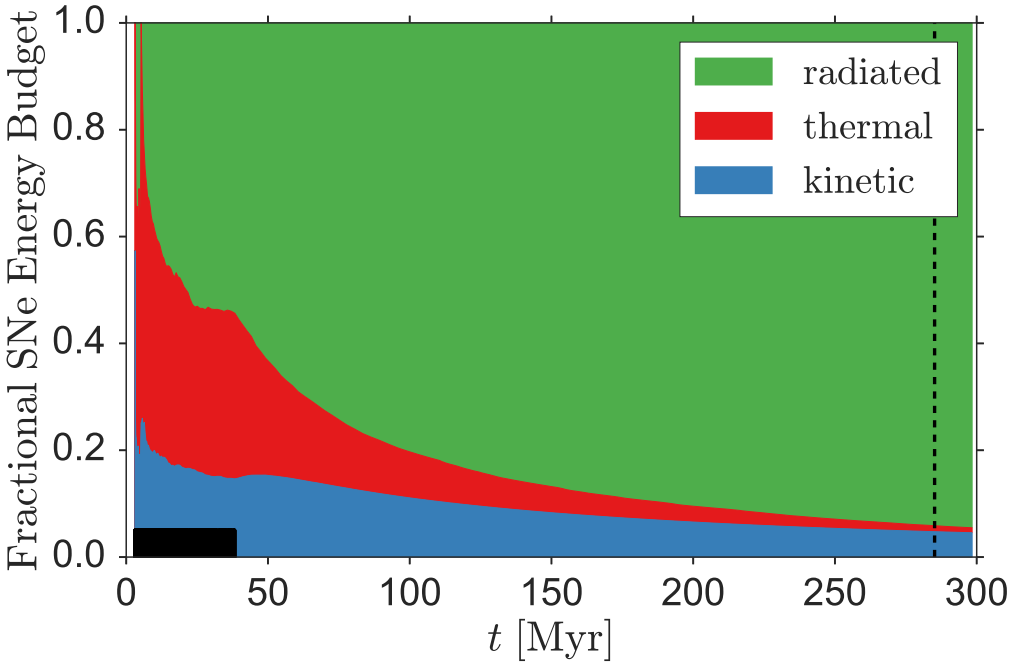

Figure 2.4: The evolution of the cumulative energy budget of the cluster shown in Figures 2.1 through 2.3. The time of maximal momentum is marked by a vertical black dashed line; SNe times are denoted by solid black ticks. The kinetic component is measured directly from the simulation; the radiated component is inferred by comparing the decrease in total energy compared to the decrease that would have occurred if there were no SNe and the background cooled anyway; the thermal component is assumed to be whatever remains.

Table 2.1: Overview of Numeric Results. The table shown here is only a stub; it is provided in its entirety as a Machine-Readable Table in the published version. The Machine-Readable Table also includes a column with an id unique to each row, to allow cross-referencing with Table 2.2, which has been hidden here to save space.

ρ ($1.33\ m_{\rm H}\ {\rm cm}^{-3}$)	Z ($Z_\odot$)	$M_{\rm cluster}$ ($M_\odot$)	$N_{\rm SNe}$	t (Myr)	R (pc)	$M_{\rm R}$ ($M_\odot$)	p (g cm s^{-1})	$E_{\rm R,kin}$ (ergs)	$E_{\rm R,int}$ (ergs)	flag
10^{+0}	$10^{+0.0}$	$10^{2.0}$	1	5.9	56.0	2.42×10^{4}	5.22×10^{43}	3.14×10^{49}	1.77×10^{48}	0
10^{+0}	$10^{+0.0}$	$10^{2.5}$	3	55.3	294.3	3.51×10^{6}	1.10×10^{45}	9.47×10^{49}	2.73×10^{50}	0
10^{+0}	$10^{+0.0}$	$10^{3.0}$	11	91.8	533.0	2.08×10^{7}	6.72×10^{45}	5.94×10^{50}	1.63×10^{51}	0
10^{+0}	$10^{+0.0}$	$10^{4.0}$	104	173.3	1149.0	2.09×10^{8}	7.18×10^{46}	6.67×10^{51}	1.63×10^{52}	0
10^{+0}	$10^{+0.0}$	$10^{5.0}$	1008	285.2	2133.0	1.34×10^{9}	4.88×10^{47}	4.79×10^{52}	1.04×10^{53}	0

Table 2.2: Momentum Evolution. The table shown here is only a stub; it is provided in its entirety as a Machine-Readable Table in the published version.

ID	t (Myr)	$R_{\rm shock}(t)$ (pc)	$M_{\rm R}(t)$ ($M_\odot$)	$p(t)$ (g cm s^{-1})	$E_{\rm R,kin}(t)$ (ergs)	$E_{\rm R,int}(t)$ (ergs)	$N_{\rm SNe}(<t)$
25451948-485f-46fe-b87b-f4329d03b203	4.0	1.3	6.92×10^{0}	0.00×10^{0}	0.00×10^{0}	1.00×10^{51}	1
25451948-485f-46fe-b87b-f4329d03b203	5.0	61.4	3.20×10^{4}	1.73×10^{44}	2.58×10^{50}	5.84×10^{50}	2
25451948-485f-46fe-b87b-f4329d03b203	10.2	152.9	4.92×10^{5}	1.24×10^{45}	8.26×10^{50}	1.14×10^{51}	5
25451948-485f-46fe-b87b-f4329d03b203	15.4	209.2	1.26×10^{6}	2.18×10^{45}	9.88×10^{50}	1.40×10^{51}	7
25451948-485f-46fe-b87b-f4329d03b203	20.5	249.3	2.13×10^{6}	2.83×10^{45}	1.07×10^{51}	2.43×10^{51}	9
25451948-485f-46fe-b87b-f4329d03b203	25.6	283.1	3.12×10^{6}	3.56×10^{45}	1.06×10^{51}	1.61×10^{51}	9
25451948-485f-46fe-b87b-f4329d03b203	35.3	335.1	5.18×10^{6}	4.59×10^{45}	1.05×10^{51}	2.84×10^{51}	11

We run a total of 672 simulations, sampling a three-dimensional parameter space composed of density ρ, metallicity, Z and cluster mass $M_{\rm cluster}$. As an example of the outcome of our simulations, in Figure 2.1 we show the density profile immediately after the last SN occurs in the simulation with $\rho = 1.33\ m_{\rm H}\ {\rm cm}^{-3}$, $Z = Z_{\odot}$ and $M_{\rm cluster} = 10^5\ M_{\odot}$. In Figure 2.2 we show the density profile for this simulation at the time when the radial momentum reaches its maximum, and we show the momentum as a function of time in Figure 2.3 and the cumulative energy budget as a function of time in Figure 2.4.

In Table 2.1 we provide an overview of our results, extracting the following key parameters when all SNe have occurred and the momentum reaches a maximum: the peak momentum p; the time, t, at which the momentum reaches a maximum (defining $t = 0$ as the time of cluster formation); the radius of the shock, R, at this time (defined by the furthest zone with an over-density compared to the background); the mass of the remnant, $M_{\rm R}$, enclosed by the shock radius at this time; and the kinetic and internal energies, $E_{\rm R,kin}$ and $E_{\rm R,int}$, enclosed by the shock radius at this time; finally, we also include a flag for untrustworthy results, which we explain in the next paragraph. In Table 2.2, we provide the time-dependent evolution of these parameters for every simulation and each snapshot before the time of peak momentum.

Not all of our simulations are trustworthy. In some runs, a strong reverse shock reaches the inner boundary before the SNR momentum peaks. In these simulations the shock reflects off our hard inner boundary, whereas in reality the shock converging on the origin would certainly become unstable and would not reflect. In these cases, we

cannot reasonably measure a maximum momentum. Fortunately, this behavior only occurs in a small part of our parameter space (40/672 runs), and in what follows we will exclude these runs from our analysis. We also exclude any other realizations of the same initial conditions (an additional 99/672 runs) so as not to bias ourselves by only allowing atypical realizations. In subsubsection 2.4.1.1 we explain the astrophysical causes and implications of these flagged runs.

The quantities $M_{\rm R}$ and $E_{\rm R,int}$ need to be interpreted with some care. At late times (when these quantities are extracted), the shock has weakened and is becoming a linear sound wave moving through a uniform medium, as illustrated in Figure 2.2. At this time, the bubble-shell decomposition no longer is a good description, and $M_{\rm R}$ and $E_{\rm R,int}$ are becoming increasingly dominated by background material which has simply had a sound wave pass through it, but has not been irreversibly affected by a shock. It would be inaccurate to include this material and its internal energy in SN-driven "feedback," and it is difficult to meaningfully disentangle SN-dominated material and background-dominated material as the SNR is merging into the ISM. It is easier to disentangle kinetic variables since the background is static; all momentum and kinetic energy must be a result of the SNe. The kinetic energy $E_{\rm R,kin}$ does not asymptote, but it varies slowly at late times, as illustrated in Figure 2.4.

As an example of how these results vary across our parameter space, we plot the asymptotic momentum per SN in two cuts through this parameter space in Figure 2.5 (momentum per SN as a function of ρ and $N_{\rm SNe}$ at fixed Z) and Figure 2.6 (momentum per SN as a function of Z and $N_{\rm SNe}$ at fixed ρ). In Figure 2.7 and Figure 2.8 we provide

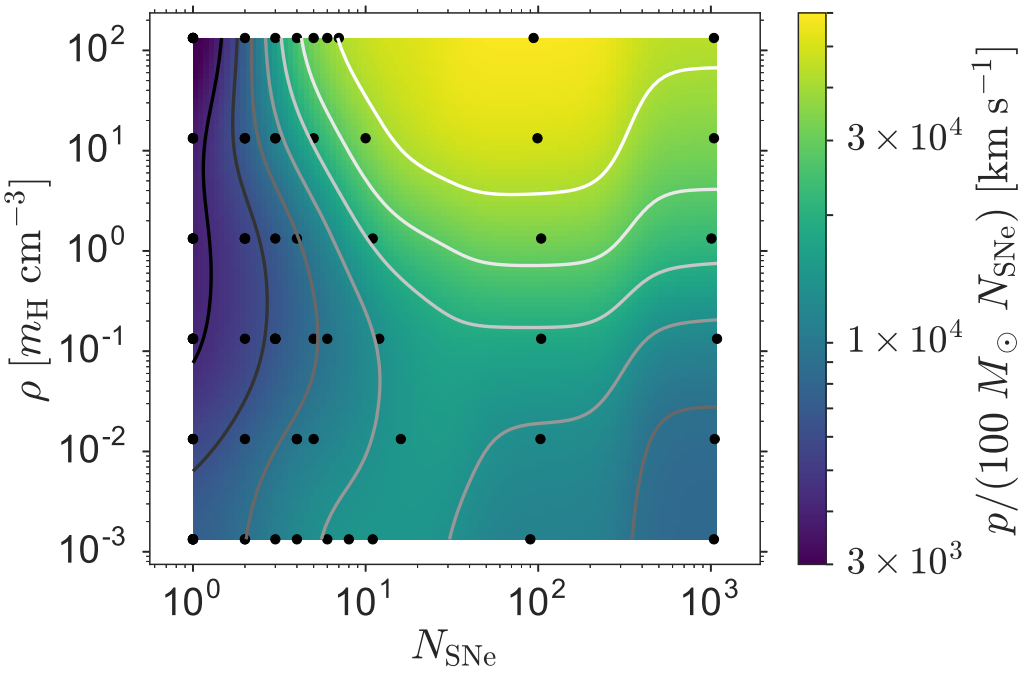

Figure 2.5: An overview of the final momentum per SN and how it varies with the number of SNe and gas density at fixed metallicity ($Z = Z_\odot$). The locations of our simulations in parameter space are marked by black scatter points (excluding flagged runs), which are not a perfect grid because the numbers of SNe are drawn stochastically. The color image is an interpolation of our simulation results using a Gaussian radial basis function, evaluated on a 100 by 100 grid with 7 greyscale contours logarithmically spaced between 3×10^3 and 6×10^4 km s^{-1} (exclusive).

analogous figures for the asymptotic kinetic energy, which is typically 1–10% of the injected SN energy.

2.4 The Momentum Budget of Clustered Supernovae

Figure 2.5 and Figure 2.6 show significant structure in the momentum as a function of number of SNe, gas metallicity and density. In particular, we note three

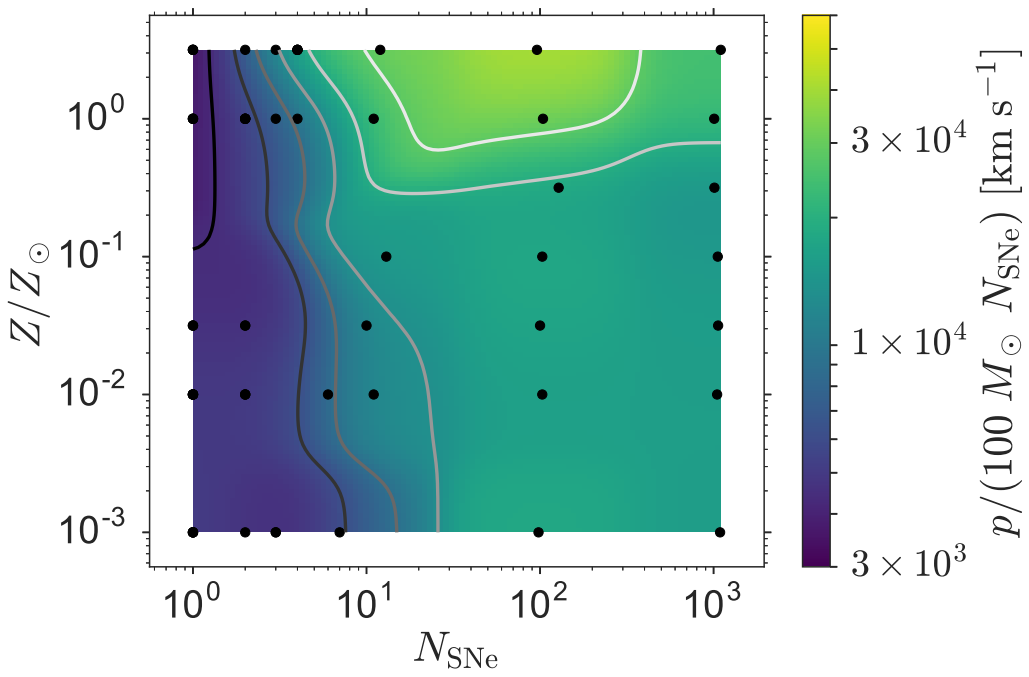

Figure 2.6: The same as the Figure 2.5, except now allowing metallicity to vary while holding density fixed at 1.33 $m_{\rm H}$ cm^{-3}. The top contour level shown in Figure 2.5 is not shown here, as the dynamic range of the data is not as large.

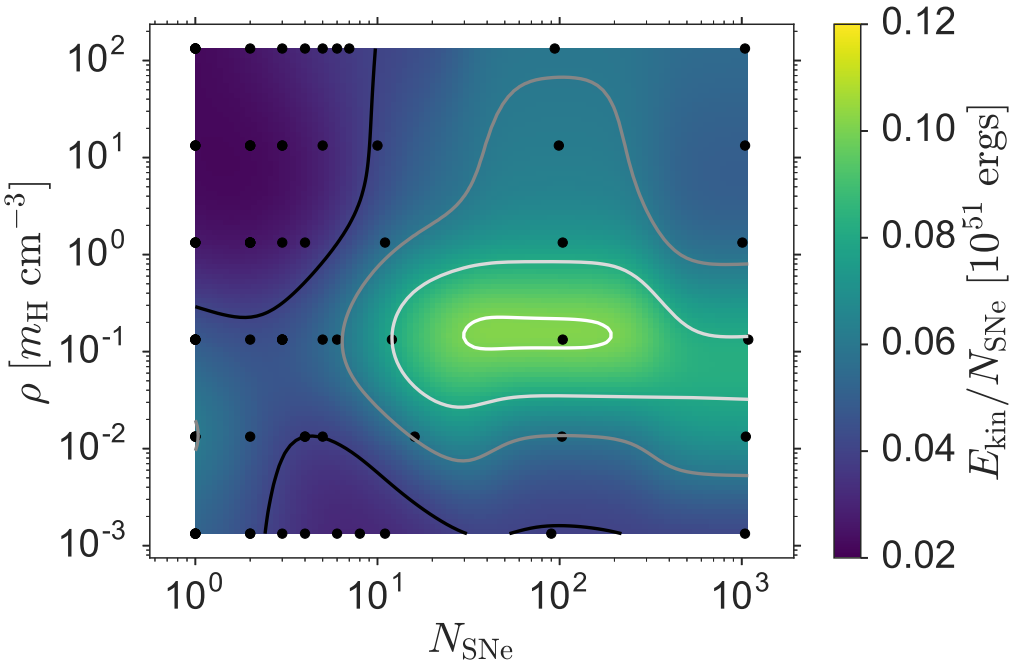

Figure 2.7: Same as Figure 2.5, except now the color image shows final kinetic energy with 4 greyscale contours linearly spaced between 2×10^{49} and 1.2×10^{50} ergs (exclusive).

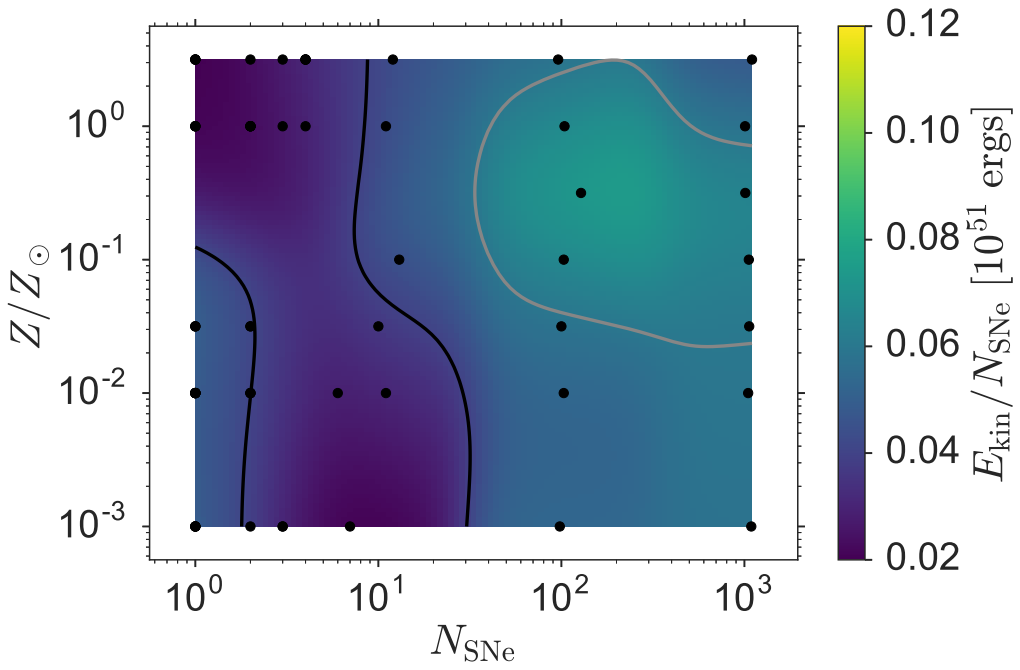

Figure 2.8: The same as Figure 2.7, except now allowing metallicity to vary while holding density fixed at 1.33 $m_{\rm H}$ cm^{-3}. The top two contour levels shown in Figure 2.7 are not shown here, as the dynamic range of the data is not as large.

behaviors: (1) For fixed density and metallicity, starting with a few SNe the momentum per SN initially increases with increasing number of SNe, reaches a maximum between 10-100 SNe (the exact location depends on density and metallicity), then decreases. (2) For a few SNe, the momentum per SN increases with decreasing density and gas metallicity. (3) For many SNe the opposite is true, as momentum per SN increases with increasing density and, to a smaller extent, metallicity. In this section we show how these primary behaviors are a consequence of clustered SNR evolution falling into one of two physical regimes: the small-N regime and the superbubble regime.

2.4.1 Qualitative Analysis

2.4.1.1 The Small-N Regime

To understand how SN momentum budgets act when the number of clustered SNe is relatively small, we start with its limiting case: single, isolated SNe. Feedback from isolated SNe has been well explored, as discussed in section 2.1. In particular, Thornton et al. (1998) found that lower gas metallicities and densities resulted in higher energy feedback, which is what we expect physically; lower gas metallicity and density results in weaker cooling, sapping less energy from a SNR, increasing the amount of energy feedback. This is also expected to apply for momentum feedback, and in Figure 2.9 we show that our results match the scaling between momentum and density expected by Cioffi et al. (1988).

As the number of SNe increases, the picture is similar to a series of isolated SNe, but with each successive SN occurring in a lower density bubble. As discussed

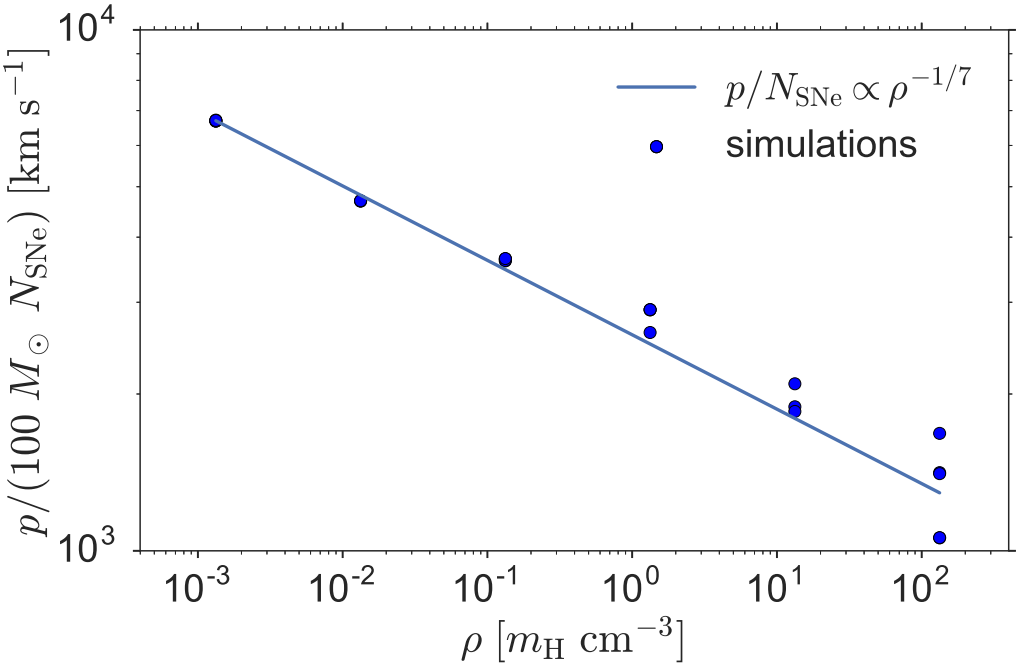

Figure 2.9: The scaling of momentum with background density, for $Z = Z_\odot$ and $N_{\rm SNe} = 1$, compared to the $p \propto \rho^{-1/7}$ scaling (normalized to the mean momentum of the lowest density clusters) expected for isolated SNe in a homogeneous background (Cioffi et al. 1988).

above, this leads to progressively more efficient momentum production as the region is progressively evacuated. Figure 2.10 illustrates this process directly, by plotting the momentum versus time for a simulation in which two SNe occur. The first SN occurs 20 Myr after cluster formation and its remnant quickly asymptotes to a momentum $\approx 3 \times 10^5\ M_\odot$ km s^{-1}, in agreement with the usual value found for single SNe. The second SN occurs 5 Myr later, and thanks to the vastly lower density inside the bubble, experiences much smaller radiative losses. This leads it to inject $\approx 2 \times 10^6\ M_\odot$ km s^{-1} of momentum by the time the momentum peaks, which is almost 10 times more momentum than injected by the first SN.

This breaks down for the clusters with the fewest SNe embedded in the highest density backgrounds, which behave more like multiple, isolated SNe. As density increases, SNRs evolve more rapidly, quickly cooling, lowering their internal pressure, and then being crushed by the pressure of the surrounding ISM. For the most dense gas with the fewest SNe, the SNR bubbles can be destroyed between SNe. As subsequent SNe are no longer occurring in the bubble of previous SNe, we believe clustering effects should be minimal.

Unfortunately, since the bubble has collapsed before all SNe have been injected, our numerical methods break down due to the reverse shock propagating all the way to the origin and undergoing unphysical reflection (see section 2.3). We are therefore forced to exclude this regime from our analysis. In our three-dimensional parameter

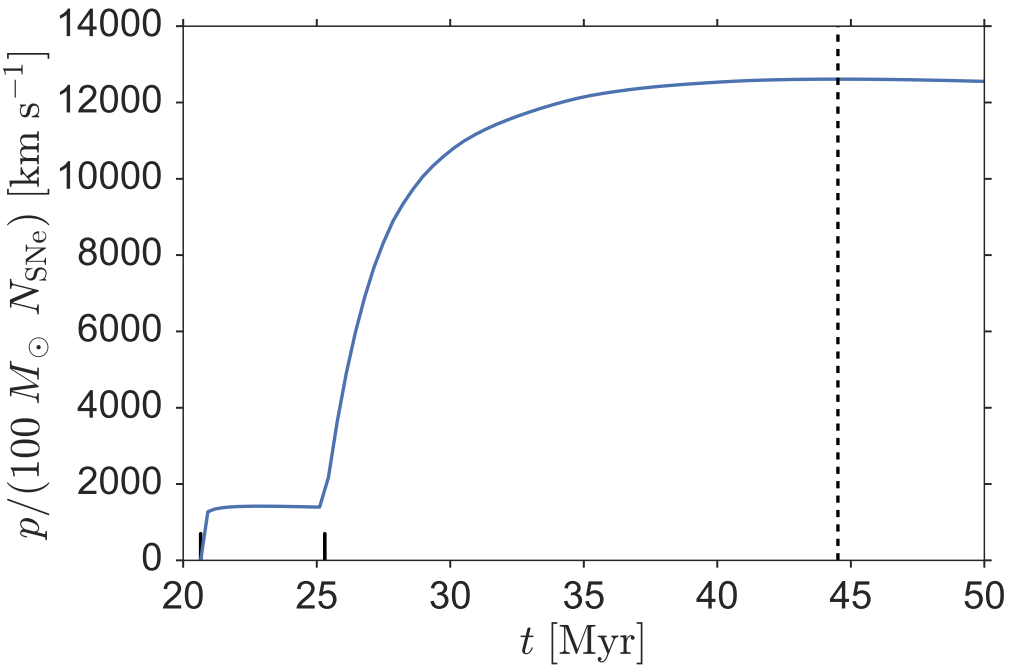

Figure 2.10: The evolution of the momentum per SN of a $Z = Z_\odot$, $\rho = 1.33\ m_{\rm H}$ cm^{-3} and $N_{\rm SNe} = 2$ cluster. The moment of maximal momentum is marked by the vertical dashed black line; the times of SNe are denoted by solid black ticks.

space, the excluded region is roughly defined by the parameters

$$\rho \geq 1.33\, m_{\rm H}\ {\rm cm}^{-3} \qquad \text{and} \qquad N_{\rm SNe} < 10 \qquad \text{and} \qquad Z < 0.1 Z_{\odot}.$$

While we cannot simulate this part of parameter space directly, the above analysis suggests that there should be no clustering effects present in it, and thus for the purpose of subgrid modeling it is likely safe to adopt a momentum budget of $3 \times 10^5\ M_{\odot}$ km s^{-1} per SN, the same as in the isolated SN regime.

2.4.1.2 The Superbubble Regime

While the few SNe model predicts that momentum efficiency increases as the number of SNe increases, our data show a turnover after about 10-100 SNe, beyond which the momentum efficiency begins to drop as the number of SNe increases. This can be understood within the framework of a superbubble powered by a continuous wind (see Figure 2.3 for an example of the momentum evolution of a large cluster). As more SNe occur, the bubble density decreases while the bubble temperature increases, both of which lead to less efficient cooling. While this leads to strong momentum feedback for a few SNe, it eventually saturates; if most of the energy is already being retained, suppressing cooling even further will only have a marginal effect.

Castor et al. (1975) provide a simplified bubble model which allows us to begin to understand superbubble evolution. They assume a constant energy injection rate, but if that energy is injected over a period of time that is the same for all clusters

(effectively assuming stellar evolution models do not depend strongly on metallicity or density), then their approach is also valid for an energy injection rate that is a power law with respect to time. Using their bubble solution, we can find the momentum per SN at the time of the last SN:

$$p(t_{\text{last SN}})/N_{\text{SNe}} \propto N_{\text{SNe}}^{-0.2} \rho_0^{0.2}. \tag{2.15}$$

We compare this predicted scaling to our numerical results for the lowest density simulated (thereby ensuring we are as close as possible to the adiabatic limit) in Figure 2.11. As the plot shows, the analytic scaling is in reasonable agreement with the numeric results.

As shown in Figure 2.3, a significant amount of momentum evolution occurs after the last SN. During this phase, the superbubble expands adiabatically until the bubble pressure equals the ISM pressure, P_0, at which point the shell's momentum reaches a maximum, since the pressure gradient switches direction. For an adiabatic index $\gamma = 5/3$ for the gas inside the SNR, this results in a final momentum per SN

$$\begin{aligned} p_{\text{final}}/N_{\text{SNe}} &\propto P_0^{-1/(2\gamma)} N_{\text{SNe}}^{-0.2+(0.2/\gamma)} \rho_0^{0.2+(0.3/\gamma)} \\ &\approx P_0^{-0.3} N_{\text{SNe}}^{-0.08} \rho_0^{0.38} \end{aligned} \tag{2.16}$$

This analytic scaling with respect to number of SNe can be compared to our numeric data in Figure 2.12; the scaling with respect to gas density is shown in Figure 2.13.

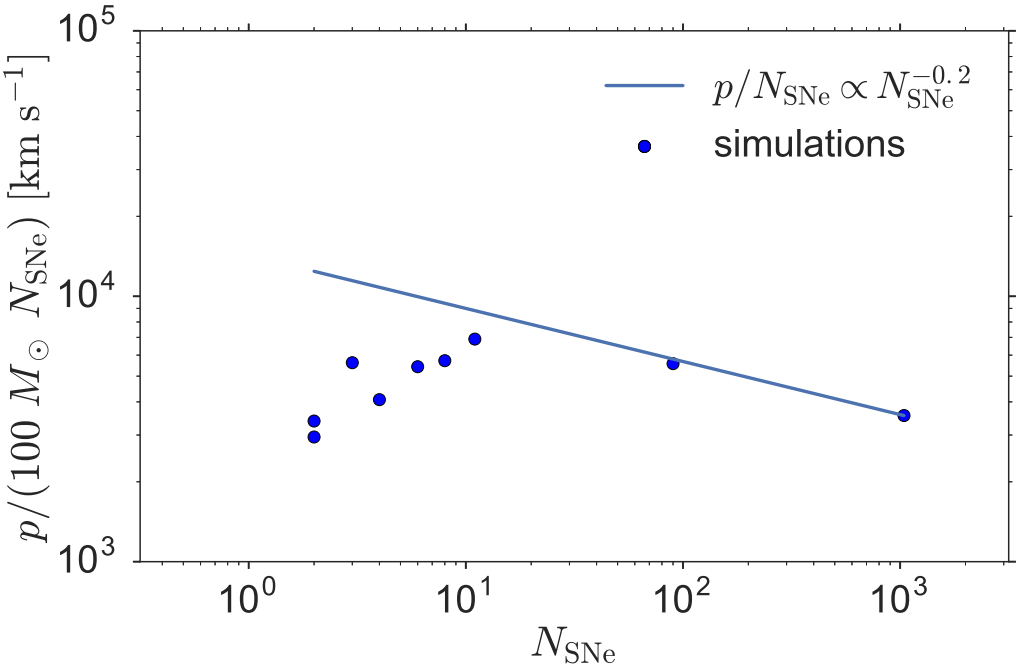

Figure 2.11: The scaling of momentum per SN with number of SNe, evaluated at the time of the last SN for each cluster. The clusters shown all have solar metallicity and the lowest density simulated, 1.33×10^{-3} $m_{\rm H}$ cm^{-3}. We plot the theoretical scaling for an adiabatic superbubble (Equation 2.15), normalized to the cluster with the most SNe (the cluster which is expected to best correspond to the adiabatic case).

It is not surprising that these scalings are not perfect; Sharma et al. (2014) predict that even 10^3 SNe are not enough to satisfy assumptions behind models like those of Castor et al. (1975). Specifically, Sharma et al. (2014) predict no wind-dominated region (where $\rho \propto r^{-2}$) which ends in a stable termination shock before the pressure-dominated bubble begins; these predictions are in agreement with our results (see Figure 2.1 and Figure 2.2). Our simulations do not satisfy all of the assumptions of superbubble models like those of Castor et al. (1975); these models are sufficient for a qualitative analysis, but they are insufficient for a quantitative understanding.

2.4.2 Quantitative Model

Informed by the qualitative understanding developed in subsection 2.4.1, we now construct a quantitative parametric model which we constrain using our simulation results (Table 2.1). In both the small-N and superbubble limits, we expect the results to behave like a power law in number of SNe, gas density and metallicity, but we expect these to be different power laws. Therefore, we choose to construct a model of two power laws with a smooth break. In the few SNe (small-N) limit, we use a model of the form

$$\left(\frac{p}{N_{\rm SNe}}\right)_{\rm few} = \left(\frac{p}{N_{\rm SNe}}\right)_{0,\rm few} \left(\frac{Z}{Z_\odot}\right)^{\eta_{Z,\rm few}} \times \left(\frac{\rho}{m_{\rm H}\ {\rm cm}^{-3}}\right)^{\eta_{\rho,\rm few}} \left(\frac{N_{\rm SNe}}{1}\right)^{\eta_{N,\rm few}} \quad (2.17)$$

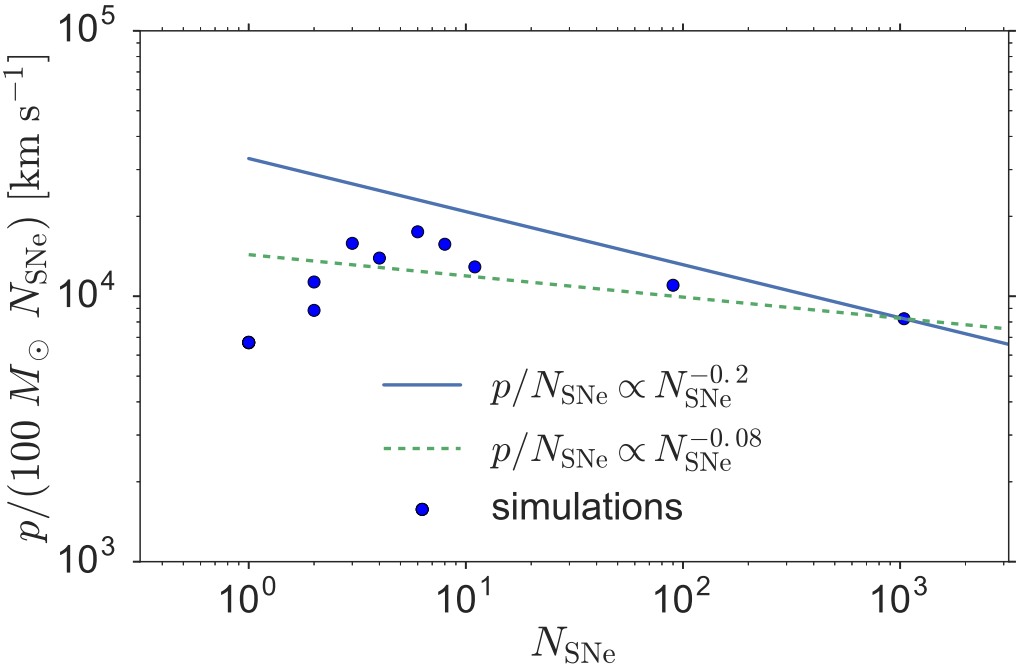

Figure 2.12: The scaling of asymptotic momentum per SN with number of SNe. These are the same clusters as those shown in Figure 2.11 (1.33×10^{-3} $m_{\rm H}$ cm^{-3}; $Z = Z_\odot$) but evaluated at a different time. We plot theoretical scalings for an adiabatic superbubble at time of the last SN (blue solid line; Equation 2.15) and at the time the interior pressure equals the exterior pressure (green dashed line; Equation 2.16).

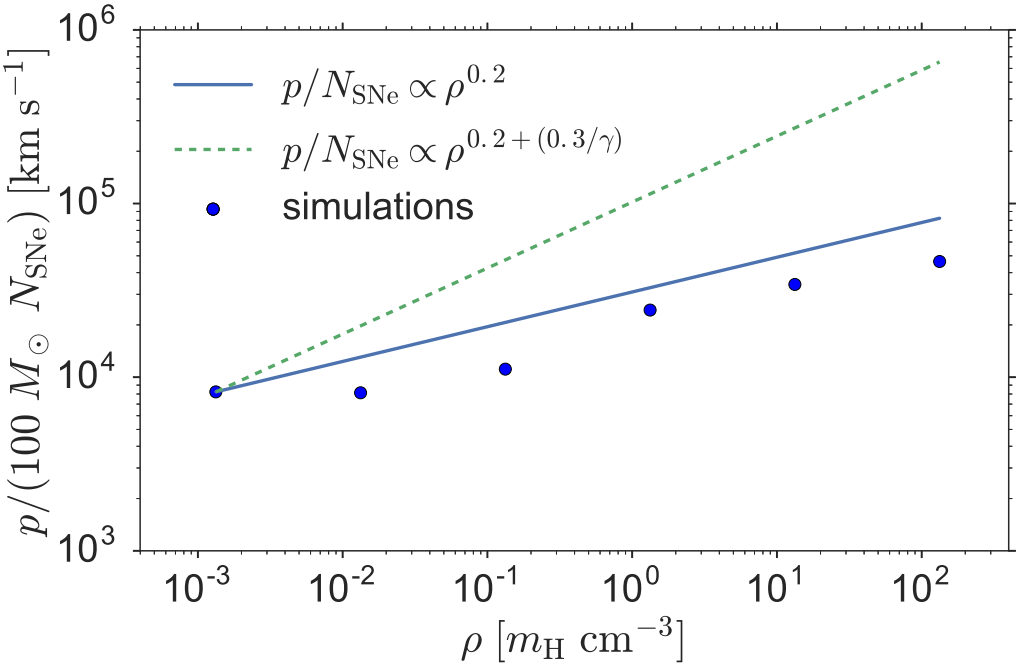

Figure 2.13: The scaling of asymptotic momentum per SN with background density, for $Z = Z_\odot$ and $M_{\rm cluster} = 10^5\ M_\odot$ ($N_{\rm SNe} \approx 10^3$) clusters, compared to the superbubble predictions for the end of the SNe injection phase ($p/N_{\rm SNe} \propto \rho^{0.2}$) and when the interior pressure equals the exterior pressure ($p/N_{\rm SNe} \propto \rho^{0.2+(0.3/\gamma)}$).

and in the many SNe (superbubble) limit we use a similar form,

$$\left(\frac{p}{N_{\rm SNe}}\right)_{\rm many} = \left(\frac{p}{N_{\rm SNe}}\right)_{0,\rm many} \left(\frac{Z}{Z_\odot}\right)^{\eta_{Z,\rm many}} \times \left(\frac{\rho}{m_{\rm H}\ {\rm cm}^{-3}}\right)^{\eta_{\rho,\rm many}} \left(\frac{N_{\rm SNe}}{1000}\right)^{\eta_{N,\rm many}} \tag{2.18}$$

which are smoothly combined using:

$$\frac{p}{N_{\rm SNe}} = \frac{\left(\frac{p}{N_{\rm SNe}}\right)_{\rm few}\left(\frac{p}{N_{\rm SNe}}\right)_{\rm many}}{\left(\frac{p}{N_{\rm SNe}}\right)_{\rm few} + \left(\frac{p}{N_{\rm SNe}}\right)_{\rm many}} \tag{2.19}$$

$$\approx \min\left[\left(\frac{p}{N_{\rm SNe}}\right)_{\rm few}, \left(\frac{p}{N_{\rm SNe}}\right)_{\rm many}\right] \tag{2.20}$$

Assuming our simulation results have a random additive gaussian noise of variance σ^2, we can construct a gaussian likelihood function for the results of each simulation. Even though σ^2 is unknown, this allows us to determine a maximum likelihood estimate (MLE) for our best-fitting model parameters. We would also like to understand the uncertainties in those parameters. For that we need the posterior, which through Bayes' theorem requires a prior distribution, π on those parameters. Since we do not have strong prior information on most of these parameters, we choose uniform, independent priors on our parameters: $\log\left(\sigma^2\right)$, $\log\left(p/N_{\rm SNe}\right)_{0,\rm few}$, $\eta_{Z,\rm few}$, $\eta_{\rho,\rm few}$, $\eta_{N,\rm few}$, $\log\left(p/N_{\rm SNe}\right)_{0,\rm many}$, $\eta_{Z,\rm many}$, $\eta_{\rho,\rm many}$, $\eta_{N,\rm many}$.

Combining this prior with a gaussian likelihood for our data results in the posterior distribution of our model parameters. We sample this posterior distribution

using a Markov Chain Monte Carlo (MCMC) scheme, using Gibbs sampling to draw samples of σ^2 from an inverse gamma distribution and using a Metropolis-Hastings random walk for the remaining parameters. We use the MLE as the starting guess, discard the first 10000 steps as burn-in steps and save the next 100000 steps. Using these samples, we can now estimate uncertainties on our model parameters: for each parameter we use the median as our best-fitting value, and the 16^{th} and 84^{th} percentiles as our uncertainty interval (effectively marginalizing over all other parameters) resulting in

$$\left(\frac{p}{N_{\text{SNe}}}\right)_{0,\text{few}} = 4249^{+741}_{-683} \cdot 100\ M_\odot\ \text{km s}^{-1} \tag{2.21}$$

$$\eta_{Z,\text{few}} = 0.05^{+0.05}_{-0.06} \tag{2.22}$$

$$\eta_{\rho,\text{few}} = -0.06^{+0.03}_{-0.03} \tag{2.23}$$

$$\eta_{N,\text{few}} = 2.20^{+0.24}_{-0.23} \tag{2.24}$$

$$\left(\frac{p}{N_{\text{SNe}}}\right)_{0,\text{many}} = 23546^{+1072}_{-1073} \cdot 100\ M_\odot\ \text{km s}^{-1} \tag{2.25}$$

$$\eta_{Z,\text{many}} = 0.15^{+0.01}_{-0.01} \tag{2.26}$$

$$\eta_{\rho,\text{many}} = 0.14^{+0.01}_{-0.01} \tag{2.27}$$

$$\eta_{N,\text{many}} = -0.07^{+0.02}_{-0.02} \tag{2.28}$$

$$\sigma = 6075^{+214}_{-202} \cdot 100\ M_\odot\ \text{km s}^{-1} \tag{2.29}$$

Our posterior samples are also useful for estimating the uncertainty in the

predicted momentum from a particular cluster. For a given gas metallicity, density and number of SNe, each posterior sample predicts a slightly different momentum and an uncertainty σ on that momentum. For each posterior sample, we then generate realizations of the noise with variance σ^2. For N posterior samples and M noise realizations, this gives us $N \times M$ samples of the momentum predictive distribution, which allows us to estimate the median and the 16^{th} and 84^{th} percentiles of the predictive distribution for a given cluster. We compare this predictive model and its uncertainties to our a subset of our numeric data in Figure 2.14.

2.5 Discussion

Using our numeric results and quantitative model, we can now comment on the significance of these results in the context of previous works. We first examine the implications of our results for models of momentum-regulated star and galaxy formation in subsection 2.5.1. We then compare our results to those of previous authors in subsection 2.5.2, and in subsection 2.5.3 we discuss the potential importance of physical processes we have omitted.

2.5.1 Implications of High Momentum Efficiency of Clustered SNe

In models of momentum-driven feedback, the key parameter is p/m_*, the amount of momentum injected per unit mass of stars formed in a given system. Non-clustered models of SNe momentum production usually assume $p/m_* \approx p/(100 M_\odot N_{\rm SNe})$ $\approx 1000 - 3000$ km s^{-1} (with a weak dependence on density) for a mass m_* of stars

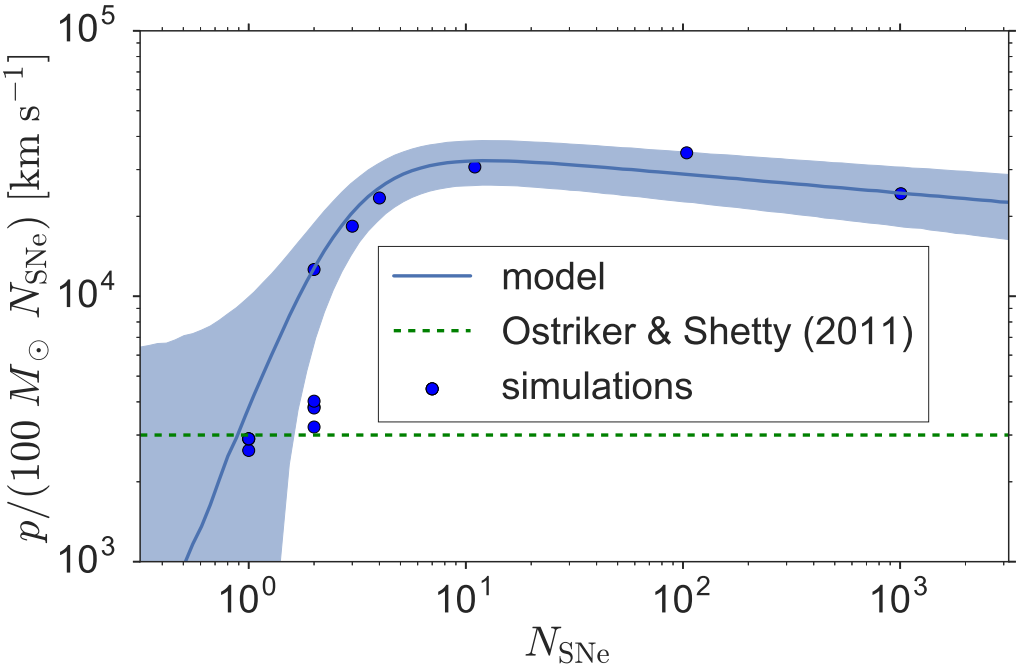

Figure 2.14: Comparison of a slice of our simulation results ($Z = Z_\odot$, $\rho = 1.33\ m_{\rm H}$ cm^{-3}) to our model with an uncertainty envelop bounding the 16$^{\rm th}$ and 84$^{\rm th}$ percentiles of our predictive momentum model. Some slices fit better and some slices fit worse, but overall this is a representative slice. For comparison, we also plot a typical unclustered model, $p/(100 M_\odot N_{\rm SNe}) = 3000$ km s^{-1} (Ostriker & Shetty 2011; green dashed line).

formed (Thompson et al. 2005; Ostriker & Shetty 2011; Shetty & Ostriker 2012; Dekel & Krumholz 2013; Faucher-Giguère et al. 2013; Hopkins et al. 2014; Kim & Ostriker 2015; Kimm et al. 2015; Hayward & Hopkins 2017). For a star cluster with a single SN ($M_{\rm cluster} \approx 100 M_\odot$), our best fit model is a little higher than but still consistent with $1000 - 3000$ km s^{-1} given the uncertainties in our model. For higher mass clusters, the discrepancy becomes significant. The most extreme difference is found for $M_{\rm cluster} = 10^3 - 10^4 M_\odot$ ($N_{\rm SNe} = 10^1 - 10^2$), for which our value of p/m_* can be greater than the unclustered value by an order of magnitude (see Figure 2.5 and Figure 2.14). But these are just the extremes; for a typical distribution of cluster masses found in a galaxy, what is the average effect?

To evaluate the mean value of p/m_* for star formation on galactic scales, we must integrate our model for individual clusters, $p/M_{\rm cluster}$, over a cluster mass function $dN/dM_{\rm cluster}$. The resulting mean momentum yield per unit mass of stars formed is

$$\frac{p}{m_*} = \int \left(\frac{p}{M_{\rm cluster}}\right) \frac{dN}{d\ln M_{\rm cluster}} dM_{\rm cluster}. \tag{2.30}$$

If we adopt a typical mass distribution $dN/dM_{\rm cluster} \propto M_{\rm cluster}^{-2}$ over the range $M_{\rm cluster} = 10^2 - 10^5\ M_\odot$, comparable to what is observed in nearby galaxies (Krumholz 2014 and references therein), and use our fitting formula (Equation 2.19) to evaluate $p/N_{\rm SNe}$ as a function of $M_{\rm cluster}$ (for $N_{\rm SNe} \approx M_{\rm cluster}/100 M_\odot$), this yields a value of $p/m_* \approx 1{-}2{\times}10^4$ km s^{-1} over the metallicity range $Z/Z_\odot = 0.01 - 1$ and density range $\rho/m_{\rm H} = 0.1 - 10^5$. This is $\sim 0.5 - 1$ dex higher than the value usually adopted based on single SN models.

This result is only logarithmically sensitive to the adopted limits on the cluster mass function.

This increased momentum yield will significantly alter the conclusions of analytic models in which star formation is regulated primarily by SN momentum input (e.g., Ostriker & Shetty 2011; Shetty & Ostriker 2012; Faucher-Giguère et al. 2013; Hayward & Hopkins 2017). The same is true for models where SN momentum is primarily responsible for launching galactic winds (e.g., Dekel & Krumholz 2013; Hayward & Hopkins 2017; Thompson & Krumholz 2016). In general, the higher momentum yield we obtain will shift such models to predict lower star formation rates for fixed galactic surface densities, which may require re-tuning of other parameters to bring the models back into agreement with observed relationships between star formation rate and gas content.[2] The models will also predict stronger outflows, though these are significantly less constrained by observations.

The momentum yield per SN is also a critical input to numerical methods that handle subgrid feedback through explicit momentum injection (e.g., Kim et al. 2011; Hopkins et al. 2014; Kimm et al. 2015; Goldbaum et al. 2016). These models should be also be rerun using our updated estimates of the SN momentum yield. Even models that do not use explicit momentum injection, but that attempt to include SN feedback by explicitly resolving the Sedov phase (e.g., Hopkins et al. 2011), may need to be reconsidered, at least for simulations of galaxies large enough for there to be significant

[2]The situation is more complex for models that include regulation of star formation by FUV radiation instead of or in addition to SN feedback (e.g., Krumholz et al. 2009; Ostriker et al. 2010; Krumholz 2013). In these models, the effects of enhanced momentum injection will be more modest or absent, depending on the details of the individual model.

numbers of clustered SNe.

2.5.2 Comparison to Previous Work and Convergence Study

We find that clustered SNe generally lead to an increase in the momentum injected by SNe, in some cases by an order of magnitude. The results of previous authors diverge strongly, as noted in section 2.1, with some finding that clustering leads to an enhancement in momentum per SN and others finding a decrease, but none finding an increase as large as an order of magnitude. To understand this discrepancy, we need to understand the role of mixing and how it enters various simulations.

In SNe-driven bubbles, the cooling rate plays a significant role in setting the dynamics of the system. This cooling rate is itself affected by the mixing rate at interfaces between hot diffuse gas (which dominates the thermal energy) and cold dense gas (which is most radiatively efficient). If the mixing of energy and matter increases, the cooling rate can increase and the final momentum can decrease. This mixing can be increased by both non-physical sources (i.e., due to numerical diffusion) and physical sources (i.e., conduction or hydrodynamic instabilities that transport energy across the contact discontinuity). While these two channels have very different causes, they can have similar effects on the momentum and energy evolution of the system (Fierlinger et al. 2016). We will look at these two channels in turn.

Hydrodynamic solvers that advect mass between adjacent cells fundamentally introduce mixing errors. These errors can be decreased by improving the resolution or decreasing the mass advected between cells (as we have done by moving our numerical

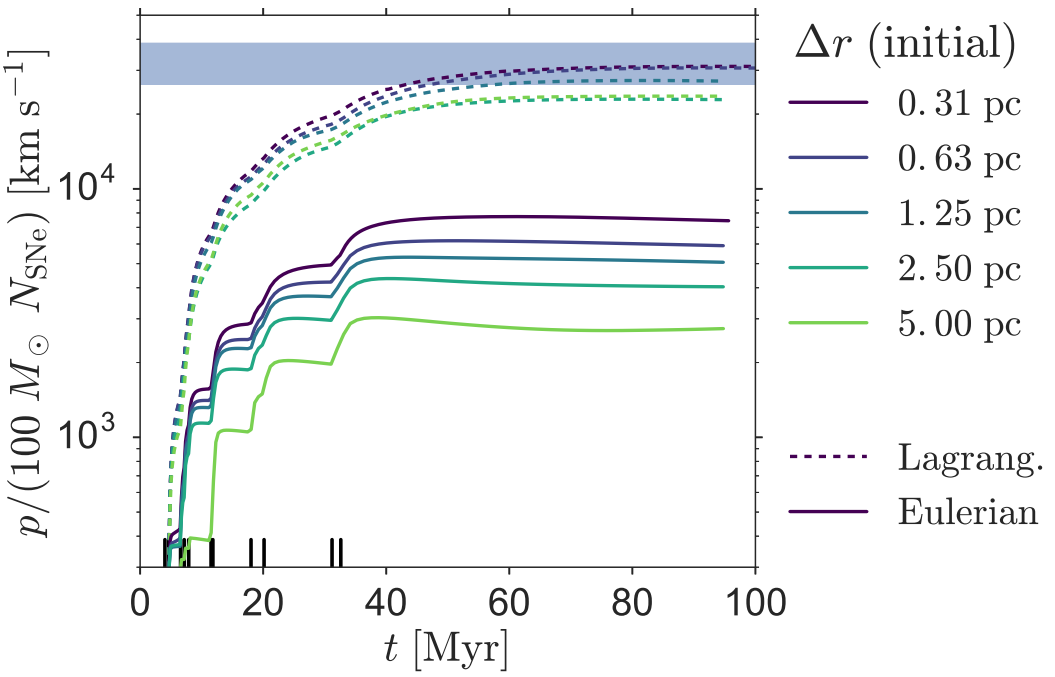

Figure 2.15: The momentum evolution of a $\rho = 1.33\ m_{\rm H}\ {\rm cm}^{-3}$, $Z = Z_\odot$, $M_{\rm cluster} = 10^3\ M_\odot$ ($N_{\rm SNe} = 11$) cluster, rerun with a range of initial resolutions, using both Eulerian and Lagrangian methods. The asymptotic momentum predicted by our model is shown by the blue horizontal band which bounds the $16^{\rm th}$ and $84^{\rm th}$ percentiles of the predictive distribution.

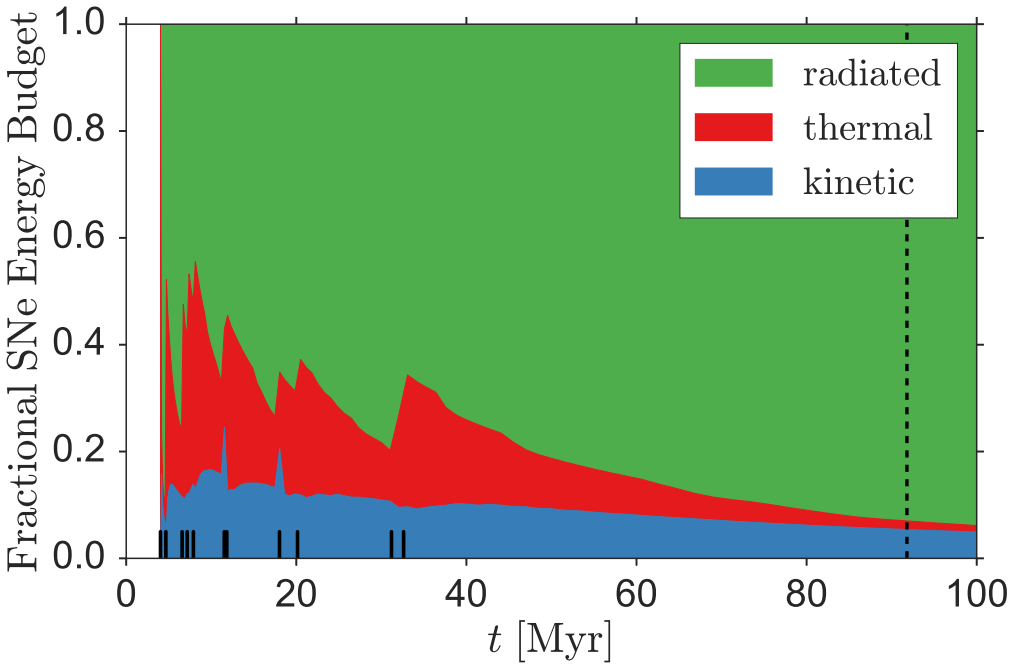

Figure 2.16: Same as Figure 2.4, except now for a $\rho = 1.33\ m_{\rm H}\ {\rm cm}^{-3}$, $Z = Z_{\odot}$, $M_{\rm cluster} = 10^3\ M_{\odot}$ ($N_{\rm SNe} = 11$) cluster, evolved using Lagrangian methods with an initial resolution of 0.6 pc.

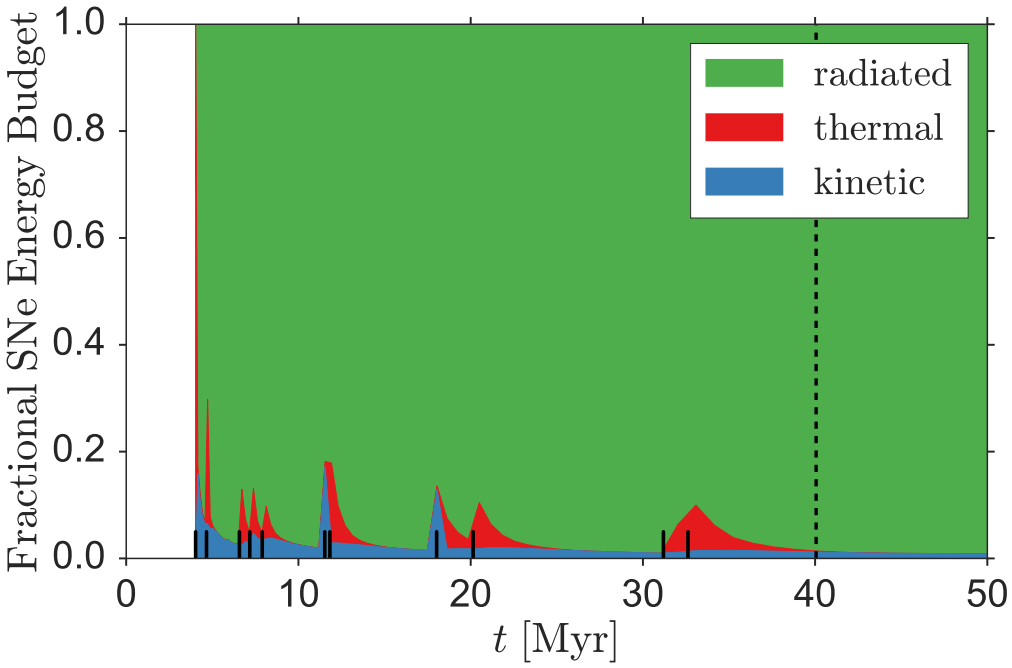

Figure 2.17: Same as Figure 2.16, except with the resolution degraded to 2.5 pc, and using a fixed Eulerian mesh.

mesh with the fluid). In order to understand how the choice of resolution and numerical methods affects our results, we re-ran one of our clusters ($\rho = 1.33\ m_{\rm H}\ {\rm cm}^{-3}$, $Z = Z_{\odot}$, $M_{\rm cluster} = 10^3 M_{\odot}$, $N_{\rm SNe} = 11$; this particular cluster will be useful in later comparisons to Kim & Ostriker (2015)) at a number of initial resolutions, using both a fixed mesh and a moving mesh. (A fixed mesh corresponds to Eulerian hydrodynamics – fixing $w = 0$ in our methods discussed in section 2.2 – which is less accurate and more diffusive than our pseudo-Lagrangian methods.) We show the results in Figure 2.15 and note a few key observations. First, the Lagrangian runs appear to be converged (within the uncertainties of our predictive model) by the resolution used for this cluster in our parameter study (an initial resolution of 0.6 pc). Second, the Eulerian runs introduce larger errors (as expected) and converge much more slowly. This suggests that Eulerian and low resolution simulations could have greater errors than high resolution, Lagrangian simulations. We can better understand these errors by comparing the energy evolution of a high resolution Lagrangian run (Figure 2.16) and a low resolution Eulerian run (Figure 2.17). While the same amount of energy is injected for each SN in both simulations, that energy is radiated away much more rapidly in the low resolution (Eulerian) simulation, draining the bubble of the energy which drives the momentum growth seen in Figure 2.15. (This connection between cooling time and resolution for multiple SNe was also found by Krause et al. (2013).) The amount of mixing can significantly impact the final momentum, but given the convergence seen in Figure 2.15, the results we have obtained appear to be converged.

While our resolution study suggests that high resolution, low diffusion simula-

tions are required to achieve accurate results, that conclusion might not apply if there are stronger, physical diffusive processes are present (Fierlinger et al. 2016). For a SNR or a superbubble, a number of processes can mix gas between the hot bubble interior and the cool shell; for a review see Appendix B of Fierlinger et al. (2016). Our 1D code cannot simulate many of these processes directly[3], but higher dimensional simulations can. In order to test the effects of these more complex mixing interfaces, we will compare our results to existing 3D simulations of multiple SNe in an inhomogeneous background.

Martizzi et al. (2015), Walch & Naab (2015) and Kim & Ostriker (2015) all tested the effects that a turbulent or multi-phase background might have on SNR evolution. For 1 SN, they all found that an inhomogeneous background makes a relatively small difference: a change of less than 60%. They also test multiple SNe in an inhomogeneous background, but none compare the results to multiple SNe in a homogeneous background. So in order to understand the effect of mixing in the case of multiple SNe, we will compare one of our clusters ($\rho = 1.33\ m_{\rm H}\ {\rm cm}^{-3}$, $Z = Z_{\odot}$, $M_{\rm cluster} = 10^3 M_{\odot}$, $N_{\rm SNe} = 11$; the cluster used in our resolution study) with a multi-phase multiple SNe cluster from Kim & Ostriker (2015) ($\rho = 1.4\ m_{\rm H}\ {\rm cm}^{-3}$, $Z = Z_{\odot}$, $N_{\rm SNe} = 10$). Note that although these clusters were chosen to be as similar as possible (except with a difference in background media), they also differ in SN delay time distributions, ejecta and mass loss prescriptions. Nevertheless, we can compare our results to those of Kim & Ostriker (2015) and find that their cluster cools much more rapidly than ours, leading to an

[3]There exist prescriptions for approximating mixing instabilities in 1D codes (e.g., Duffell 2016), but none were incorporated in our work.

asymptotic momentum per SN which is a factor of 20 lower than ours.

We believe that this difference is due to physical mixing that is present in their simulations but not ours. While our results can be brought into agreement with theirs by degrading our resolution and using less accurate numerical methods (as shown by our convergence study), subsequent simulations have shown their results to be converged with respect to resolution (Kim et al. 2017). If a two-phase background results in significantly increased physical mixing, that would explain how their converged simulations could appear similar to our less-accurate, unresolved simulations: a strong source of physical mixing can appear similar to strong artificial mixing (Fierlinger et al. 2016), while being less sensitive to the resolution.

One might ask at this point whether the enhanced momentum injection we find is solely a result of our use of 1D simulations, which necessarily suppressing mixing. Such a conclusion might be comforting, but is far from warranted. The overall lesson to draw from this comparison is that the cooling rate and momentum budget for bubbles produced by multiple SNe is exquisitely sensitive to the amount of mixing, whether physical or numerical. For a homogeneous background, very high resolution is required to get a converged value for the asymptotic momentum. This resolution requirement can be rendered irrelevant if strong, physical mixing occurs, allowing convergence at much lower resolution. But accurate results require more than just convergence; to be confident in the accuracy of a set of results, one must be confident that the physical mixing processes have been properly captured. In their multiple SNe simulations Kim & Ostriker (2015) include a two-phase background, but not magnetic fields, which are known to suppress

mixing across contact discontinuities in other contexts (e.g., Markevitch & Vikhlinin 2007). Thus their results should probably be regarded as lower limits. Sharma et al. (2014) do include magnetic fields, but only the context of a uniform medium. Given the state of the field, and the resolution requirements we have obtained in the uniform case, it seems clear that there is an urgent need to re-examine the momentum budget of clustered SNe in a multi-dimensional context, properly including all the mechanisms that can both enhance and suppress mixing.

2.5.3 The Effects of Additional Physics

In order to render the problem as clean as possible, we have focused only on type II SN feedback in a uniform medium. We now consider how other physical processes that we have heretofore neglected might alter our results.

2.5.3.1 Pre-SN Radiative Feedback

Before any SNe occur, we expect pre-SN feedback to already be sculpting the region. In particular, ionizing radiation from young stars can create an overpressured, expanding bubble, lowering the density in which SNe occur. In addition, expansion of the H II region will by itself add some momentum to the gas. Neither effect is included in our model, and we would like to understand if this significantly biases our results.

The ionizing luminosity of a cluster of mass $M_{\rm cluster}$ is $Q = 10^{49.6}\ M_{\rm cluster}/(10^3 M_\odot)$ s^{-1} (Leitherer et al. 1999). These photons will ionize a bubble of gas around the cluster, raising the temperature to 10^4 K. For density $\rho < m_{\rm H}$ cm^{-3}, this ionized region is not

much hotter than the background (which has a temperature only slightly below 10^4 K, appropriate for warm neutral gas). This means the ionized bubble will not be significantly over-pressured compared to the background, so it will not expand significantly. For clusters in backgrounds of density $\rho < m_{\rm H}$ cm^{-3}, we therefore do not expect pre-SN radiative feedback to affect our results.

For higher densities, the equilibrium temperature of the gas is well below 10^4 K, so the 10^4 K ionized bubble is significantly over-pressured compared to its surroundings. This will allow it to expand, lowering the density in which SNe occur. For uniform density, and neglecting the small range of parameter space where radiation pressure effects will be significant (Krumholz & Matzner 2009), the H II bubble radius $r_{\rm II}$ will be governed by the classical Spitzer (1978) solution[4] (Krumholz 2017, chapter 7),

$$r_{\rm II} \approx r_{S,0} \left(\frac{7t}{2\sqrt{3}t_{S,0}} \right)^{4/7}, \tag{2.31}$$

where $r_{S,0}$ is the Strömgren radius at the start of expansion, given by[5]

$$\begin{aligned} r_{S,0} &= \left[\frac{3Q\mu^2 m_{\rm H}^2}{4(1.1)\pi\alpha_B \rho^2} \right]^{1/3} \\ &= 3.1 Q_{49}^{1/3} n_2^{-2/3} T_{\rm II,4}^{0.272+0.007 \ln T_{\rm II,4}} \text{ pc}, \end{aligned} \tag{2.32}$$

and where $Q_{49} = Q/10^{49}$ s^{-1}, $\mu = 1.33$, $n_2 = \rho/(\mu m_{\rm H})/100$ cm^{-3}, $T_{\rm II}$ is the temperature

[4]This solution assumes $t \gg t_{S,0}$, otherwise expansion is expected to be negligible. For the cluster masses considered, this assumption typically fails for $n < 1$ cm^{-3}, but we already expected minimal expansion in such backgrounds.

[5]Note that this and the following expressions assume that He is singly ionized.

of the ionized gas, $T_{\rm II,4} = T_{\rm II}/10^4$ K, $\alpha_B \approx 2.54 \times 10^{-13} T_{\rm II,4}^{-0.8163-0.0208 \ln T_{\rm II,4}}$ cm^3 s^{-1} is the case B recombination coefficient (Draine 2011), $t_{S,0} = r_{S,0}/c_{\rm II}$,

$$c_{\rm II} = \sqrt{2.2 \frac{k_{\rm B} T_{\rm II}}{\mu m_{\rm H}}} = 12 T_{\rm II,4}^{1/2} \text{ km s}^{-1}, \tag{2.33}$$

is the ionized gas sound speed. The corresponding density inside the H II region is

$$\rho_{\rm II} = \rho \left(\frac{r}{r_{S,0}} \right)^{-3/2} = \rho \left(\frac{7t}{2\sqrt{3} t_{\rm S,0}} \right)^{-6/7}. \tag{2.34}$$

The mass and velocity of the swept-up shell are

$$v_{\rm II} = \frac{4}{7} \frac{r_{\rm II}}{t} \tag{2.35}$$

$$M_{\rm sh} = \frac{4}{3} \pi r_{\rm II}^3 \rho. \tag{2.36}$$

We are interested in the properties of the H II region at a time of ≈ 4 Myr, when the first supernova occurs; these are

$$r_{\rm II} = 27 M_{\rm cluster,3}^{1/7} n_2^{-2/7} \left(\frac{t}{4 \text{ Myr}} \right)^{4/7} T_{\rm II,4}^{0.402+0.003 \ln T_{\rm II,4}} \text{ pc} \tag{2.37}$$

$$\frac{\rho_{\rm II}}{\rho} = 0.077 M_{\rm cluster,3}^{2/7} n_2^{-4/7} \left(\frac{t}{4 \text{ Myr}} \right)^{-6/7} T_{\rm II,4}^{-0.195+0.006 \ln T_{\rm II,4}} \tag{2.38}$$

$$p_{\rm II} = 7.7 \times 10^5 M_{\rm cluster,3}^{4/7} n_2^{-1/7} \left(\frac{t}{4 \text{ Myr}} \right)^{-3/7} \times T_{\rm II,4}^{1.41+0.02 \ln T_{\rm II,4}} M_\odot \text{ km s}^{-1} \tag{2.39}$$

where $M_{\rm cluster,3} = M_{\rm cluster}/10^3 M_\odot$, $p_{\rm II}$ is the momentum of the shell. For the rest of this section we assume $T_{\rm II,4} = 1$.

Based on these results, for $M_{\rm cluster} \approx 100\ M_\odot$ the extra momentum in the H II region ($\approx 2\times10^5\ M_\odot$ km s^{-1}) is $\sim 50\%$ smaller than that injected by SNe ($\approx 3\times10^5\ M_\odot$ km s^{-1}), and the fractional contribution drops fairly rapidly for more massive clusters thanks to the sublinear scaling of $p_{\rm II}$ with $M_{\rm cluster}$ (Equation 2.39). Thus the extra momentum injected directly by the H II region is mostly negligible.

The second effect, lowering the density of the medium into which the SNe expand, matters if the region of lowered density encompasses where a SNR would otherwise experience significant cooling: beyond the shell formation radius of the first SN (when the remnant exits the Sedov phase). This radius is (Kim & Ostriker 2015; their equation 8, assuming an energy budget of 10^{51} ergs per 100 $M_\odot$ of stars)

$$r_{\rm sh} = 6.4 M_{\rm cluster,3}^{0.29} n_2^{-0.42}\ {\rm pc}. \tag{2.40}$$

Given the weak scaling of both $r_{\rm sh}$ and $r_{\rm II}$ with $M_{\rm cluster}$ and n, and the lower coefficient for $r_{\rm sh}$, this means that the presence of an H II region can at least potentially affect the evolution at densities $n \gtrsim 1$ cm^{-3}.

To quantify the effect of this radiative feedback, we can use the results of Walch & Naab (2015), who ran simulations with and without pre-SN ionization for $N_{\rm SNe} = 1$ and $\rho \approx 100\ m_{\rm H}$ cm^{-3}. They found that ionization led to a pre-SNe bubble of density $\rho \approx 10\ m_{\rm H}$ cm^{-3}, which resulted in a final momentum 50% higher than their simulation

without ionization (runs "HCI" and "HC" respectively.) This change in momentum can be explained well by re-scaling the results from the non-ionized run to the density within the bubble of the ionized run (using the $p \propto \rho^{-1/7}$ scaling of Cioffi et al. (1988)), with the added extra momentum injected by the H II region directly. Thus for single SNe, it appears that the relevant density for a momentum feedback model is the ionized bubble density rather than the background density.

Assuming that this conclusion can be extended to the case of multiple SNe, we can quantity the effects of pre-SN ionizing radiation simply by combining the density scaling in Equation 2.38 with our best-fitting density dependences, $p \propto \rho^{-0.06}$ in the few-SN regime and $p \propto \rho^{0.14}$ in the superbubble regime. In the few-SN regime, this implies an increase in the momentum yield per SN by a factor of

$$f_{\rm II,few} \approx 1.17 M_{\rm cluster,3}^{-0.017} n_2^{0.034}. \tag{2.41}$$

The corresponding effect in the superbubble regime is a *decrease* in the momentum yield by a factor of

$$f_{\rm II,many} \approx 0.70 M_{\rm cluster,3}^{0.040} n_2^{-0.080}. \tag{2.42}$$

Thus in general we expect that the effect of a pre-SN H II region will be to alter the final momentum yield at the tens of percent level, with the sign of the effect depending on the whether we are in the few-SN or the superbubble regime.

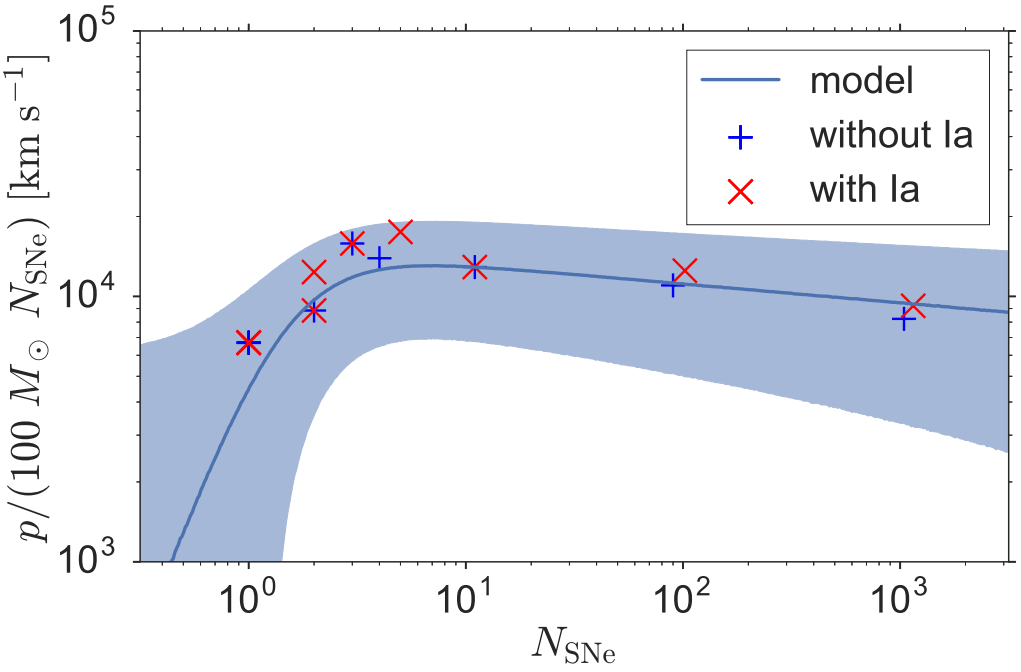

Figure 2.18: Comparison of simulations with just core-collapse SNe (marked by a blue +), and simulations with core-collapse and Type Ia SNe (marked by a red ×) for a set of simulations with $Z = Z_\odot$ and $\rho = 1.33 \times 10^{-3}\ m_{\rm H}\ {\rm cm}^{-3}$.

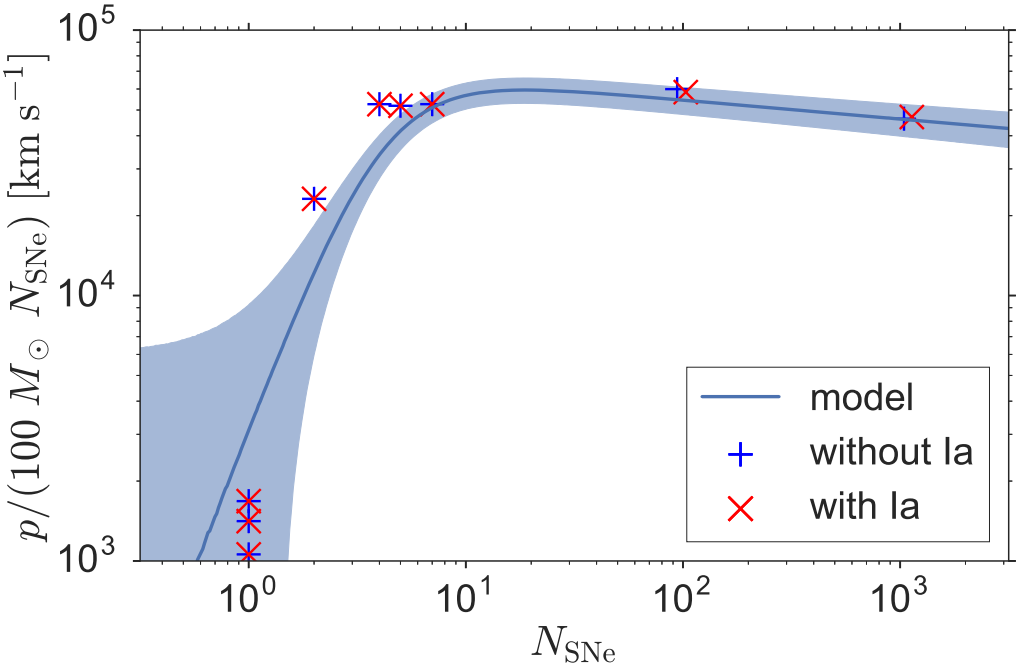

Figure 2.19: Same as Figure 2.18, except now for $\rho = 1.33 \times 10^2\ m_{\rm H}\ {\rm cm}^{-3}$ clusters.

2.5.3.2 Type Ia SNe

We just discussed pre-SN alterations to our results – what about changes after the core-collapse SNe occur? In particular, could subsequent Type Ia SNe rejuvenate an old superbubble?

To test this we rerun a subset of our highest and lowest density simulations with Type Ia SNe added after all the core-collapse SNe have occurred. We add 9.75×10^{-4} Type Ia SNe per $M_{\odot}$ of stars, with the exact number sampled from a Poisson distribution; this rate is taken from Kim et al. (2014), rescaled to a Kroupa (2002) IMF. We draw Type Ia SN delay times from a t^{-1} distribution, beginning at $t = 40$ Myr and extending to 100 Myr. This is not meant to be a complete accounting of the full effect of Type Ia SNe; most Type Ia SNe occur after much longer delays, and within 40-100 Myr the delay time distribution is poorly constrained (Maoz et al. 2012); this is simply to test the effects of SNe that might occur while the bubble still exists.

We show results for the low and high density runs in Figure 2.18 and Figure 2.19. For relatively short-delay Type Ia SNe, we find that they are consistent with our model if $N_{\rm SNe}$ is increased accordingly ($N_{\rm SNe} = N_{\rm core\text{-}collapse} + N_{\rm Type\ Ia}$). For long-delay SNe we caution against using our model; it is not guaranteed that both the progenitor will remain within the cluster and that the bubble will survive much longer than 100 Myr.

2.5.3.3 Self-Gravity

Previous studies of SNRs and superbubbles have differed in regards to whether they include or exclude gravitational forces. For example, Martizzi et al. (2015) and Kim & Ostriker (2015) do not include gravity, while Thornton et al. (1998) and Walch & Naab (2015) do include self-gravity. We chose not to include any gravitational forces in our main simulations, and now we estimate what effect that has on our results. In this section we focus on the effects of the self-gravity of the simulated gas, rather than external gravitational forces.

First, it is useful to understand why we did not include gravity ... in our simulations. This is partly a philosophical choice. The momentum budget we are attempting to compute is frequently used as an input to models of feedback-regulated star formation or wind generation. In such models, the feedback is compared to the force of gravity either analytically or numerically. For this purpose, the momentum budget in which we are interested is that before the effects of gravity are applied; to include gravity would be in effect to double-count it, by inserting it once into the subgrid feedback model and then a second time into the overall model. However, there is also a practical reason that we omit gravity. Strictly speaking, the self-gravitational potential of an infinite, uniform medium is undefined. We could assume an external potential dominates, but only if it were spherically symmetric about the cluster, which rules out many potentials of interest, like a galactic potential. Additionally, including a gravitational force, especially self-gravity, would cause our uniform background to

collapse. This was not a problem for Thornton et al. (1998), who simulated much less than a free-fall time, but caused Walch & Naab (2015) to limit their simulations to 1 Myr in duration. Unfortunately, limiting the duration of our simulations was not an option; we needed to simulate SNe over a period of at least 30 Myr, much longer than a free-fall time for most of our densities. Rather than artificially require a pressure gradient that ensures equilibrium, we chose to exclude gravitational forces.

Still, gravity exists in real systems, so we should try to understand its effect on our work. First, we will use analytic, simplified arguments to predict the effects of self-gravity on our simulations. We then use a simplified prescription to include self-gravity directly in our numeric simulations. By comparing those results we can begin to understand the effects of self-gravity and the limitations of our analytic model.

For an arbitrarily thin shell dominated by mass swept up from a constant density background the force of self-gravity is

$$F_{\rm grav} = \frac{GM_{\rm shell}^2}{2R_{\rm shell}^2} = \frac{8\pi^2}{9} G\rho_0^2 R_{\rm shell}^4. \tag{2.43}$$

When this inward force becomes greater than the force exerted by the pressure of the hot bubble, the momentum will stop increasing, ending the simulation (unless the bubble would have already mixed into the ISM and the simulation is already completed). This competing force from the hot bubble gas is

$$F_{\rm gas} \approx 4\pi R_{\rm shell}^2 P_{\rm bubble} = 3(\gamma - 1)\frac{E_{\rm R,int}}{R_{\rm shell}}. \tag{2.44}$$

These two forces become equal at a radius

$$R_{\rm max} = \left(\frac{27}{8\pi^2}(\gamma-1)\frac{E_{\rm R,int}}{G\rho_0^2}\right)^{1/5} \tag{2.45}$$

$$\approx 1200\left(\frac{N_{\rm SNe}}{1000}\right)^{1/5}\left(\frac{\rho_0}{1.33\times m_{\rm H}\ {\rm cm}^{-3}}\right)^{-2/5}\ {\rm pc} \tag{2.46}$$

where $R_{\rm max}$ is the maximum radius the bubble could reach, having assumed all the injected energy is retained as internal energy ($E_{\rm R,int} = N_{\rm SNe}\times 10^{51}$ ergs). This provides a simple way to include the effects of gravity via post-processing: using the $R_{\rm max}$ determined by Equation 2.45, we can truncate a simulation at that radius using the data in Table 2.2. If the bubble mixes into the ISM at a radius smaller than $R_{\rm max}$, then gravity is assumed to have no effect.

We can also re-run a subset of simulations including gravity explicitly. To do this, we calculate the force of self-gravity and the force due to a central point source of mass $M_{\rm cluster}$, and use these forces to compute the appropriate source terms for momentum and energy. These source terms are then only applied to the shocked gas (cells with $r < R_{\rm shock}$, where $R_{\rm shock}$ is the radius of the overdensity furthest from the center). By only applying gravity to shocked gas, we are able to approximate our analytic approach (which only considers gravity of the shell), and avoid the problem of our background collapsing. This effectively assumes the background is kept in equilibrium by a corresponding thermal or dynamic pressure gradient. We could have explicitly included this pressure gradient in our simulations, but chose not to, so that our with-gravity simulations would better correspond to our without-gravity simulations. A more sophisticated

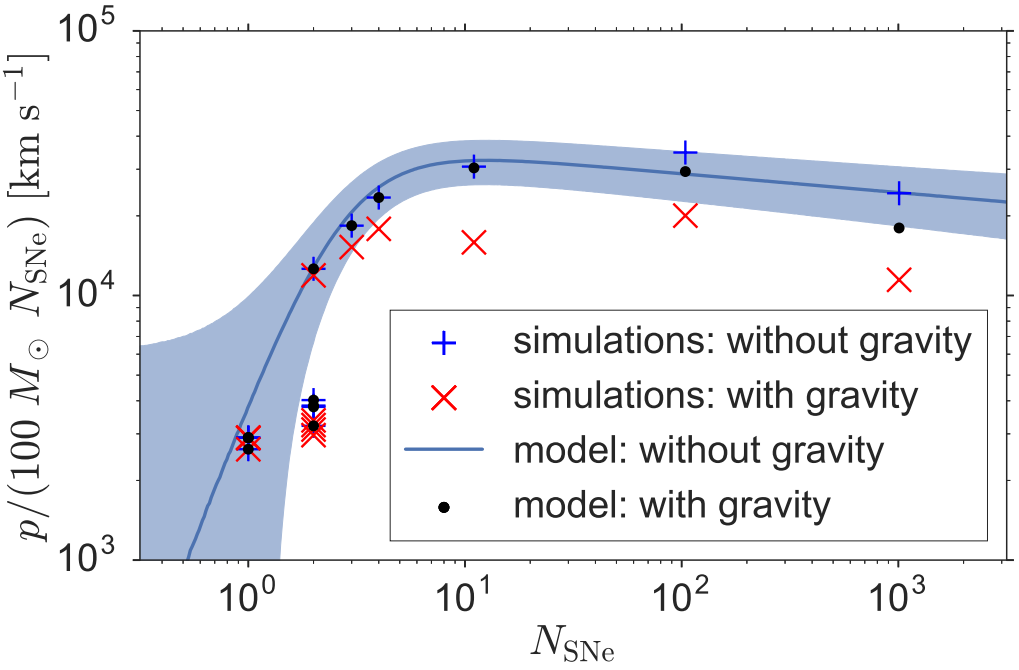

Figure 2.20: Comparison of simulations with no gravity (marked by a blue +) and simulations with gravity applied to the remnant (marked by a red ×) for all of our simulations with $Z = Z_\odot$ and $\rho = 1.33 \times 10^{-3}\ m_{\rm H}\ {\rm cm}^{-3}$. We also include a simple post-processing prediction for the effect of self-gravity, which truncates our without-gravity simulations if and when they reach $R_{\rm max}$ (Equation 2.45).

treatment of gravity using higher dimensional simulations would be worthwhile.

In Figure 2.20 we compare the results for simulations with and without gravity, and the results of our post-processing model which truncates the without-gravity simulations. We see that including gravity in simulations can decrease the final momentum, and that this decrease can be significant (compared to the uncertainties in our without-gravity model), and as large as a factor of 2.3. We also see that the post-processing truncation model typically over-predicts the final momentum (under-predicting the effects of gravity), relative to the simulations which incorporate gravity directly. This is expected; our post-processing model assumes all the SN energy remains as thermal energy, but in our simulations some SN energy is converted into kinetic energy, some into potential energy and most is radiated away. This causes simulations to stop at smaller radii than predicted, leading to lower final momenta than the post-processing model predicts.

We find that self-gravity can significantly change the final momentum, especially for clusters of many SNe, but that momentum is nevertheless still enhanced by a factor of about 4 compared to the single SN case.

2.5.3.4 Galactic Environment

In subsubsection 2.5.3.3 we investigated the effects of self-gravity, but in some cases a galactic gravitational potential might play a larger role in shaping the late-time dynamics of large bubbles. In this section we will estimate the effects of a galactic gravitational potential, as well as the effects of rotational shear and disc breakout. For

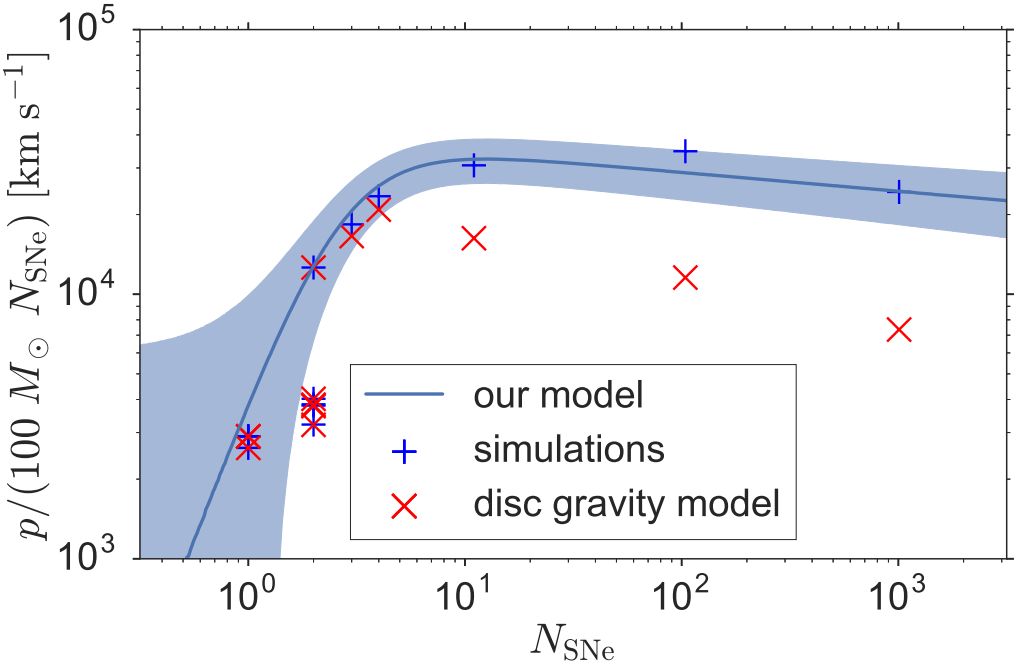

Figure 2.21: Comparison of simulations with no galactic gravity (marked by a blue +) and our simple post-processing model of galactic gravity in a disc for all of our simulations with $Z = Z_\odot$ and $\rho = 1.33 \times 10^{-3}\ m_{\rm H}\ {\rm cm}^{-3}$.

each of these effects, we will use a post-processing correction similar to that used in subsubsection 2.5.3.3: calculate a radius or time where our assumptions break down, and then use the data in Table 2.2 to truncate the bubble evolution at that radius or time.

First, we consider the gravitational force perpendicular to a galactic disc, produced by the galactic gravitational potential. The acceleration due to this force can be written as:

$$g = \frac{z}{r_{\rm g}} \frac{v_{\rm c}^2}{r_{\rm g}} \tag{2.47}$$

where v_c is the circular velocity at a galactocentric radius r_g, and z is the distance from the midplane. For simplicity, we will assume all of the mass of the shell is in a plane at height $z = R_{shock}$. This results in a force

$$F_{grav} = M_{shell} \frac{R_{shell}}{r_g} \frac{v_c^2}{r_g}. \tag{2.48}$$

For a Milky Way-like galaxy with $v_c = 200$ km s^{-1} and a cluster at $r_g = 3$ kpc, this gravitational force is equal to the force from the bubble pressure when the shock is at a radius

$$R_{max} = \left(\frac{9(\gamma - 1)}{4\pi} \frac{1}{\rho_0} \frac{v_c^2}{r_g^2} E_{R,int} \right)^{1/5} \tag{2.49}$$

$$\approx 700 \left(\frac{N_{SNe}}{1000} \right)^{1/5} \left(\frac{\rho_0}{1.33 \times m_H \text{ cm}^{-3}} \right)^{1/5} \times \left(\frac{v_c}{200 \text{ km s}^{-1}} \right)^{-2/5} \left(\frac{r_g}{3 \text{ kpc}} \right)^{2/5} \text{ pc}. \tag{2.50}$$

As with our self-gravity model, we create a model which truncates our simulations if and when they reach this radius. In Figure 2.21, we compare the results of this galactic gravity model to a subset of our simulations. Similar to the self-gravity results shown in Figure 2.20, galactic gravity has the largest effect for the largest clusters. We also find that this galactic gravity model is able to reduce the momentum by a greater factor than our self-gravity model; the self-gravity model never reduced the momentum by more than a factor of 1.4, whereas the galactic gravity model can reduce the momentum by up to a factor of 3.3. As with self-gravity, however, we caution that

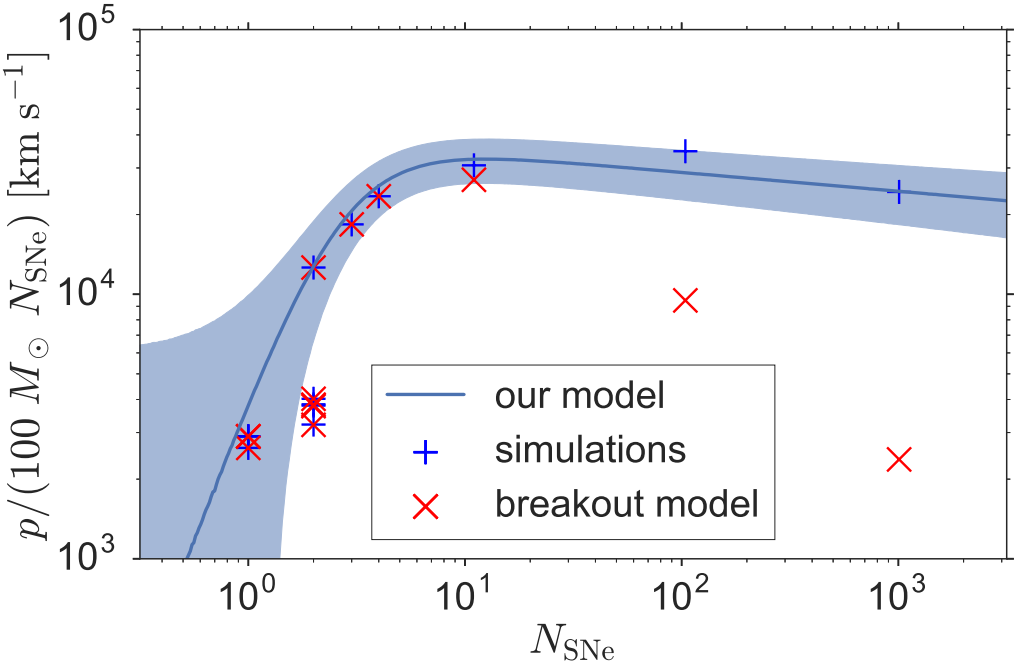

Figure 2.22: Comparison of simulations with no disc breakout (marked by a blue +) and our simple post-processing model of disc breakout for all of our simulations with $Z = Z_\odot$ and $\rho = 1.33 \times 10^{-3}\ m_{\rm H}\ {\rm cm}^{-3}$.

for many applications, the momentum budget of interest will be that excluding the effects of disc gravity, since if disc gravity is included in the overall model, it should not be double-counted by also including it in the subgrid feedback model.

Our disc gravity model is incomplete; among other things, it ignores the density and pressure gradients that result from this gravitational potential. As a superbubble expands vertically, it will find less dense and lower pressure gas; as it expands horizontally, it will not experience such gradients in the background gas. This will break the spherical symmetry of the bubble expansion, and can lead to disc breakout where the

bubble expansion becomes predominantly vertical. When studying this phenomenon with hydrodynamic simulations, Mac Low et al. (1989) found that spherical symmetry is often broken and shell mixing instabilities are significant by the time the bubble has expanded 3-4 scale heights in the vertical direction. Applying this to a Milky Way-like galaxy, using a thin-disc scale height of 100 pc, we can create a model which cuts off the momentum growth when a bubble reaches $R_{\rm max} = 400$ pc. The results of this model can be seen in Figure 2.22. As with our previous models, the effect of this model is stronger for larger clusters, but this model predicts effects which increase much more rapidly as cluster size increases. While there is no significant effect on our 11 SNe simulation, it has the largest effect for the 1008 SNe simulation for all of the models we have tested, decreasing the momentum by a factor of about 10. It is important to understand that this behaviour is not a result of the total momentum decreasing for large bubbles. It is simply that, for the largest clusters, the bubble expands to the breakout radius before a significant fraction of the SNe occur, and, by assumption, these additional SNe then contribute no additional momentum. This causes the average momentum per SN to fall, because the extra SNe are counted in the denominator but not the numerator of our average.

In addition to considering distortions caused by expansion perpendicular to the disc plane, we can also consider distortions due to shear within the disc plane. Adopting a shear timescale

$$t_{\rm shear} = \Omega_{\rm orb}^{-1} \frac{r_{\rm g}}{R_{\rm shell}} \tag{2.51}$$

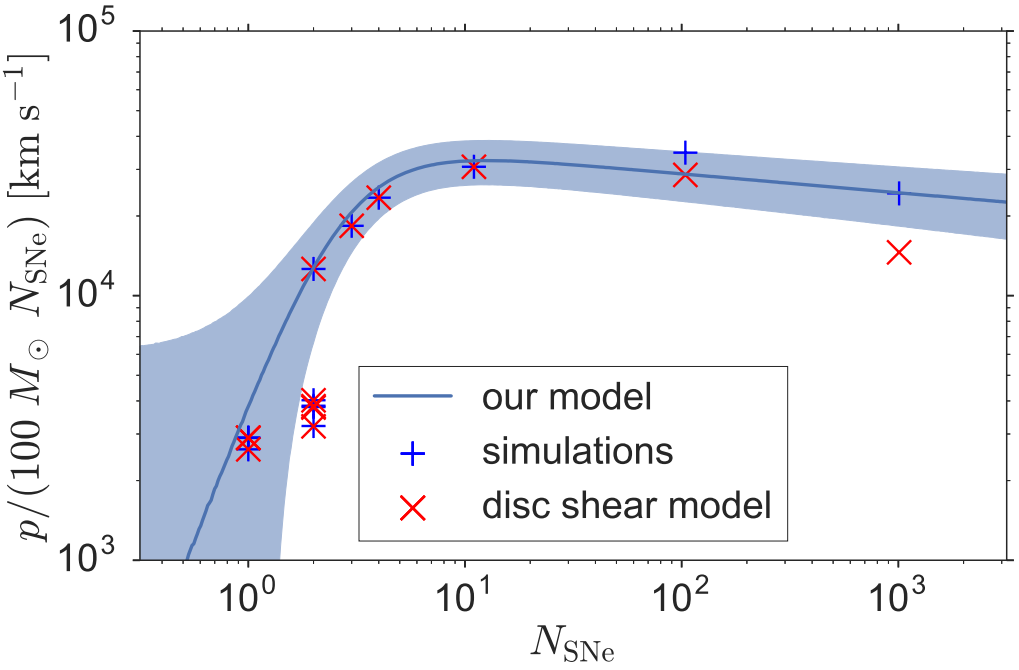

Figure 2.23: Comparison of simulations with no rotational shear (marked by a blue +) and our simple post-processing model of rotational shear in a disc for all of our simulations with $Z = Z_\odot$ and $\rho = 1.33 \times 10^{-3}\ m_{\rm H}\ {\rm cm}^{-3}$.

for a disc orbital frequency $\Omega_{\rm orb} = v_{\rm c}/r_{\rm g}$, we can truncate our bubble evolution when the bubble age ($t - t_{\rm first\ SN}$) exceeds the shear time. Results can be seen in Figure 2.23. Once again, only the largest clusters are significantly affected, with the largest effect being a decrease of a factor of 1.75 for the 1008 SNe cluster.

Comparing these post-processing models, we find that all of the galactic models (gravity, breakout and shear) are predicted to have larger effects than the self-gravity model, but these galactic models introduce a number of free parameters which we have not explored (disc circular velocity, galactocentric radius and disc scale height). It is important to remember that these models are only very rough estimates. When investigating self-gravity, we found that directly including self-gravity in simulations led to a much stronger effect than predicted by our simple model. The same is likely true for these galactic effects, which we cannot directly include in our simulations. Still, even though every model was structured to decrease the final momentum, we always saw that clustering can lead to an increased momentum efficiency, compared to our single SN results. Moreover, all of these corrections are relatively small for the most efficient clusters, those producing ~ 10 SNe ($M_{\rm cluster} \sim 1000 M_{\odot}$).

Overall, this suggests that superbubble models are less cleanly separable from galactic dynamics than single SNR models. Single, isolated SNRs might only expand to 100 pc over 2 Myr, allowing us to largely ignore effects like disc shear and galactic gravity. As we have shown, these effects can play a significant role for models of clustered SNe and superbubbles. Testing these effects self-consistently goes beyond the capability of our 1D simulations, but we have already gained some insight from our simplified

post-processing models. By design, all of these models lowered the final momenta of our bubbles (by as much as a factor of 10, for the disc breakout model applied to our largest bubble), but every model still predicted that clustered SNe lead to enhanced momentum feedback (for instance, our disc breakout model still predicts an average momentum per SN 4 times larger than the single SN value). Both clustering of SNe and effects from the host galaxy seem to play significant roles in the overall SNe momentum budget; it would be useful for higher dimensional simulations to explore these two effects simultaneously in the future.

2.6 Conclusions

We perform several hundred 1D simulations to study the momentum delivered to the interstellar medium by clustered supernova explosions over a wide range of star cluster sizes, gas densities and metallicities. Our simulations use a realistic IMF paired with realistic stellar lifetimes, and we evolve them at very high numerical resolution until the momentum of the expanding shell reaches a maximum. At the end of our simulations, we find that our clusters typically retain $1-10\%$ of the injected SN energy (similar to isolated SNe), but clustered SNe produce significantly more momentum per SN than isolated SNe, i.e., the momentum injected by a star cluster is a superlinear function of the number of SNe that explode within it. Clustering has the largest impact for $10-100$ SNe, leading to an order of magnitude increase in the momentum per SN. When integrating over the observed cluster mass function, our findings suggest that the

mean SN momentum budget per mass of stars formed is $p/m_* \approx 1 - 2 \times 10^4$ km s^{-1}, which is $\sim 0.5 - 1$ dex higher than the value of ≈ 3000 km s^{-1} most commonly adopted in the literature.

The increased momentum budget for SNe will have strong implications for any simulation or analytic model that relies on SN momentum injection. In galaxies where the overall star formation rate is high enough for clustering of SNe to be significant for quenching star formation and producing galactic winds (i.e., perhaps not in dwarfs, but almost certainly in more massive galaxies), using our updated SN momentum budget may cause these models' predicted star formation rates to decrease by the same factor of $\sim 0.5 - 1$ dex by which the SN momentum budget increases. This may render the models inconsistent with the observed relationship between gas and star formation rate, which will require changes in other free parameters to bring the models back into agreement. The implications for galactic wind launching are less clear, since the increased SN momentum budget will be offset by overall lower star formation rates.

To facilitate the implementation of our results in 3D numerical simulations that include explicit supernova momentum injection, we provide a fitting formula for the momentum per SN as a function of cluster size, ambient density and metallicity. This formula is suitable for implementation in galaxy simulations capable of resolving individual star clusters, typically $\sim 10^2 - 10^5$ $M_\odot$. We also provide tabulated outputs from our simulations, for those who wish to calibrate subgrid models at a range of size scales. For lower resolution simulations we recommend the value of $p/m_* \approx 1 - 2 \times 10^4$ km s^{-1} we obtain by integrating over the cluster mass function. Regardless of resolution,

it is clear that the existing body of simulations may need to be revisited, and some of the strong assumptions that previous authors have adopted to make feedback more effective (e.g., assuming efficient trapping of infrared radiation) may be relaxed. Other numerical schemes, such as turning off radiative cooling for an extended period of time (e.g., Stinson et al. 2006) or stochastically injecting thermal energy in order to delay cooling (e.g., Dalla Vecchia & Schaye 2012), may prove to be closer to reality than had previously been assumed.

Properly capturing the effect of clustering in a 1D simulation requires very high numerical resolution to avoid over-cooling through numerical mixing. The resolution requirements may be less severe, and the momentum injection rate lower, in higher dimensions where instabilities may produce mixing at large physical scales. However, the size scale and mixing rate due to instabilities likely depends strongly on the properties of the host galaxy, the nature of the background into which the SNR is propagating and the microphysical details like magnetic fields and conduction near the contact discontinuity. Since we are unable to directly include these effects in our simulations, we create simple models to estimate the strengths of some of these effects. While these models are able to decrease the momentum of the most-massive cluster by a factor of 10, lower mass (more common) clusters are less affected. For each of our models, the average momentum per SN remains greater than the fiducial, unclustered value by a factor of 4, assuming a realistic cluster mass distribution. But these are just simple estimates; no present simulation includes all these effects, and thus the correct momentum budget for clustered SNe in multiple dimensions remains uncertain. While the correct momentum

budget may not be as high as $\sim 0.5 - 1$ dex greater than the commonly-adopted single SN value (as we find in one dimension), it is still likely greater than the single SN value. There is clearly an urgent need for further study.

Chapter 3

3D simulations of clustered SNe: The momentum budget of clustered SN feedback in a 3D, magnetized medium

(Much of this text is going to come directly from Gentry et al. (2019); no changes to the substance have been made.)

While the evolution of superbubbles driven by clustered supernovae has been studied by numerous authors, the resulting radial momentum yield is uncertain by as much as an order of magnitude depending on the computational methods and assumed properties of the surrounding interstellar medium (ISM). In this work, we study the origin of these discrepancies, and seek to determine the correct momentum budget for a homogeneous ISM. We carry out 3D hydrodynamic (HD) and magnetohydrodynamic (MHD) simulations of clustered supernova explosions, using a Lagrangian method and

checking for convergence with respect to resolution. We find that the terminal momentum of a shell driven by clustered supernovae is dictated primarily by the mixing rate across the contact discontinuity between the hot and cold phases, and that this energy mixing rate is dominated by numerical diffusion even at the highest resolution we can complete, 0.03 $M_\odot$. Magnetic fields also reduce the mixing rate, so that MHD simulations produce higher momentum yields than HD ones at equal resolution. As a result, we obtain only a lower limit on the momentum yield from clustered supernovae. Combining this with our previous 1D results, which provide an upper limit because they allow almost no mixing across the contact discontinuity, we conclude that the momentum yield per supernova from clustered supernovae in a homogeneous ISM is bounded between 2×10^5 and 3×10^6 $M_\odot$ km s^{-1}. A converged value for the simple homogeneous ISM remains elusive.

3.1 Introduction

Feedback from supernovae (SNe) is an important component of understanding the interstellar medium (ISM), galactic winds, and galactic evolution (e.g., McKee & Ostriker 1977; Dekel & Silk 1986; Murray et al. 2005; Ostriker & Shetty 2011; Kim et al. 2011; Jenkins & Tripp 2011; Hopkins et al. 2012; Dekel & Krumholz 2013; Faucher-Giguère et al. 2013; Creasey et al. 2013; Thompson & Krumholz 2016). Unfortunately, the processes governing the strength of SN feedback operate non-linearly and at small scales. This makes it difficult to include the effects of SNe in analytic models or large

galactic simulations without a simplified prescription for SN feedback.

In the past, most investigations of the key factors governing SN feedback strength have focused on single, isolated SNe (e.g., Chevalier 1974; Cioffi et al. 1988; Thornton et al. 1998; Iffrig & Hennebelle 2015). In reality, however, core collapse SNe are clustered in space and time: massive stars are born in clusters, and explode after $\sim 3 - 40$ Myr, before these stars can significantly disperse. The few studies that have looked at the feedback from multiple clustered, interacting SNe have found conflicting results. While some studies have found relatively small changes in the momentum from clustering SNe (Kim & Ostriker 2015; Walch & Naab 2015; Kim et al. 2017), others have found that it could increase the average momentum per SN to 5-10 times greater than the traditional yields for isolated SNe (Keller et al. 2014; Gentry et al. 2017).

It has been suggested that the differences in results for clustered SN simulations could stem from the different levels of mixing in the simulations, from both physical and non-physical sources. Unfortunately, each recent simulation was idealized in significantly different ways, which makes it difficult for us to directly isolate which aspects were the primary drivers of the differences. Our goal in this paper is to identify the causes of the discrepancies between different published results, and resolve whether, when including appropriate physics, clustering does in fact lead to significant changes in the terminal momentum of supernova remnants.

One of the key issues that we investigate is dimensionality and resolution. We found that clustering produces an order of magnitude enhancement in momentum (Gentry et al. 2017), but these results were based on 1D spherically symmetric simulations.

Assuming spherical symmetry is potentially misleading because we know of fluid instabilities (such as the Vishniac instabilities and the Rayleigh–Taylor instability) that affect SNR morphologies (Vishniac 1983, 1994; Mac Low & McCray 1988; Mac Low & Norman 1993; Krause et al. 2013; Fierlinger et al. 2016). Even small perturbations can be amplified and noticeably change key properties of SNR evolution. For isolated SN simulations, 1D and 3D simulations do not produce significantly different terminal momenta (e.g., Martizzi et al. 2015, Kim & Ostriker 2015, and Walch & Naab 2015 all find differences of less than 60% between 1D and 3D), but it is worth re-investigating the issue for clustered SNe. It could be that the longer time frame allows the instabilities to grow to have larger effects.

Conversely, our 1D simulations achieved much higher resolution than in any of the 3D simulations found in the literature. We found that the terminal momentum for clustered SNe did not converge until we reached peak resolutions better than 0.1 pc, far higher than the resolutions of published 3D simulations. Moreover, we achieved this convergence only by using pseudo-Lagrangian methods that minimized numerical diffusion across the contact discontinuity at the inner edge of the superbubble, whereas many of the published 3D results are based on Eulerian methods that, for fronts advecting across the grid at high speed, are much more diffusive. Indeed, it is noteworthy that the one published 3D result that finds a significant momentum enhancement from clustering (Keller et al. 2014) uses a Lagrangian method, while all the papers reporting no enhancement are based on Eulerian methods. Clearly, given the conjoined issues of resolution and dimensionality, further investigation is warranted.

Since the suppression and enhancement of mixing is a key unknown for the feedback budget of clustered SNe, we also explore the role of magnetic fields, which may reduce the amount of physical mixing. Our interest in this possibility comes primarily from the example of cold fronts in galaxy clusters, where magnetic fields draped across a shock front have been used to explain the stability of these cold fronts against fluid instabilities (Vikhlinin et al. 2001; Markevitch & Vikhlinin 2007; although see also Churazov & Inogamov 2004 who show that magnetic fields might not be necessary for stabilizing cold fronts).

In this paper, we first test the effects of bringing our simulations from 1D to 3D and carry out a 3D convergence study, and then we test the effects of adding magnetic fields into our 3D simulations. In section 3.2, we discuss our computational methods. In section 3.3, we discuss the results of our simulations, with a more detailed physical analysis of the significance of those results in section 3.4. In section 3.5, we discuss our conclusions and compare to other works.

3.2 Computational Methods

For this work we repeat one of the 1D simulations from Gentry et al. (2017), and conduct 3D simulations of the same set-up at a range of resolutions and including or excluding magnetic fields. For the most part our 1D simulations reuse the code developed by Gentry et al. (2017), with minor changes that we discuss in subsection 3.2.1. In subsection 3.2.2, we discuss the methodology for our 3D simulations, for which we

use the GIZMO code (Hopkins 2015; Hopkins & Raives 2016). We use GIZMO for this work because it has a Lagrangian hydrodynamic solver; in our previous 1D simulations, we found that Lagrangian methods were more likely to converge for simulations of clustered SNe (Gentry et al. 2017).

3.2.1 1D simulation

The methods for our 1D simulation are very similar to those used in our previous work (Gentry et al. 2017), with only slight modifications. First, we disable the injection of pre-SN winds, because injecting small amounts of mass over extended periods is impractical at the resolutions we are able to achieve in the 3D simulations. Second, we initialize the ISM to be at an equilibrium temperature ($T \sim 340$ K or a specific internal energy of $e_{\rm int} \sim 3.5 \times 10^{10}$ erg g^{-1} for an initial ISM density of $\rho_0 = 1.33\ m_{\rm H}$ cm^{-3} and gas-phase metallicity of $Z = 0.02$, rather than $T \sim 15\,000$ K).[1] This simplifies the analysis, as changes in energy now only occur in feedback-affected gas. Furthermore the initial temperature makes little difference as the gas would otherwise rapidly cool to its equilibrium state (the 15 000 K gas had a cooling time of a few kyr). Using these modified methods we reran the most-studied cluster from our previous work, one that had a stellar mass of $M_\star = 10^3\ M_\odot$ (producing 11 SNe) and was embedded in an ISM of initial density $\rho_0 = 1.33\ m_{\rm H}$ cm^{-3} and an initial gas-phase metallicity of $Z = 0.02$.[2] These changes allowed for more direct comparison with our 3D simulations,

[1] Throughout this paper we will quote temperatures calculated by GRACKLE which accounts for temperature dependence in the mean molecular weight, μ (Smith et al. 2017).

[2] This cluster can be found in the tables produced by Gentry et al. (2017) using the id 25451948-485f-46fe-b87b-f4329d03b203.

and do not affect our final conclusions.

The remainder of our methodology is identical to that of Gentry et al. (2017), which we summarize here for convenience. To generate a star cluster of given mass, we used the `SLUG` code (da Silva et al. 2012; da Silva et al. 2014; Krumholz et al. 2015) to realistically sample a Kroupa (2002) IMF of stars. We assume every star with an initial mass above $8M_{\odot}$ explodes as a core collapse SN. The lifetimes of these massive stars are computed using the stellar evolution tracks of Ekström et al. (2012); the SN mass and metal yields are computed using the work of Woosley & Heger (2007) while all SNe are assumed to have an explosion energy of 10^{51} erg. This cluster of stars is embedded in an initially static, homogeneous ISM, with each SN occurring at the same location. The resulting superbubble is evolved using a 1D, spherically symmetric, Lagrangian hydrodynamic solver first developed by Duffell (2016). Cells are split (merged) when they become sufficiently larger (smaller) than the average resolution. Metallicity-dependent cooling (assuming collisional ionization equilibrium) is included using `GRACKLE` (Smith et al. 2017). The simulation is evolved until the radial momentum reaches a maximum, at which point it is assumed that the superbubble mixes into the ISM.

3.2.2 3D simulations

Rather than adapt our 1D code to work in 3D, we instead chose to use the `GIZMO` simulation code (Hopkins 2015; Hopkins & Raives 2016), which includes a Lagrangian hydrodynamic solver with additional support for magnetohydrodynamics (MHD). For all of our runs, we used the Meshless Finite Mass solver on a periodic

Table 3.1: Initial Conditions. The mass resolution Δm is not included for the 1D run, as it is neither constant in space nor time.

Name	1D/3D	$B_{z,0}$ (μG)	β	Δx_0 (pc)	Δm ($M_\odot$)	L (pc)
`1D_06_HD`	1D	0	∞	0.6		1200
`3D_07_HD`	3D	0	∞	0.7	0.01	300
`3D_10_HD`	3D	0	∞	1.0	0.03	600
`3D_13_HD`	3D	0	∞	1.3	0.08	400
`3D_20_HD`	3D	0	∞	2.0	0.26	600
`3D_40_HD`	3D	0	∞	4.0	2.10	600
`3D_20_MHD`	3D	5	0.05	2.0	0.26	1200

domain, while ignoring the effects of gravity. We assume the gas follows an ideal equation of state with a constant adiabatic index $\gamma = 5/3$, as in our 1D simulation. When including magnetic fields, we used `GIZMO`'s standard solver for ideal MHD, as detailed in Hopkins & Raives (2016).

We modify the standard `GIZMO` code in two ways.[3] First, we added metallicity-dependent cooling using `GRACKLE` (Smith et al. 2017). Second, we inject SN ejecta, distributed in time, mass, and metal content using the same realization of SN properties as our 1D simulation. At the time of each SN, we inject new gas particles (each with mass approximately equal to the average existing particle mass) at random locations using a spherical Gaussian kernel with a dispersion of 2 pc centred on the origin. Each new particle has equal mass and metallicity, which are determined by the SN ejecta yields.[4] For simulations which include magnetic fields, we linearly interpolate the magnetic field vector from nearby existing particles to the origin, and initialize the new feedback

[3] Our modifications of `GIZMO` and our analysis routines can be found at: `github.com/egentry/gizmo-clustered-SNe`.

[4] While this approach leads to a well-sampled injection kernel at our higher resolutions, the kernel is only sampled by about five new particles for each SN in our lowest resolution run, `3D_40_HD`. This undersampling is not ideal and might slightly alter the bubble's evolution, but the stochasticity this introduces does not appear to affect our conclusions.

particles with that interpolated magnetic field vector. This procedure does not exactly preserve $\nabla \cdot \boldsymbol{B} = 0$, but GIZMO's divergence cleaning procedure rapidly damps away the non-solenoidal component of the field produced by our injection procedure.

We initialize the 3D simulations with the same static ($\boldsymbol{v} = 0$) homogeneous ISM as our 1D simulations ($\rho = 1.33 m_{\rm H}$ cm^{-3}, $Z = 0.02$ and $T \sim 340$ K). For simulations with magnetic fields, we include a homogeneous seed field, with $\boldsymbol{B} = (0, 0, 5)$ μG (identical to Iffrig & Hennebelle 2015), corresponding to a plasma $\beta \approx 0.05$. We place particles of mass Δm on an evenly spaced grid of spacing Δx_0, which extends for a box size of L. Particle locations are perturbed on the mpc scale in order to avoid pathologies in GIZMO's density solver caused by the symmetric grid. In Table 3.1, we present the parameters of the initial conditions. We typically[5] run each 3D simulation for 40 Myr, by which point the radial momentum, the quantity of primary interest for our study, had stabilized. We also carry out a smaller set of simulations in which we give the ISM a larger initial perturbation, whose magnitude shows a proper physical dependence on resolution. We describe these simulations in Appendix B.1, where we show that their results are nearly identical to those of our fiducial simulations. For this reason, we will not discuss them further in the main text.

3.3 Results

[5]The only exceptions are simulations 3D_07_HD and 3D_13_HD, which cannot be run to completion because they have smaller box sizes in order to minimize computational expense. These simulations are run until the shock approximately reaches the edge of the box. They are not meant to provide final values, but rather to enable us to investigate convergence of the results up to the times when these runs end.

Table 3.2: Results.

Name	N_{SNe}	t (Myr)	R_{eff} (pc)	M_{affected} ($10^6 M_\odot$)	$p_{\mathrm{max}}/N_{\mathrm{SNe}}$ ($100 M_\odot$ km s^{-1})	$p_{\mathrm{ratchet}}/N_{\mathrm{SNe}}$ ($100 M_\odot$ km s^{-1})	$p_{t=6.46\ \mathrm{Myr}}/N_{\mathrm{SNe}}$ ($100 M_\odot$ km s^{-1})	E_{kin} (10^{49} erg)	ΔE_{int} (10^{49} erg)
1D_06_HD	11	94.8	552	23.2	33978	33978	5987	65.0	26.3
3D_07_HD	11	–	–	–	–	–	1027	–	–
3D_10_HD	11	30.7	218	1.7	2425	2474	948	8.7	0.8
3D_13_HD	11	–	–	–	–	–	911	–	–
3D_20_HD	11	30.7	200	1.5	2128	2182	862	7.4	0.8
3D_40_HD	11	31.7	209	1.8	2007	2039	901	6.8	7.0
3D_20_MHD	11	29.6	423	10.5	1213	2418	1020	12.6	13.1

In Table 3.2, we provide a summary of the key numeric results of each simulation. First, we extract the time of maximum momentum after the last SN (only accurate within about 0.5 Myr.) At that time we extract the effective radius of the region affected by the bubble (particles with speeds greater than 1 m s^{-1})

$$R_{\text{eff}} = \left(\frac{3}{4\pi} \sum_{i:|\boldsymbol{v}_i|>1\text{m s}^{-1}} \text{Volume}_\text{i} \right)^{1/3} \tag{3.1}$$

and the total mass of those particles, M_{affected}.[6] Next we extract the kinetic energy E_{kin} and the change in the internal energy ΔE_{int} of the entire domain (which should be approximately equal to the values for the bubble-affected region). Finally, we extract the radial momentum using three approaches: one by simply measuring the radial momentum at the same time as the previous quantities (denoted p_{max}), another using a "ratchet" approach justified and explained in subsection 3.3.2 (denoted p_{ratchet}), and third by extracting the momentum at the last time achieved by all simulations ($t = 6.46$ Myr).

In the following subsections we discuss the results in greater detail. First, we compare our 1D and 3D results in subsection 3.3.1. Second, we look at the effect of including magnetic fields in our 3D simulations in subsection 3.3.2.

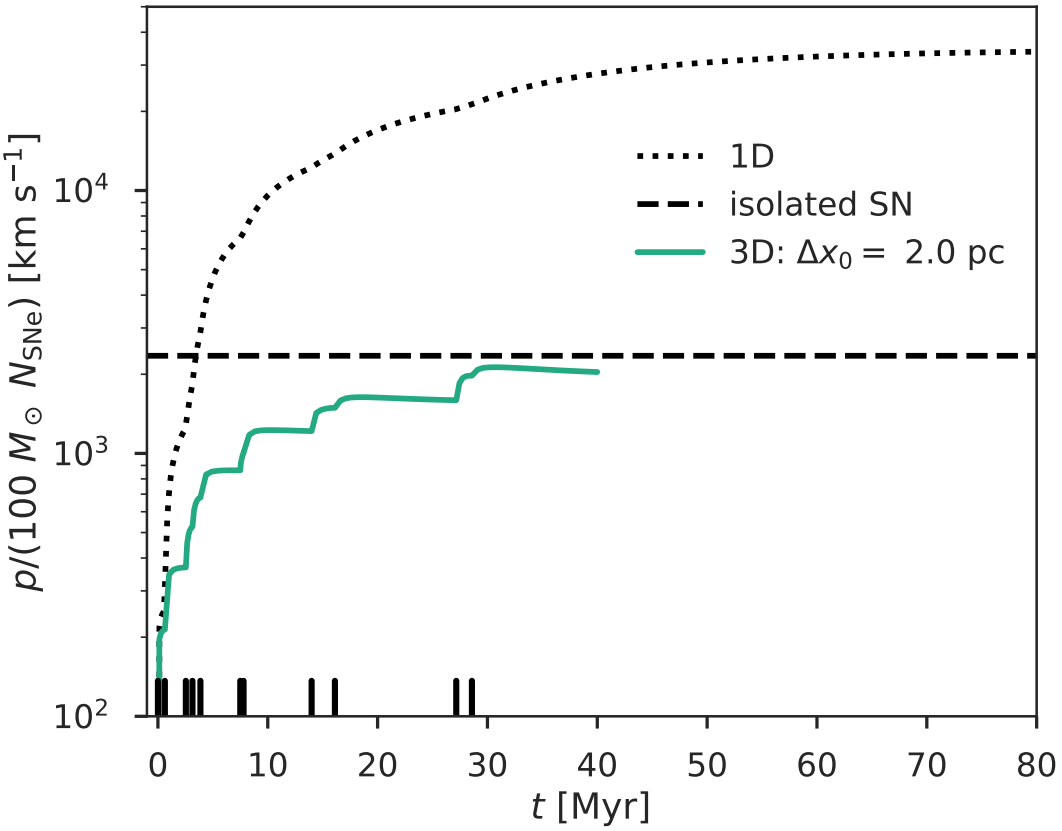

Figure 3.1: Comparison of the momentum evolution of 1D and 3D simulations of the same cluster (simulations `1D_06_HD`, and `3D_20_HD`, respectively). The 'isolated SN' value is estimated using the first SN of the `3D_20_HD` simulation, although it does not vary substantially between any of our 3D simulations.

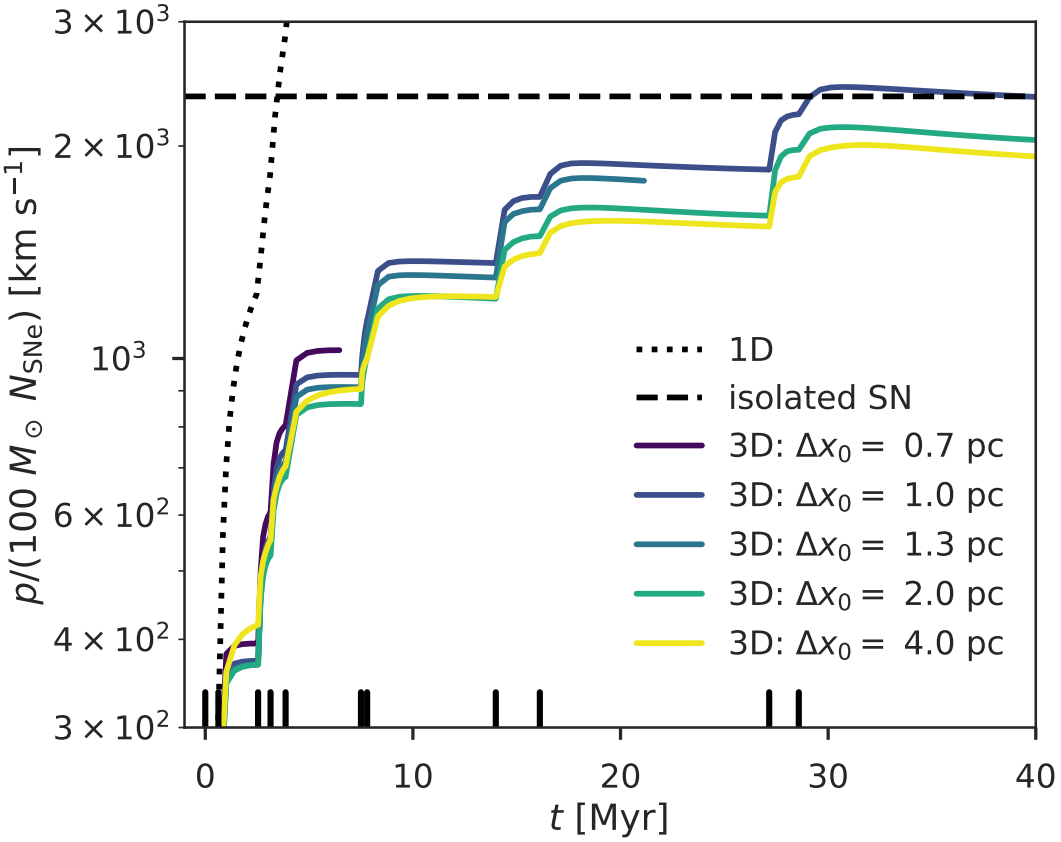

Figure 3.2: Resolution study of our 3D HD simulations.

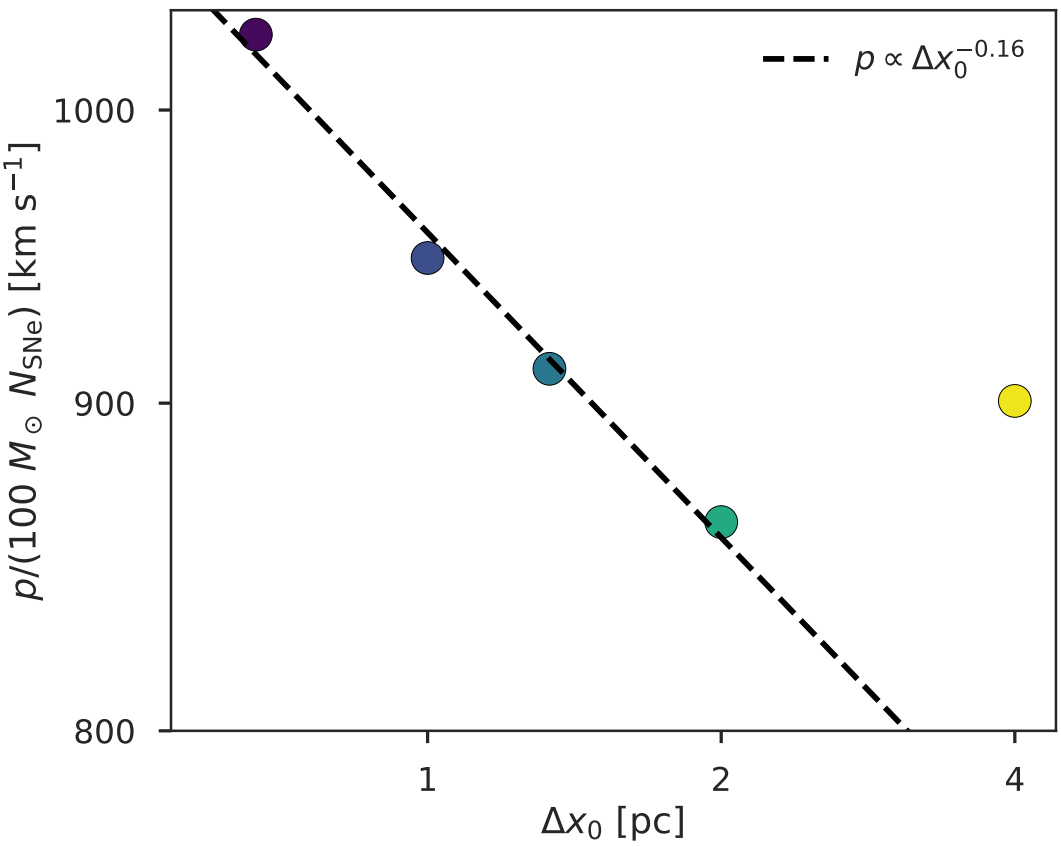

Figure 3.3: Resolution study of our 3D HD simulations at the last time achieved by all simulations. Colours are consistent with the resolution study figures above. The black dashed line shows the best power-law fit to all 3D HD simulations *except* the worst resolution simulation (`3D_40_HD`). Both axes are plotted using log scales.

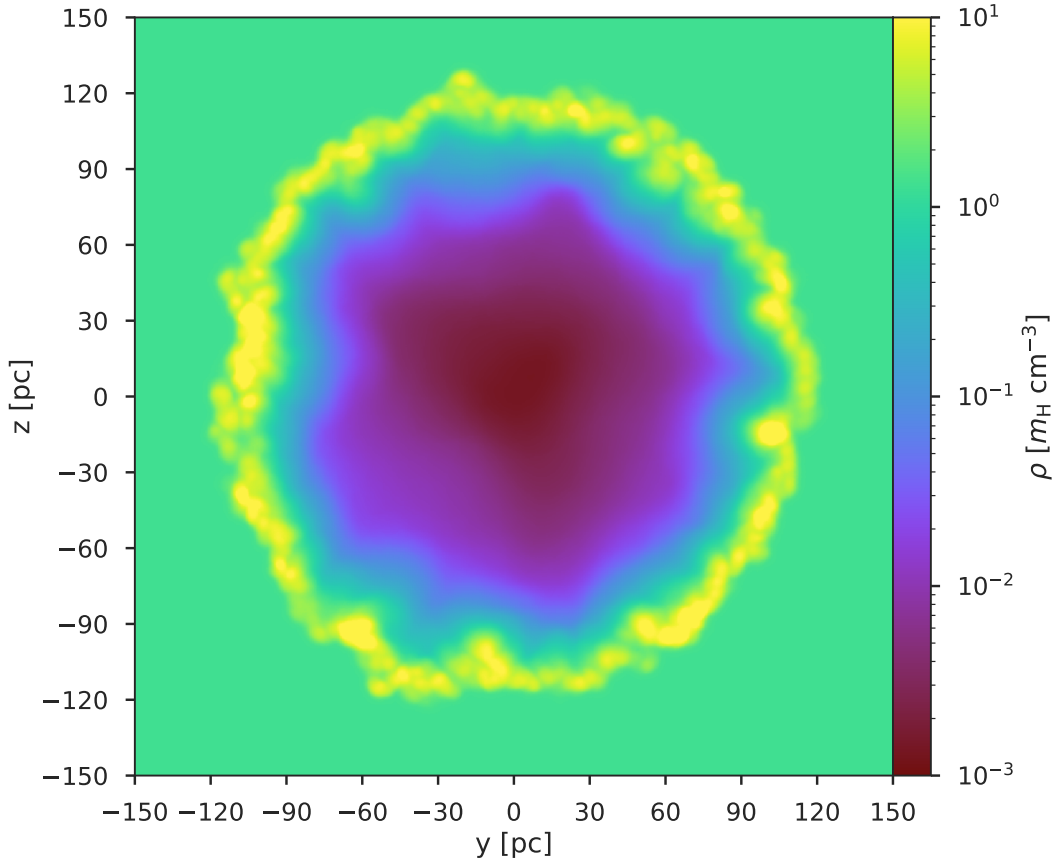

Figure 3.4: Reference density slice of the median resolution completed 3D simulation (`3D_20_HD`) at $t = 7.53$ Myr, approximately 0.03 Myr after the sixth SN.

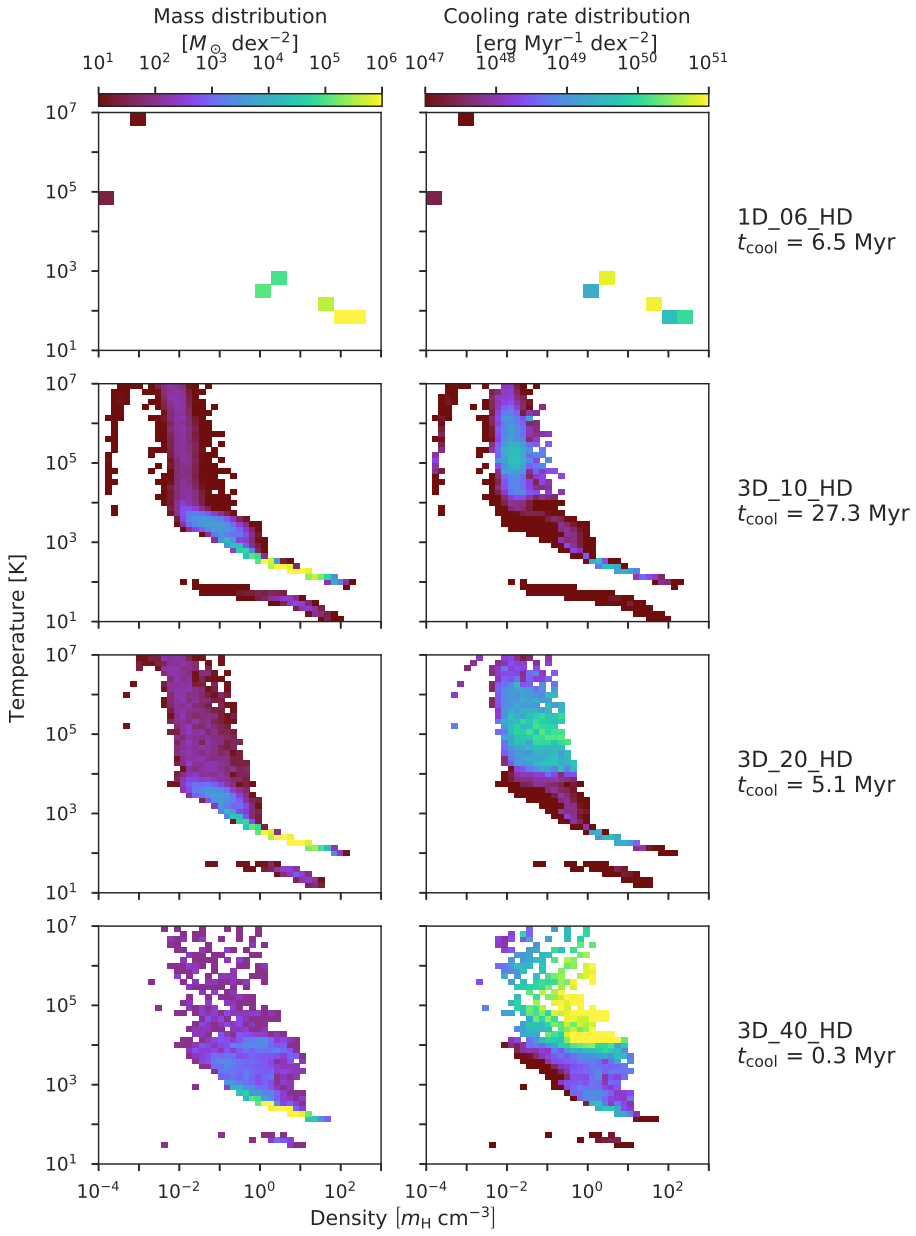

Figure 3.5: Phase diagrams for our completed HD simulations at $t = 7.53$ Myr, about 0.03 Myr after the sixth SN, when all simulations still retain almost all of the energy from the most recent SN. The left column shows the distribution of mass within temperature-density space, and the right column shows the cooling rate distribution within the space. The rows show the non-magnetized simulations with initial resolution worsening from top to bottom. To the right of each row, we give the cooling time of each simulation, $t_{\rm cool} \equiv E_{\rm int}/\dot{E}_{\rm cool}$, for reference.

3.3.1 Hydrodynamic results and convergence study

In Figure 3.1, we show the radial momentum evolution of our median-resolution completed 3D simulation without MHD (`3D_20_HD`), and compare it to our 1D simulation. As can be seen in that figure, we observe a significant difference between the final momenta in our 1D and 3D simulations. While our 1D simulation of clustered SNe shows a large gain in momentum per SN compared to the isolated SN yield,[7] our 3D simulation shows no such gain. That discrepancy needs to be addressed.

This cannot be explained just by the fact that the 3D simulation has a lower initial resolution. In our previous work we tested the resolution dependence in our 1D simulations, and found that even with an initial spatial resolution of 5 pc, we still measured a terminal momentum yield roughly 10 times higher than what we find in our 3D simulation here as long as we ran our code in pseudo-Lagrangian rather than Eulerian mode (Gentry et al. 2017, Figure 14). So the problem is not convergence in our 1D simulation, but we have not yet shown whether our 3D results are converged.

To test for convergence in our 3D simulations, we compare our simulations which differ only in resolution (`3D_07_HD`, `3D_10_HD`, `3D_13_HD`, `3D_20_HD` and `3D_40_HD`); in Figure 3.2, we show the momentum evolution of each simulation. From that figure, we conclude that our 3D simulations do not appear converged, unlike our 1D simulations. The terminal momentum yield is increasing monotonically as we increase the resolution,

[6] The exact velocity threshold is somewhat arbitrary, leading to roughly 10 per cent uncertainty in the affected mass depending on the chosen threshold.

[7] We estimate the isolated SN momentum yield, 2.4×10^5 $M_\odot$ km s^{-1}, using the first SN of our `3D_20_HD` simulation, although all of our 3D simulations would give the same value within a few percent. This is approximately consistent with previous single SN simulations (e.g. Martizzi et al. 2015; Kim & Ostriker 2015).

so our 3D results are converging in the direction of our 1D results, but even at the highest resolution we can afford the momentum yield remains well below the 1D case. Thus we do not know if the 3D results would converge to the same value as the 1D case, even with infinite resolution.

We further illustrate the non-convergence in Figure 3.3, which shows the momentum of the shell at 6.46 Myr, the latest time we are able to reach at all resolutions. As the figure shows, with the exception of the lowest resolution run there is a clear trend of increasing momentum at higher resolution; we discuss possible explanations for the anomalous behaviour of the lowest resolution run in Appendix B.2. A simple power-law fit to the points at resolutions of $\Delta x_0 = 2$ pc or better suggests that the momentum is increasing with resolution as $p \propto \Delta x_0^{-0.16}$. If we naively extrapolate this trend to the peak initial resolution of 0.03 pc achieved in our 1D simulations, the predicted momentum would be a factor of ~ 2 larger than the highest resolution run shown, though this may well be an underestimate since Figure 3.3 shows that the momentum appears to increase with resolution somewhat faster than predicted by a simple power-law fit. In any event, it is clear that, even at a resolution of 0.7 pc, our results are not converged.

To gain additional insight into the resolution-dependence of our results, and the differences between the 1D and 3D runs, we show a density slice through the centre of simulation `3D_20_HD` at $t = 7.53$ Myr shown in Figure 3.4. Clearly in 3D, the interface between the hot bubble interior and the cold shell is not spherically symmetric. These anisotropies are the result of physical instabilities (such as the Vishniac instabilities and the Rayleigh–Taylor instability) amplifying numerical inhomogeneities in the back-

ground ISM (Vishniac 1983, 1994; Mac Low & McCray 1988; Mac Low & Norman 1993; Krause et al. 2013; Fierlinger et al. 2016; Yadav et al. 2017). To see how this might affect the terminal momentum, we turn to density–temperature phase diagrams which are shown in Figure 3.5. These phase diagrams correspond to a time soon after the sixth SN, with a delay long enough to allow the injected energy to spread throughout the bubble but sufficiently short to avoid significant energy losses due to cooling in any simulation. All 3D simulations have about 1.1×10^{51} erg more total energy than the start of the simulations, but `1D_06_HD` retains more energy from previous SNe, and contains about 2.7×10^{51} erg of total energy relative to the simulation start. When we look at the mass-weighted phase diagram for our highest resolution completed simulation (`3D_10_HD`), we see that the mass is dominated by a cold dense shell, with a minority of mass in less-dense warm and hot phases ($> 10^3$ K).[8] Even when we vary the resolution, we only find negligible changes in the fraction of mass in the cold phase; the cold phase ($T < 10^3$ K) contributes 99.2% of the affected mass in every completed simulation of our resolution study (`3D_10_HD`, `3D_20_HD`, and `3D_40_HD`). What does change is the density and temperature distribution of the warm/hot gas ($T > 10^3$ K). As we increase the resolution, the warm/hot gas shifts to lower and lower densities. This effect is very apparent for gas near the peak of the cooling curve (specifically 3×10^4 K $< T < 3 \times 10^5$ K), which has a mass-weighted median density of $\sim 10^{-1}$ $m_{\rm H}$ cm^{-3} in our lowest resolution run, and $\sim 10^{-2}$ $m_{\rm H}$ cm^{-3} in our highest resolution completed run.

[8] We also see a negligible amount of mass at unusually low temperatures, < 100 K. These particles are SN ejecta, which have very high metallicities that have been frozen-in due to the Lagrangian nature of the code.

This has a significant impact on the overall cooling times of the simulations. The right column of Figure 3.5 shows that while the cold, dense phase dominates the mass, the minority of mass in the warm/hot phases dominates the cooling rate. This is important because resolution primarily affects these warm/hot phases, and it affects those phases by shifting them to higher densities at lower resolution, causing each particle to become more efficient at cooling. This results in significantly shorter cooling times: from 27 Myr at the best resolution to 0.3 Myr at the worst resolution, nearly two orders of magnitude difference. This increase primarily occurs in the warm/hot phases; at all resolutions gas warmer than 10^3 K constitutes slightly less than 1% of the total mass, but this mass is responsible for 81% of the cooling at our highest resolution completed run, and $> 99\%$ of the cooling at our lowest resolution.

When we look at the phase diagrams for our 1D simulation, we see significant differences in the distributions of mass and cooling rate, leading to the very different behaviour of the 1D simulation. In particular, the 1D simulation completely lacks material at intermediate densities ($\sim 10^{-2} - 10^{0}$ $m_{\rm H}$ cm^{-3}) due to how well the 1D simulation retains the contact discontinuity. The diffuse bubble-dense shell transition occurs within only a few cells, and the entire dense shell is resolved by just 5-10 cells. In our 3D simulations, these intermediate densities contribute a negligible amount of mass, but are responsible for much of the cooling. Without this intermediate phase material, almost all of the cooling in the 1D simulation occurs in the dense shell. We defer further discussion about the physical nature of the intermediate-temperature gas, and to what extent its properties are determined by physics versus numerics in the

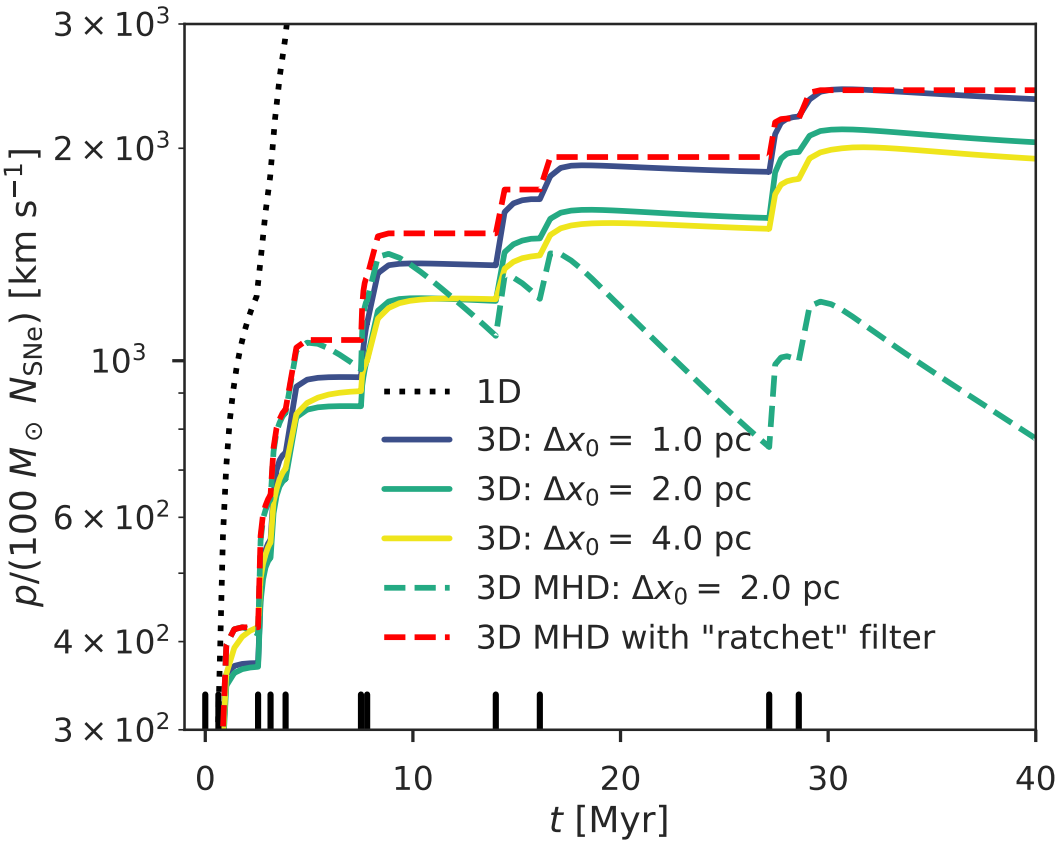

Figure 3.6: Same as Figure 3.2, except now including the momentum evolution of our MHD simulation (`3D_20_MHD`; blue dashed curve), as well its "ratchet"-filtered momentum evolution, $p_{\rm ratchet}$ (red dashed curve), and excluding our incomplete HD runs (`3D_07_HD` and `3D_13_HD`).

various simulations, in section 3.4.

3.3.2 Magnetic fields

In subsection 3.3.1, we showed that our numerical methods and resolution are not sufficient to achieve converged values of final radial momentum and other key parameters due to physical instabilities that develop within the superbubble shell. As

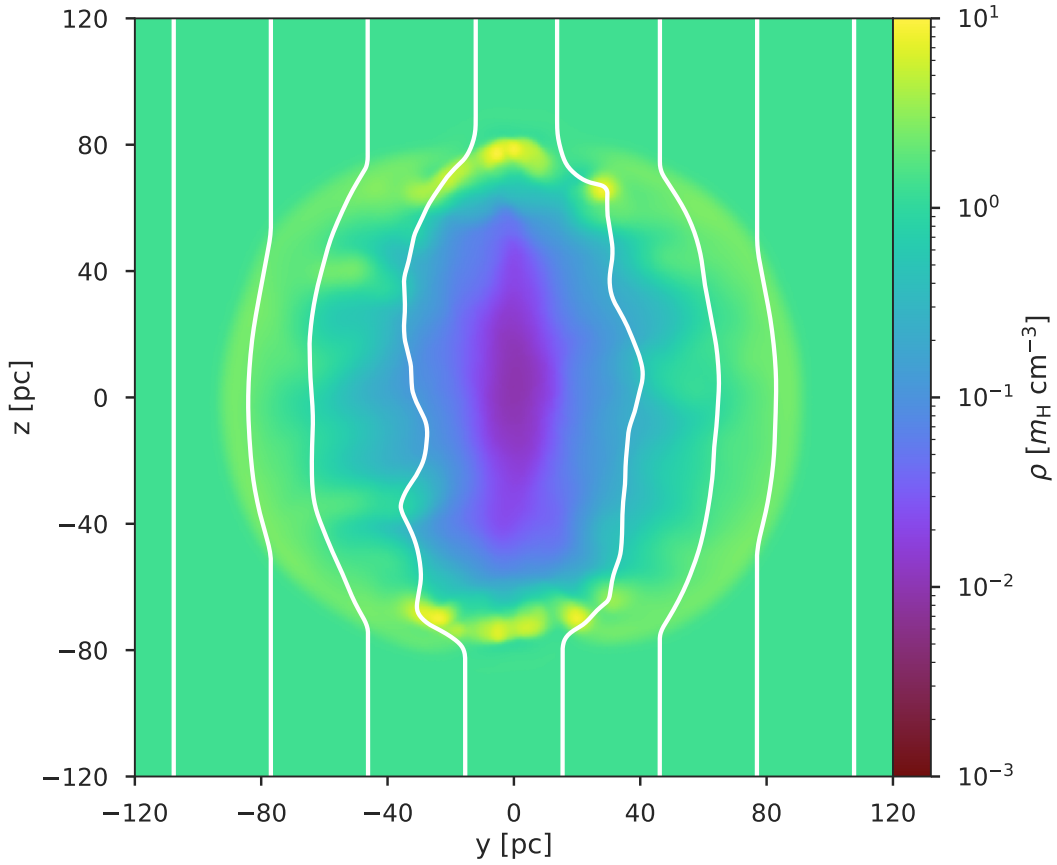

Figure 3.7: Same as Figure 3.4, except now for simulation `3D_20_MHD` with *approximate* magnetic field lines overplotted, and at an earlier time ($t = 2.56$ Myr; approximately 0.02 Myr after the third SN).

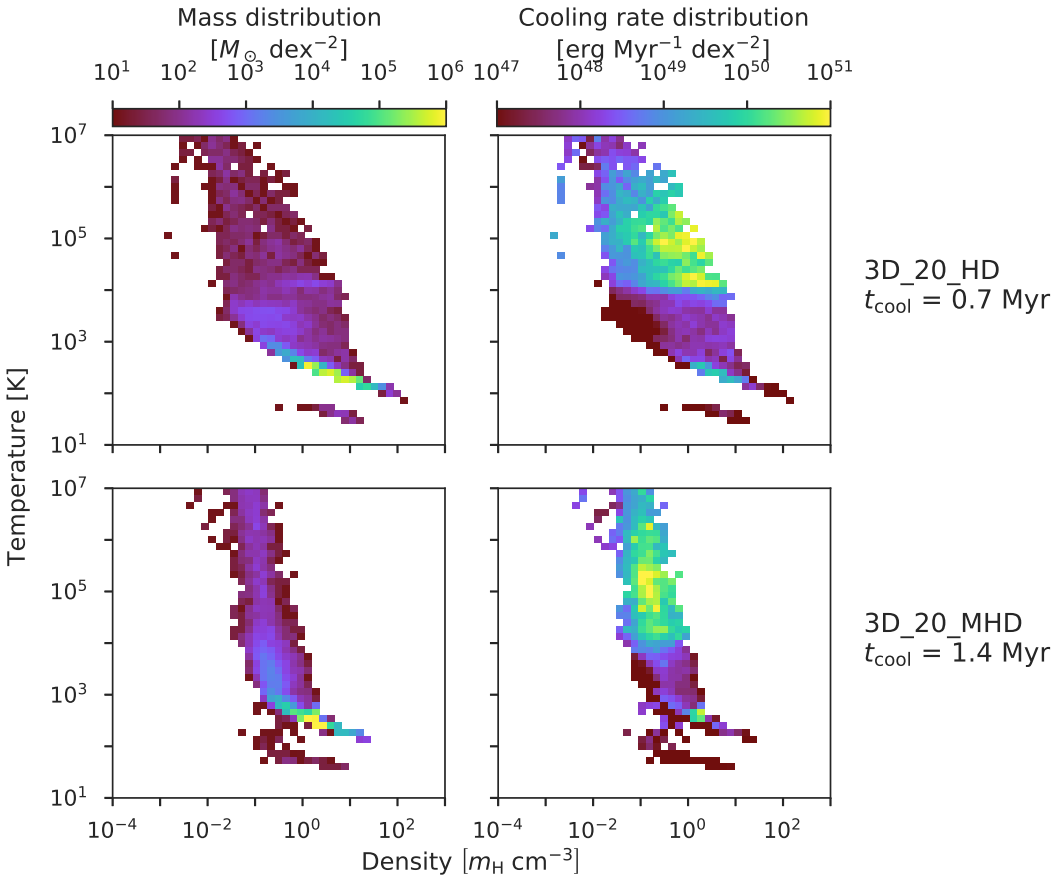

Figure 3.8: Same as Figure 3.5, except at the earlier time shown in Figure 3.7 ($t = 2.56$ Myr), and only showing the magnetized simulation (`3D_20_MHD`) and the corresponding non-magnetized simulation with the same resolution (`3D_20_HD`).

described in section 3.1, we expect that magnetic fields might affect the growth of physical instabilities, so we also run an MHD simulation as described in subsection 3.2.2 to test the impact of magnetic fields on the final momentum. While the more standard method of extracting momentum, $p_{\rm max}$, quoted in Table 3.2 appears to show that the inclusion of magnetic fields significantly decreases the final momentum, in this subsection we show that that method for estimating the asymptotic momentum (finding the maximum momentum following the last SN) is an oversimplification for simulations with magnetic fields. When we better isolate the momentum added by SNe, we find that adding magnetic fields can actually increase the momentum yield at fixed resolution. Indeed, our $\Delta x_0 = 2.0$ pc MHD run produces a larger momentum injection than our $\Delta x_0 = 1.0$ pc HD run.

First, to illustrate why the interpretation of the MHD simulation is more complex, in Figure 3.6 we compare its momentum evolution to those of the non-magnetized simulations. The MHD simulation initially shows an *increased* momentum yield relative to the corresponding simulation without magnetic fields at the same resolution (`3D_20_MHD`), but then the momentum decreases due to magnetic tension forces. The reason for this is obvious if we examine a density slice at an earlier time, [9] as shown in Figure 3.7: the expanding shell bends magnetic field lines outward, and the field lines exert a corresponding magnetic tension that reduces the radial momentum of the

[9] We chose to look at an earlier snapshot, when the magnetization has only perturbed the bubble structure, rather than the later time shown in Figure 3.4, when the magnetization would have caused a strong, non-linear change in the structure which could not be treated as a perturbation. In both cases the magnetic tension is present, but the earlier time makes it more straightforward to compare to the non-magnetized runs.

expanding shell. This effect is so strong that the momentum peaks after just seven SNe; the remaining four SNe clearly add momentum but not enough to overcome the steady decline.

Due to this effect, the quantity $p_{\rm max}$ (the maximum momentum after the last SN) that we have used to characterize the hydrodynamic simulations is somewhat misleading, since our goal is to study the momentum injected by SNe, not the combined effects of SNe and magnetic confinement. To avoid this, we define an alternative quantity $p_{\rm ratchet}$. To compute this quantity we sum any positive changes in radial momentum between snapshots, while ignoring any negative changes. We plot $p_{\rm ratchet}$ in Figure 3.6, and report the final value in Table 3.2. As expected, for the non-magnetic runs $p_{\rm ratchet}$ and $p_{\rm max}$ are essentially the same, and thus examining $p_{\rm ratchet}$ allows us to make an apples-to-apples comparison between the magnetic and non-magnetic results.

This comparison is revealing, in that it shows that our simulation with magnetic fields (`3D_20_MHD`) injects *about 10% more* momentum than the analogous simulations without magnetic fields (`3D_20_HD`). The full explanation for this difference will likely be complicated – for example, the bubble morphology and phase structure are significantly altered at late times relative to the non-magnetized runs – but we can see if our results are at least consistent with the hypothesis that magnetic fields could inhibit the growth of instabilities, leading to less phase mixing and cooling. To test this hypothesis, we compare phase diagrams for the resolution-matched magnetized and non-magnetized runs in Figure 3.8, shown at the same time ($t = 2.56$ Myr) as Figure 3.7. There we see that magnetic fields have an effect similar to that of increasing resolution in

Figure 3.5: both suppress the growth of fluid instabilities, causing the material near the peak of the cooling curve to stay at lower densities where it cools less efficiently. For gas near the peak of the cooling curve (specifically 3×10^4 K $< T < 3 \times 10^5$ K), the median density of the non-magnetized run is 1.7×10^{-1} $m_{\rm H}$ cm^{-3}, while in the magnetized run it is 1.4×10^{-1} $m_{\rm H}$ cm^{-3} – a modest change, but a change in the predicted direction. As a result the overall cooling time is about two times longer in the MHD run. Thus by suppressing the growth of instabilities, the inclusion of magnetic fields results in a longer overall cooling time which should contribute to a higher yield of momentum.

3.4 Analysis

In section 3.3, we showed our broad results, which have three key features: (1) Our 1D Lagrangian simulation finishes with about 10 times more momentum than our 3D simulations, and is converged with respect to resolution. (2) Our 3D HD simulations show a general increase in momentum as resolution improves, but are not converged even at the highest resolutions we can reach [similar to the 1D Eulerian simulations of Gentry et al. (2017), which are not converged even at a resolution of 0.31 pc]. (3) Our MHD simulations show less momentum than the resolution-matched HD simulation when the momentum is estimated directly, but more momentum when our "ratchet" filter is used.

The phase diagrams shown in Figure 3.5 and Figure 3.8 reveal that the changes in momentum budget appear to be associated with changes in the total mass and mean density of gas at temperatures of $\approx 10^5$ K, near the peak of the cooling curve, which

dominates the cooling budget. In this section we seek to understand the physical origin of these differences, with the goal of understanding whether the converged 1D or non-converged 3D results are likely to be closer to reality.

3.4.1 What determines convergence or non-convergence?

As a first step in this analysis, we investigate why our 1D Lagrangian simulations are converged while our 3D simulations are not. A simplistic view of superbubble cooling is one where the diffuse bubble interior contains most of the thermal energy but is radiatively inefficient, while the cold dense shell *is* radiatively efficient but does not have significant amounts of thermal energy to radiate. The cooling rate is then set by how quickly energy can transfer from one phase to the other.

The minimum amount of energy transfer comes from the fact that the hot overpressured bubble is doing work on the shell, transferring thermal energy from the interior into kinetic energy of the shell. As the shell sweeps up and shocks new material, some of this kinetic energy will be transferred into thermal energy within the shell, where it can be easily radiated. To lowest order, we predict this mechanical process would result in the following cooling rate:

$$\dot{E}_{\rm cool,mechanical} = 4\pi R_{\rm shock}^2 V_{\rm shock} \rho_0 \left(\frac{V_{\rm shock}^2}{2} \right). \tag{3.2}$$

This expression assumes a supersonic shock, and that all of the energy that is converted from kinetic to internal energy is immediately radiated away. At each

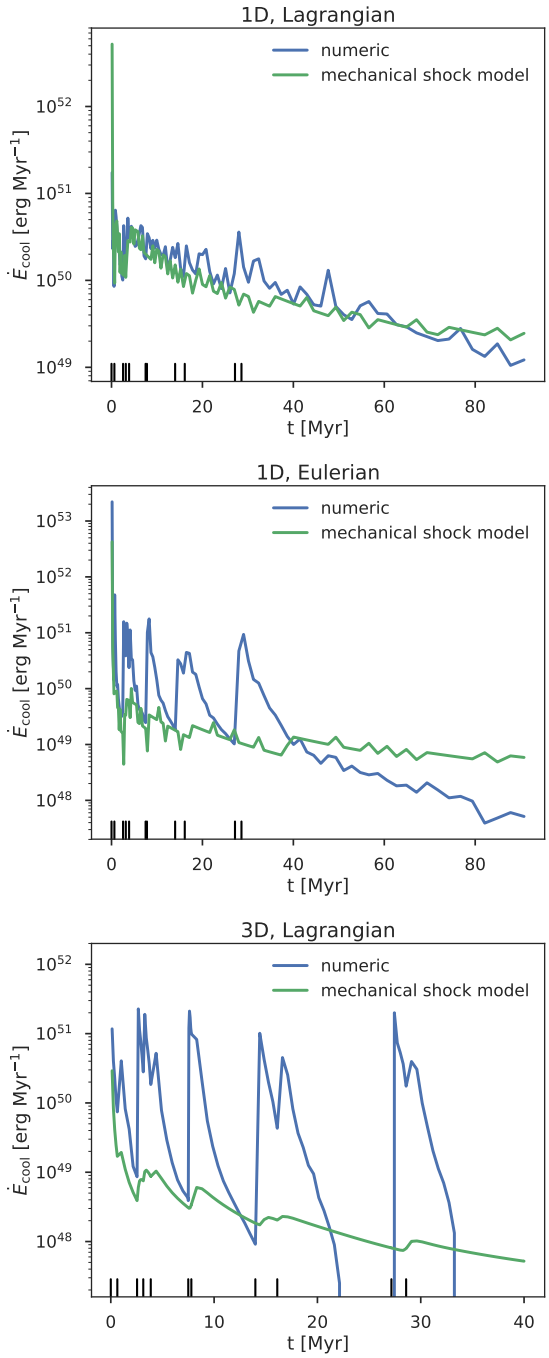

Figure 3.9: Comparison of the numeric cooling rate with the cooling rate predicted by our mechanical shock model, Equation 3.2, for our 1D Lagrangian simulation (`1D_06_HD`; *top*), our 1D Eulerian simulation (`1D_06_HD`, but run with the code in Eulerian mode; *middle*), and our 3D HD simulation with 2 pc initial resolution (`3D_20_HD`; *bottom*).

simulation snapshot, we can compute $R_{\rm shock}$ and $V_{\rm shock}$ [10] and compute the expected cooling rate using Equation 3.2. We can then compare that to the observed cooling rate, calculated by `GRACKLE` for each snapshot.

We begin our predicted-versus-actual cooling rate comparisons with our 1D Lagrangian simulation (`1D_06_HD`) shown in the top panel of Figure 3.9. In that figure we can see that even though our mechanical shock model is simple, it does a generally good job predicting the observed cooling rate. On the other hand, we can repeat this with a simulation that is identical to `1D_06_HD` except it uses an *Eulerian* hydrodynamic solver, leading to the results shown in the middle panel of Figure 3.9. This reveals a very different picture: there are many times when the observed cooling rate is over an order of magnitude greater than our mechanical shock model would predict. And when the observed rate is lower than predicted, it is because the shell has already transitioned from a non-linear shock to a linear sound wave, for which we know Equation 3.2 should not hold. While the mechanical shock model can explain most of the behaviour behind the 1D Lagrangian simulation, in the 1D *Eulerian* simulation the chosen numerical methods lead to much higher cooling rates which must be powered by additional thermal energy being pumped into the shell. When we apply this same approach to one of our 3D Lagrangian simulations (specifically simulation `3D_20_HD`, shown in the bottom panel of Figure 3.9), we find a behaviour similar to the 1D Eulerian simulation and very different from the 1D Lagrangian simulation: the actual cooling rates often far exceed the rate

[10] For our 3D simulations, we estimate $R_{\rm shock}$ as the mean radius of the overdense particles and $V_{\rm shock}$ as the mean radial velocity of the overdense particles. For our 1D simulations, we determine $R_{\rm shock}$ as the outermost overdense cell (see Gentry et al. 2017), and determine $V_{\rm shock}$ by taking the difference of $R_{\rm shock}$ between snapshots.

predicted by our mechanical shock cooling model.

This analysis makes it clear why the 1D Lagrangian simulations are converged: the radiative cooling rate has reached the minimum allowed by the physical situation of an adiabatic fluid doing work on a medium with a short radiative cooling time. Consequently, increasing the resolution cannot further reduce the rate of radiative loss; it is already as low as physically allowed. If we run the same problem in 1D Eulerian mode, or in 3D but at much lower resolution, the cooling rate is far in excess of the minimum. Cooling is powered not primarily by adiabatic compression of the cold gas followed by radiative loss, but by direct transfer of thermal energy between the hot and cold phases without doing any mechanical work. The rate of transfer is clearly resolution-dependent, which explains why the 1D Eulerian and 3D simulations are not converged.

3.4.2 Conduction and numerical mixing across the interface

Since the key difference between the converged 1D Lagrangian simulations and the unconverged 3D simulations is the relative importance of energy transfer by mechanical work versus other mechanisms, we next investigate the expected rate of non-mechanical energy transfer in reality, and how that compares to the rate in our simulations.

In a bistable radiative medium such as the one we are simulating, conductive transfer occurs across an interface whose characteristic width, known as the Field length,

is given by (Begelman & McKee 1990)

$$\lambda_F = \left(\frac{\kappa T}{n^2 \Lambda} \right)^{1/2}, \tag{3.3}$$

where κ is the thermal conductivity and Λ is the cooling function. The conductive heat flux is $F \sim \kappa T / \lambda_F$, so the total rate at which energy conducts across an interface of area A and is lost to radiation is

$$\dot{E}_{\rm cond} \sim \dot{E}_{\rm cool} \sim A \frac{\kappa T}{\lambda_F}. \tag{3.4}$$

Figure 3.5 shows that, for simulation `3D_40_HD` at time $t = 7.53$ Myr, typical values for the gas that dominates the cooling are $n = 1$ cm^{-3} and $T = 4 \times 10^5$ K. Using Begelman & McKee (1990)'s expression for thermal conductivity, assuming no suppression by magnetic fields and no saturation, together with the approximate cooling function Λ from Koyama & Inutsuka (2002) (their equation 4, which we use for simplicity, rather than performing the full `GRACKLE` calculation), we find $\lambda_F \approx 0.003$ pc. Using the lower density $n \approx 10^{-1}$ cm^{-3} found in our highest resolution completed 3D simulation (`3D_10_HD`) would increase this to $\lambda_F \approx 0.03$ pc. By contrast, our best 3D simulation resolution is an order of magnitude larger; only our 1D Lagrangian simulation approaches this resolution. Thus the true physical width of the interface is far from resolved in any of our 3D simulations.

In our simulations, as opposed to reality, the width of the interface is set by

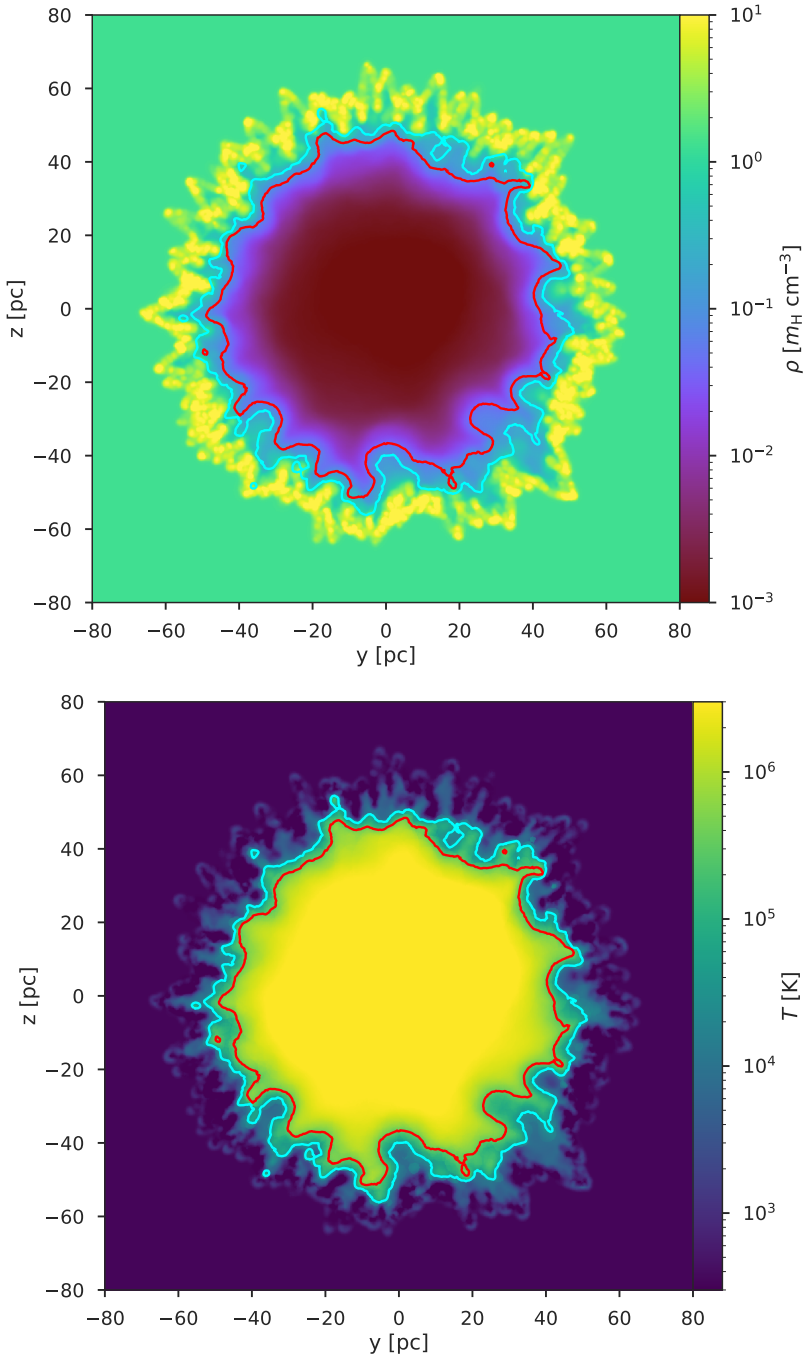

Figure 3.10: Density (top) and temperature (bottom) slices of the highest resolution 3D simulation (`3D_07_HD`) at $t = 1.01$ Myr, approximately 0.5 Myr after the second SN. The lighter cyan and darker red contours correspond to temperatures 3×10^4 K and 3×10^5 K, respectively, which are roughly the bounds of the peak of the cooling curve.

Table 3.3: Interface properties at $t = 1.01$ Myr (0.5 Myr after the second SN). Here $N_{\rm paticles}$ and m are the number of particles and total mass in the interface (defined as the set of particles with temperatures in the range $3 \times 10^4 - 3 \times 10^5$ K, near the peak of the cooling curve), $r_{\rm median}$ is the median radius of interface particles, $h_{\rm median}$ is the median scale length of interface particles, $\Delta r_{\rm IQR}$ is the interquartile range of interface particle radii.

Name	$N_{\rm particles}$	m ($M_\odot$)	$r_{\rm median}$ (pc)	$h_{\rm median}$ (pc)	$\Delta r_{\rm IQR}$ (pc)	$\Delta r_{\rm IQR}/h_{\rm median}$
`3D_40_HD`	16	33.6	37.0	15.4	8.5	0.6
`3D_20_HD`	159	41.8	39.1	9.8	8.3	0.8
`3D_13_HD`	485	37.8	43.3	6.6	6.4	1.0
`3D_10_HD`	1303	42.8	45.4	4.9	5.1	1.0
`3D_07_HD`	4436	43.2	47.4	3.3	4.3	1.3

numerical resolution. We illustrate this point in Figure 3.10, which shows temperature and density slices from our highest resolution simulation (`3D_07_HD`) shortly after the second SN. We summarize the physical properties of the hot–cold interface, defined as material between 3×10^4 K and 3×10^5 K, roughly corresponding to the peak of the cooling curve, in Table 3.3; we include `3D_40_HD` for completeness, but warn that, at this early time, its interface is poorly sampled by only 16 particles, and thus the results for it are not particularly meaningful. The main conclusion to make from Figure 3.10 and Table 3.3 is that the physical width of the interface region is of order a particle smoothing length, so the width of the interface is determined by numerics rather than physics.

What is the impact of this underresolution on the rate of radiative loss? Though we do not include explicit conduction (nor would it matter if we did, since our failure to resolve λ_F would lead us to greatly underestimate the true conduction rate), any finite-resolution numerical method necessarily has some conduction-like dis-

sipation at the resolution scale. It is convenient to characterize this dissipation in terms of the effective Péclet number of the method, which is related to the effective thermal conductivity of the numerical scheme $\kappa_{\rm num}$ by

$$\mathrm{Pe} \sim \frac{Lvnk_B}{\kappa_{\rm num}}, \tag{3.5}$$

where L and v are the characteristic length and velocity scales. The exact value of the Péclet number will depend on the numerical method. In Eulerian methods where effective conduction is due to fluids mixing at the resolution scale Δx, we expect to have $\mathrm{Pe} \sim 1$ for $L \sim \Delta x$. In a Lagrangian method Pe will be substantially larger, since mixing is suppressed. Ignoring this complication, if we replace κ with $\kappa_{\rm num}$ and λ_F with the interface width λ_I in Equation 3.4, the numerical conductive transport and cooling rates are then

$$\dot{E}_{\rm cond,num} \sim \mathrm{Pe}^{-1} A n k_B T v \left(\Delta x / \lambda_I\right). \tag{3.6}$$

Using the same values of n and T given above, the approximate velocity $v \approx 40$ km s^{-1} for the shock at the time shown in Figure 3.10, and our empirical finding that $\Delta x/\lambda_I \sim 1$, we find that for a method with $\mathrm{Pe} = 1$ at the resolution scale, $\dot{E}_{\rm cond,num} \approx 10\dot{E}_{\rm cond}$, i.e., underresolving the interface causes us to overestimate the rate of energy loss by a factor of ~ 10. This overcooling problem is substantially reduced for the 1D Lagrangian simulation, since compared to the other simulations it has both a smaller value of $\Delta x/\lambda_I$ (due to its high resolution) and a larger value of Pe (due to its Lagrangian method). Conversely, this analysis strongly suggests that the ultimate reason for non-convergence

in our 3D simulations is that their rates of energy loss are dominated by resolution-dependent artificial conduction. Thus the terminal momentum in these simulations must be regarded as a lower limit on the true value.

3.4.3 The role of 3D instabilities

Before accepting the conclusion that artificial conduction is the culprit for our non-convergence, however, we should examine an alternative hypothesis. The total conductive transport rate (Equation 3.4) depends not just on the conductive flux, but on the area of the interface. Examining Figure 3.10, it is clear that the area of the interface is affected by 3D instabilities that are not properly captured in the 1D simulations. Could the non-convergence in 3D be a result of the area not being converged, rather than the conductive flux not being converged? This hypothesis might at first seem plausible, because many instabilities (such as the Rayleigh–Taylor, Richtmyer–Meshkov and Vishniac) initially grow fastest at the smallest scales (e.g., Taylor 1950; Richtmyer 1960; Vishniac 1983; Michaut et al. 2012). If the area of the interface is determined by the amount of time that it takes perturbations to grow from the resolution scale that might explain why our highest resolution simulation has the lowest cooling rate: because it had the smallest perturbations to start, and it has the smallest interface area later on, and thus the smallest rate of conduction.

However, we can ultimately rule out this hypothesis for two reasons. First, if the rate of mixing and radiative loss were set by processes developing from grid-scale perturbations, then changing the initial perturbation strength and scaling should have

a noticeable impact on the cooling rate. However, as shown in Appendix B.1, our non-convergence is quite robust to the details of the grid-scale perturbations. The results are not any more converged when we impose perturbations whose power spectral density is independent of resolution over all resolved scales, and increasing the initial perturbation strength by a factor of > 25 has negligible effects on the outcome. Second, once they are strongly non-linear, interface instabilities are typically dominated by larger rather than smaller modes. Examining Figure 3.10, it is clear that even just after the second supernova we already have strongly non-linear perturbations in the shell, with each spike well resolved by many particles. If linear growth of instabilities from the grid scale were the source of our non-convergence, we would expect to see the greatest resolution dependence at early times, when the perturbations are smallest, and convergence between the runs at later time, when the instabilities reach non-linear saturation. Examining Figure 3.2, however, shows exactly the opposite pattern: resolution matters more at later times than at earlier ones.

However, simply because we can rule out the hypothesis that the non-convergence of the 3D simulations is a result of our failure to capture the growth of 3D instabilities, it does not follow that the instabilities are not important. Figure 3.10 clearly shows that the area of the interface in 3D is clearly larger than $4\pi R_{\rm shock}^2$, and thus the rate of conduction across the interface should be higher than it is in our 1D simulations. Thus while our 3D simulations represent a lower limit on the terminal momentum, we must regard the 1D simulations as representing an upper limit, since the interface in 1D has the smallest possible area.

3.5 Conclusions

In this paper, we revisit the question of whether clustering of SNe leads to significant differences in the amount of momentum and kinetic energy that supernova remnants deliver to the ISM. This question is strongly debated in the literature, with published results offering a menu of answers that range from a relatively modest increase or decrease (Kim & Ostriker 2015; Walch & Naab 2015; Kim et al. 2017) to a substantial increase (Keller et al. 2014; Gentry et al. 2017). We investigate whether this discrepancy in results is due to numerical or physical effects, and to what extent it might depend on whether the flow is modelled as magnetized or non-magnetized.

Our results offer some encouragement and also some unhappy news regarding the prospects for treating supernova feedback in galactic and cosmological simulations. The encouraging aspect of our findings is that we have identified the likely cause of the discrepancy between the published results. We find that the key physical mechanism driving the differences between our runs, and almost certainly between other published results, is the rate of mixing across the contact discontinuity between the hot interior of a superbubble and the cool gas in the shell around it. Our 1D Lagrangian results (Gentry et al. 2017) maintain the contact discontinuity nearly perfectly, and give it the smallest possible area, and this explains why they produce large gains in terminal momentum per supernova due to clustering. However, these results likely represent an upper limit on the momentum gain, because they do not properly capture instabilities that increase the area of the contact discontinuity and thus encourage mixing across it.

In 3D, both physical instabilities and numerical mixing produce intermediate temperature gas that radiates rapidly and saps the energy of the superbubble, lowering the terminal momentum. Due to this mixing, we are unable to obtain a converged result for the terminal momentum; we find that the terminal momentum continues to increase with resolution even at the highest resolution that we complete (1 pc initial linear resolution, 0.03 $M_\odot$ mass resolution). The cause of this effect is clear: as we increase the resolution, we find that the mean density and total mass of gas near the peak of the cooling curve continuously decreases (indicating a decrease in mixing), and this typically leads to a decrease in the amount of energy lost to radiation. Consequently, we are forced to conclude that even at our highest resolution in 3D, the mixing and energy transfer rate across the contact discontinuity is dominated by numerical mixing. As a result, our estimate of the momentum per supernova is only a lower limit.

Our tests with magnetic fields reinforce this conclusion. We find that magnetic fields suppress the growth rate of physical instabilities. This leads an magnetized simulation to inject more momentum per supernova than a non-magnetized simulation, but both still inject far less than the no-mixing case. This is consistent with the conclusion that physical mixing is present in our simulations but numerical mixing is the dominant source. In the real ISM, magnetic fields are doubtless present, so this effect should not be neglected, especially in simulations that are not dominated by numerical mixing.

Our findings cloud the prospects for obtaining a good first-principles estimate of the true supernova momentum yield in a homogeneous ISM. Our peak spatial resolution is higher than that achieved in previous 3D simulations, and we used Lagrangian

methods rather than Eulerian methods. We note our choice of Lagrangian rather than Eulerian methods was based on a 1D rather than 3D experiment, and that our results are likely affected by multiple definitions of resolution, such as the mass resolution of ejecta, and not just the peak spatial resolution. None the less, we are unable to reach convergence. We are forced to conclude that the true momentum yield from clustered SNe in a homogeneous ISM remains substantially uncertain. At this point we can only bound it between $\approx 2.4 \times 10^5\ M_\odot$ km s^{-1} per SN (our non-converged 3D result) and $\approx 3.4 \times 10^6\ M_\odot$ km s^{-1} per SN (our converged but 1D result). The 1D result certainly produces too much momentum, since 3D instabilities must enhance the conduction rate at least somewhat by increasing the area of the hotcold interface. Similarly, our 3D results produce too little momentum, since our 3D results remain dominated by numerical conduction even at the highest attainable resolution; we do not know how close a converged 3D result would lie to the 1D, no-mixing limit.

We conclude by noting that we have not thus far investigated the effects of using a realistically turbulent, multiphase ISM. The presence of density inhomogeneities could well lead to higher rates of mixing across the contact discontinuity, and thus a reduction in the supernova momentum yield. However, we urge caution in interpreting the results of any investigations of these phenomena, since we have shown that even state-of-the-art simulation methods operating at the highest affordable resolutions cannot reach convergence in what should be substantially simpler problems. It is conceivable that the more complex density field of a realistic ISM might make it easier to reach convergence, but such a hope would need to be demonstrated rigorously using convergence studies in

multiple numerical methods.

Chapter 4

Momentum Injection by Clustered Supernovae: Testing Subgrid Feedback Prescriptions

This is meant to be submitted shortly, but is not ready quite yet, so we do not have an arXiv number towards which to point you.

Using a 1D Lagrangian code specifically designed to assess the impact of multiple, time-resolved supernovae (SNe) from a single star cluster on the surrounding medium, we test three commonly used feedback recipes: delayed cooling (e.g., used in the `GASOLINE-2` code), momentum-energy injection (a resolution-dependent transition between momentum-dominated feedback and energy-dominated feedback which is, for example, used in the `FIRE-2` code), and simultaneous energy injection (e.g., used in the `EAGLE` simulations). Our work provides an intermediary test for these recipes, since we

analyse a setting that is more complex than the simplified scenarios many were designed for, but one more controlled than a full galactic simulation. In particular, we test how well these models can reproduce the enhanced momentum efficiency seen for an 11 SNe cluster at high resolution (0.6 pc; a factor of 12 enhancement relative to the isolated SN case) when these subgrid models are run at low resolution (20 pc). We find that the delayed cooling model can be finely tuned to a given stellar cluster (resulting in at least 9 times the momentum efficiency of the fiducial isolated SN value), but that those tuned parameters may require a yet-to-be-calibrated dependence on cluster mass. The momentum-energy model requires no tuning, while still producing good results (a factor of 5 boost in efficiency). Injecting the energy from all SNe simultaneously does little to prevent over-cooling and greatly under-produces the momentum deposited by clustered SNe (resulting in a factor of 3 *decrease* in momentum efficiency on average).

4.1 Introduction

Energy and momentum injection from supernovae (SNe) are thought to be one of the key ingredients regulating galaxy formation and the thermodynamics of the interstellar and circumgalactic media. Without feedback processes that reheat and redistribute gas in galaxies, simulated galaxies are found to be too cold and too centrally compact (e.g., Katz & Gunn 1991). Despite its importance, however, a proper treatment of feedback in 3D hydrodynamics simulations remains elusive; at high resolutions ($\lesssim 7$ pc for $n = 1$ cm^{-3} and 1 SN; Kim & Ostriker 2015), we can simply inject 10^{51} erg of energy

and get reasonable results, but at lower resolutions this direct injection approach yields incorrect asymptotic properties for the SN remnant (SNR), and consequently erroneous galaxy properties (Smith et al. 2018).

The fundamental cause of this is a phenomenon known as over-cooling (Katz 1992): the rate of radiative cooling in a hot astrophysical plasma is a highly non-linear function of density and temperature, and when the energy deposited by an exploding SN is spread over too large a volume or mass as a result of low resolution, this non-linearity leads to a dramatic overestimate of the cooling rate. Since the full, complex physics that describes the interaction of SN ejecta with the interstellar medium (ISM) cannot be captured at currently-realistic resolutions in large galactic and cosmological simulations, simulators have adopted a variety of simpler subgrid models that we hope can capture the "main" behaviours of a SNR. This approach cannot hope to capture every dependence on the environment or context, so we typically start with a simple model, and then only add new dependencies as they are shown to be necessary.

One potentially strong dependence is on the clustering of core collapse SNe. Depending on the frequency of SNe, it is possible that one or more SNe can occur within the bubble of a previous SNR, forming a superbubble. Some models have shown that feedback from superbubbles can be much more efficient than isolated SNe at ejecting mass and adding momentum to the ISM (Roy et al. 2013; Sharma et al. 2014; Keller et al. 2014; Gentry et al. 2017; see Dekel et al. 2019 for a general discussion of regimes of SN clustering). The amount of enhancement from clustering appears to depend sensitively on the level of mixing across the interface between the hot SNR interior and the cool

shell around it (Gentry et al. 2019; El-Badry et al. 2019) and the specific superbubble regime being studied. The turbulent mixing rate is uncertain, and likely depends on both the pre-existing clumpiness of the ISM and on the presence of magnetic fields, which suppress instabilities such as the Rayleigh-Taylor instability that promote mixing (Gentry et al. 2019). At present we lack detailed magnetohydrodynamic simulations of SNRs including conduction with enough resolution to quantify the mixing rate, and thus the amount of boosting due to clustering is uncertain (see the review by Krumholz & Federrath 2019 for further discussion).

Since many traditional subgrid models for SNe do not directly account for clustering, it is important to investigate whether this could constitute a significant error in how we model feedback in galactic simulations. This is a question with at least three parts: how well does a particular subgrid model approximate a given superbubble, how well does it approximate each of the various regimes of superbubbles, and what is the relative occurrence frequency for each superbubble regime? For simplicity, in this paper we focus on the first question: how well is the behaviour of a particular 11 SNe bubble captured by existing subgrid models at low resolution? We focus on this particular case because the simulations of Gentry et al. (2017) show that it has near maximal effects in terms of boosting the terminal momentum of the SNR, and thus can be used to set an upper limit on the potential error that subgrid models make by ignoring the effects of clustering.

In this paper, we use 1D spherically symmetric hydrodynamic simulations to study what happens when a sample of common, traditional subgrid models are applied

to a series of clustered SNe. We use these models to simulate our 11 SNe case at both low resolution ($\Delta r_0 = 20$ pc, typical of isolated galaxy or the best-resolved zoom-in cosmological simulations) and high resolution ($\Delta r_0 = 0.6$ pc, sufficient to obtain a converged terminal momentum without an explicit subgrid model). While it would be ideal for each subgrid model to perfectly match the final momentum of a converged high-resolution 3D simulation, such a simulation is unfortunately not available for clustered SNe. In 1D, Gentry et al. (2017) show that the converged result for the total momentum enhancement due to clustering is a factor of ≈ 10 increase over simply adding up the momentum of single SNe (though this can be reduced if one includes an explicit model for turbulent conduction – see El-Badry et al. 2019), but Gentry et al. (2019) show that 3D simulations remain unconverged even at ≈ 1 pc resolution. However, the 3D result does provide a lower limit to the amount by which clustering enhances the terminal momenta of SNRs: by extrapolating the Field length, they show that the expanded enhancement in terminal momentum per SN is at least a factor of $2 - 3$. This provides our primary criterion for the success of a subgrid model: when applied to a low resolution 11 SNe cluster, does it result in a momentum efficiency *at least* 2 times greater than the fiducial isolated SN momentum efficiency, thus reproducing the lower limit implied by the 3D simulations?

Finally, we needed to choose specific subgrid models to test. For this work we chose a sample of 3 commonly-used approaches: "delayed cooling" (specifically mimicking the "blastwave" feedback method available in the `GASOLINE-2` code; Stinson et al. 2006; Wadsley et al. 2017), "momentum-energy injection" (specifically mimicking the

implementation used by the FIRE-2 simulations; Hopkins et al. 2018b) and finally "simultaneous energy injection" (specifically mimicking the implementation used by the EAGLE simulations; Dalla Vecchia & Schaye 2012; Schaye et al. 2015; Crain et al. 2015). This is not meant to be an exhaustive list, but covers some of the most common approaches which could be tested by our code[1].

In section 4.2 we introduce the ideas behind each of these subgrid models at a high level. In section 4.3 we discuss the numerical methods used in our 1D simulations, and then the implementation details needed for each subgrid model. In section 4.4 we show the results of each simulation and briefly comment on the differences. We put these results in context in section 4.5 and then conclude in section 4.6.

4.2 Overview of Feedback Models

Before getting into the implementation of each subgrid model, we give a conceptual overview of the physical motivation of each. In section 4.3 we cover these models again at a lower level, specifying the implementation details of each step.

4.2.1 Delayed cooling feedback

Since the underlying problem is that at low resolutions SNRs cool too quickly, before they are able to accelerate enough mass, one early approach was to simply "turn

[1]For example, we cannot meaningfully test the subgrid model used in the IllustrisTNG simulations in a homogeneous ISM, as that model prescribes "wind" particles that travel from regions of high density and only inject their energy into the ISM in regions of low density (Vogelsberger et al. 2013; Pillepich et al. 2018). Our homogeneous ISM has no regions of low density, so these wind particles would never re-couple with the ISM.

off" radiative cooling, allowing the SNR to develop until the correct radiative cooling timescale, and then turn cooling back on (at which point most of the energy is likely rapidly radiated away). Since this delays the cooling, we will refer to this as the "delayed cooling" model.

To focus on a specific example, we will mimic the model proposed by Stinson et al. (2006) (although this idea dates back at least to Gerritsen 1997). This approach is based on the well-studied stages of SNR evolution (allowing it incorporate effects like a density dependence to the cooling shutoff duration; Chevalier 1974; McKee & Ostriker 1977), but it still includes a crucial free parameter: how much energy should be injected (which we will refer to as $E_{\rm blast}$ although they call it $E_{\rm SN}$). Even for a high resolution simulation (which should not experience *over*-cooling), there will always be some cooling, leading to a time-averaged energy less than 10^{51} erg. Thus if one injects 10^{51} erg of energy, and delays cooling until late times, the resulting explosions will be too powerful because the gas retains too much energy during the crucial, early stages. Thus in practice the blast energy within this delayed cooling model is treated as a free parameter, less than the nominal 10^{51} erg. Stinson et al. (2006) ultimately make a recommendation of 10^{50} erg, which we adopt for our tests here.

4.2.2 Momentum-energy feedback

A second class a models takes a different tack: rather than focusing on intermediary steps (like mimicking the instantaneous cooling rate), we could instead focus on directly prescribing the key final results. In particular, radial momentum is one of those

key quantities. The radial momentum of a single SN expanding into a cold medium asymptotes to a value (Cioffi et al. 1988) that is a function of the SN energy and the density and metallicity of the ambient medium, and this momentum is expected to be a key driver of small scale limits on star formation (e.g., by driving turbulence that maintains the scale height of a galactic disc; Ostriker & Shetty 2011; Faucher-Giguère et al. 2013), as well as large scale limits on star formation (e.g. galactic winds removing gas from galaxies; Murray et al. 2005; Hopkins et al. 2012; Dekel & Krumholz 2013; Creasey et al. 2013; Thompson & Krumholz 2016).

The specific model we will mimic is from the `FIRE-2` methods (described in greatest depth in Hopkins et al. 2018a and with more context in Hopkins et al. 2018b), although numerous earlier authors adopted very similar approaches (e.g., Kim et al. 2011; Kimm & Cen 2014; Kimm et al. 2015; Simpson et al. 2015; Goldbaum et al. 2016). This is effectively a hybrid method: at high resolution it is primarily a direct energy injection method (which requires very few assumptions), but at low resolution it transitions to a momentum injection method when needed.

The `FIRE-2` approach achieves this by first directly adding the ejecta to a neighbourhood around the SN location, calculating if that region has sufficient resolution to resolve the SNR evolution, and if not, adjusting the injected quantities (i.e. increasing the injected momentum and decreasing the injected energy), to approximate the late-time state of the SNR. When the expected cooling radius of a SN is unresolved, which is the case in almost all cosmological or galaxy-scale simulations focusing on spiral

galaxies,[2] this recipe essentially reduces to injecting a fixed amount of radial momentum per SN.

4.2.3 Simultaneous energy injection

The final class of models we will consider are those that explicitly harness the clustering of SNe by releasing all of the cluster's SN energy at one time, making it more likely that the gas is heated beyond the peak of the cooling curve. For a specific example, we will mimic the feedback model proposed by Dalla Vecchia & Schaye (2012) and used in the `EAGLE` simulations (Schaye et al. 2015; Crain et al. 2015). In their method, Dalla Vecchia & Schaye go one step beyond just injecting the energy simultaneously and hoping it heats the affected resolution elements above the peak of the cooling curve; they add a stochastic element that *guarantees* that any elements receiving energy are heated beyond the peak of the cooling curve. (Dalla Vecchia & Schaye note that their model can be extended to stochastically release energy across *multiple* timesteps rather than simultaneously, but since their work focuses on the simultaneous case, we classify it as a "simultaneous" model.)

The key parameter in the `EAGLE` approach is $\Delta\epsilon$, the increase in specific thermal energy by a cell receiving SN energy (mostly equivalent to a desired change in temperature, ΔT). By making this a prescribed parameter, they can ensure that any resolution element that receives energy is sufficiently hot ($\gtrsim 10^{7.5}$ K) that it is radia-

[2] Though not in dwarfs, where lower densities yield larger cooling radii, and which can be simulated at much higher resolution due to their smaller overall size – e.g., Forbes et al. (2016); Wheeler et al. (2018).

tively inefficient. This comes at a cost; in order to inject the right amount of energy (on average), this sets a limit on the amount of mass (on average) in the resolution elements that can receive the energy. This is especially difficult at low resolution (for fixed cluster mass and SN energy), where we cannot easily partition the injection kernel into the correct amount of mass that does and does not receive energy. To solve this, Dalla Vecchia & Schaye (2012) prescribe a stochastic approach.

Within an injection kernel, each cell has a probability of receiving energy, p; this probability depends on the mass within the kernel and the total blast energy, but does not vary between cells. For the ith cell within the kernel, a thermal energy $m_i \Delta\epsilon$ is added with probability p. By choosing a sufficiently high $\Delta\epsilon$ we can ensure that *on average* we inject the correct energy, even if a particular cluster injects more or less than the desired amount. They recommend choosing a value of $\Delta\epsilon$ corresponding to $\Delta T = 10^{7.5}$ K, but note that for fixed total SN energy, higher resolutions will require higher values of $\Delta\epsilon$ (or else the model would calculate a value $p > 1$ which leads to *always* injecting too little energy). In Appendix C.1 we discuss how resolution, total blast energy and the choice of $\Delta\epsilon$ affect the variance of the injected energy, which can guide the choice of $\Delta\epsilon$.

4.3 Numerical Methods

All the simulations we present here use the `clustered_SNe` code described by Gentry et al. (2017), with a few modifications. We will briefly describe the general

framework of the code and then clearly state the modifications, before moving onto the specific implementation details of each model added for this paper.

The `clustered_SNe` code hydrodynamically evolves blasts in a 1D (spherically symmetric) environment, with the assumption that all SNe occur at the same location $r = 0$. The ISM is assumed to have an initial spatially constant density (we use $\rho = 1.33 m_{\rm H}$ cm^{-3} for all the simulations presented here) and metallicity ($Z = 0.02$ for all simulations here); we only track total metallicity, not any specific species. The inner boundary is a zero-flux, zero-velocity boundary; the outer boundary condition does not matter since we choose a large enough domain so that the SNR does not reach the outer boundary. This ISM mass and metallicity is placed within moving mesh cells, with an initial spacing of Δr_0 that depends on the specific simulation and no initial velocity.

For a given cluster mass, the `clustered_SNe` code uses the `SLUG` code (da Silva et al. 2012; da Silva et al. 2014; Krumholz et al. 2015) to directly sample a Kroupa (2002) IMF of stars, and then explodes any stars with an initial mass greater than $8 M_\odot$ after a mass-dependent lifetime predicted by the Geneva stellar evolution models (Ekström et al. 2012). Explosion mass and metal yields follow the results of Woosley & Heger (2007), while each explosion is assumed to yield a constant $E_{\rm blast} = 10^{51}$ erg of energy (unless otherwise specified by a feedback model).

Each SN is injected into the innermost cell (unless otherwise specified by a feedback model), and then the cells are evolved using an approximately Lagrangian HLLC solver (Toro et al. 1994, with the specific implementation by Duffell 2016). Optically-thin, metallicity-dependent radiative cooling is included using the `GRACKLE` cooling li-

brary (Smith et al. 2017) assuming equilibrium chemistry and a Haardt & Madau (2012) extragalactic UV background.

We also made some modifications since the version described by Gentry et al. (2017). The biggest is that we changed the initial ISM temperature so that it matches the equilibrium temperature of the initial density and metallicity. For $\rho = 1.33 m_{\rm H}$ cm^{-3}, $Z = 0.02$, and a fixed $\gamma = 5/3$, this corresponds to a specific internal energy of 3.50×10^{10} erg g^{-1} (about 340 K as calculated by `GRACKLE`). Although this choice has no significant effect on the simulation outcome, which is the same as long as the ambient temperature is much smaller than the temperature of the hot gas in the SNR interior, starting the gas in thermal equilibrium simplifies the analysis. The second change is that, unlike in Gentry et al. (2017), we do not include pre-SN stellar winds; now the only mass that is injected is from the SN itself. We disable winds because they are generally not included in the feedback prescriptions we are testing.

For each feedback model we do simulations with 1 SN and 11 SNe, ensuring that the same SNe properties are drawn for all 1 and 11 SNe simulations respectively. The 1 SN simulations are run until a cluster age of 20 Myr (roughly 10 Myr after the only SN); the 11 SNe simulations are run until a cluster age of 100 Myr (roughly 97 Myr after the first SN). For each model and cluster size we also carry out both a high resolution run ($\Delta r_0 = 0.3$ for 1 SN; $\Delta r_0 = 0.6$ for 11 SNe to match the reference run in Gentry et al. 2017) and a low resolution run ($\Delta r_0 = 20$ pc). The particular resolution of our low resolution run is chosen so that the region in the inner "ghost" cell (the innermost 20 pc, which we do not hydrodynamically evolve) contains $\sim 10^3 M_\odot$ of material, which

would be expected to produce ~ 11 SNe if it were completely converted to stars. This mass and spatial resolution is also comparable to the typical highest values achieved in modern zoom-in cosmological simulations of spiral galaxies.

4.3.1 Delayed cooling implementation

As described in subsection 4.2.1, the key idea behind the delayed cooling model is that there should be a spatial scale around the location of a SN into which the SNR can expand before losing a significant amount of energy to radiative cooling; associated with this expansion should also be a characteristic time scale. The approach of the delayed cooling model is to temporarily disable radiative cooling for any resolution elements initially within this spatial scale, $R_{\rm E}$ for an appropriate time scale $t_{\rm E}$, explicitly delaying cooling.

These key scales are given as analytic expressions by Stinson et al. (2006, their Equations 9 and 10) that depend on the local ISM density, ρ_0, and pressure, P_0 as well as the blast energy, $E_{\rm blast}$:

$$\begin{aligned} R_{\rm E}(\rho_0, P_0) =& 10^{1.74} \left(\frac{E_{\rm blast}}{10^{51}{\rm erg}}\right)^{0.32} \left(\frac{\rho_0}{\mu m_{\rm H}\ {\rm cm}^{-3}}\right)^{-0.16} \\ & \times \left(\frac{P_0}{10^4 k_{\rm B}\ {\rm K\ cm}^{-3}}\right)^{-0.20} {\rm pc} \end{aligned} \tag{4.1}$$

$$t_{\rm E}(\rho_0, P_0) = 10^{5.92} \left(\frac{E_{\rm blast}}{10^{51}{\rm erg}}\right)^{0.31} \left(\frac{\rho_0}{\mu m_{\rm H}\ {\rm cm}^{-3}}\right)^{0.27} \times \left(\frac{P_0}{10^4 k_{\rm B}\ {\rm K\ cm}^{-3}}\right)^{-0.64} {\rm yr} \quad (4.2)$$

where μ is the mean molecular weight, $m_{\rm H}$ is the mass of the hydrogen atom, $k_{\rm B}$ is the Boltzmann constant. (Note: Stinson et al. (2006) end up adopting a slightly different, slightly longer timescale for their final model, but in their Section 5.3.1 they ultimately conclude it should not make a significant difference.)

It is easy to evaluate these expressions for the first SN when there is a single, clear value for ρ_0 and P_0 due to our initially-homogeneous ISM; for subsequent SNe it becomes more ambiguous due to the non-uniform bubble that has formed. To handle this, we solve for $R_{\rm E}$ iteratively, starting from the centre of the simulation and stepping outwards until the volume-weighted average density and pressure result in an $R_{\rm E}$ that matched the current radius. Fortunately, preliminary tests (using the results from an 11 SNe reference run) found that there should typically be a single, unique $R_{\rm E}$ for each SN; in practice we take the first valid $R_{\rm E}$ (i.e. $R_{\rm E}(\rho_0(< r), P_0(< r)) < r$). Within this radius, all cells have their cooling disabled for a time $t_{\rm E}$, even if they later move beyond this radius. Cells that have already had their cooling disabled from a previous SNe have their cooling turned off for the longer of: the current SN's $t_{\rm E}$ and the remaining duration from the previous SNe's shutoff periods ($t_{{\rm E},i} - t_{{\rm SN},i}$).

Since the cooling rate of a SNR does not truly go to 0 at high resolution, Stinson et al. (2006) leave $E_{\rm blast}$ as a free parameter that can be decreased to compensate,

suggesting a typical value of $E_{\rm blast} = 10^{50}$ based off their initial experiments. For each cluster we simulate, we will run two variants in order to explore the effect of this free parameter: one using $E_{\rm blast} = 10^{50}$ and another using $E_{\rm blast} = 10^{51}$.

In addition to $R_{\rm E}$ and $t_{\rm E}$, we must compute where to deposit the SN energy and mass. Following Stinson et al. (2006), we assume a fixed kernel mass ($M_{\rm kernel} = 3 \times 10^5 M_\odot$), and solve for the corresponding radius, $R_{\rm kernel}$, that encloses that mass. Within this radius, we inject mass and energy using a Gaussian kernel with 1D dispersion $\sigma = 0.1 R_{\rm kernel}$, weighted by the cell masses and truncated at $R_{\rm kernel}$. This kernel mass was chosen so that $R_{\rm kernel}$ is always larger than $R_{\rm E}$; this means some energy will be injected outside the cooling-disabled region and will be lost rapidly, but the amount so affected is minimal due to the sharp drop off in the Gaussian profile.

Finally, we point out that it is important that when the first SN occurs, the ISM is near its equilibrium temperature rather than significantly above it (which was originally the default of the `clustered_SNe` code; Gentry et al. 2017). If the ISM is far from equilibrium with a short cooling time, then an artificial discontinuity would rapidly develop near $R_{\rm E}$ after the first SN. Within $R_{\rm E}$ the gas would stay hot and over-pressured, while just beyond $R_{\rm E}$ the gas would cool and drop in pressure, leading to an outward-propagating shock.

4.3.2 Momentum-energy feedback implementation

Momentum-energy models are characterised by a common thread: at high resolution their key active component is the injected energy, while at low resolution the

active component is injected radial momentum, with a continuous transition between these regimes. For this work, we base our implementation off the `FIRE-2` algorithm (Hopkins et al. 2018a), but make a few necessary alterations to match the different geometry of our cells. Unlike Hopkins et al. (2018a) we neglect stellar winds in this work.

First, we define the total SN yields. The SN-ejected mass and metallicity will be consistent with our reference simulations, along with the times at which these SNe occur; this is in contrast to the SN mass yields and delay times suggested by Hopkins et al. (2018a) although we expect this makes relatively little difference. Next, we keep the SN blast energy at the fiducial $E_{\rm blast} = 10^{51}$ erg. The biggest difference between this model and our reference simulations is that this model also injects momentum. At high resolution, this momentum is determined by assuming the blast energy is fully kinetic:

$$p_{\rm ejecta} = \sqrt{2 m_{\rm ejecta} E_{\rm ejecta}} \tag{4.3}$$

but this momentum will be increased at lower resolution, depending on the properties of the cells into which the blast is injected.

The injection kernel is probably the most significant departure from the algorithm described by Hopkins et al. (2018a) owing to the different geometry of our simulation (we use rigidly structured 1D, nested shells, whereas the `FIRE-2` simulations use unstructured 3D moving particles). We identify the injection kernel to comprise the innermost $N_{\rm ngb} = 3$ cells ($\approx 32^{1/3}$, as opposed to their 32 nearest neighbours). Next, we

weight all cells equally, $w_i = N_{\rm ngb}^{-1}$ (using index i for the ith cell); this is in contrast to their solid angle-based weighting (Hopkins et al. 2018a, Eq. 2) which would only ever inject into 1 cell given our enforced spherical symmetry.

We now can specify the amount of mass, metals, momentum and energy added to each individual cell. Mass and metals are easy; they follow the weights:

$$\Delta m_i = w_i m_{\rm ejecta} \tag{4.4}$$

and

$$\Delta m_{Z,i} = w_i m_{Z,\rm ejecta}. \tag{4.5}$$

The injected momentum is slightly more complicated. As mentioned above, we start with a base amount of total momentum, $p_{\rm ejecta}$, but as the resolution decreases the amount of momentum is increased to mimic the SNR evolution below the resolved scales. At arbitrarily low resolution, all stages of the SNR evolution will be unresolved, so the momentum should approach the expected terminal momentum. This expected terminal momentum is calculated for each cell based on the cell's gas density and metallicity:

$$\begin{aligned}\frac{p_{{\rm t},i}}{M_\odot\ {\rm km\ s^{-1}}} =& 4.8\times10^5 \left(\frac{E_{\rm blast}}{10^{51}{\rm erg}}\right)^{13/14} \\ &\times \left(\frac{\rho_i}{\mu_i m_{\rm H}\ {\rm cm^{-3}}}\right)^{-1/7} f(Z_i)^{3/2}\end{aligned} \tag{4.6}$$

where

$$f(Z) = \begin{cases} 2 & Z/Z_\odot < 0.01 \\ (Z/Z_\odot)^{-0.14} & \text{otherwise} \end{cases} \tag{4.7}$$

and $Z_\odot = 0.02$. The two extremes—arbitrarily high resolution for which $\Delta p_i = w_i p_{\rm ejecta}$ and arbitrarily low resolution for which $\Delta p = w_i p_{{\rm t},i}$—are tied together by:

$$\Delta p_i = w_i p_{\rm ejecta} \min\left(\sqrt{1 + \frac{m_i}{\Delta m_i}}, \frac{p_{{\rm t},i}}{p_{\rm ejecta}}\right) \tag{4.8}$$

The injected energy similarly starts with a high resolution proposal: $\Delta E_i = w_i E_{\rm blast}$ which is then corrected at low resolutions. The motivation for this correction is to avoid adding energy at unphysical large distances from the SN. While a SNR will not be able to directly add energy beyond the cooling radius of the SNR, at very low resolution, some of the $N_{\rm ngb}$ neighbours in which we deposit energy might be beyond this cooling radius. Therefore, for each cell we first compute the expected SNR cooling radius:

$$R_{{\rm cool},i} = 28.4 \left(\frac{\rho_i}{\mu_i m_{\rm H}\ {\rm cm}^{-3}}\right)^{-3/7} \left(\frac{E_{\rm blast}}{10^{51}{\rm erg}}\right)^{2/7} f(Z_i)\ {\rm pc}. \tag{4.9}$$

If the cell's distance from the SN, r_i, is larger than $R_{{\rm cool},i}$, we then calculate the proposed change in the cell's internal energy, ΔU_i, and decrease it by a factor $(r_i/R_{{\rm cool},i})^{-6.5}$.

4.3.3 Simultaneous energy injection implementation

As introduced above in subsection 4.2.3, the simultaneous injection model attempts to explicitly harness the clustering of SNe by forcing every SNe from a cluster

to occur simultaneously. This loss of time-resolution changes the dynamics, hopefully in a positive way. Rather than each SN failing to heat the nearby material past the peak of the cooling curve (and quickly overcooling as a result), injecting all SN energy at the same time makes it more likely that the affected material will be heated beyond the peak of the cooling curve. Dalla Vecchia & Schaye (2012) go one step further, introducing a stochastic component that guarantees material is heated past the peak of the cooling curve; it is this particular algorithm that we will try to follow as closely as possible.

The first part, defining the SN yields and delay time distribution and yields, is easy. The yields are the same as in our reference model, in particular $E_{\rm blast} = 10^{51}$ erg. For the delay time distribution, if there are multiple SNe we modify the explosion times to all occur at $t = 30$ Myr, matching Dalla Vecchia & Schaye (2012); if there is only 1 SN, we do not modify the explosion time (typically $t \approx 10$ Myr), since that corresponds to an arbitrary shift of when we define $t = 0$ and does not affect the results in any way.

The injection kernel comprises the innermost $N_{\rm ngb} = 3$ cells[3], into which mass and metals are injected deterministically, while energy is injected stochastically. The mass and metals are distributed evenly between each cell:

$$\Delta m_i = N_{\rm ngb}^{-1} m_{\rm ejecta} \tag{4.10}$$

and

$$\Delta m_{Z,i} = N_{\rm ngb}^{-1} m_{{\rm ejecta},Z}. \tag{4.11}$$

[3]Conveniently, in our low resolution 11 SNe simulations, our ghost cell mass matches the cluster mass, $m_{\rm star}$, and the nearest 3 cells enclose $m_{\rm kernel} \approx 70 m_{\rm star}$, the closest we can get at this resolution to the value $m_{\rm kernel} = 48 m_\star$ used by Dalla Vecchia & Schaye (2012).

The stochastic energy injection within this kernel is more complicated. First, the mass within the kernel, $m_{\rm kernel}$ is computed, after adding the SN ejecta material. Then, given a value of $\Delta\epsilon$, the desired increase in specific thermal energy, we can calculate the probability of any cell within the kernel receiving energy:

$$p = \frac{E_{\rm blast} N_{\rm SNe}}{\Delta\epsilon} \frac{1}{m_{\rm kernel}} \tag{4.12}$$

At high resolution, the restriction $p < 1$ becomes a problem given the small mass within the kernel (coming predominantly from the ejecta mass: $\sim 14 M_\odot$ for 1 SN and $130 M_\odot$ for 11 SNe). So at high resolution, we adopt a value $\Delta\epsilon$ corresponding to a $\Delta T = 10^9$ K.

At low resolution, this constraint is not a problem. For 11 SNe, since the kernel mass to cluster stellar mass roughly matches the ratio expected by Dalla Vecchia & Schaye (2012) we can use the $\Delta\epsilon$ corresponding to the recommended $\Delta T = 10^{7.5}$ K. For 1 SN at low resolution, the kernel mass is not as well matched. In order to have a reasonable probability of injecting *any* energy we have to reduce $\Delta\epsilon$ to the value corresponding to $\Delta T = 10^6$ K. (Even with a ΔT as low as 10^6 K, there is still an $\approx 70\%$ chance that no cell receives energy for our 1 SN, low resolution simulations.) This value of ΔT is low enough to begin to become uncomfortable, but should allow us to avoid the worst overcooling; also, the main focus of this paper is on the low resolution *11 SNe* simulations, not the 1 SN simulations.

Although in a galactic simulation we would select cells independently and

stochastically with probability p, in this controlled test we will simply run all 8 possible realisations deterministically (enumerating the 2^3 possibilities of selecting or not selecting 3 cells). Each of these realisations will be weighted with their respective Bernoulli probability: $p^N(1-p)^{3-N}$ for a total of N cells being selected for energy injection.

4.4 Results of Feedback Models

Table 4.1: Simulation results. "Model" refers to the injection model; $N_{\rm SNe}$ refers to the number of SNe from the cluster; $E_{\rm blast}$ denotes the energy added *per SN*; Δr_0 gives the initial spatial resolution; $t_{\rm end}$ indicates when we extract the final momentum and energy of the simulation relative to the time of cluster formation; $p_{\rm end}$ gives the momentum at that time; $E_{\rm kin}$ is the final total kinetic energy within the computational domain while $\Delta E_{\rm int}$ is the change in total internal energy within the computational domain relative to the moment before the first SN. For the simultaneous energy injection model, the final momentum and energy results are presented as the mean and standard deviation of our realizations (but it should be remembered that the distributions of these results are very non-gaussian; see Figures 4.5 and 4.6).

Model	$N_{\rm SNe}$	$E_{\rm blast}$ (erg)	Δr_0 (pc)	$t_{\rm end}$ (Myr)	$p_{\rm end}/N_{\rm SNe}$ ($100 M_\odot$ km s^{-1})	$E_{\rm kin}$ (10^{49} erg)	$\Delta E_{\rm int}$ (10^{49} erg)
Reference	1	10^{51}	0.3	20	2763	0.620	-0.071
	1	10^{51}	20.0	20	265	0.008	-2.856
	11	10^{51}	0.6	100	34037	61.483	23.919
	11	10^{51}	20.0	100	9153	11.677	3.354
Delayed cooling	1	10^{50}	0.3	20	3803	0.972	-0.002
	1	10^{50}	20.0	20	2139	0.320	-1.252
	1	10^{51}	0.3	20	37058	18.274	-0.406
	1	10^{51}	20.0	20	31609	12.833	0.159
	11	10^{50}	0.6	100	3939	4.065	2.335
	11	10^{50}	20.0	100	1805	1.458	0.585
	11	10^{51}	0.6	100	34185	60.960	26.573
	11	10^{51}	20.0	100	25169	35.023	315.989
Momentum-energy	1	10^{51}	0.3	20	2666	0.597	-0.063
	1	10^{51}	20.0	20	4047	0.873	-0.275
	11	10^{51}	0.6	25	16687	112.626	159.605
	11	10^{51}	20.0	100	14010	18.326	5.409
Simultaneous	1	10^{51}	0.3	20	2781 ± 1414	0.663 ± 0.371	-0.075 ± 0.059
	1	10^{51}	20.0	20	133 ± 134	0.002 ± 0.003	-1.759 ± 0.447
	11	10^{51}	0.6	100	2070 ± 870	2.427 ± 0.982	0.527 ± 0.265
	11	10^{51}	20.0	100	874 ± 2700	7.845 ± 33.433	-21.528 ± 13.502

In this section we give the results and discuss each model in turn. In the next section (section 4.5), we compare the results between models. A summary of the simulation results is given in Table 4.1.

4.4.1 Direct injection results

We start with the simplest simulations which directly inject thermal energy and no momentum to the innermost cell. The evolution of the momentum with respect to time for these 4 simulations is shown in Figure 4.1. These will provide the "reference" results, against which we will compare the 3 competing models.

Looking first at the 1 SN results, we see the standard picture: at high resolution, we recover the standard terminal momentum ($\sim 3000\ M_\odot$ km s^{-1}), but at low resolution, overcooling becomes a problem, leading to far too little momentum (in this case, an order of magnitude too little momentum).

This result is mirrored in the 11 SNe simulations, but there it is slightly mitigated. As subsequent SNe occur, the density near the location of the blast drops, causing cooling to become less efficient and the resolution requirements to be loosened. This is not a perfect solution; it takes multiple SNe before a well-defined superbubble is inflated at low resolution, and even then each SN blast eventually propagates out to the dense shell where overcooling can occur. Still, the low resolution 11 SNe run only contains 3-4 times less momentum than the high resolution simulation, whereas the 1 SN simulations exhibit a factor of 10 deficit in momentum when the resolution is worsened.

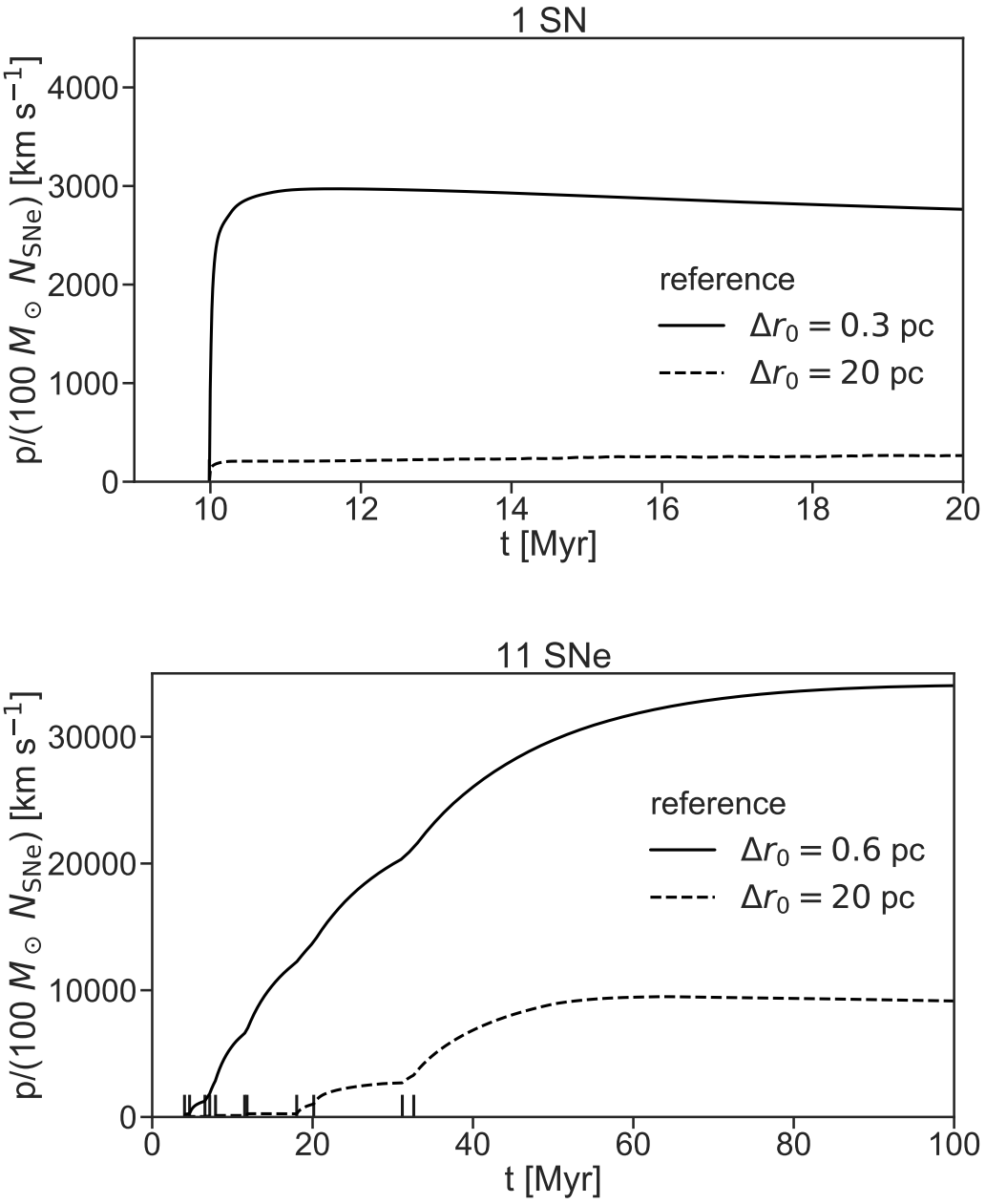

Figure 4.1: Comparison of the momentum as a function of time for our direct injection simulations, which we later use as the reference results. The top shows a comparison between different resolution 1 SN simulations; the bottom shows the different resolution 11 SNe simulations. Note the difference in both horizontal and vertical scale between the two panels, both here and in subsequent similar plots.

4.4.2 Delayed cooling results

Similar to the reference case, we ran simulations of 1 and 11 SNe at high and low resolution. Since the blast energy is a free parameter within the Stinson et al. (2006) algorithm, we carry out this experiment for two different blast energies: $E_{\rm blast} = 10^{50}$ erg and 10^{51} erg. The resulting momentum evolution of each simulation can be seen in Figure 4.2.

Starting with the 1 SN simulations, we see the expected behaviour. When injecting the *recommended* blast energy (10^{50} erg), both the high resolution *and* the low resolution simulations do a good job reproducing the terminal momentum of the high resolution reference simulation. This is what the method was designed to do. Conversely, when we inject a blast of 10^{51} erg and also shut off cooling, we observe a terminal momentum that is too large by a factor of ~ 10. This is unsurprising for the high resolution case; if we add the same amount of energy, but delay the onset of cooling, the result will be too much momentum. It is slightly more interesting that the low resolution simulation *also* results in too much momentum. However, Stinson et al. (2006) were aware of this potential problem, and this is what motivated them to recommend decreasing the blast energy.

The situation for the 11 SNe simulations is very different. In this case using $E_{\rm blast} = 10^{50}$ erg results in far *too little* momentum—at high resolution it even does worse than our low resolution reference simulation (which used $E_{\rm blast} = 10^{51}$ erg). This is because for the later SNe, the superbubble approaches near-adiabatic behaviour.

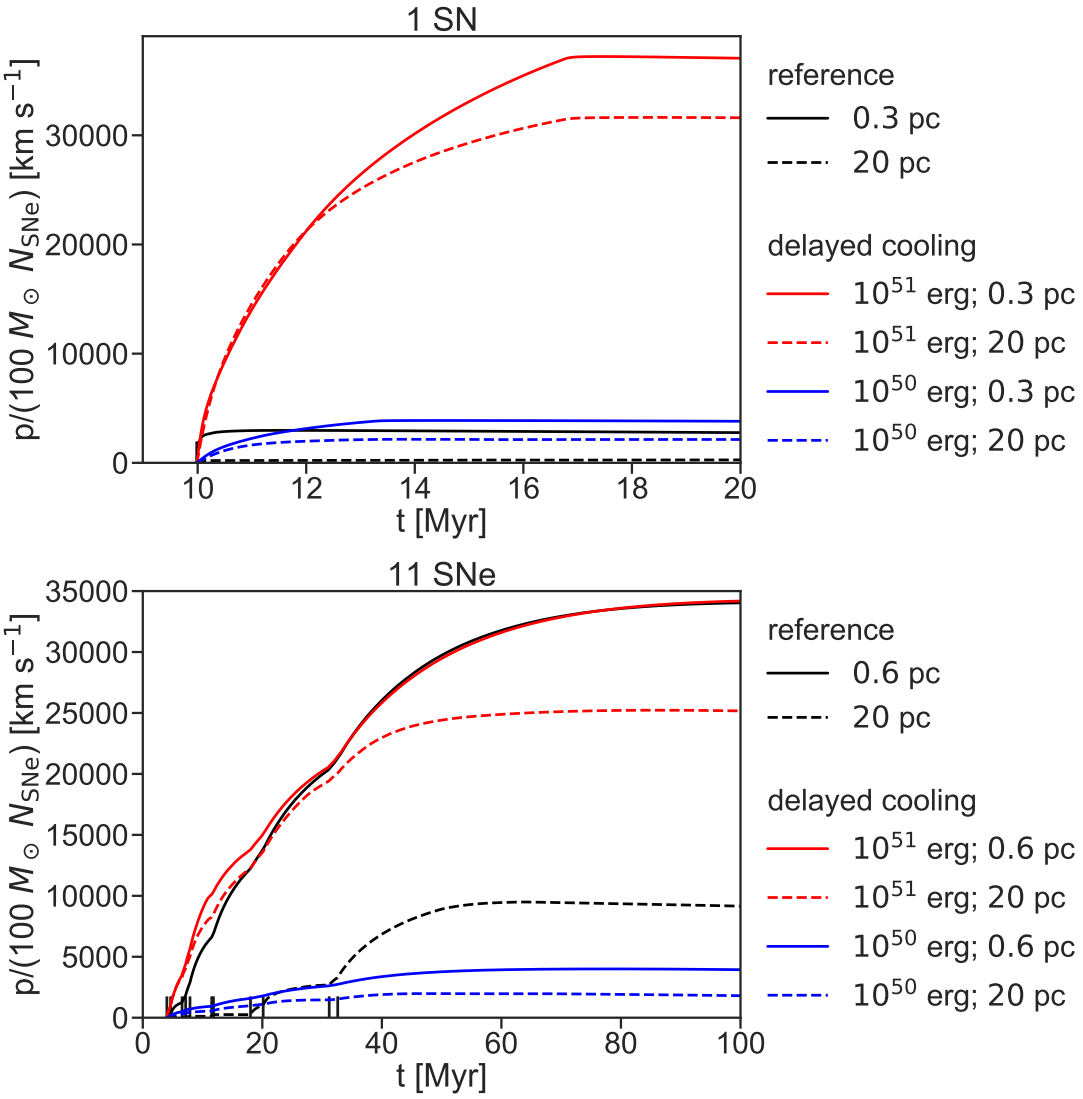

Figure 4.2: Same as Figure 4.1, except now also overplotting the delayed cooling simulations, with the different resolutions and $E_{\rm blast}$ parameters denoted in the legends.

Since the cooling radius is within the bubble where cooling is already relatively low, the cooling shut off switch in the delayed cooling prescription has little effect. However, the reduction in injection energy that Stinson et al. (2006) recommend in order to fix the single-SN case then results in an under-powered superbubble and too little momentum even at high resolution.

When we use stronger blasts ($E_{\rm blast} = 10^{51}$ erg), we find better results for the 11 SNe case. For the first few SNe, the momentum starts too high (as expected from our 1 SN simulation results), but as the superbubble becomes more adiabatic, the delayed cooling approach starts to approach our direct injection behaviour. It is remarkable just how well the delayed cooling approach does with $E_{\rm blast} = 10^{51}$ erg for 11 SNe. Despite starting with more momentum at early times, and still shutting off cooling for each subsequent SNe, the high resolution 11 SNe simulation with $E_{\rm blast} = 10^{51}$ differs in terminal momentum from the high resolution reference simulation by less than 1%. The lower resolution delayed cooling simulation ($E_{\rm blast} = 10^{51}$) does not do quite as well, but still is relatively close (differing by only about 25%).

Thus this method is successful at being *less resolution-dependent* than the reference, direct injection method. Whether this resolution-robust method produces accurate momenta is a more complicated question. We have found that a fixed $E_{\rm blast}$ is unable to handle both 1 SN and 11 SNe clusters; an energy of $E_{\rm blast} = 10^{50}$ erg as recommended by Stinson et al. (2006) gives a good fit to the single SN case, but fails for 11 SNe, while injecting the full SN energy $E_{\rm blast} = 10^{51}$ erg succeeds for 11 SNe but fails for 1. The fundamental reason for this is easy to understand: the mean

amount of radiative loss per SN is not a single number, but instead depends on the SN environment, and in particular on whether a SN is going off inside an already low-density, hot cavity carved by a previous SN. Both the *ad hoc* reduction from $E_{\rm blast} = 10^{51}$ erg to 10^{50} erg and the density- and pressure-dependence built into Equation 4.1 and Equation 4.2 for $R_{\rm E}$ and $t_{\rm E}$ attempt to capture the complex dependence of radiative loss on environment, but they do not do so with sufficient accuracy to reproduce the results of the high resolution simulation across a factor of 10 in cluster size. That said, this analysis suggests that it might be possible to find a prescription for $E_{\rm blast}(N_{\rm SNe})$, or to choose a value of $\langle E_{\rm blast} \rangle$ averaged over the cluster mass function, that performs better than the current approximation of picking a single $E_{\rm blast}$. This would require a campaign of simulations similar to ours, to quantity the amount of radiative loss as a function of cluster size.

This is a clear indication, and a reminder, that these subgrid models can behave differently at different cluster sizes. This both means that it is useful to check how well each method is able to handle clustered SNe, but also provides a warning that our conclusions likely depend on the cluster sizes we test here (1 SN and 11 SNe). Dekel et al. (2019) identify as many as seven possible SN-driven superbubble regimes, of which we have tested only two.

4.4.3 Momentum-energy feedback results

We tested the momentum-energy injection prescription with high and low resolution simulations of 1 and 11 SNe clusters. The momentum evolution is compared to

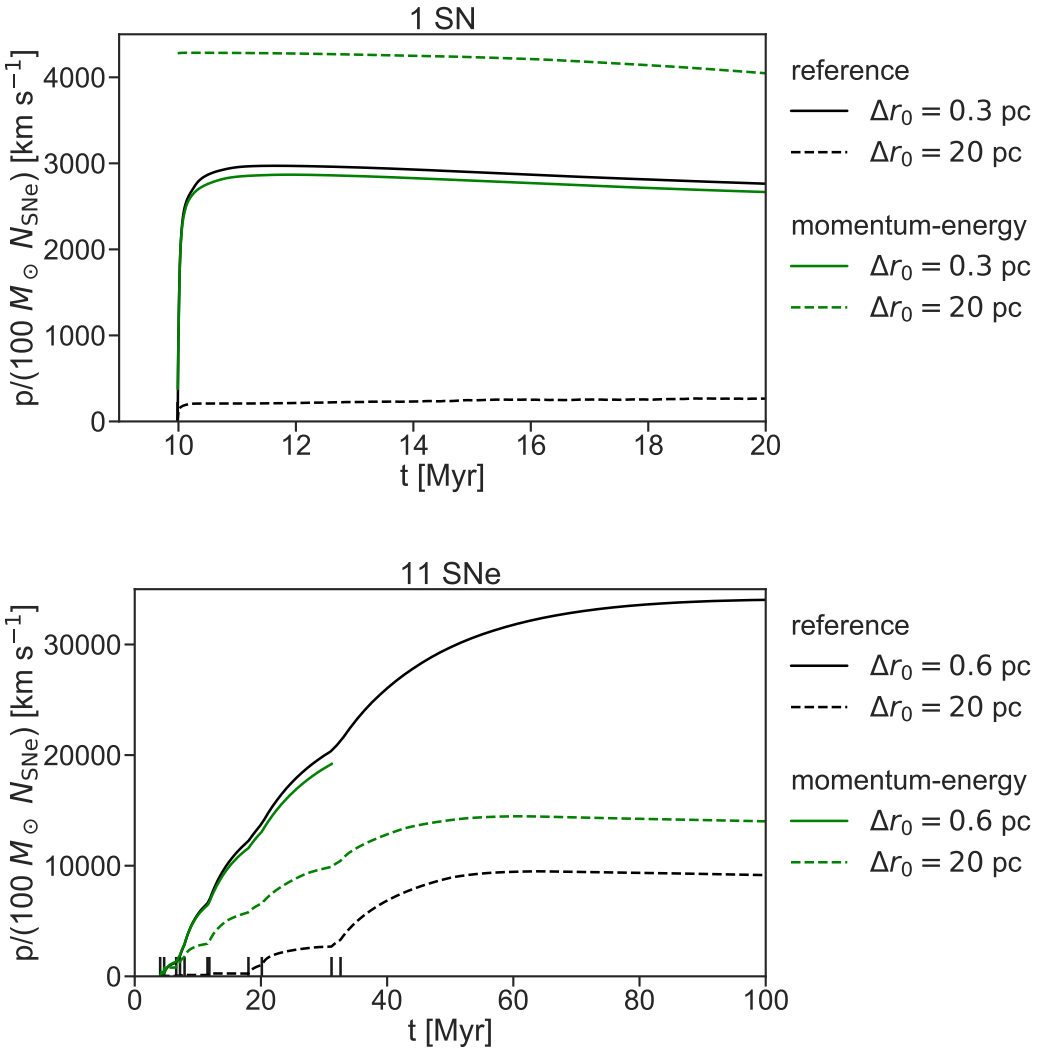

Figure 4.3: Same as Figure 4.1, except now overplotting the simulations using the momentum-energy feedback model. While it is difficult to see, the high resolution momentum-energy run is plotted starting at the first SN with the others; it simply matches the fiducial model so well that it is difficult to visually distinguish until $t > 10$ Myr.

our reference simulations in Figure 4.3.

The clearest discrepancy between the results of this scheme and our high resolution fiducial results is for the 1 SN, low-resolution simulation, which has a terminal momentum which is too high (by $\sim 30\%$). However, this is relatively easy to diagnose; Hopkins et al. (2018b) prescribe $\sim 5 \times 10^5\ M_\odot\ N_{\rm SNe}$ km s^{-1} of momentum per SN (at $\rho = 1.33 m_{\rm H}$ cm^{-3}), so it is not surprising at that low resolution this assumption yields more than the $3 \times 10^5\ M_\odot\ N_{\rm SNe}$ km s^{-1} recovered at high resolution. Hopkins et al. adopt this value from the earlier 1D simulations of Cioffi et al. (1988), and to be consistent with the earlier `FIRE-1` simulations that used this value. Our 1D simulations improve on those of Cioffi et al. (1988) in many ways – for example by use of modern cooling tables and by adoption of a pseudo-Lagrangian high-order hydrodynamics method – and thus our somewhat lower value is likely more reliable. However, for our purposes here the offset between our high resolution results and the results of the momentum-energy feedback method at low resolution are not particularly significant, since they result from a particular numerical parameter choice, which can easily be changed.

At 11 SNe, we find that adoption of the momentum-energy injection method helps mitigate the effects of low resolution, yielding a terminal momentum that is a factor of ≈ 1.5 higher than a naive low resolution direct injection method, though still a factor of ≈ 2.5 too small compared to the high resolution results. Although the momentum-energy injection results do not match as well as the best delayed cooling results using a value of $E_{\rm blast}$ tuned to match the 11 SNe case, the momentum-energy

prescription does significantly better than the delayed cooling model using the lower $E_{\rm blast}$ recommend by Stinson et al. (2006).

4.4.4 Simultaneous energy injection results

Finally, we compare the simultaneous energy injection approach to our reference simulations, with the momentum evolution shown in Figure 4.4. Since there are so many realisations (8 high resolution and 8 low resolution), we simplify these figures by only showing the probability-weighted mean, along with mean ± 1 standard deviation at each point in time. The full distribution of final momenta can be seen in Figures 4.5 and 4.6 for the 1 SN simulations and 11 SNe simulations respectively. No substantial difference was observed in the shape of the time evolution of the momentum of each realisation besides the overall normalisation.

First, for 1 SN at high resolution, we see in Table 4.1 and Figure 4.4 that the mean momentum corresponds well to the results from our reference simulations. There is some scatter (see Figure 4.5), as some realisations inject $> 10^{51}$ of energy and others inject $< 10^{51}$, but the probability-weighted mean result is close to the standard expected value. This is unsurprising, because for 1 SN most theoretical studies predict a roughly linear scaling between momentum and energy at high resolution (e.g., Cioffi et al. 1988, Draine 2011), and the expected energy is unbiased by construction ($\langle \Delta E \rangle = E_{\rm blast}$). At low resolution for 1 SN, we see the simultaneous energy injection model does not do significantly better than the direct injection model, suggesting that it is still susceptible to overcooling. This is not very surprising. Our reference method is to inject 10^{51} erg

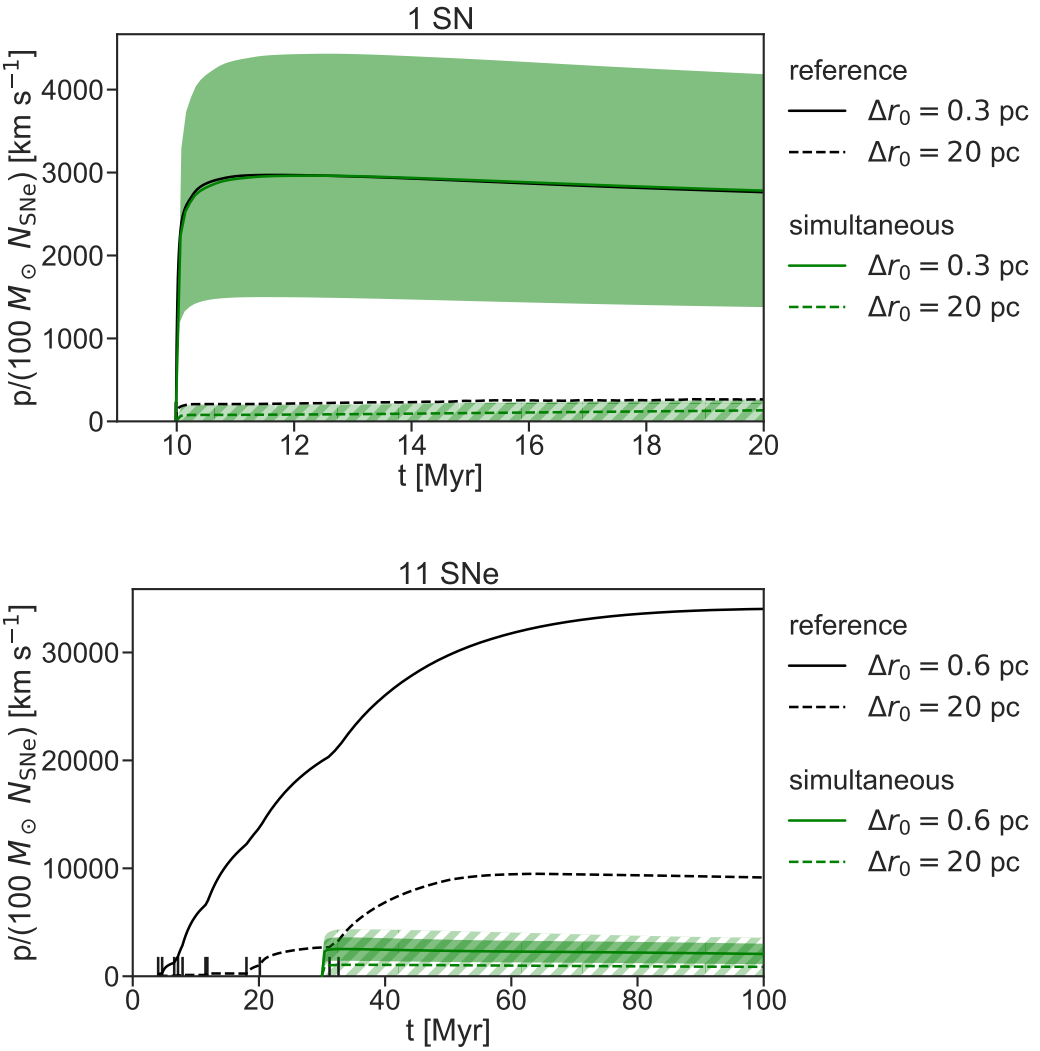

Figure 4.4: Same as Figure 4.1, except now overplotting the simultaneous energy injection simulations. For each simulation, we ran all eight possible realisations of the stochastic injection process. In this figure for each timestep we show the probability-weighted mean with a central green line and shade ±1 [probability-weighted] standard deviation around the mean—the high resolution run uses a solid shading, while the low resolution run uses a hatched shading.

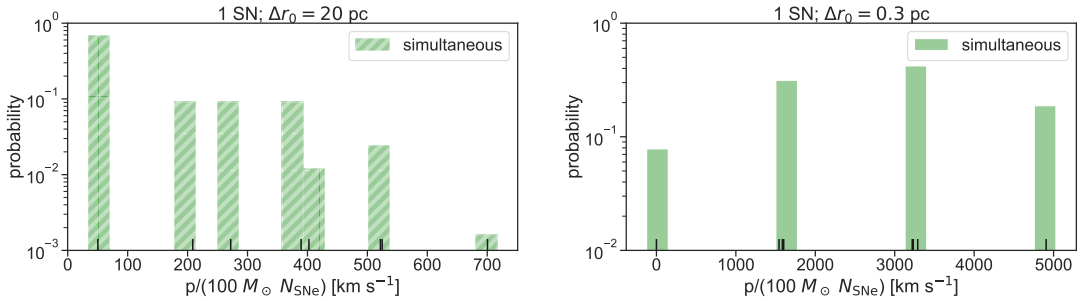

Figure 4.5: The final ($t = 20$ Myr) momenta for all realisations of the 1 SN simultaneous energy injection model, with the low resolution realisations on the left and the high resolution realisations on the right. The exact momentum of each realisation is marked by a solid black tick mark; for the histogram, each realisation contributes its Bernoulli probability of occurrence to the binned probability mass function.

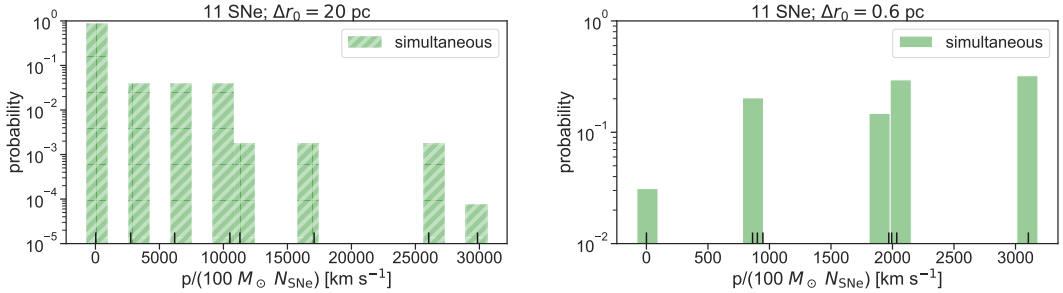

Figure 4.6: Same as Figure 4.5, except now for our 11 SNe simulations using the simultaneous injection model extracted at $t = 100$ Myr.

into the innermost cell; the closest analogue is a realisation which injects 0.95×10^{51} erg into the innermost cell (and no energy elsewhere). In that case the only other difference between the two methods is that the reference approach adds all the ejecta mass into the innermost cell, whereas the simultaneous energy injection model spreads it equally among the 3 innermost cells, but this mass contribution is negligible at $\Delta r_0 = 20$ pc resolution. Thus the final outcome largely reduces to a question of the amount of added energy; the realisation that adds 0.95×10^{51} erg results in less final momentum than the low resolution reference simulation, whereas realisations that add $> 10^{51}$ erg result in more momentum. When properly weighted, the average added energy is 10^{51} erg, so the average resulting momentum is consistent with the low resolution reference simulation. Ultimately this suggests that it is not enough to only prevent overcooling at the initial time of injection (by ensuring a fixed ΔT); overcooling in the subsequent shock front is also an important consideration.

Moving to 11 SNe we find somewhat worse results. At high resolution, we find the average momentum per SN for 11 SNe is roughly consistent with our results for 1 SN; this is bad because even our low resolution reference simulation showed a substantial momentum efficiency boost. Even when we look at the high resolution realisation which adds the most energy ($\approx 1.6 \times 10^{52}$ erg, i.e. $\approx 50\%$ above the mean value), it still results in less momentum than the 11 SNe low resolution reference simulation.

At low resolution, while we do see a momentum efficiency boost for the 11 SNe simultaneous injection realisations relative to their 1 SN simultaneous injection counterparts, it still has not clearly improved beyond the momentum efficiency of high

resolution single SN simulations. This general situation is especially concerning since the cluster mass ($\approx 10^3 M_\odot$) is actually well paired to the kernel mass at low resolution ($70 \times 10^3 M_\odot$), as this implementation intended.

So in conclusion, the simultaneous energy injection model does not appear to be very effective at producing the correct final momentum. For the 1 SN cluster its performance is comparable to the direct injection model, but for the 11 SNe cluster it does worse than the direct injection model.

4.5 Discussion

4.5.1 Comparison of injection methods

In this section we focus on our primary question: which subgrid models produce a momentum efficiency for 11 SNe at low resolution that is at least a factor of 2 greater than the fiducial momentum efficiency of an isolated SN (i.e., $p_{\rm end}/(100 M_\odot N_{\rm SNe}$ km s$^{-1}) > 5500$)? This is a necessary but not sufficient test; we predict the true momentum efficiency boost for this cluster is roughly a factor of 2, but these 1D simulations might overpredict the momentum that application of the same subgrid model in 3D would yield due to artificially suppressing asymmetries from hydrodynamic instabilities that drive mixing. Therefore, a subgrid model that cannot produce a factor of 2 boost in momentum efficiency in 1D would also necessarily underproduce momentum in 3D. However, we cannot assume the converse: if a model produces greater than a factor of 2 enhancement in momentum efficiency in 1D, it still might produce too little momentum

in 3D.

Looking at Table 4.1, we can easily answer this question for the three subgrid models. The delayed cooling model clears this threshold for $E_{\rm blast} = 10^{51}$ but not $E_{\rm blast} = 10^{50}$, the momentum-energy model clears this threshold, and the simultaneous injection model does not clear this threshold on average (although an individual realisation has a $\sim 10\%$ probability that it will clear this threshold; see Figure 4.6).

As noted in Sections 4.1 and 4.4.2, this is not a perfect test. First, we do not precisely know the true momentum efficiency for this cluster; we can only extrapolate existing 3D simulations. Second, here we have only tested two cluster sizes ($N_{\rm SNe} = 1$ and 11), whereas we have shown that a particular choice of model parameters might perform well at one cluster scale and poorly at another (in particular, see subsection 4.4.2). Therefore we cannot be sure how these models might perform across a realistic distribution of cluster sizes without a more methodical study, although by choosing an approximately maximal enhancement cluster ($N_{\rm SNe} = 11$), we hope to bound the problem. Third, by using 1D simulations we are artificially suppressing asymmetries produced by hydrodynamic instabilities that increase mixing and can decrease the final momentum. For instance, our reference model at low resolution clears our factor of 2 enhancement threshold, even though it almost certainly would result in too little momentum in 3D. This illustrates that our threshold could be too lenient, but to raise it higher would require a prediction about how additional mixing would affect each subgrid model. We cannot measure that directly from these simulations for each subgrid model, but we can make qualitative predictions.

4.5.2 Predicted effects of mixing

Although the low resolution (20 pc), 11 SNe simulation using our reference model shows a factor of ~ 3 increase in momentum efficiency relative to the fiducial isolated SN momentum efficiency, some 3D simulations of the same cluster (Gentry et al. 2019) showed an apparent *decrease* in momentum efficiency relative to the fiducial isolated value, even at much higher resolutions (e.g., 2 pc) due to the presence of increased mixing in 3D. This mixing is entirely numerical: the same simulation run at higher resolution shows a momentum increase, with the momentum continuing to increase even at the finest resolution available. Thus, while we do not have corresponding 3D simulations of this cluster for the subgrid models tested here, it is nonetheless important to consider what impact mixing might have in 3D at the resolutions likely to be typical of the simulations in which these recipes are deployed.

Delayed Cooling For the 11 SNe, delayed cooling simulations with $E_{\rm blast} = 10^{51}$ erg, no cooling can occur until the shock moves beyond $R_{\rm E} \approx 100$ pc or $t_{\rm E} \approx 7$ Myr pass. In our low resolution simulation, the radius restriction passes first, after about 1 Myr has elapsed, at which point we can observe noticeable radiative cooling. Unfortunately the momentum efficiency at this time only reached about $p(t)/(10^2 N_{\rm SNe} M_\odot \text{ km s}^{-1}) \sim 1500$; it has not yet achieved the isolated SN momentum efficiency let alone the desired factor of 2 enhancement. Most of the energy is still contained within radiative cooling-disabled cells, so our 1D simulation is still able to gain significant amounts of momentum beyond this time, but the same might not hold in 3D, especially if hydrodynamic in-

stabilities are able to mix mass and energy across the interface of resolution elements with enabled and disabled radiative cooling. Therefore it is possible that although this model performs well in 1D, it could result in too little momentum in 3D.

Momentum-energy feedback The momentum-energy model has a useful safeguard: at low resolution, even if all the thermal energy is radiated immediately after each SN, the deposited momentum is prescribed to approximately match the isolated SN terminal momentum. This means that it is unlikely that the momentum-energy model would result in a lower momentum efficiency than the fiducial isolated SN efficiency, even in the presence of strong mixing in 3D. Furthermore, this model already prescribes slightly too much momentum (see subsection 4.4.3), and if the deposited momentum is enough to open a low density bubble, then the momentum deposited with subsequent SNe can increase as the local ISM density decreases (Equation 4.6). Using our low resolution single SN simulation with this model model as a lower limit on the efficiency ($p/(10^2 N_{\rm SNe} M_\odot$ km s$^{-1}) \approx 4400$), we feel confident that this model would exhibit some enhancement relative to the fiducial momentum efficiency, but cannot be sure that it would achieve the desired factor of 2 enhancement. While it might be possible to tune this model by increasing the deposited momentum per SN for larger clusters, it is unclear how to best do this for a distribution of cluster sizes and across a range of resolutions for which a bubble might or might not be opened.

Simultaneous energy injection The simultaneous energy injection model provides no direct means to prevent enhanced cooling due to mixing, but it does provide an

indirect benefit: there is no time between SNe for the shock to weaken, which is when 3D instabilities like those seen by Gentry et al. (2019) are especially likely to develop and strengthen. Still, it is unlikely that this model will produce *more* momentum in 3D, and given that in 1D it already produces too little momentum $\sim 90\%$ of the time, we do not expect this model would be able to capture the effect of clustering in 3D.

4.6 Conclusions

We set out to answer a primary question: which of the subgrid models commonly-used to model SN feedback in galactic and cosmological-scale simulations can reproduce the factor of > 2 increase in the terminal radial momentum delivered per SN when SNe occur in a cluster of ≈ 10 SNe rather than as single, isolated event. This question is crucial because both observations and simulations suggest that superbubbles driven by multiple SNe play an important role in regulating the formation of galactic winds and ejecting mass and metals from galaxies. Our study therefore illuminates which subgrid models can at least potentially capture this phenomenon.

The three methods we tested meet this goal with varying degrees of success. A delayed cooling model (similar to that used in the `GASOLINE-2` code) can mimic the increase in terminal momentum of superbubbles, but only when we increase the deposited energy per SN from $E_{\rm blast} = 10^{50}$ erg (as recommended) to 10^{51} erg; the price of this choice is that it overestimates the terminal efficiency produced by a single SN. A momentum-energy model (similar to that used in the `FIRE-2` simulations) achieves

this as well, without requiring any changes. A simultaneous energy injection model (similar to that used in the `EAGLE` simulations) fails with $\sim 90\%$ probability in any given stochastic realisation, as well as in the mean of the stochastic results. While there are other ways these model could be tested, this test provides a useful window into how these subgrid models behave in realistic situations where SNe are clustered, unlike the isolated SN configuration for which many of these models were explicitly designed.

Our results show the value of performing these tests on common SNR / superbubble regimes, but this also hints at the limitations of exploring only two possible SNR regimes, as we have here. For example, under the delayed cooling model $E_{\rm blast} = 10^{50}$ erg is strongly favoured over $E_{\rm blast} = 10^{51}$ erg for 1 SN, but for 11 SNe the oppostive is true. Unfortunately, having tested only two cluster sizes, we cannot directly prescribe a solution. It is unclear if a simple switch between these two energy values, depending on whether SN are overlapping, would be sufficient to capture the main effects of clustering, or if we need a complex formula for $E_{\rm blast}(N_{\rm SNe}, \rho, Z, ...)$. As another example, with the momentum-energy model, we saw that for 1 SN at low resolution, we can directly prescribe the terminal momentum, but at 11 SNe we ended with a different, higher momentum efficiency. If this model needs to be tuned stronger or weaker to properly account for clustering, it is unclear how to do so precisely; for a single SN, we could just turn a knob, but for a superbubble more complex behaviour dynamically emerges.

In order to understand subgrid models better we would at least need to run similar tests in 3D, which is how these models are most commonly applied and where these simulations would experience stronger mixing leading to stronger cooling and

lower momenta. In subsection 4.5.2 we made qualitative predictions on how each model might be affected by mixing in 3D, but it is hard to know a quantitative value without running the experiment directly. However, even if we ran these models in 3D (and in an ISM representative of those present in low resolution galactic simulations) our test would only be as powerful as our knowledge of the true momentum efficiency.

With regard to this last point, we remind the reader that the true momentum efficiency of clustered SN feedback is currently unknown, inherently limiting any test like this. While we are making progress towards understanding the effects of clustering on SN feedback, discrepancies as large as a factor of 5 still exist within current literature. These discrepancies are primarily because we do not know the true level of physical mixing present in a superbubble expanding into a realistic ISM. Solving this problem likely requires converged simulations that include thermal conduction and a 3D, multi-phase, turbulent, magnetised ISM. While various simulations of clustered SNe have touched on each of these in turn, none have done so simultaneously, let alone across a range of superbubble regimes. Until that point, our ability to test, diagnose problems within and improve subgrid models of SN feedback will remain fundamentally limited.

Chapter 5

Conclusion

Where does this leave us?

First, it is unclear what the actual, quantified effect of clustering is on the efficiency of SN momentum feedback. Our 1D simulations (chapter 2) suggest that the effect could be *up to* a factor of ~ 10 in the momentum efficiency (comparing $N_{\rm SNe} = 1$ and $N_{\rm SNe} = 11$ for $\rho = 1.33 m_{\rm H}$ cm^{-1} and $Z = 0.02$). When we try to repeat the same $N_{\rm SNe} = 11$ cluster simulation in 3D, we also find there is an *increase* in efficiency, but can only directly measure an increase of about 1%; if we extrapolate to higher resolutions we can predict a factor of ~ 2 increased efficiency.

For our 1D simulations, we can go beyond the efficiency for a single cluster, and explicitly average the efficiency across a typical cluster mass distribution and find that on average clustering might increase efficiency by a factor of ~ 4. Since we only ran 1 cluster size in 3D, we cannot directly average across scales, but given that going from 1D to 3D reduced the momentum of our 11 SNe cluster, it is possible that the

average momentum is also decreased in 3D (although Keller et al. 2014 do find higher efficiencies in 3D in line with our 1D results for more massive but less common clusters).

In order to be able to quantify the impact more precisely we need a combination of better numerical methods (or computational resources) and a better model for the background ISM. We want to get to the point where the artificial diffusivity of our simulations is less significant than that of physical mixing processes such as thermal conduction and hydrodynamic instabilities. This will likely not be solved *just* by improving numerical methods or computational resources in the near future; in order to achieve a resolution similar to the Field length[1] estimated in chapter 3, we would need to increase our linear spatial resolution by a factor of $\gtrsim 30$ above what we have currently achieved (i.e., we would need to go from 1 pc to 0.03 pc). Assuming a naïve scaling, this would increase the required CPU time by $(30)^4 \approx 10^6$; given that our high resolution simulations already take $\sim 10^5$ CPU hours (not including the cost of actually solving for conduction at that resolution), an increase of 10^6 goes way beyond what we can expect to access. It is therefore unlikely that we will be able to directly quantify the effect of clustering on SN momentum yields in a homogeneous ISM; fortunately, the true ISM is not homogeneous.

To truly solve this problem, we will need the results to hold in a realistic ISM, not just a homogeneous ISM. In chapter 3 we took baby steps in that direction by including magnetic fields, which helped prevent the onset of hydrodynamic instabilities,

[1] Note that the Field length can vary over time and space, *and* the spatial resolution of our Lagrangian methods vary over time and space. Here I simply use the initial [spatially-constant] spatial resolution and the Field length at the shell estimated in chapter 3 for simplicity.

increasing the momentum injection (depending on how you measure momentum). On the other hand, Kim et al. (2017) use a slightly more complex, static 2-phase ISM in their study of clustered SNe, and find that when the SNR collides with cold clouds in this ISM it leads to increased mixing and lower final momenta. Perhaps this provides hope; if physical mixing is much higher in a realistic ISM than a homogeneous ISM, then we would not need significantly improved resolution or methods. However, we should point out that Keller et al. (2014) also tested a static, clumpy ISM and found a significant increase in final momentum, in conflict with the clumpy ISM simulations of Kim et al. (2017). Therefore, it is safe to say we do not yet understand what role a clumpy ISM plays in stabilizing or enhancing mixing, and if we do not understand a static clumpy ISM, we certainly do not understand the effects a realistic ISM has on superbubble evolution.

No one has completed a study of clustered SNe in a truly realistic ISM. Perhaps Martizzi et al. (2015) get closest, but even their simulations are lacking. First, their periodic boundary conditions mean that their simulation corresponds to a cluster of indeterminate mass. Second, they leave out magnetic fields for simplicity, acknowledging: "Future work incorporating magnetic fields and anisotropic thermal conduction into simulations analogous to those presented [by Martizzi et al. (2015)] would be very valuable." Ideally, in order to solve this problem (even at fixed initial average density and metallicity), one would need simulations that are 3D, include a turbulent, multiphase ISM, and include magnetic fields (or at least would ideally show that the neglected elements are not crucial). Realistically, that costs a lot of time and computa-

tional resources, and I would not be able to confidently predict the required resolution for convergence. It is easy to write down this "wishlist" of items here, but harder to achieve in practice.

What should galactic simulations do in the meantime? Since we do not have a specific number for the average momentum yield from clustered SNe, we cannot give a definite, straightforward prescription.

One approach, if your simulation uses a momentum-based subgrid model for feedback, would be to "bracket" the possible range; run a simulation with the fiducial $p/N_{\rm SNe} \approx 3 \times 10^5 M_\odot$ km s^{-1}, one with a lower yield (representing the Kim et al. 2017 results) and one with a higher yield (representing our results). This, by itself, will not be terribly useful for determining which is correct, but does help reflect the uncertainty in the momentum yield if you are trying to be careful. (Kim et al. (2017) suggest that higher momentum yields might lead to galaxy properties which do not match observations, but this does not show that the problem is in the momentum yield as opposed to the other prescriptions.)

Another approach, whether or not you use momentum-based subgrid models is to ensure that the SNe locations are clustered in space and time. For example, Fielding et al. (2018) do this, followed by a significant amount of analysis about how clustering would affect a real simulation. More generally, many approaches already achieve this with "star" particles which help keep track of where core collapse SNe "should" occur. Nonetheless, remember that clustering the locations is not necessarily enough; chapter 4

was an entire project devoted to exploring the ways that existing subgrid models might differ from high resolution simulations, even when every model assumed all SNe go off in the exact same location with a realistic delay time distribution. The details of the subgrid model can make a significant difference in whether the physical effect of SNe clustering is captured properly.

Part II

Machine Learning with Galaxy Images

Chapter 6

Introduction

Feedback (such as SN feedback discussed in Part I) can have a significant impact on the mass distribution of galaxy and its halo; for example Pontzen & Governato (2014) show that bursty SN feedback can change the shape of the dark matter density profile at the centers of galaxies. This is crucial, as it is one way to possibly solve the "core-cusp" problem within cosmology: in Cold Dark Matter (CDM) only simulations, the central density slope ($d\rho/dr$ at $r \lesssim 1$ kpc) is too steep compared to observations (for a more comprehensive review, see de Blok 2010). While it is reassuring that strong, bursty star formation feedback is *one* way to solve this problem, it is not the only proposed solution; it might instead be an indication that our standard model of CDM is incorrect and needs to be changed (see, for example, Schive et al. 2014 for how "fuzzy" dark matter might relieve the tension between dark matter-only simulations and observations). Since many of these proposals have been crafted to reproduce the inner density slope, we need a different measurement which can help test these various

models. One measurement that could help break the degeneracy is the total halo mass, measured out to scales > 100 kpc (Leauthaud et al. 2019)—much beyond the 1 kpc scales at which the density slope is measured.

Weak lensing provides one pathway to measuring the total halo mass, and recent forecasts suggest it will be able to measure the *average* halo mass of the lowest mass galaxies to date (galaxies with *stellar* masses as low as $10^8 - 10^9 M_\odot$; Leauthaud et al. 2019), and this is likely possible with currently-running surveys such as the Hyper Suprime Cam survey (HSC; Aihara et al. 2018a,b). Since the weak lensing signal depends on mass of the target object, and we are trying to study the lowest mass target objects ever, we will need to "stack" our analysis of these galaxies, and we will need to stack *many* galaxies to get a usable signal. HSC is expected to identify $\sim 10^9$ galaxies, and roughly 10^6 of them will be nearby dwarf galaxies ($z < 0.15$; $10^8 < M_\star/M_\odot < 10^9$)—but how will we find these diamonds in the rough?

In chapter 7 we start with the basic test: how well can we identify these dwarf galaxies using only data from the HSC collaboration and relatively "off-the-shelf" machine learning tools? Since that initial test will leave something to be desired, in chapter 8 we implement a cutting-edge machine learning tool to help artificially augment our training sample size. chapter 9 takes a more typical astronomy approach; in that chapter we discuss plans to create a potential narrowband H_α survey crafted to identify these galaxies. In that survey, we expect some contamination from higher redshift galaxies with different emission lines that happen to fall within the same observed wavelength window, so we test how well machine learning methods can distinguish the

target population from the contaminant population. In chapter 10 we give some advice to researchers seeking to use machine learning, to researchers interested in some ideas of creating original machine learning designs, and for students seeking to get machine learning jobs in industry.

Chapter 7

Initial Test

7.1 Introduction

In order to identify dwarf galaxies within the HSC survey, we need some model of what these dwarf galaxies should "look" like (at least in reduced photometric quantities, but ideally also in their morphology on the sky). One possibility is building a galaxy formation and evolution model from first principles. Unfortunately that takes a significant amount of time—more time that I was able to fit around Part I of this thesis. So instead we turned to machine learning models which could be "trained" using data from known, labelled galaxies and then applied to the rest of the HSC survey. This known, labelled data will come from galaxies in the COSMOS field, where we can match HSC objects to the nearest labelled galaxy from previous studies (if any exist within 1 arcsec of the HSC object). Fortunately, this labelling is pretty complete within the COSMOS field; almost every HSC galaxy in this field can be matched to a relatively

precise stellar mass and redshift[1].

The size of the HSC survey complicates things. Although we can download images for every galaxy within the COSMOS field, that will not be feasible for the entire survey which is expected to be > 100 TB in disk storage. Instead, we *can* download and store all the *reduced* photometry for every object (mostly just the magnitudes of the 5 broad bands, *grizy*). Then, we can develop a 2-stage classifier that makes an initial pass using just the reduced photometric data, and then downloads the top $\sim 1\%$ of images and constructs a "second-opinion" using a Convolutional Neural Network (CNN) applied to the images, ideally gleaning additional information not captured by the overall object magnitudes.

At this point, it is good to point out we *do not know* that the images will provide significant useful information beyond what is already contained in the overall magnitudes. The targets we are trying to infer—redshift and stellar mass—are not something that a human could accurately estimate by-eye just from 5 broad band images, but the neural networks we plan to use were developed to emulate classifications made visually by humans. This has been a problem in past works; for example Hoyle (2016) find that while deep neural networks using SDSS galaxies images can predict the galaxy redshift, a Random Forest model using *no* morphological information can do just as well. Still, there is a possibility of hope: the HSC survey is able to reach unprecedented surface brightness limits for a survey of its size, showing much more faint

[1]These stellar mass and redshift estimates are technically only *photometric* estimates, not full spectral measurements. However, the estimates were made using images through ~ 30 filters of which ~ 15 were narrowband filters. Even though this is not a true spectrum, it is a relatively high resolution spectral energy distribution, which leaves us confident in the accuracy of these estimates.

structures in galaxies than previous surveys. Therefore, there might be more information in HSC images than images from previous surveys which were found to not add valuable morphological information.

7.2 Data; Features and Targets

We are using data primary from two sources:

1. The HSC survey, outlined by Aihara et al. (2018a,b). The publicly released data for this survey includes both images as well as "reduced" quantities (such as object magnitudes and photometric redshift estimates (Tanaka et al. 2018)). There are multiple subsets of the HSC survey; we in particular are looking at the "wide" layer; this layer gives us access to the most number of galaxies, but means we will not always have access to useful information like narrow band fluxes.

2. The COSMOS2015 catalog from Laigle et al. (2016). This will be treated as our "truth" table. For this work we are only interested in reduced physical parameters, in particular the inferred redshift and stellar mass (along with the sky coordinates of each object for matching against the HSC survey catalog) and not the underlying photometric flux measurements.

Typically our *features* (denoted $\boldsymbol{X}$) are going to be from the HSC survey. This is because we want to be able to apply our model across the entire HSC survey, which means incorporating no outside data.

Typically our *targets* (denoted t) are from the COSMOS2015 dataset. This is

because we want to add something of value to the HSC data; it (generally) would not be worthwhile to be trying to predict something that the HSC survey can extract, since they would have extracted it in whatever fields were possible.

To be more specific, unless stated otherwise, our features are:

- i mag (cmodel – a composite of exponential and de Vaucouleurs model fits)
- $g-r$, $r-i$, $i-z$, $z-y$ colors (cmodel as well)
- `FRANKENZ` `photoz_best` (best-guess photometric redshift, *photo-z*) and `photoz_risk_best` (the statistical risk that `photoz_best` is outside the range $z_{\text{true}} \pm 0.15(1+z_{\text{true}})$)
- g, r, i, z, y images. We download a $20'' \times 20''$ image from the deep layer[2] with preprocessing that will be described in subsection 7.2.1.

Our specific target is the following binary target:

- $t = (8 < \log_{10} M_\star/M_\odot < 9)$ AND $(z < .15)$

7.2.1 Image preprocessing

Before even opening the images, we remove any objects with COSMOS2015-estimated photo-zs outside the range $(0, 8)$, any objects missing the reduced cmodel magnitude for any of its *grizy* photometric bands, and any objects which raised flags during the reduction of those photometric fluxes. When training and testing the CNN,

[2]For this test we use images from the HSC *deep* layer, even though we are primarily interested in applying this technique to the *wide* layer. The reason is simple: wide-layer images were not yet ready for the COSMOS field, although they will be released in future HSC public data releases.

we remove any objects for which we were unable to download all 5 bands (which can happen even if all 5 bands of reduced magnitudes are in the database).

The first preprocessing step for the images is to apply a pixel-wise arcsinh scaling (Lupton et al. 1999) using the HSC survey-recommended parameters[3]. This helps the CNN deal with the large dynamic range of intensities in astronomical images. We generally do not apply any additional re-normalization, although this could also be accomplished within the CNN by adding an initial batch normalization layer as the first layer.

Next, if this image is being used for training, we perform data augmentation that will be described in subsection 7.3.2. In general, this typically means random affine transforms (such as rotations, reflections and translations to the image).

Then, we crop the image down to about half its original size (i.e., to $(10'')^2 = (75\text{px})^2$ unless otherwise noted). It is helpful to do this *after* the data augmentation; if you crop before data augmentation, then apply an affine transform, you typically will need to prescribe how to "fill" the values that now appear at the edges of your transformed image which did not exist in the original, cropped form. By performing the transform on a large enough image, *and then* cropping the image to the desired size, none of the remaining pixels will have been affected by the artificial "fill" prescription.

[3] https://hsc-gitlab.mtk.nao.ac.jp/snippets/26; unfortunately there appears to be no version-controlled permalink

7.3 Methods

We take a two step approach:

1. A Random Forest classifier (*RF*; Breiman 2001) applied to expert-extracted features. This allows us to cut out most of the clearly-unwanted galaxies.

2. A convolutional neural network (*CNN*; e.g., LeCun et al. 1989) on the best candidates identified by the Random Forest to provide a second opinion, hopefully strengthening our discriminative power

7.3.1 Random Forest (RF)

This effectively provides a non-parametric classifier. It is an ensemble of many (in our case 1000) decision trees which are each constructed with a random subset of the training data. Each decision tree then partitions the feature space (with each branching allowed to come from a different random subsample of features) until only one training example remains at each leaf of the tree. When applied to new data, each new example follows the tree, and the tree predicts with the class of the training example at the leaf; the ensembled prediction is then simply the average of the individual tree predictions.

In practice we used 10-fold cross validation to measure the performance of our network and maximize the possible training set size for the neural network. To do this, we partitioned the entire dataset into 10 equal sized partitions, and then trained 10 RFs each of which used 9 partitions for its training data and 1 partition for its validation data. This ensured that *every* galaxy had a RF predicted probability and that the

galaxy was not in the training set of the RF which made the prediction. In production we plan to apply all 10 RFs to each new galaxy, and average the probabilities predicted by each forest, effectively creating 1 RF of 10,000 trees.

The final step is to "soften" the predictions of the RF. The standard RF algorithm allows for predicted probabilities of exactly 0 or 1 (if *all* trees agree on the classification), but there is always *some* chance that its predictions are incorrect. To avoid these overly-confident predictions, we add pseudo-trees (analogous to the pseudo-observations from a Bayesian conjugate pair like the Beta-Binomial pair), effectively creating 1 additional tree that always predicts $p = 0$ and 1 additional tree that always predicts $p = 1$:

$$p_{\mathrm{RF,softened}} = \frac{p_{\mathrm{RF}} \times N_{\mathrm{trees}} + 1}{N_{\mathrm{trees}} + 2}. \tag{7.1}$$

When applied to a new set of galaxies in production, we apply 2 possible outcomes depending on the reported probability for a given galaxy within the production set:

1. If the galaxy is one of the best 1000 per sq. deg. candidates, we download its image and feed it through a deep neural network to get a second-opinion on the data, in hopes of increasing the final purity of the sample

2. If it is not within the top 1000 per sq. deg. of candidates, label the galaxy to be the undesired class.

The exact cutoff, i.e. the best 1000 per sq. deg., is somewhat arbitrary. We choose it since it is a relatively round number, gave reasonably high completeness values,

and does not require downloading an unreasonably large number of galaxy images.

7.3.1.1 Possible Improvements

The results will show that our RF probabilities are pretty accurate; if you applied the Random Forest to a held-out training set and bin by predicted class probability, bins of higher predicted probability also have higher purity (and predicted probability $\approx$ actual purity). Therefore if you wanted to optimize for the best completeness and could tolerate a certain level of impurity, you could automatically accept galaxies above a certain predicted probability, and then have the RF focus on the most *uncertain* candidates (not the *most likely*) candidates.

In practice we chose not to do this, since it is not clear what this auto-accept purity threshold should be. Even if you had a *desired, final* purity, it is not a priori clear how much the neural network will affect purity, so it's possible the auto-acceptance threshold would have to be treated as an unknown hyperparameter, and for each different value the neural network would have to be retrained.

7.3.1.2 Alternatives?

We tested two alternative (non-deep learning) methods, which did worse than the Random Forest classifier; we will briefly mention them here.

The first was a simple i magnitude cut. This effectively assumes that the rank ordering of our galaxies i band magnitude matches the rank order of their probability of matching our target classification. This might make sense if our target was just a

redshift cut or just a mass cut; unfortunately looking for nearby *and* low mass galaxies with this approach is not likely to be optimal.

A generalization of the 1-dimensional cut is *logistic regression*. Instead of just 1 feature dimension, we also included the 4 colors and the 2 photo-z related features. Then logistic regression constructs a linear model predicting the log odds of a new example being the target class ($\boldsymbol{\beta X} = \log p/(1-p)$ for features $\boldsymbol{X}$, parameters $\boldsymbol{\beta}$ and predicted probability p). Just like the 1-dimensional cut, this assumes there exists a direction along which the rank ordering of examples matches the rank ordering of their probabilities (and now with an additional assumption that distance along that direction is meaningfully connected to the probability).

Note that both of these models make a *parametric* assumption about our model. This linear parameterization is certainly not exactly true, given that we are trying to create an inverse model of photometry to galaxy redshift and mass. Furthermore, it is not clear that it will be a good *approximate* description of that mapping.

7.3.2 Convolutional Neural Network

We created a fairly simple deep convolutional neural network (CNN), based off a prior architecture by Huertas-Company et al. (2015). It has 3 convolutional blocks each consisting of:

- a convolutional layer (with each kernel twice the size of the previous block in each spatial dimension, but always 16 filters)

- RELU activation
- max pooling (2x2)
- dropout (only for training; typically using a dropout fraction of .5)

It then finishes off with 3 fully-connected layers; there is RELU activation after the first two, and a logistic sigmoid activation applied to the output of the final fully-connected layer, transforming the output from a log odds to a probability.

We then train the network using the Adam optimizer (Kingma & Ba 2014), which tries to decrease the binary cross entropy over the training set.

7.3.3 Combining Models

Hopefully our RF model and our CNN will provide complementary information about whether a given object is the desired galaxy type or not. This means we ideally want to combine the outputs from both models when making our final selection; if we only use the CNN-reported probability, we lose any marginal information contained in the RF-reported probability. In order to combine these models, we will assume the reported probabilities are independent of each other, conditional on the true class of the object; in other words we will assume that what has been "learned" by each model is independent. The rest of this subsection discusses how we did this in depth.

First, we have mentioned our *assumption* that the RF features, $\boldsymbol{X}_{\mathrm{RF}}$, and the

CNN features, $\boldsymbol{X}_{\rm CNN}$, are independent conditional on t, i.e.:

$$p(\boldsymbol{X}_{\rm RF}, \boldsymbol{X}_{\rm CNN}|t) = p(\boldsymbol{X}_{\rm RF}|t)p(\boldsymbol{X}_{\rm CNN}|t). \tag{7.2}$$

This is driven by convenience, not principle. While we could try to enforce aspects of this, for example by normalizing the CNN input images so that they do not contain direct information on the overall i magnitude, that still does not rule out implicit correlations. And trying to model the dependence between the two types of inputs goes well beyond the scope of this project. There are empirical ways to calibrate the effect of this unhandled dependence (such as bagging and boosting), but they typically require an additional layer of held out, CNN-worthy validation data, of which we are already in extremely short supply (or significantly larger computational costs). So we will simply *assume* that the reduced photometric features are independent of the pixel-level information.

Given conditional independence (Equation 7.2), we can construct a likelihood ratio that is simply the product of each model's likelihood ratio:

$$\frac{p(\boldsymbol{X}_{\rm RF}, \boldsymbol{X}_{\rm CNN}|t=1)}{p(\boldsymbol{X}_{\rm RF}, \boldsymbol{X}_{\rm CNN}|t=0)} = \frac{p(\boldsymbol{X}_{\rm RF}|t=1)}{p(\boldsymbol{X}_{\rm RF}|t=0)}\frac{p(\boldsymbol{X}_{\rm CNN}|t=1)}{p(\boldsymbol{X}_{\rm CNN}|t=0)}. \tag{7.3}$$

But neither of our models directly reports the log likelihood ratio; instead we must infer it using Bayesian methods. For a generic discriminative model, under Bayes

rule the output posterior class probability is:

$$p(t|\boldsymbol{X}) = \frac{p(t)p(\boldsymbol{X}|t)}{p(\boldsymbol{X})} \tag{7.4}$$

so the log likelihood ratio is:

$$\frac{p(\boldsymbol{X}|t=1)}{p(\boldsymbol{X}|t=0)} = \left(\frac{p(t=1)}{p(t=0)}\right)^{-1} \frac{p(t=1|\boldsymbol{X})}{p(t=0|\boldsymbol{X})} \tag{7.5}$$

We do not necessarily know how to decompose the prior from the posterior, but it is fairly common to assume $p(t=1) = N^{-1}\sum_i t_i$. This is not truly Bayesian since it depends on the observed class balance, but we *could* construct a hierarchical model where we assume the number of target galaxies within a field of fixed number of total galaxies is binomial distributed and use a Beta prior for the class balance. That effectively takes the $p(t=1)$ estimator for our discriminative model, and shrinks it towards a prescribed value (typical 1/2 for a weakly informative prior). But if the prior is truly *weakly* informative it should have little-to-no impact when our sample sets are in the thousands (for the CNN) or hundreds of thousands (for the RF). So we will simply assume $p(t=1) = N^{-1}\sum_i t_i$ throughout.

That is all the theoretical justification we need. We are not doing anything special beyond this point; we are simply *assuming* conditional independence in our models' input features, and we are *assuming* we can decompose our model class priors from the output posterior by using the sample mean class balance of our training sets.

Now explicitly show how this combining process is done, given the assumptions above. First it is useful to define the logistic transform:

$$\text{logit}(p) = \log\left(\frac{p}{1-p}\right) \tag{7.6}$$

i.e. the log odds, and its inverse:

$$\text{expit}(x) = \frac{e^x}{1+e^x} = \frac{1}{1+e^{-x}}. \tag{7.7}$$

Using these definitions, along with the decomposition of Equation 7.5, we can see:

$$\log\frac{p(\boldsymbol{X}|t=1)}{p(\boldsymbol{X}|t=0)} = \text{logit}(p_{\text{posterior}}) - \text{logit}(p_{\text{prior}}) \tag{7.8}$$

and if we want to combine the information from multiple models using *independent* features:

$$\begin{aligned}\log\frac{p(\boldsymbol{X}_1,\boldsymbol{X}_2|t=1)}{p(\boldsymbol{X}_1,\boldsymbol{X}_2|t=0)} = &[\text{logit}(p_{\text{posterior},1}) - \text{logit}(p_{\text{prior},1})] \\ &+ [\text{logit}(p_{\text{posterior},2}) - \text{logit}(p_{\text{prior},2})]\end{aligned} \tag{7.9}$$

One can see that this has a desirable quality: if one model does not update the posterior for a given object (e.g., a galaxy is not passed through the CNN) then that model's logistic terms cancel and we recover the single model form (Equation 7.8).

Finally, to get back to an overall posterior probability, we then use our assumed

class prior:

$$p_{\text{combined}} \equiv p(t=1|\boldsymbol{X}_1, \boldsymbol{X}_2) \tag{7.10}$$

$$= \text{expit}\left(\log \frac{p(t=1|\boldsymbol{X}_1, \boldsymbol{X}_2)}{1-p(t=1|\boldsymbol{X}_1, \boldsymbol{X}_2)}\right) \tag{7.11}$$

$$= \text{expit}\left(\log \frac{p(t=1|\boldsymbol{X}_1, \boldsymbol{X}_2)}{p(t=0|\boldsymbol{X}_1, \boldsymbol{X}_2)}\right) \tag{7.12}$$

$$= \text{expit}\left(\log \frac{p(t=1)}{p(t=0)} \frac{p(\boldsymbol{X}_1, \boldsymbol{X}_2|t=1)}{p(\boldsymbol{X}_1, \boldsymbol{X}_2|t=0)}\right) \tag{7.13}$$

$$p_{\text{combined}} = \text{expit}\left(\text{logit}(p_{\text{prior,overall}}) + \sum_{i\in\{1,2\}} \text{logit}(p_{\text{posterior,i}}) - \text{logit}(p_{\text{prior,i}})\right) \tag{7.14}$$

7.3.4 Loss Functions and Metrics

In order to compare results between models, we need some kind of *loss function*, which we would like to minimize.

Binary Cross-Entropy The first measure we will use is *binary cross-entropy*, a common metric from information theory:

$$H(t,p) = -E_t[-\log p] \tag{7.15}$$

$$= -\frac{1}{N}\sum_{i}^{N} t_i \log p_i + (1-t_i)\log(1-p_i) \tag{7.16}$$

for predicted probabilities, p and true binary target labels t. This is a convenient measure for a few reasons:

1. maximizing the binary cross entropy is equivalent to maximizing the likelihood of

a model's ability to predict categorical labels

2. this measure rewards models which can both make accurate predictions, *as well as* predictions with accurate uncertainties

3. binary cross entropy is differentiable with respect to the predicted probability (whereas thresholded metrics such as accuracy are not), allowing the possibility for gradient-informed optimization techniques.

Completeness / Recall Completeness (also called recall within data science and machine learning circles) is pretty simple. *Given a probability threshold*, $p_{\text{threshold}}$ completeness is then:

$$\text{completeness} = \frac{\sum_i \mathbb{1}_{t_i=1} \times \mathbb{1}_{p_i > p_{\text{threshold}}}}{\sum_i \mathbb{1}_{t_i=1}} \tag{7.17}$$

where $\mathbb{1}$ is an indicator function (1 if the subscripted condition is true, 0 otherwise), and the subscript i refers to different samples which your model was not trained on.

Purity / Precision Similarly purity (more commonly called precision within data science and machine learning fields) just changes the denominator:

$$\text{purity} = \frac{\sum_i \mathbb{1}_{t_i=1} \times \mathbb{1}_{p_i > p_{\text{threshold}}}}{\sum_i \mathbb{1}_{p_i > p_{\text{threshold}}}}. \tag{7.18}$$

The expressions above are defined for a single threshold, but we could imagine incrementally increasing that threshold, and keeping track of how precision and recall (or purity and completeness) evolve. This results in a parametric curve, called the

Precision-Recall curve.

True Positive Rate / False Positive Rate Related are the true and false positive rates. The true positive rate is actually just the completeness/recall:

$$\text{True Positive Rate} = \text{completeness} = \frac{\sum_i \mathbb{1}_{t_i=1} \times \mathbb{1}_{p_i > p_{\text{threshold}}}}{\sum_i \mathbb{1}_{t_i=1}} \tag{7.19}$$

while the false positive rate is new:

$$\text{False Positive Rate} = \frac{\sum_i \mathbb{1}_{t_i=0} \times \mathbb{1}_{p_i > p_{\text{threshold}}}}{\sum_i \mathbb{1}_{t_i=0}} \tag{7.20}$$

Just as above, we can create a curve of true positive rates and false positive rates, parameterized by $p_{\text{threshold}}$. This results in a *receiver operating characteristic* curve (typically just abbreviated as a *ROC* curve)[4].

By defining these two curves (the Precision-Recall curve and the ROC curve), we can explore the results of a particular model by only training it *once*, but then considering a whole range of possible thresholds.

Areas under the curve (AUCs) When *comparing* multiple models, having access to a full curve (either Precision-Recall or ROC) is often too much information; we want a single value that defines the "power" of the classifying model. One category of metrics that attempt this is areas under the curve (AUCs). This is simply the integral under the Precision-Recall curve (the PR-AUC) or the area under the ROC curve (the ROC-

[4]For a good introduction to ROC curves and their many properties, see Fawcett (2006).

model	cross entropy	PR AUC	ROC AUC
RF	0.0068	0.47	0.972
i-mag ordering	0.0119	0.03	0.952
LR	0.0099	0.11	0.950

Table 7.1: A summary of our metrics for the traditional machine learning approaches which only used reduced photometric data (not images themselves). The probability for each galaxy comes from the cross-validation iteration in which that galaxy was in the held-out validation set; this gives us a held-out probability for every galaxy, and we compute these metrics using all of the galaxies. Although the i magnitude ordering simply creates a rank ordering, we can estimate probabilities with a 1-feature logistic regression model which preserves rank ordering.

AUC). For both, a higher AUC corresponds to a generally more powerful model. We will show the AUCs in a few plots as a diagnostic, but *we will not use it for our ultimate model selection and evaluation.* This is because these AUCs are not as useful as our other options. Binary cross-entropy tests if probabilities are well calibrated while AUCs only care about the *rank ordering* of the probabilities. Also AUCs are designed for situations where you *do not have a way to choose a specific threshold*; in cases where we can, it is better to just look at the specific completeness and purity at that threshold.

7.4 Initial results

7.4.1 Random Forest Results

We begin by comparing the performance of the Random Forest model against the alternatives (a logistic regression model using the same features, as well as a simple i-band magnitude threshold as a classifier) in Figures 7.1 and Figure 7.2, with a summary of the metrics given in Table 7.1. Clearly the Random Forest outperforms the competing alternatives; for that reason, we will only focus on the Random Forest model from now

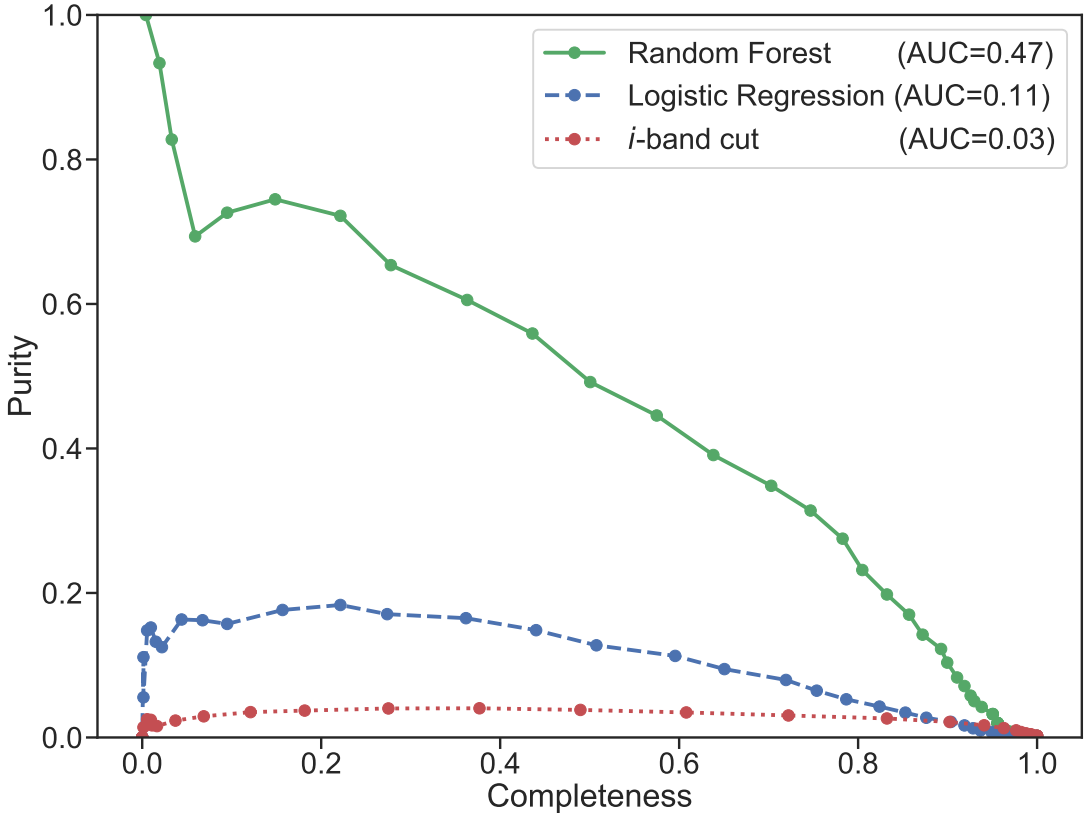

Figure 7.1: A comparison of the traditional classifiers which use only reduced photometric information. The Random Forest model clearly dominates the others at basically any fixed completeness or purity level. (This plot is equivalent to a precision-recall plot that is more common outside of astronomy.)

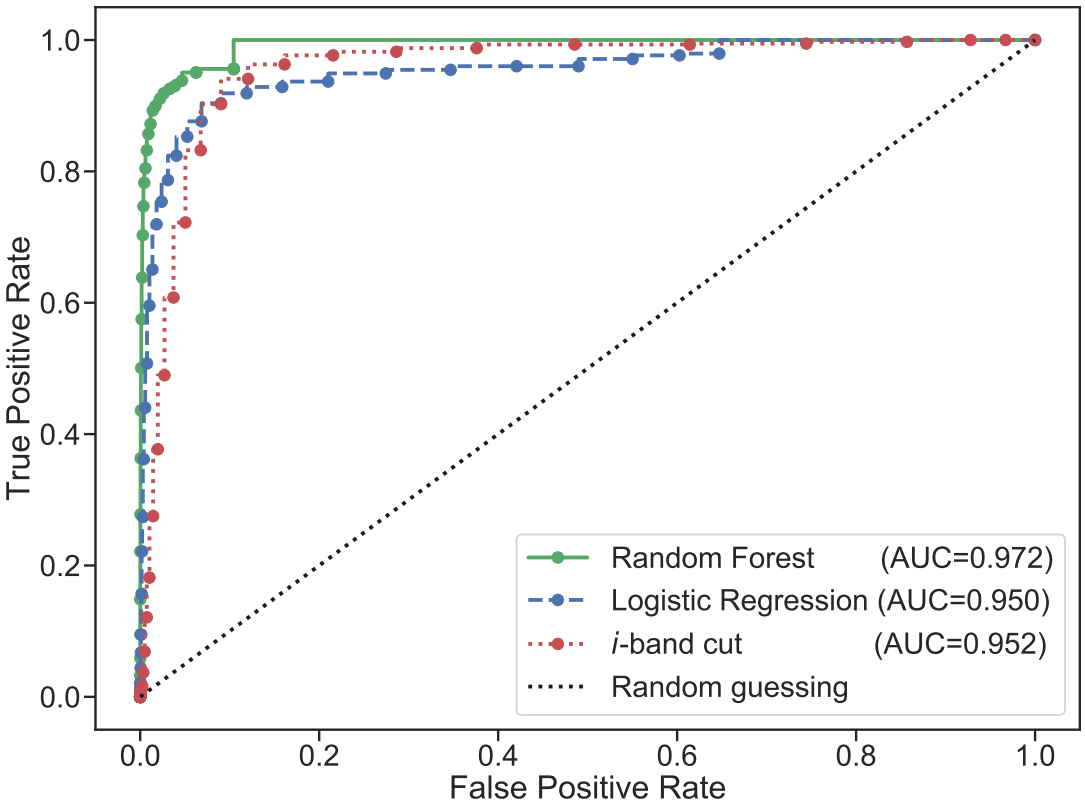

Figure 7.2: Same as Figure 7.1, except now showing the ROC curve.

on.

While that purity-completeness curve is a convenient way to compare models, it is not the best way to explore the results from a single model. That is because we do not plan to use a fixed purity or completeness goal for choosing the threshold that defines whether a galaxy is "selected" or not by the Random Forest. Instead, we are driven by more practical limitations: we can only follow up (with a CNN or otherwise) a certain number of objects.

Since a purity-completeness curve is simply a sequence of increasing more stringent selection thresholds, the number of "selected" objects changes monotonically along the purity-completeness curve. This means purity and completeness can be separately parameterized as a function of the number of selected objects. Figure 7.3 shows this visually.

Splitting the purity-completeness curve into two plots is very useful. Now if we are considering a new follow up method capable of following a certain density of objects, we can quickly look up how pure and complete the Random Forest selection would be, without having to retrain anything.

In practice we choose to set a threshold corresponding to 10^3 selected objects per sq. deg. This is large enough to theoretically allow every true dwarf to be selected (if somehow our Random Forest could be improved to be optimal), while still being small enough to be reasonable to process with the deep learning classifier[5]. This threshold is denoted in Figure 7.3 with the vertical dashed line.

[5]This would correspond to roughly 10^6 selected galaxies across the entire HSC Wide layer, which would take ~ 1 TB to store.

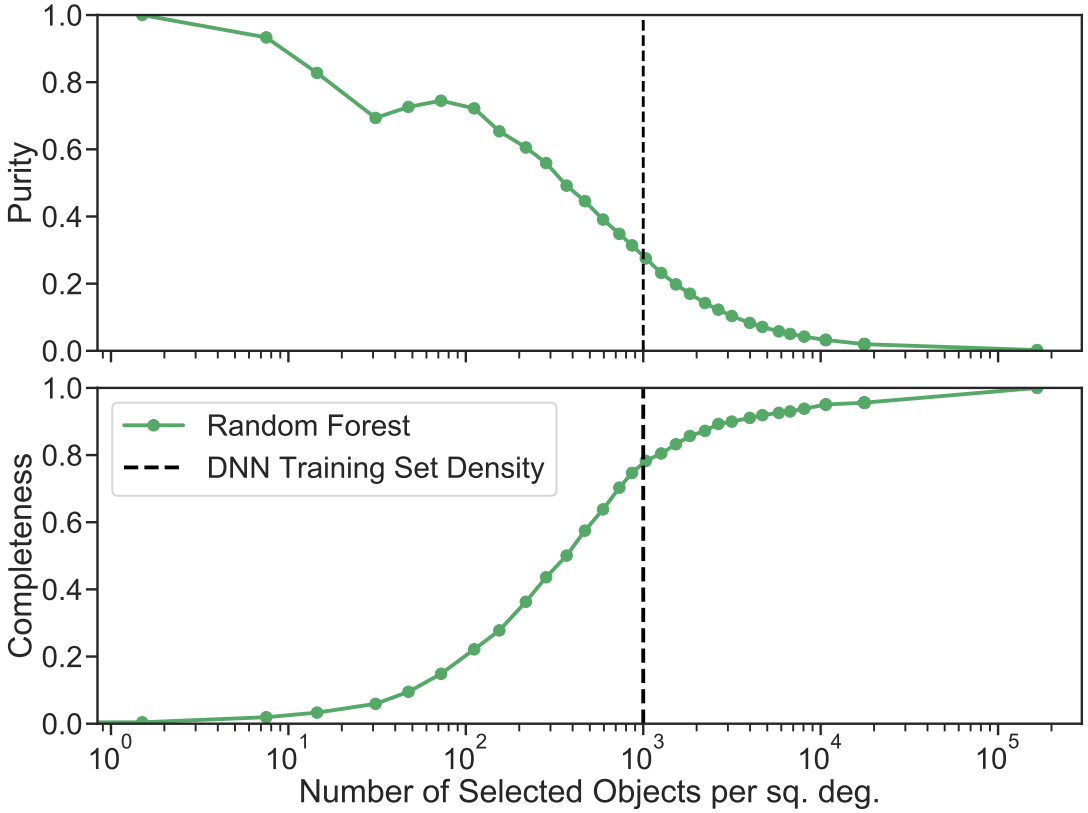

Figure 7.3: The purity-completeness curve of the Random Forest classifier, now parameterized by the number of selected objects at each threshold. This was trained over a 2 sq. deg. area.

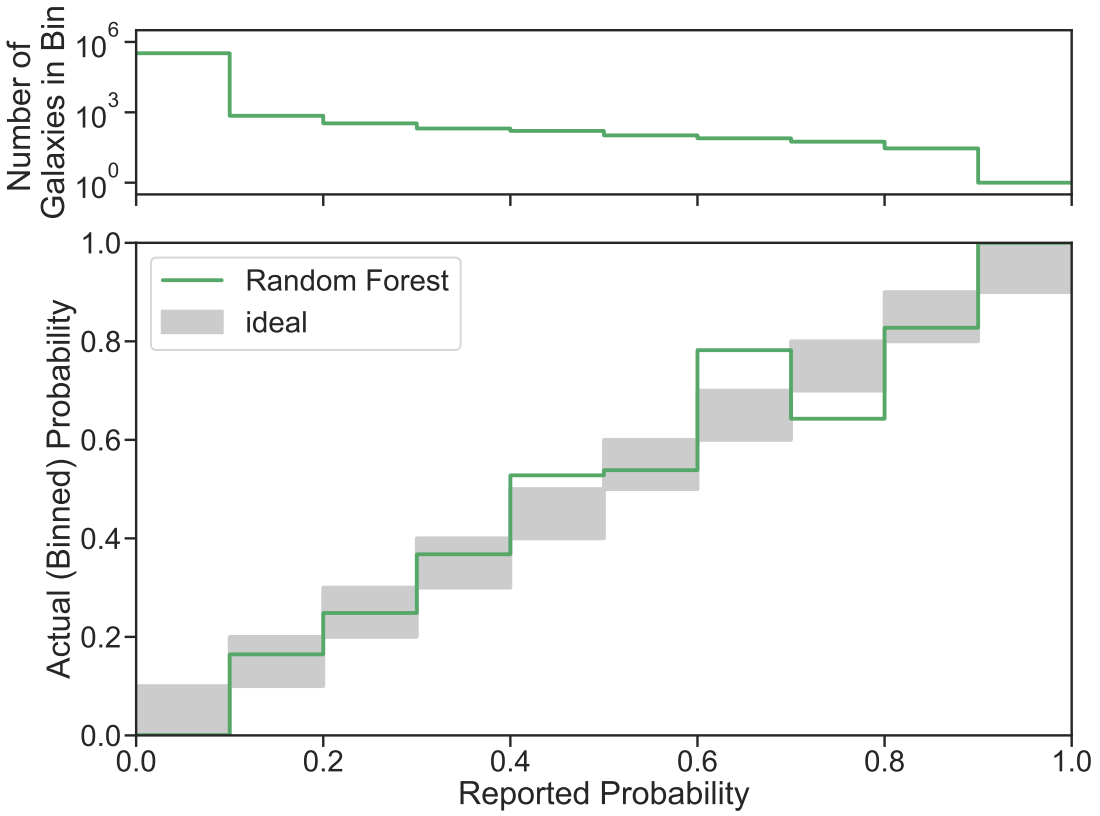

Figure 7.4: A visualization of how well calibrated the Random Forest model's predictions are. A perfectly calibrated model would follow the "ideal" range on the lower panel; a perfect model would also separate the distribution in the top panel into two delta functions at 0 and 1. The mean binary cross entropy is a measure of how well a model has done *both*.

Finally, it will be useful for us to consider a few other diagnostic plots. In particular, I will look at how accurately the predicted probabilities reflect the uncertainty of the model. In Figure 7.4, I first bin validation galaxies by their predicted probabilities, then look at the purity within that bin (i.e., the actual probability that any member matches our target).

Overall the probabilities look reasonably well calibrated, which is good. This means that even if our model is not able to definitively classify an object, it can at least indicate its uncertainty. This also leaves open the possibility that we might be able to take the objects with the highest predicted probabilities and accept them as-is, without any follow-up. So far we have chosen to follow up all objects above a simple probability threshold (even the most-confident matches), but we note that this is a possible way to improve the efficiency of the follow-up process.

When we measure the binary cross entropy of our cross-validated Random Forest model, we find a value of 0.006868.

7.4.2 Convolutional Neural Network Results

We will look at the CNN results in two passes: first we will look at its performance by itself (only look at objects which were selected for follow up), then we will look at how it does *in combination* with the Random Forest.

In order to understand how the CNN is performing, the first diagnostic to look at is the learning curve, shown in Figure 7.5. There we see the typical behavior: the training loss is almost always lower than the validation loss, and although the training

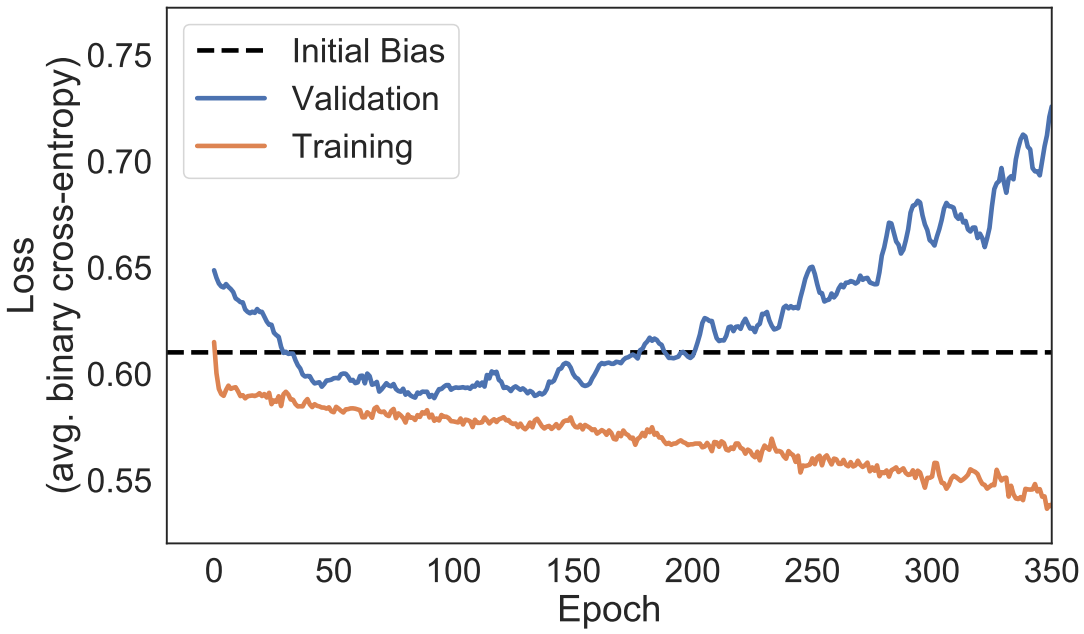

Figure 7.5: The learning curve of the CNN's performance, comparing the loss over the training set and the loss of the held-out validation set. (Both curves have been smoothed using a 5-element box-car kernel to help see the underlying trend.) The horizontal "initial bias" line shows how well we would do if we only knew what fraction of the labeled training set was the desired galaxy type, and used that fraction as the predicted probability for every galaxy in the held out testing set; this is what we would expect if the CNN was not able to learn *anything* from the images.

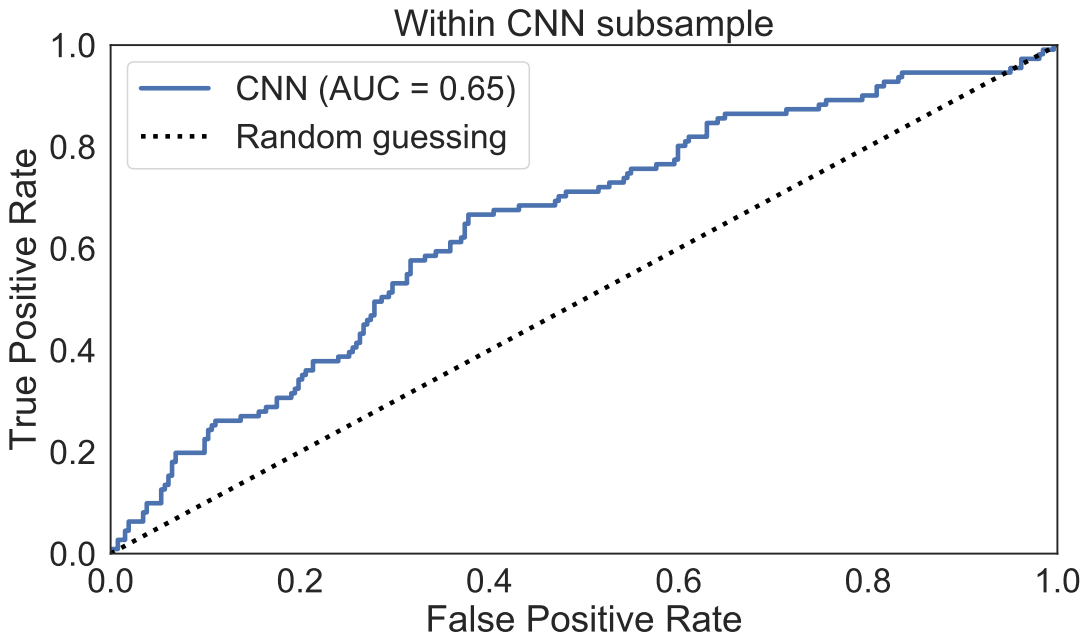

Figure 7.6: The ROC curve for the trained CNN, evaluated for just the validation samples in our RF-selected subsample.

loss continues to decrease, the validation loss either plateaus or starts to rise again at late times (indicating some level of overfitting). For the rest of the results, we will use the state of the neural network after 200 epochs of training.

The training curve is the first indication that this model *is* learning useful, discriminative features within the images, but still it is not quite as powerful as we would like.

7.4.3 Combined Results

Here we combine the results from Sections 7.4.1 and 7.4.2 using the method described in subsection 7.3.3 (i.e. Equation 7.14).

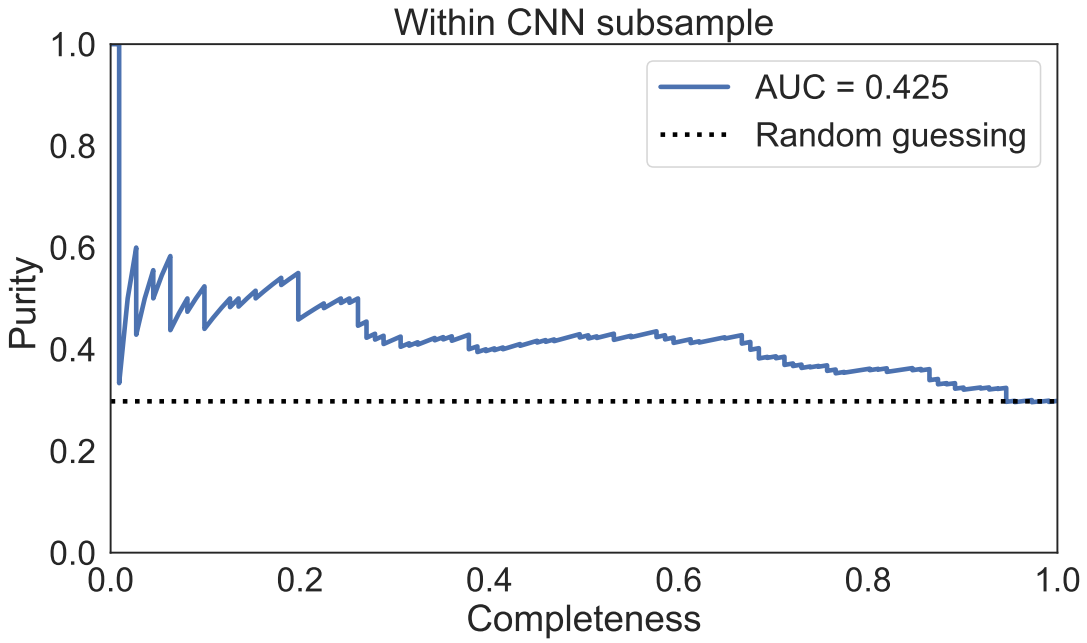

Figure 7.7: The completeness-purity curve for the trained CNN, evaluated for just the validation samples in our RF-selected subsample.

model	cross entropy	PR AUC	ROC AUC
RF	0.00702	0.4822	0.97277
RF + CNN	0.00704	0.4816	0.97274

Table 7.2: A comparison of the metrics for the RF-only model and the model which combines RF and CNN predicted probabilities. Note that the RF-only values do not exactly match those in Table 7.1. This is because we did not create a cross-validated version of the CNN, and therefore we do not have held-out CNN-predicted probabilities for all of the best candidates—those only exist for about 20% (randomly selected) of the best candidate population. These galaxies were upweighted by a factor of 5 (compared to the non-best candidate population) to account for this, causing a minor difference between these values and those in Table 7.1 which used equally-weighted probabilities.

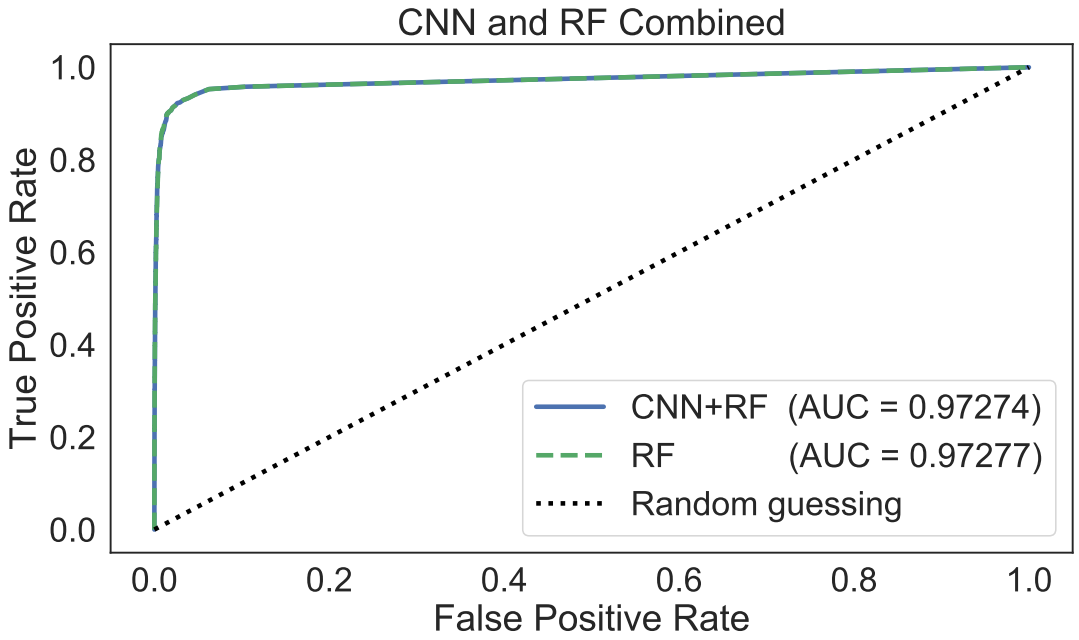

Figure 7.8: Same as Figure 7.2 and Figure 7.6, except now showing the ROC curve for the "combined" (CNN and RF) prediction versus the purely RF prediction. The curves are so close they are basically identical.

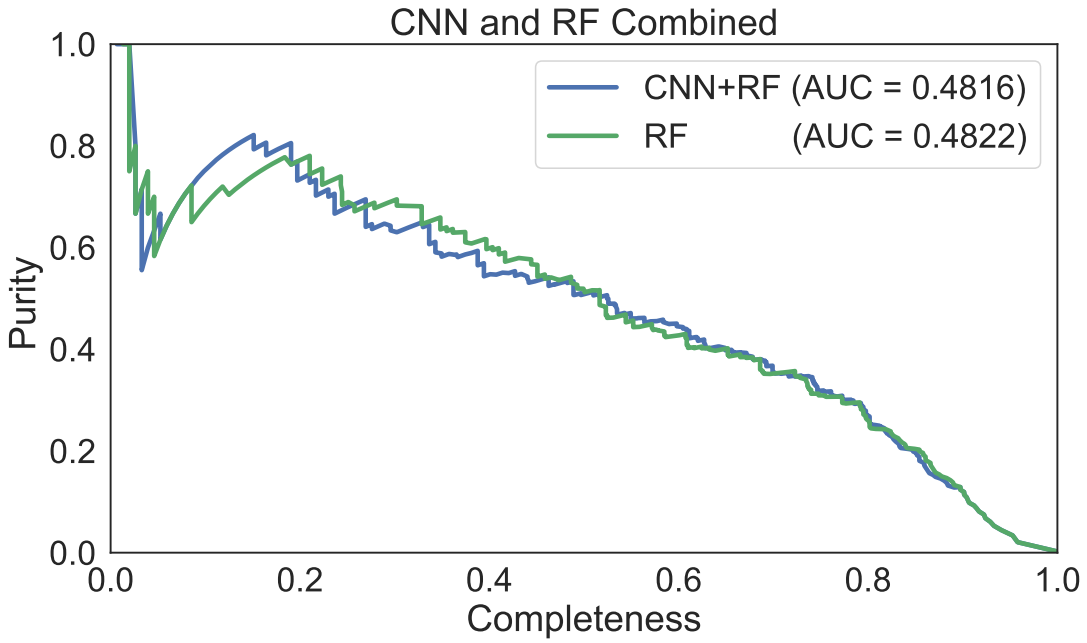

Figure 7.9: Same as Figure 7.1 and Figure 7.7 except now showing the ROC curve for the "combined" (CNN and RF) prediction versus the purely RF prediction. The "random guessing" line has been omitted, as it would be down at purity=0.003.

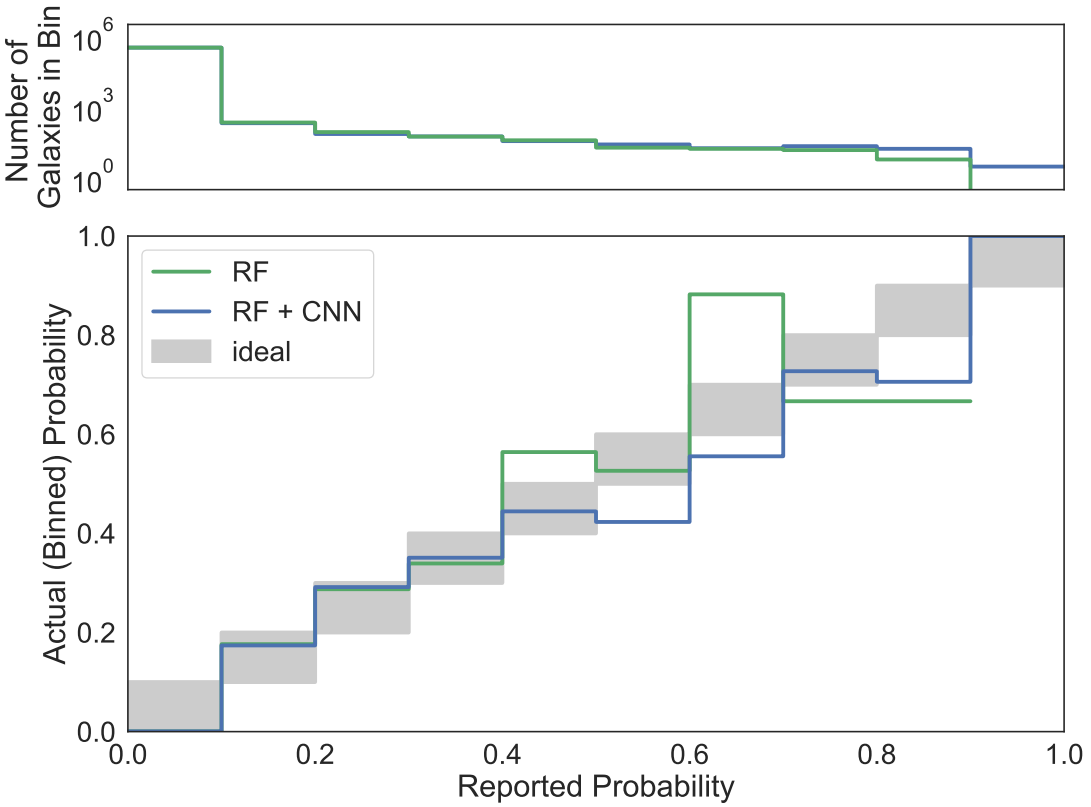

Figure 7.10: Same as Figure 7.4 except now comparing the probability calibration for the "combined" (CNN and RF) prediction along with the purely RF prediction.

When we do this, we can run through the same metrics as before. (Note: for this step, since we do not have cross-validated CNN probabilities, we will not include the 80% of the randomly selected training set. To compensate for this, we will up-weight the CNN validation set by a factor of 5). These metrics are shown in Table 7.2.

The results are shown in Figure 7.8 for the ROC curve, Figure 7.9 for the completeness-purity curve and Figure 7.10 for the probability calibration diagnostic plot. The ROC curve shows basically no substantial difference, which is unsurprising for such a class-imbalanced problem. The completeness-purity curve on the other hand does show improvement for *some* thresholds. Unfortunately this improvement is at relatively low completenesses, while we really would have preferred to see the largest gains at high completenesses ($\gtrsim 80\%$). If we were to accept a final classification with very low completenesses, it might be that we were choosing a relatively biased set of galaxies; at high completeness, we can virtually ensure that we have as representative of a sample as possible.

7.5 Conclusions

1. In every standard metric (ROC AUC, PR AUC, cross entropy), the CNN does not add appreciably compared to just using the RF. Part of this is expected, since by design, most objects are never passed through the CNN, so some of the metrics will be dominated by galaxies which only have an RF score (and are thus unchanged by adding the CNN). Still, for some metrics (especially the binary cross

entropy), the contribution from RF-only objects is identical, so any difference *must* be driven galaxies which have been passed through the CNN. Unfortunately, we see a net *decrease* in performance as measured by the cross entropy. That means that *worsening* would be even more clear if we only looked at the galaxies which had been passed through the CNN (in particular, we would see a difference in the cross entropy of ≈ 0.02 rather than 10^{-5} as reported in Table 7.2).

2. Still, many of the metrics above (specifically the AUCs) are for a *range* of final probability thresholds. In reality, we will ultimately just care about a single threshold. So even though *on average* the combined model might be worse, there are certain regions where the combined model is better (see the purity-completeness curve comparison in Figure 7.9). Unfortunately, the only locations that show significant lift tend to be at very low completeness and very high purity. This region is not great for our science, as it leaves open the possibility that the final selection is strongly biased, but the general uninterpretability of neural networks means we cannot easily understand the shape of the selection function. For example, a hypothetical neural network that could perfectly identify cuspy dwarf galaxies (but not cored galaxies) would have a region of 100% purity, but the network would be worthless for a stacked lensing analysis, because it would not represent the true dwarf population and we would not know how to re-weight or correct for this biased selection.

So it appears that the combination of our data, our CNN architecture and our

training method is too limited to add any useful predictive power. Where do we go from here? There are a few (non-exhaustive) paths which we will explore in later chapters:

1. Artificially expand our training set using Generative Adversarial Networks (GANs; chapter 8)

2. Truly expand our training set by designing a new narrowband survey to get more labels for galaxies outside the COSMOS field. The main effort of this will be continued by others in Alexie Leauthaud's group, but in chapter 9 I will explore two approaches of how machine learning can help clean up the sample identified by a hypothetical narrowband survey (comparing the results from an RF and a CNN).

3. Improve our CNN architecture. There are more powerful architectures in the literature, and many of them come with publicly released pre-trained weights. We can use transfer learning to start with an existing architecture and state, and then adapt it for our use, rather than re-inventing basic architectures *and* starting from randomly-initialized weights. This pre-trained approach will be introduced in chapter 9 for use with the narrowband survey.

Chapter 8

Data Augmentation GAN

This is primarily taken from my final project for UCSC's CMPS290C: Advanced Topics in Machine Learning. We had hopes of turning this into a Research Note of the AAS, but we never got a chance to tune it as much as would have been necessary, since that was when the referee report for (Gentry et al. 2019) started taking more time than hoped.

8.1 Introduction

Neural networks can be effective, efficient prediction models, but they generally require a relatively large amount of training before they sufficiently converge.

The simplest approach to address this is to use multiple epochs through a training set, allowing the optimizer to see each training example multiple times. Unfortunately this can lead to overfitting: the optimizer might think it has found a correlation between input data and output label that holds across multiple training examples, but

in fact it has just seen the noise pattern from a single example multiple times. Fortunately, this suggests a possible solution: at each epoch, change the training examples so that their noise is less correlated between epochs.

These approaches to increase the effective size of your training data are termed "data augmentation" transformations (Krizhevsky et al. 2012). Common examples include affine/geometric transformations, and noise "whitening" (Krizhevsky & Hinton 2009), but care must be taken to ensure that these transformations do not change the true label of the image. For example, when classifying handwritten characters, applying a horizontal reflection to the letter "p" would require changing the label to "q"—the label is not invariant to horizontal reflections. On the other hand, astronomical images tend to have arbitrary axes, so the label should remain invariant to a horizontal reflection. But the brightness of the main astronomical object relative to the noise might be physically meaningful—this means the label might not be invariant to whitening transformations, which would have been valid for handwritten characters.

So rather than rely on domain experts to define and implement the appropriate transformations, it would be interesting to see whether neural networks could learn to indefinitely generate new examples given a label, based off a finite training set size.

Using Generative Adversarial Networks (GANs) to generate these new examples is one method that has shown promise recently, but it is still unclear how generally applicable this approach is. Shrivastava et al. (2016) had success when their GAN was conditioned on a synthetic image, and the generated need only to learn to apply a noise model to the image. Antoniou et al. (2017) relax the need to condition on an entire

image, and instead only condition their GAN on a lower dimensional representation extracted from an image. Both sets of researchers find that training their target neural network on GAN-generated data can often improve results, but a key question remains: Will this still work if we only condition on the target labels, rather than condition on synthetic images or low dimensional representations of real images?

8.2 Data

We use images from the Hyper Suprime-Cam (HSC) survey (Aihara et al. 2018a), and labels from a galaxy catalog of the COSMOS survey (Laigle et al. 2016). In particular we are interested in two labels for each galaxy:

1. redshift
2. stellar mass

From these, we create a derived label: is a galaxy closer than a threshold redshift and within a certain low mass interval. Ultimately, we want a classifier that can predict this derived label for new galaxy images.

The training dataset we have is relatively small. It consists of roughly 2000 images, each of which has 3 bands (gri), and is $(50\text{px})^2$. These images are each centered on a specific galaxy, but there may be other background or foreground objects within the field-of-view.

These galaxies are not an unbiased sample of the galaxies within the available survey field—instead they have already been preselected as the galaxies most likely to

be the targeted nearby and low-mass galaxies (see section 7.4). Most of the details of this pre-selection are unimportant, but it is good to remember that it already took into bulk features like the total brightness and the average color of each galaxy. This makes our classification task more difficult, because the most useful discriminating information has already been used. This does not complicate our validation, as we validate using images with the same pre-selection function.

When training the GAN we will use all of the images, but when training the classifiers we will hold out 20% of the data as a validation set. We ensure that the same data is held out for the classifier trained on real images and the classifier trained on GAN-generated images.

8.3 Architecture

8.3.1 GAN

The main focus of our project is on the GAN. In particular, we build a conditional GAN that is conditioned on the two continuous labels: distance and mass (transformed as redshift and log stellar mass, both standardized to have mean=0 and standard deviation=1).

In this section we will broadly describe the GAN architecture and methods, but more details and the code can be found on github[1].

[1] https://github.com/egentry/galaxyCGAN/blob/master/gan.py

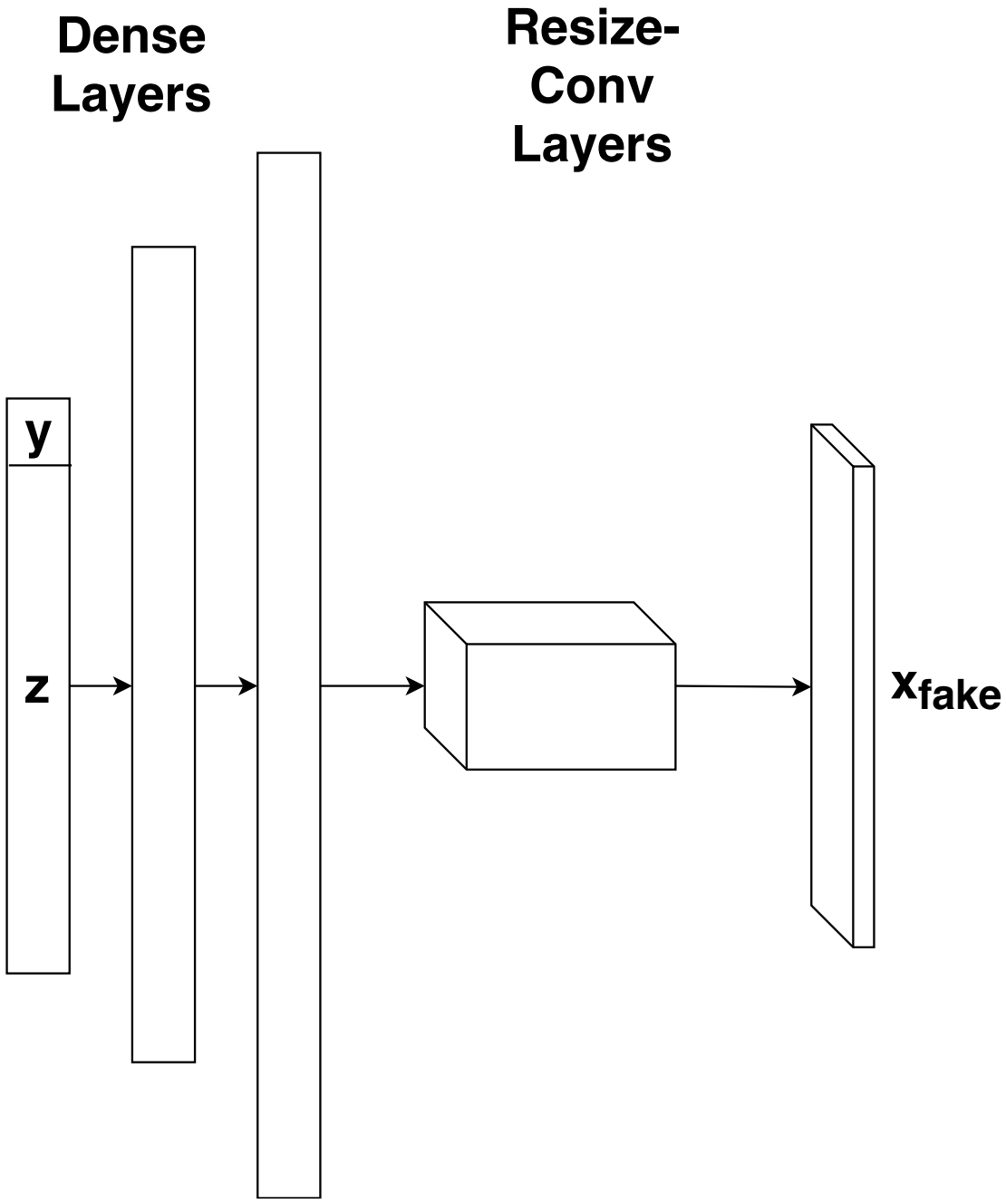

Figure 8.1: Overview of the GAN's generator architecture, producing a fake image, x_{fake}

Generator The generator[2] is a relatively standard conditional GAN. We concatenate the conditional labels with a noise vector (representing a latent set of features that should be add diversity even if the conditional labels are fixed). We then feed this input through two fully connected ("dense") layers, and then through two resize-convolution layers. After each layer except the final layer we apply batch normalization and RELU activation. We apply no activation to the final output layer. A visual summary of this network is shown in Figure 8.1.

While deconvolutional layers might be slightly more common, we choose to avoid them due to strong checkerboard artifacts in our GAN output. Following the advice of Odena et al. (2016), we instead used a resize layer (using bilinear interpolation) followed by *unstrided* convolutions.

Discriminator/Predictor The discriminator component of our GAN was much less standard. In particular, it was both a "predictor" as well as a discriminator[3].

The predictor component is uncommon but fairly straightforward. We feed in an image (real or generated), and it outputs a prediction for the distance and mass labels. It does this by first applying two convolutional layers (with batch normalization and LRELU activation after each), and then applying two fully connected ("dense") layers (with batch normalization and LRELU activation between the layers, but not after the final layer). A visual summary of this network is shown in Figure 8.2.

[2] https://github.com/egentry/galaxyCGAN/blob/9dbd59dad518a3454080a136d3ba4abdbc48b7fc/gan.py#L168

[3] https://github.com/egentry/galaxyCGAN/blob/9dbd59dad518a3454080a136d3ba4abdbc48b7fc/gan.py#L223

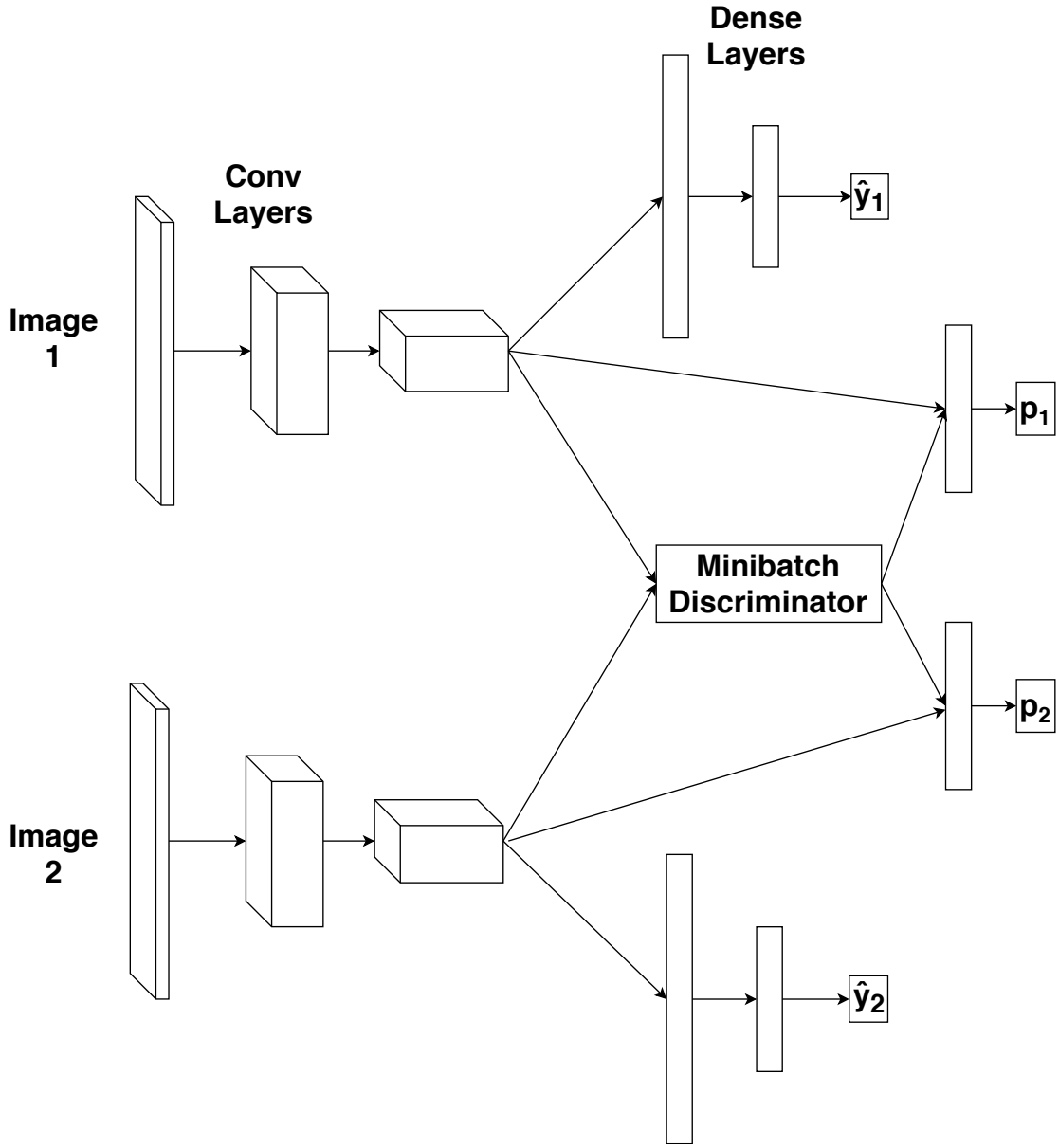

Figure 8.2: Overview of the GAN's discriminator/predictor architecture on an example batch of 2 images.

The motivation for this predictor component comes from Ravanbakhsh et al. (2016), who were a mix of computer scientists and domain experts in galaxy imaging . They found that the traditional conditional GAN architecture (Mirza & Osindero 2014), which feeds the conditional label into the discriminator, performed well on categorical conditional labels but not continuous conditional labels. When Ravanbakhsh et al. (2016) switched to a predictor they claim they found better results. Anecdotally we observed the same thing, but we are unaware of any rigorous study documenting this or any theoretical justification for the predictor.

But we cannot completely get rid of the discriminator. Ravanbakhsh et al. (2016) found that constructing a GAN with just a predictor and no discriminator often leads to mode collapse, which we confirmed in our early testing. So we followed their advice, and implemented *minibatch discrimination* (Salimans et al. 2016), which effectively looks for correlations between images in a given batch. As input features for the minibatch discrimination, we fed in the outputs from the final convolutional layer in the predictor *for an entire batch*. The minibatch discriminator then creates a new set of features for each image. The new features for each image are concatenated with the features for each image that were input into the minibatch discriminator. Finally, for each image, these combined features are passed through a single fully connected layer, with a sigmoid activation, representing the probability that a particular image is real or fake.

Loss functions We used two primary types of losses: an ℓ^2-based loss for $\hat{y}$ and a cross entropy-based loss for p.[4] In particular for the generator we minimize:

$$\ell^2 \left(y - \hat{y}(x_{\text{fake}}(y))\right) - \lambda \log(p(x_{\text{fake}}(y))) \tag{8.1}$$

where x are features (in this case images), y are targets (mass and redshift), $x_{\text{fake}}(y)$ is our generator conditioned on y, $\hat{y}(x)$ is our predictor, and λ is a hyperparameter weighting the two flavors of loss.

Similarly, for the discriminator/predictor we minimize:

$$\max\left[0, \ell^2 \left(y - \hat{y}(x_{\text{real}})\right)\right] - \lambda \log(1 - p(x_{\text{fake}}(y))) - \lambda \log p(x_{\text{real}}) \tag{8.2}$$

Ravanbakhsh et al. (2016) recommended[5] and briefly discuss this hinged-difference formulation, but do not clearly show why this is necessary.

Finally, we minimized this system using Adam (Kingma & Ba 2014). We used separate learning rates for the discriminator/predictor and the generator: .0001 and .0004 respectively. We used parameters $\beta_1 = 0.5$ and $\beta_2 = 0.999$.

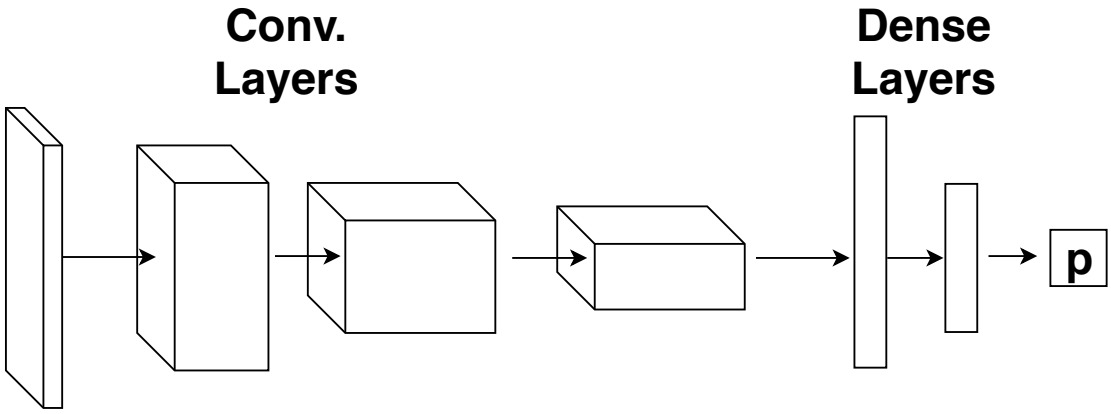

Figure 8.3: Overview of the classifier architecture, predicting the probability p that the image is of the target class.

8.3.2 Classifier

Fortunately the classifier is pretty standard. We used 3 convolutional layers, with RELU activation, max pooling and dropout after each layer. Then we used 3 fully connected layers, with RELU activation between each and a sigmoid activation after the final layer. A visual summary of this network is shown in Figure 8.3.

The loss function for the classifier was the standard binary cross-entropy loss, where a label=1 corresponds to "is a nearby, low mass galaxy" and label=0 otherwise.

We optimized the classifier using a learning rate of 0.001, $\beta_1 = 0.9$ and $\beta_2 = 0.999$.

[4] https://github.com/egentry/galaxyCGAN/blob/9dbd59dad518a3454080a136d3ba4abdbc48b7fc/gan.py#L323

[5] Actually they recommend a slightly different version in the text (where they change the max operator into a min and flip the "real" and "fake" terms), but we believe there to be an error in their text. The loss function they wrote doesn't match the behavior seen in their figures. We used the form written in this report, which would better mesh with their figures, and which gave us much better results.

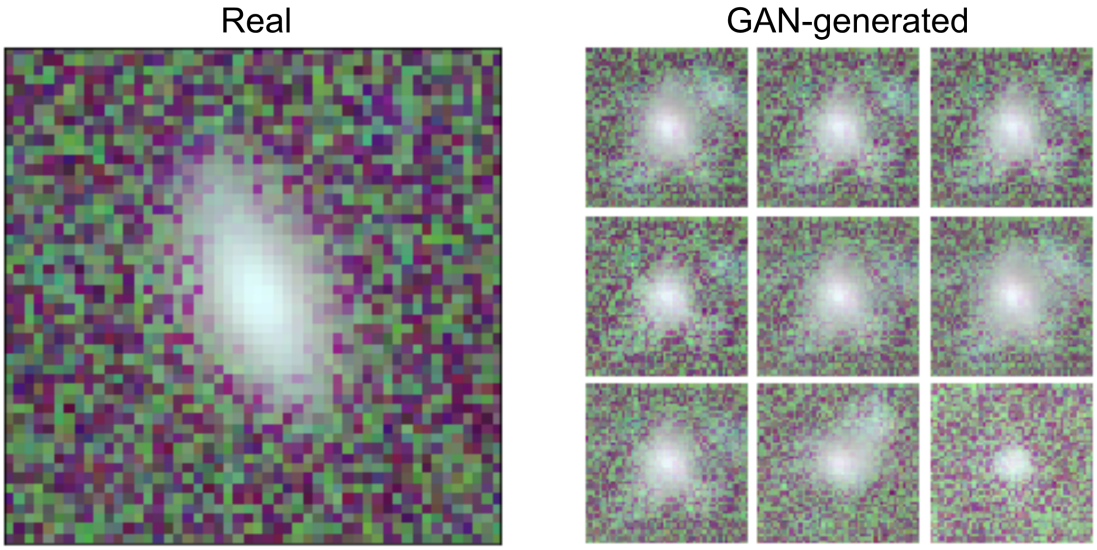

Figure 8.4: Comparison of real images and GAN-generated images for $z = 0.14$, $M_\star = 10^{8.51} M_\odot$.

8.4 Results

8.4.1 GAN

Once trained (for 400-500 epochs) the GAN can make some galaxy images that look believable (see Figure 8.4), but it does not show clear evidence of capturing the correct trends with respect to the conditional labels (see Figure 8.5).

The failure of our GAN to properly capture the correlation between images and the conditional labels suggests that our loss function is non-ideal. In particular, it might be that we need separate weighting terms for each label getting combined by the ℓ^2 loss. It could also be that we need a more complex loss function less susceptible to outliers (which we know exist in our data, leading to strong non-gaussianities in the label distributions). Finally, maybe it is simply a limitation of our small, pre-selected

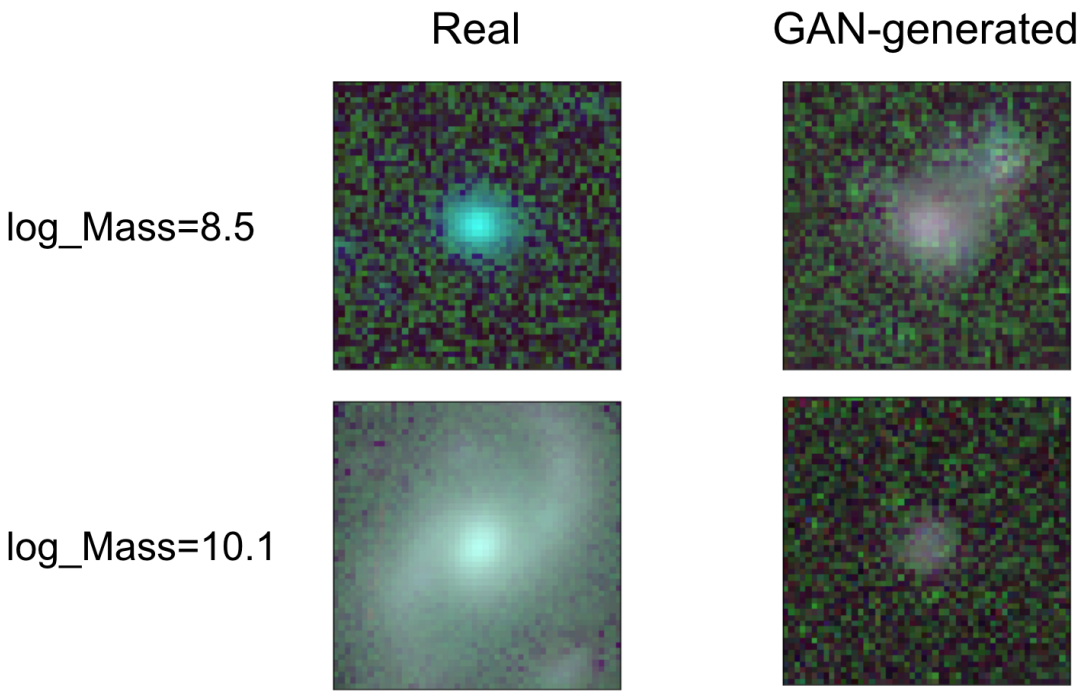

Figure 8.5: Comparison of how GAN-generated images scale with input conditional labels. Note: this image uses a slightly different stretch function from Figure 8.4, but that does not change the conclusions. Both rows show $z \approx 0.115$ galaxies.

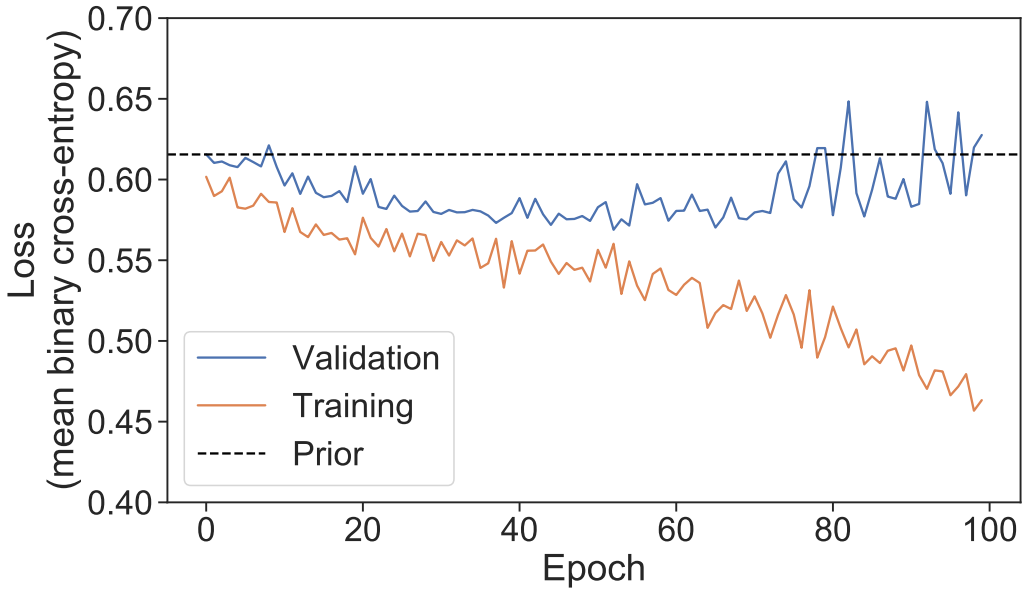

Figure 8.6: Learning curve of our classifier trained on real images using traditional data augmentation techniques.

dataset.

8.4.2 Classifier

In Figure 8.6 we show the learning curve of our classifier trained on real data with traditional data augmentation. It exhibits the standard behavior: the validation loss initially decreases, but then starts to flatten out. Near the end of the training, the training loss continues to decline by the validation loss is starting to rise, indicating over-training. We are not surprised that there can be overtraining, given the relatively small training set size.

The more surprising behavior is seen in Figure 8.7, the learning curve of the

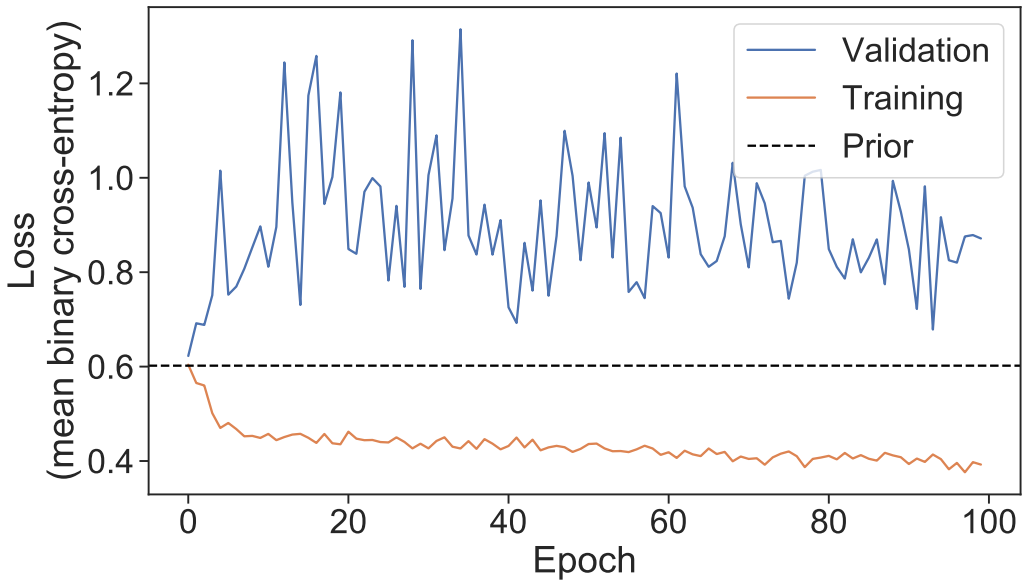

Figure 8.7: Learning curve of our classifier trained only on GAN-generated images and validated against only real images. Notice that the training loss is lower here, but the validation loss is much better in Figure 8.6.

classifier trained purely on GAN-generated images. In that case, the validation loss basically never decreases; the network parameters are "optimized" to become worse than their random initializations. The training loss typically *thinks* that it is doing better than in Figure 8.6, but that is not useful for us.

Ultimately, training on fake images and validating on real images leads to overtraining *almost immediately*.

8.5 Conclusions

Things we would try in the future:

- Training the GAN on a larger training set (even though it will not be balanced the same as our validation set)
- More tuning to balance the different components of the loss function

Answer to our main question (from section 8.1):

- Training only on our GAN images led to *clearly* worse results than using traditional data augmentation

Ultimately, using a conditional GAN for data augmentation worsened the performance of the classifier. This is not too surprising. Although Figure 8.4 shows that the GAN can sometimes make believable images, Figure 8.5 shows that it does not reliably capture the expected dependence on the conditional labels. While we might be able to improve this performance with more work, the key result is that it led to decreased classification performance relative to traditional data augmentation methods.

For this reason, our recommendation is to avoid using GANs for data augmentation when you cannot condition the GAN on an input image or at least a lower dimensional encoding on an image. Trying to condition just on a continuous label did not provide enough information to allow the GAN to outperform using existing, non-generated images.

Chapter 9

De-confusing narrowband survey results

This chapter will move a little faster, now that we have covered the basics in chapter 7.

9.1 Introduction

One question raised by our initial project was whether our CNN training set (2000 galaxies from the COSMOS field) was too limited. We hope to gain some perspective on that question in two ways. First, it would be useful to do a classification project which included a larger training set, in hopes of building intuition for how a larger training set might change the "trainability" and behavior of the network. Second, this will provide a useful step in the direction of gathering more dwarf galaxies through a narrow band survey.

So first, a brief background on the survey concept. We want to identify a sample of low redshift, low mass galaxies in the HSC fields. HSC has multiple photo-z

estimators, but in the wide layer these estimators are limited by a lack of any narrow band photometry. As shown in Figure 7.3, only having broad band photometry can go a long way towards starting to select a sample, but it is not enough; broad bands can only be coarsely sensitive to how much of a redshift has been applied to the galaxy's light and which spectral lines are now within a given observing band.

One possible way to improve this situation is by designing a custom narrow band filter, and performing a survey across the sky using that filter. The idea is that any galaxy which appears bright in this band (relative to its continuum flux, estimated using the HSC broad bands) must have a strong emission line with an observed wavelength within the band. Rather than having to take object-by-object spectra (or deal with the complications of highly multiplexed spectrometry), we need only to take an image of a field, and look for which objects are bright.

This method is not fool-proof though. In particular, if we construct a filter targeting the H_α line for galaxies at $z = 0.05 - 0.15$, we will also be sensitive to shorter rest wavelength lines being emitted by higher redshift galaxies (e.g. OIII at $\lambda_{\text{rest}} = 500.7$ nm from galaxies at $z = 0.38 - 0.51$). So how can we tell which situation is more likely for a given galaxy?

We will attempt 2 approaches to leverage existing HSC data to classify narrow-band selected galaxies as "target" or "contaminant" objects:

1. Training a RF using only reduced photometry and `FRANKEN-Z` photometric redshift estimates

2. Training a CNN using broad band images (and concatenating in the reduced photometric features and redshift estimates after the convolutional layers but before the fully connected layers).

9.2 Data

For more details, see section 7.2.

9.2.1 Features

Random Forest The input features for the RF will be the same as in section 7.2: i band magnitude; $g-r$, $r-i$, $i-z$, $z-y$ colors; `FRANKEN-Z` photometric redshift estimates.

CNN There will be 2 major differences between the features used by this CNN and the one described in chapter 7. First, we will only use 3 imaging bands: g, i and y (giving the broadest and most evenly-spaced wavelength coverage of the 3-band options). This will allow us to use the convolutional blocks of a pretrained network (reducing the number of training examples needed to constrain good weights in the network) which are typically trained on 3-band (RGB) images; by using a pretrained network, we can also use a significantly more complicated CNN architectures which would otherwise greatly increase the number of free parameters we would need to constrain. The second major difference in the data fed into the CNN is that after we pass the image through the convolutional layers, we then concatenate in the 7 photometric features (i mag, the

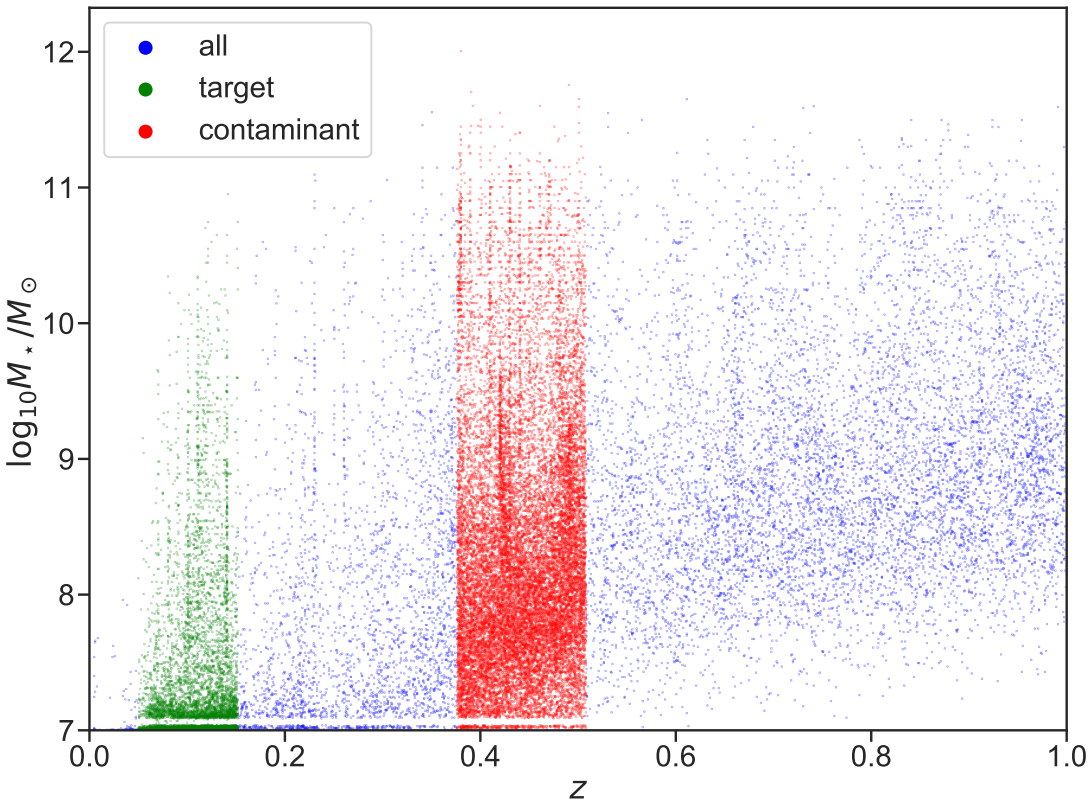

Figure 9.1: A visual representation of the "target" and "contaminant" populations. Although there are galaxies which do not fall within either redshift range, we *assume* they will not be selected by the narrowband survey, and thus would not be passed to our classifiers and thus we filter them out of our dataset.

4 colors, and the 2 `FRANKEN-Z` photo-z features), and pass all the features to the fully connected layers. This means that even though the images only contain 3 bands, the network will still have *some* knowledge of all 5 bands.

9.2.2 Targets

The targets will be the same for both classifiers. In particular we will be looking at the COSMOS field and will define a galaxy as being a "target" type if its

Laigle et al. (2016)-estimated redshift is within $z = .10 \pm 0.05$. In order to define a "contaminant" population, I assumed the "target" population was identified using an H_α sensitive filter ($\lambda_{\text{rest}} = 656$ nm) and that contamination would come from the OIII line ($\lambda_{\text{rest}} = 501$ nm) and computed the relevant redshift range: $z \in [0.38, 0.51]$.

Therefore, our dataset only consists of COSMOS galaxies with $z \in [0.05, 0.15] \cup [0.38, 0.51]$, and the target-type population ($z \in [0.05, 0.15]$) are given the target value $t = 1$ and the contaminant-type population ($z \in [0.38, 0.51]$) are given the target value $t = 0$.

For a visual representation of these two populations, see Figure 9.1. That figure makes it clear that unlike in section 7.2, stellar mass does not directly affect the target definition for this project.

The precise filter width and central wavelength might change before the survey is conducted (along with target redshift, "contaminant" population definition, etc.). That is fine. The point of this project is to lay the groundwork for the future researchers designing that survey.

9.3 Methods

Unlike in chapter 7, we will apply both classifiers to *all* objects for which we could obtain both images and reduced photometry. There will be no "pre-filtering" by the RF.

9.3.1 Random Forest (RF)

Very little is different between the RF used in this project compared to the one described in subsection 7.3.1. The biggest change is that we *do not* use a 10-fold cross-validation approach here. The reason is that here since we are not using a 2-staged approach (RF and then CNN), but rather are pitting the two models against each other, we want to use same training set and testing set for both classifiers. While in theory we could do the same cross-validation approach to get predicted probabilities for all galaxies in our dataset, in practice we do not want to spend the computational time training 10 different neural networks.

So as before we use an RF with 1000 trees, and then "soften" the predicted probabilities by adding 1 pseudo tree which always predicts "true" and another which also predicts false (in order to ensure the probability is never exactly 0 or 1).

9.3.2 CNN

This will have some significant differences from the CNN approach used in subsection 7.3.2.

First, rather than defining our own architecture, we start with `VGG-19`, the 19 layer architecture produced by Oxford's Visual Geometry Group (Simonyan & Zisserman 2014). There are 2 major sets of changes we make though. First, we change the size of the 3 fully connected layers: we use fully connected layers of 128, 64 and 1 units respectively rather than their 4096, 4096, 1000 units (respectively). The last layer *must* be different since we are doing binary classification while they were doing

1000-class multi-class classification; the other two fully connected layers were chosen fairly arbitrarily to match the fully connected layers used in section 7.3 (and to keep the number of free parameters relatively small).

By using more convolutional layers (16 in `VGG-19` as opposed to the 3 used in section 7.3) we have greatly increased the number of free parameters (on the order of 10^7 convolutional kernel weights for `VGG-19` compared to 10^4 for our previous architecture). Rather than trying to re-learn all these weights from our limited dataset, we will take advantage of their publicly released *pre-trained* network, trained on the `ILSVRC-2012` dataset created by ImageNet (Russakovsky et al. 2015).

For this project we will keep all these convolutional weights fixed; we could have instead used these pre-trained weights as initial conditions and then allowed our optimizer to try to find better values. There are 2 reasons why we chose not to do this: 1) initial tests hinted that trying keep these weights trainable led to initially worse results and would require many more training epochs (and each epoch would take significantly longer) than using fixed weights, and 2) by keeping the convolutional layers fixed, we would only need to run each image through them once and we could cache those `VGG-19`-extracted features for future training epochs (greatly speeding up training time which was crucial as I neared the end of my time in graduate school).

The only downside to using the pre-trained convolutional layers of `VGG-19` is that they were designed to handle 3-band images, whereas we have access to 5-band images. For this project, we will simply drop two of the bands (r and z, allowing us to keep the biggest span in wavelength between g, i and y), but to partially compensate

for this, we can concatenate in reduced photometric information for all bands after the convolutional layers and before the fully connected layers.

The second major change is that after the convolutional layers but before the fully connected layers we concatenate in the features used for the RF (i mag; $g-r$, $r-i$, $i-z$, $z-y$ colors; `FRANKEN-Z` best-estimated redshift and the associated risk measure of that estimate). This only adds 7 features to the 2048 features extracted by the `VGG-19` convolutional layers (a 2x2 image in 512 filters, flattened to a length 2048 feature vector), but subsection 9.5.2 will show that concatenating in these additional features dramatically increased the predictive power of the network. This extension serves two purposes:

1. It demonstrates that it is possible to take non-pixel-based information into account when using a CNN-based classifier. In particular, this means that even if you apply batch normalization to your images (a common preprocessing step in machine learning) you can still make sure the neural networks is able to take the overall magnitude into account when making its prediction. Similarly, you could also rescale galaxies of different effective radii, but still explicitly give size-based information to the network.

2. It ensures that the network has at least some information about all 5 bands, even if it does not have pixel-level information for all bands.

So in summary, we use the pretrained, non-trainable convolution layers of `VGG-19` as a *feature extractor*. On top of those 2048 features, we add our 7 additional

model	cross entropy	PR AUC	ROC AUC
RF	0.40	0.68	0.82
CNN	0.43	0.64	0.80

Table 9.1: A summary of the results from our primary RF and CNN model.

features that were used for the RF training. Then we pass all the features to 3 sequential, trainable, fully connected layers. The final result is converted into a probability using a logistic sigmoid, and the network is trained using the Adam optimizer (Kingma & Ba 2014) which tries to decrease the binary cross entropy across the training set.

(For consistency with previous sections, I will continue to call this a "CNN", even though we are not training any convolutional layers.)

9.4 Results

We trained the RF classifier and the CNN classifier; the CNN-specific learning curve is shown in Figure 9.2. We saved the CNN state at the 50th epoch (before overtraining sets in), and used that state for the remainder of this chapter. We show the purity-completeness curve in Figure 9.3, the ROC curve in Figure 9.4, and the diagnostic probability calibration plot in Figure 9.5. In Table 9.1, we compare the quantitative metrics of the two primary models (along with the models to be discussed in section 9.5.

Overall we can see that both methods show a substantial ability to learn and extract useful information from the training data. The RF outperforms the CNN, but only slightly, and the CNN shows a substantial improvement in its ability to learn

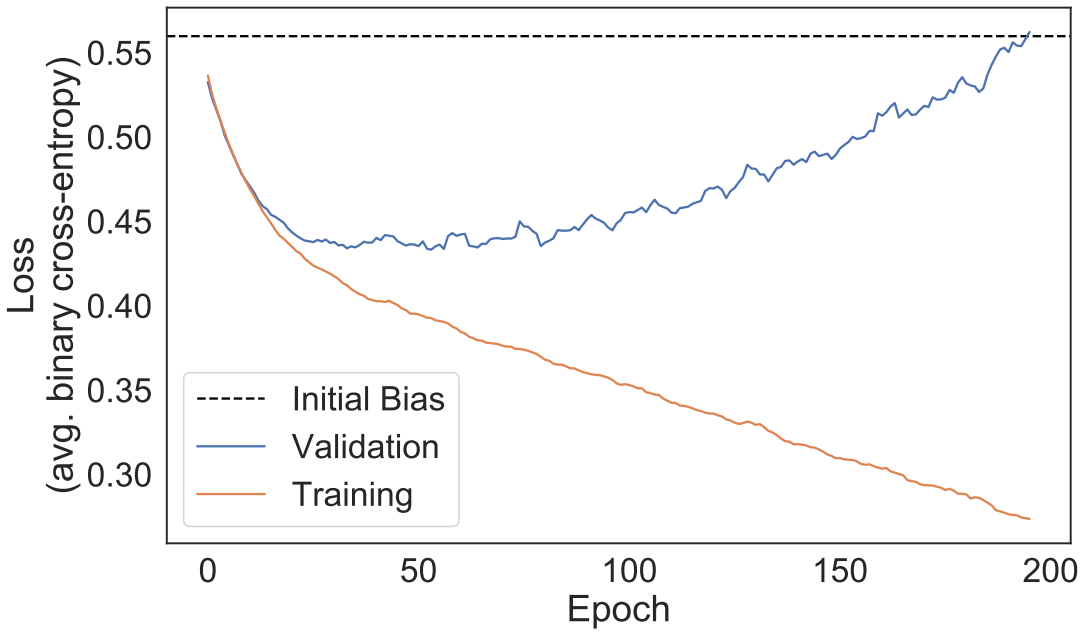

Figure 9.2: The learning curve of the CNN's performance, comparing the loss over the training set and the loss of the held-out validation set. (Both curves have been smoothed using a 5-element box-car kernel to help see the underlying trend.) The horizontal "initial bias" line shows how well we would do if we only knew what fraction of the labeled training set was the desired galaxy type, and used that fraction as the predicted probability for every galaxy in the held out testing set; this is what we would expect if the CNN was not able to learn *anything* from the images. The model was saved at the 50th epoch, and that state was used for the rest of the results.

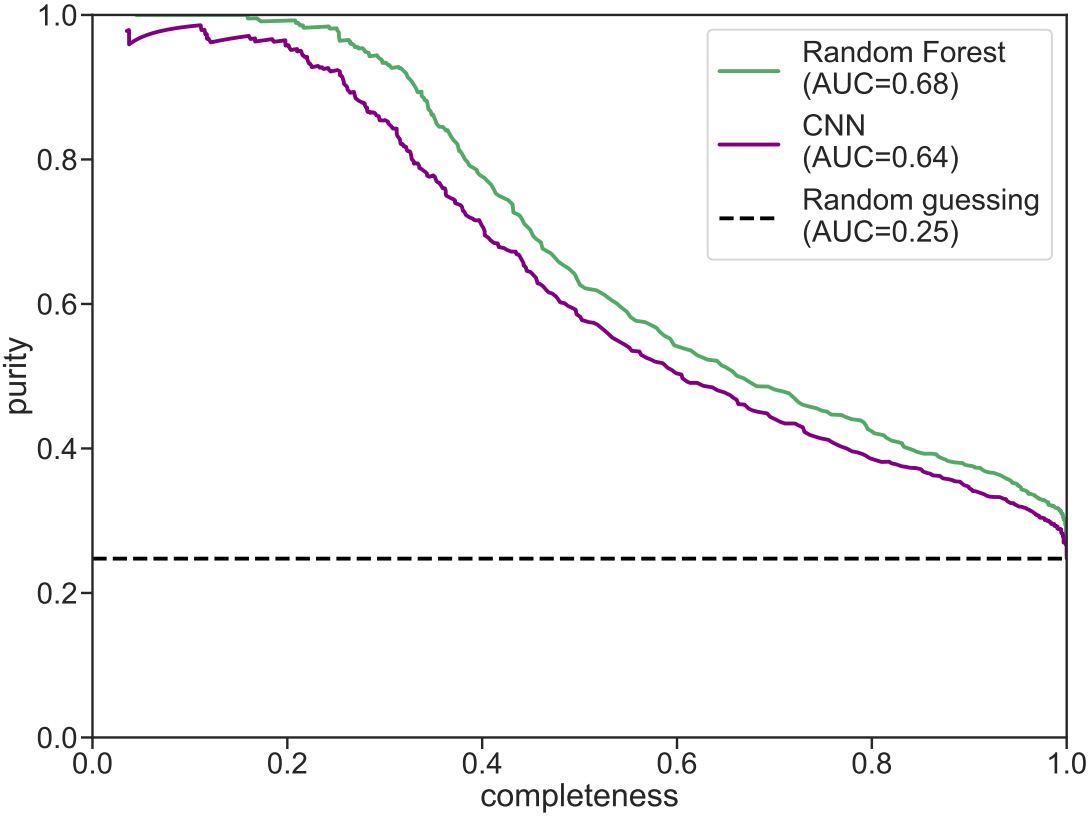

Figure 9.3: The purity-completeness curve for our two classifiers, trained and tested on the same sets of galaxies selected by a hypothetical narrowband survey. The RF classifier only uses the 7 features based on reduced photometry; the CNN classifier only uses the 2048 features extracted by `VGG-19`.

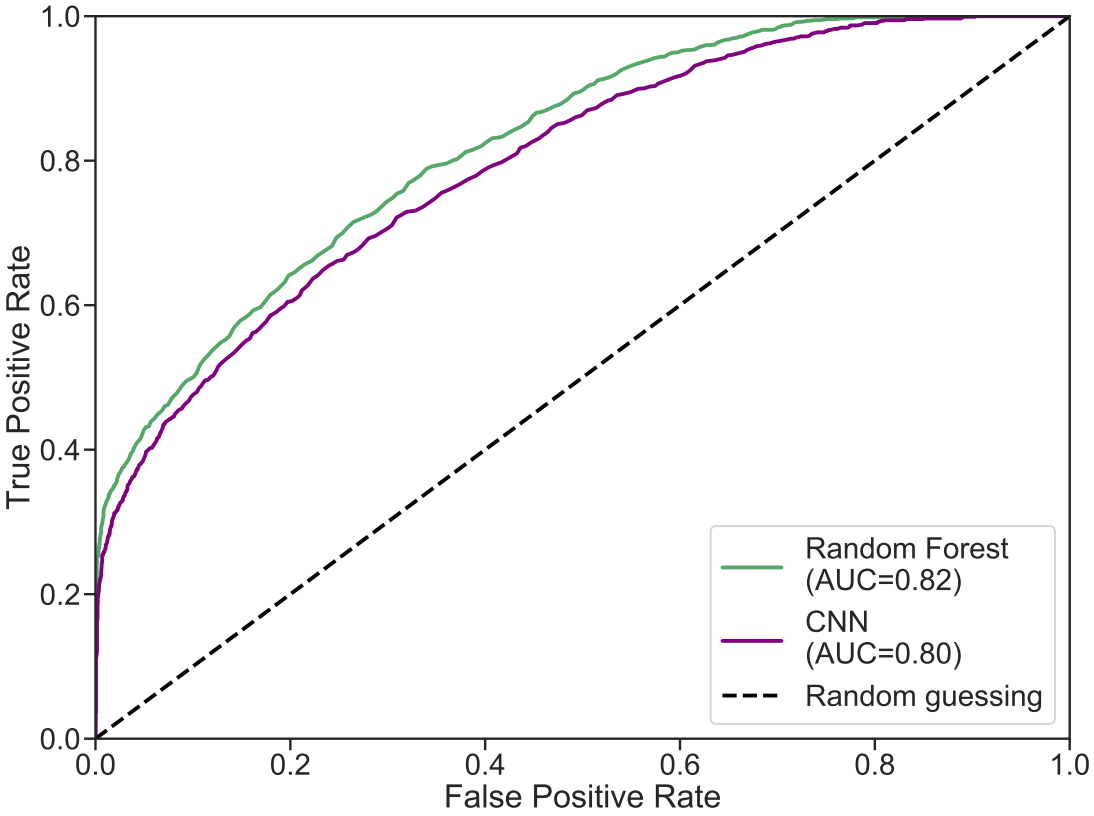

Figure 9.4: Same as Figure 9.3, except now showing the purity-completeness curve.

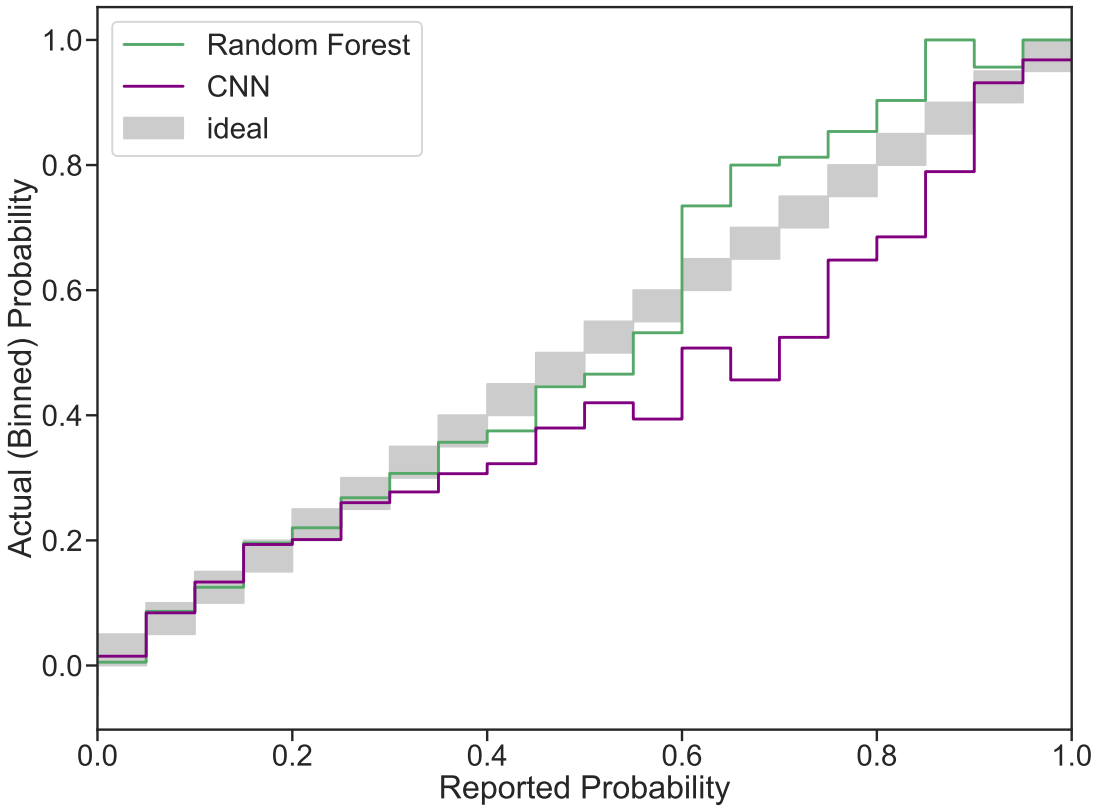

Figure 9.5: Same as Figure 9.3, except now showing the probability calibration of each classifier.

model	cross entropy	PR AUC	ROC AUC
RF	0.40	0.68	0.82
CNN	0.43	0.64	0.80
RF: no photo-z	0.42	0.65	0.80
LR: only photo-z	0.56	0.29	0.57
CNN without photometric features	0.53	0.40	0.64
RF with `VGG` ($n_{\text{split features}} = 1024$)	0.43	0.64	0.79
RF with `VGG` ($n_{\text{split features}} = \sqrt{n_{\text{features}}}$)	0.50	0.55	0.72

Table 9.2: An expanded version of Table 9.1 now including the results from our various exploratory models

compared to the results in chapter 7. Whether that is due to the larger training sets, the use of pretrained layers, or simply the more separated populations (we do not have to separate $z = z_0 + \epsilon$ galaxies from $z = z_0 - \epsilon$ galaxies for arbitrarily small ϵ) is difficult to disentangle.

9.5 Additional Explorations

For the rest of this section, we will explore a few aspects of how these results are affected by the choice of model architecture and input data. A quantitative summary of results is given in Table 9.2.

9.5.1 The role of `FRANKEN-Z` redshifts in the RF

One question we frequently receive is why we need to construct our own redshift-related estimators if HSC releases its own general purpose photo-z estimators. In the problem explored in chapter 7, part of the answer is that our target also required stellar mass information, but in this chapter we are *only* interested in redshift. Further-

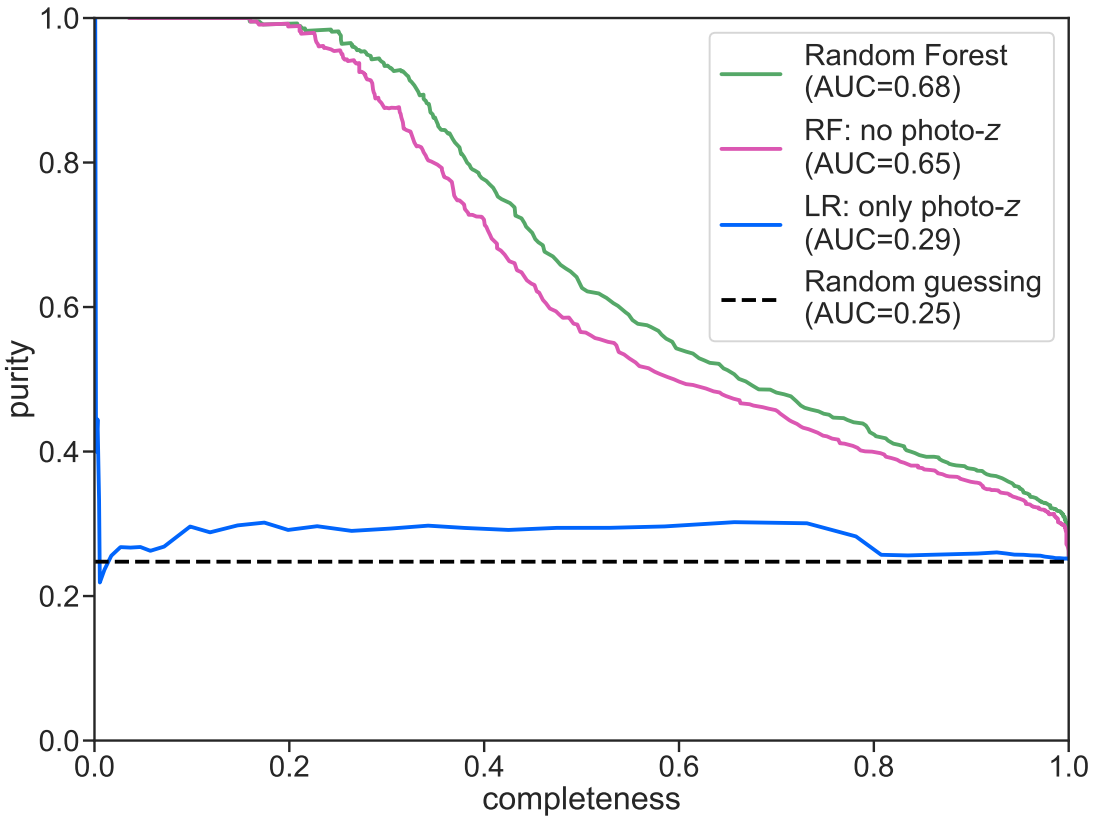

Figure 9.6: The purity-completeness curve for standard RF classifier along with the two additional models described in subsection 9.5.1: RF without `FRANKEN-Z` photo-z features, and logistic regression (LR) with only `FRANKEN-Z` photo-z features.

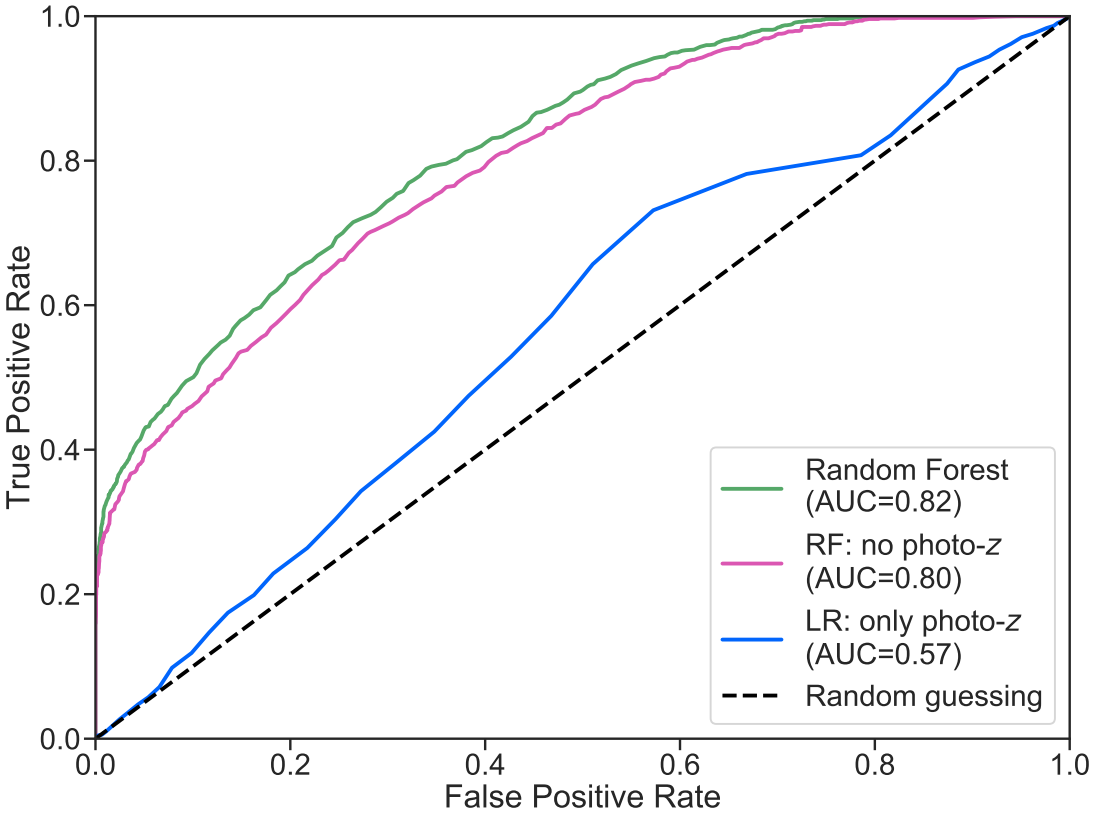

Figure 9.7: Same as Figure 9.6, except now showing the ROC curve.

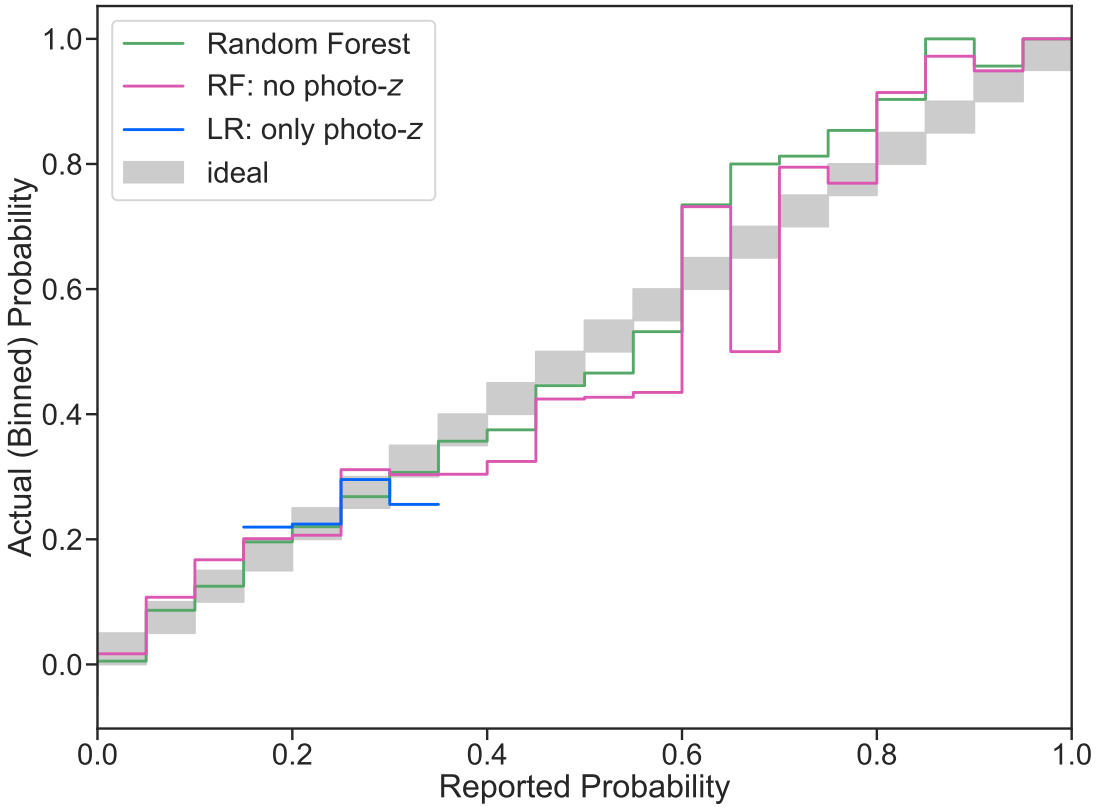

Figure 9.8: Same as Figure 9.6, except now showing the probability calibration of the classifiers. The photo-z only curve does not extend across the entire plot since that classifier only predicts probabilities within the range 17-33% on the testing set.

more, we only need to learn a single threshold in redshift, and the two population of candidate galaxies are pretty clearly seperated in redshift.

This appears to be a time when we need only use the HSC photo-zs (specifically the `FRANKEN-Z` estimator), but in practice we will find there is still great value in constructing and training our own model that is targeted to our specific application, rather than just relying on general purpose broad-band photo-z estimators. Nonetheless, as we will show in this section, including those general purpose photo-z estimators as a feature to our model *can* provide additional predictive power compared to *just* using the photometric colors and magnitudes.

To illustrate this, we ran 2 additional models:

1. We construct a logistic regression (LR) model using *only* the `FRANKEN-Z` features. This effectively asks whether using the general purpose photometric redshifts alone would be sufficiently powerful. We could have constructed a model that only looks for a threshold in the estimated photo-z, but using a LR model lets us associate probabilities with each prediction without significantly changing the interpretation.

2. We also construct a RF classifier with just the magnitude and color features (*not* including the 2 `FRANKEN-Z` features used by the reference RF). This provides an ablation test, so that we can isolate how much the photo-z features contributed to the overall performance of the RF.

In Figure 9.6 we show the purity-completeness curve for these classifiers along

with the standard RF classifier, and in Figure 9.7 we show the ROC curve. Clearly just using the photo-z features by themselves provides very poor results for this particular application. However these these features do provide value, since when we remove them from the RF, we see a small but clear decrease in performance.

This shows the value of preprocessing features, even for machine learning methods. This might not be expected for two reasons: 1) the `FRANKEN-Z` uses the same input photometry as our ablated RF; therefore, it does not provide any "new" information, it just transforms it, and 2) a RF (given infinite training data and computational resources!) should be able to learn any transformation of the data (as opposed to methods like logistic regression that can only learn linear decision boundaries). But for finite resources, RFs still have their limitations; in particular they only branch along the axis directions of the feature space. So in practice, having additional features which are pre-transformed to be more readily useful can still add predictive power, even if those features do not add any truly new information.

9.5.2 Concatenating photometric features into CNN

As mentioned in subsection 9.3.2 we are taking the slightly unusual approach of inputting multiple types of data into our CNN: we first pass in a 3-band image into the convolutional layers, but then also concatenate reduced photometric features after the convolutional layers but before the fully connected layers. Even though this only adds 7 features to the 2048 extracted by `VGG-19`, it leads to a significant improvement in performance, as seen in Figures 9.9-9.11 and in Table 9.2. Similar to the results in the

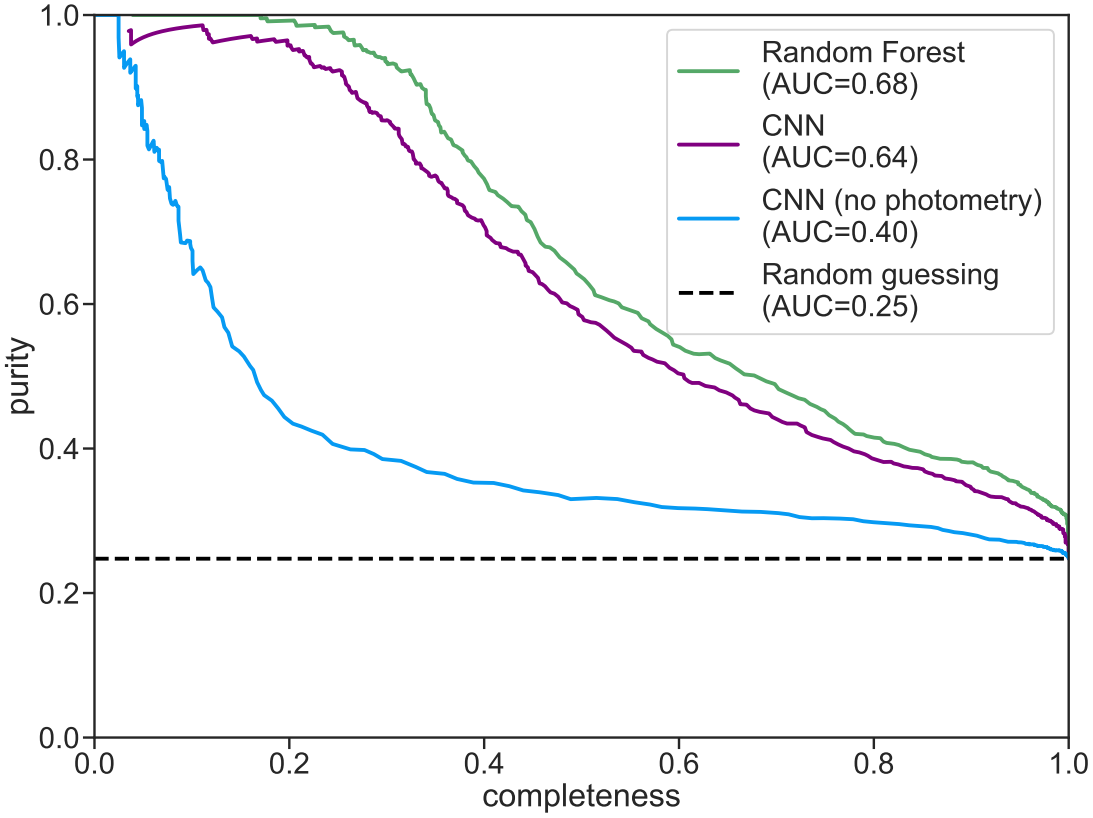

Figure 9.9: The purity-completeness curve for standard RF classifier and the standard CNN classifier, along with a CNN classifier *without* the photometric features concatenated in before the fully connected layers.

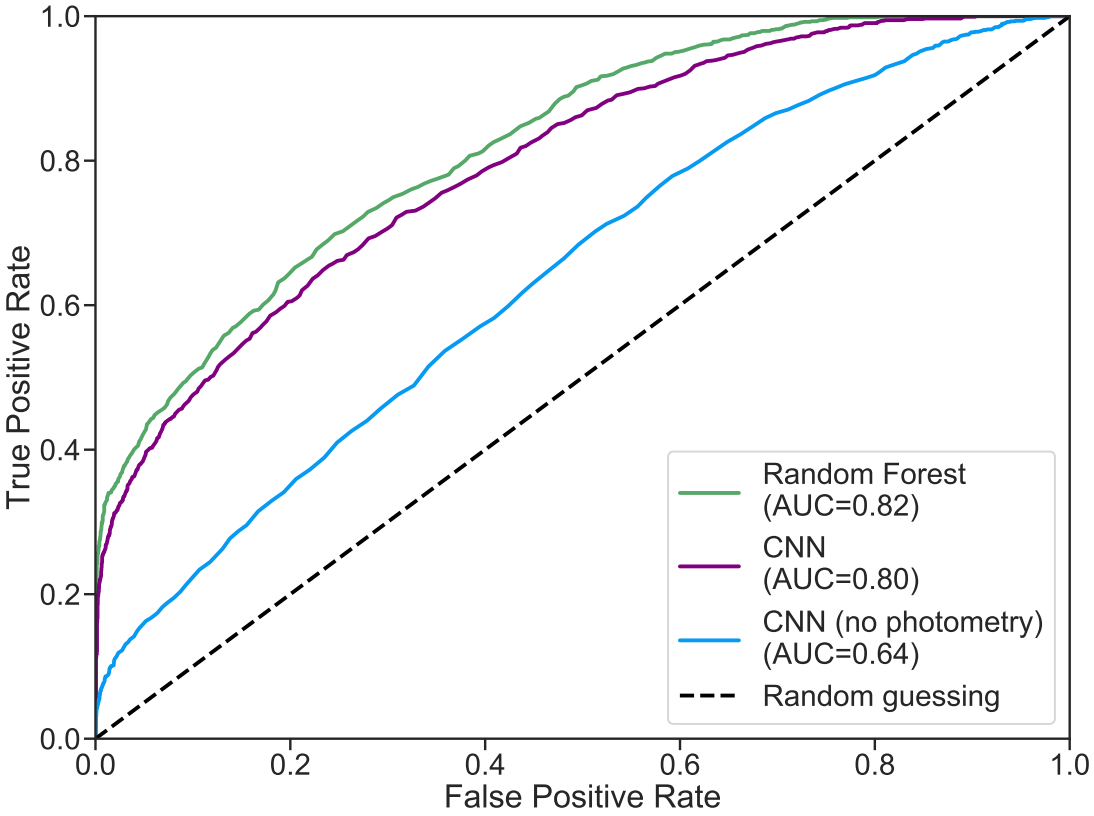

Figure 9.10: Same as Figure 9.9, except now showing the ROC curve.

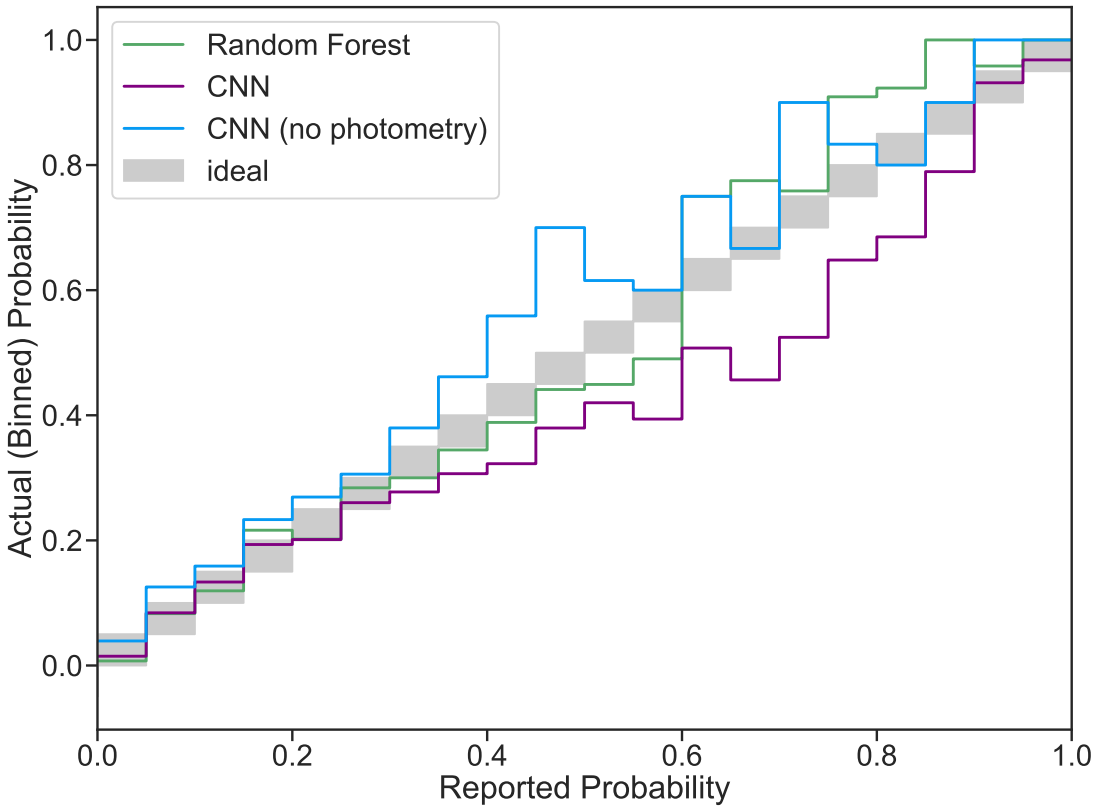

Figure 9.11: Same as Figure 9.9, except now showing the probability calibration of the classifiers.

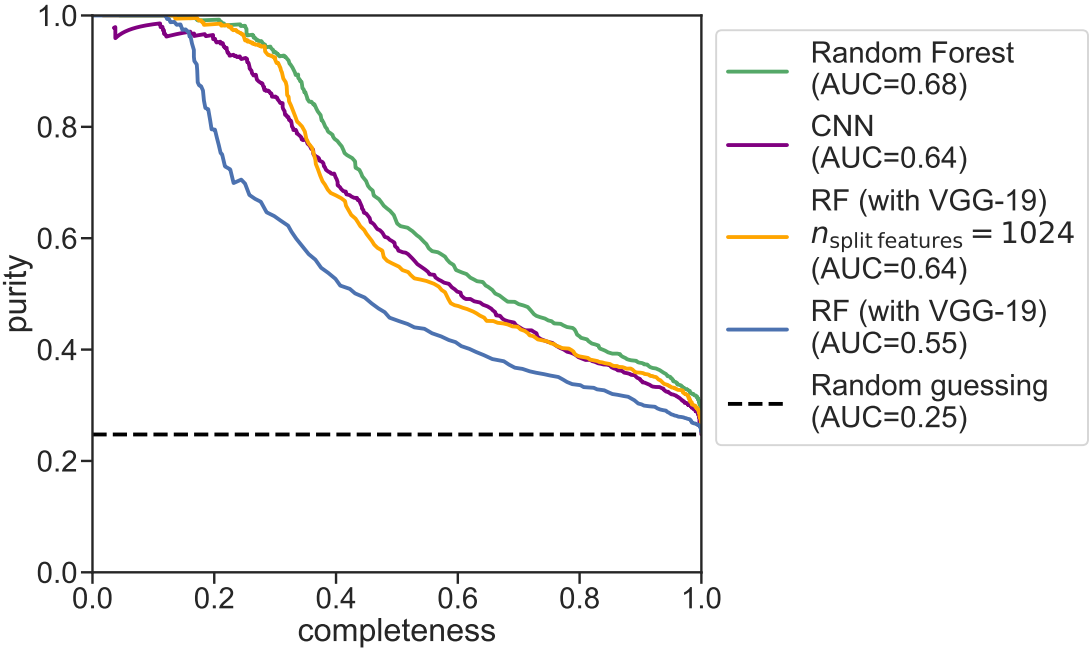

Figure 9.12: The purity-completeness curve for standard RF classifier (which only had access to our 7 reduced photometric features) as well as two feature-augmented RFs which also had access to the `VGG-19`-extracted features, and our CNN. (The blue curve represents an RF that used the standard feature selection scheme, which selected $\lfloor\sqrt{2055}\rfloor = 45$ candidate features for each split, and the orange curve represents an RF that selects 1024 candidate features for each split.)

previous experiment, subsection 9.5.1, we find that adding in expert-designed and pre-transformed features can help make it easier for a model to make powerful predictions, even if those features do not actually include any new information.

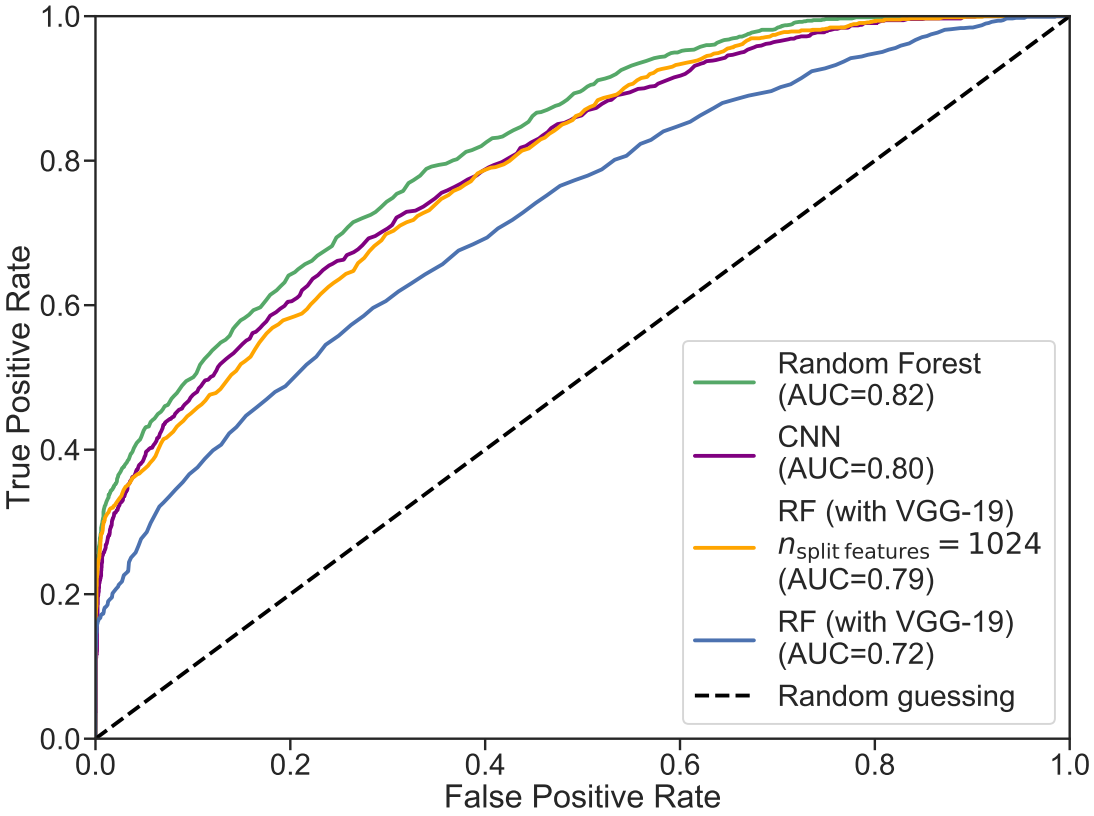

Figure 9.13: Same as Figure 9.12, except now showing the ROC curve.

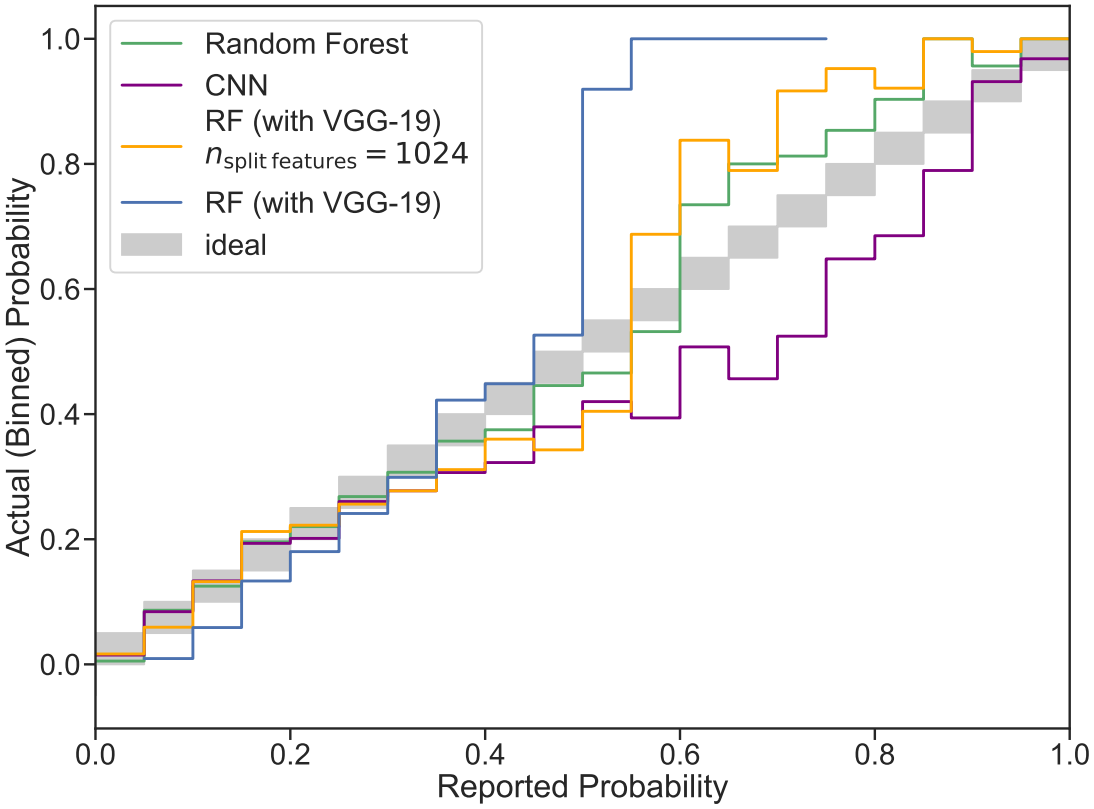

Figure 9.14: Same as Figure 9.12, except now showing the probability calibration diagnostic. The blue curve representing an RF with `VGG-19` features and the standard number of features at each split ($\lfloor\sqrt{2055}\rfloor$) cuts off early because it does not predict a probability greater than 73% for any object.

9.5.3 RF augmented by CNN-extracted features

Thus far, it appears that adding more features to models is always better, and a RF is more powerful than the CNN. From these observations we generate a hypothesis: the best performance would come from combining the 2048 `VGG`-extracted features with the 7 photometric features, and passing all of them into a new RF classifier. The results are shown in Figures 9.12-9.14. (For reasons about to be discussed, we actually created 2 RFs, one which followed the default hyperparameter of randomly selecting $n_{\text{split features}} = \sqrt{n_{\text{features}}}$ features of the total possible n_{features} at each branch of each decision tree (the features are selected independently at each split; they are not the same chosen features), and another RF which considered $n_{\text{split features}} = 1024$ randomly selected features.

The first thing that we can see is that the RF using the default hyperparameter, $n_{\text{split features}} = \sqrt{n_{\text{features}}}$, suffers a significant decrease in performance, even though it has access to strictly *more* information. One major reason why adding additional features might decrease RF performance is due to the randomness purposefully added to the process of constructing branches. Each branch randomly selects a subsample of candidate features and from those candidates, chooses the feature that allows for the most informative split. Typically, the number of features randomly considered for a given branch is the square root of the total number of features. This means that if we start with a set of low noise, high value features and start adding additional high noise, low value features, the first few added features will not significantly change things. We

will still randomly select roughly the same number of "good" features, and when a "low value" feature is chosen, it should not have the most informative split, so it will not become the actual branch in the tree. But as the number of low value features increases (for a fixed number of "good" features), it will become more likely that *all* of the candidate features will be low-value features (since the number of candidate features scales sublinearly with the total number of features), resulting in less informative splits. So by adding 2048 features that each included a low but non-zero amount of information, they swamped the 7 higher value features, resulting in worse performance.

Fortunately, this should not be too difficult to solve: if the problem is just that the number of candidate features scales sublinearly with number of features, we could just choose a different scaling. When we just manually tell the RF to consider 1024 features at each split (leading an average of 3.5 of the non-`VGG-19` features being considered in each split, compared to the 2 non-`VGG-19` features considered by our non-`VGG-19` classifier), we find better results than the model using the default hyperparameter, but still worse than the RF with *no* `VGG-19` features (see Figures 9.12-9.14 and Table 9.2).

So there is still some degradation of performance due to the additional features. Whether this is "overfitting" or not depends on how you define overfitting. For instance when we look at Figure 9.14, we can see that the problem is *not* that the model is over-confident; particularly at high probabilities the model is actually under-confident. On the otherhand it is not simply a matter of our probabilities needing to be scaled or stretched monotonically (without changing the rank ordering); the ROC

and purity-completeness curves and their respective AUCs are invariant to monotonic transformations of probability, yet still they all indicate that adding the `VGG-19` features *hurt* performance.

In practice, overfitting with RFs can be difficult to deal with. One approach would be to use the out-of-bag samples (not chosen by the initial bootstrap for a given tree) to estimate how well the tree performs, and "boost" (i.e. weight) the tree according to its performance. Another approach is to perform something like permutation feature importance. In that case, you would randomly permute the rows of a given feature in a training set (enforcing that the value of that feature is independent of the label), retrain the model, and see how well it performs on held-out data. Features are then only kept if the performance noticeably decreases when that feature is permuted. Unfortunately, both of these options go beyond the scope of this mini-experiment.

No matter what, this is an indication that our current methods could be improved upon. When we add additional information, we see a decrease in performance.

9.6 Future work: allowing greater-than-3 band images

In subsection 9.3.2 we explained that in order to use many of the most-common pre-trained CNN architectures, we can only pass in 3-band images, which is probably suboptimal given that we have access to 5 bands of imaging data. In order to make use of all those bands, we have outlined two main possibilities, but did not have time to explore them in depth within this thesis. Therefore, we will simply point out these

options, and leave precise tests for future work.

9.6.1 Run pretrained feature extractors on multiple permutations of bands

In our standard CNN, we use the convolutional layers of a pretrained, fixed network to extract features from an image made of a particular choice of 3 bands. Typically we only concatenate these features with the features extracted by the HSC photometric pipeline, but there is no reason why we could not concatenate other convolution-extracted features from *different* choices of 3-bands (e.g., concatenate the features extracted from a *gri* image with features extracted from a *izy* image).

This approach has the benefit of clearly giving the final classifier (an RF, a set of fully connected layers, etc.) access to all wavelengths. Furthermore, this method retains the ability to extract features once per image per combination of bands, and cache the convolution-extracted features to speed up subsequent training epochs.

The main downside of this approach is that it greatly increases the number of (potentially correlated, low-information) features, and in subsection 9.5.3 we showed that that can decrease the performance of a classifier (in that case an RF). Therefore, we need to keep in mind that by adding *more* wavelength information, we might actually get worse results if the classifier cannot properly deal with an increased number of low-value features (through feature selection, regularization, boosting or some other method).

9.6.2 1x1 convolutions

This idea is active *before* passing any images into the pretrained layers. The idea is that we need, for each pixel, to go from N_{bands} (in this case 5) to 3 bands; in other words, we need to perform a version of pixel-wise dimensionality reduction. In other areas of convolutional neural network architectures this is often accomplished using "1×1" convolutions. In some sense, "1×1" and "convolution" are contradictory, but it shows how it is an extension of true convolutions. True convolutions take a $N_{\text{pix}} \times N_{\text{pix}} \times N_{\text{bands, in}}$ patch, perform a tensor multiplication with a size $N_{\text{pix}} \times N_{\text{pix}} \times N_{\text{bands, out}}$ learned weight matrix, producing a length $N_{\text{bands, out}}$ output spaxel. A "1×1" convolution follows this approach, but simply using a 1 pixel $\times$ 1 pixel receptive field. Said another way, we simply perform a pixelwise matrix multiplication:

$$\begin{pmatrix} R_{\text{out}} \\ G_{\text{out}} \\ B_{\text{out}} \end{pmatrix} = \boldsymbol{W} \begin{pmatrix} g_{\text{in}} \\ r_{\text{in}} \\ i_{\text{in}} \\ z_{\text{in}} \\ y_{\text{in}} \end{pmatrix} + \boldsymbol{b} \tag{9.1}$$

where we have passed in a 5-band pixel ($grizy$ surface brightnesses) and are converting it into an RGB-like pixel using the learned weights $\boldsymbol{W}$ and $\boldsymbol{b}$ (and optionally a non-linear activation function).

This model has many benefits. First, is that it *includes* the approach we took

of just choosing the *yig* bands and setting them as RGB respectively. If that truly is the best approach, then this model should be able to find the $\boldsymbol{b} = 0$ and a suitable projection matrix for $\boldsymbol{W}$. (In fact, we recommend using this as a well-motivated initial condition.) But completely throwing away 2 bands of information is probably not optimal, so this model can also learn more complex combinations, while only increasing the number of free weights by $5 \times (3+1)$. Since the number of weights is relatively small, we could also probably afford to chain together multiple 1×1 convolutions, in order to give access to more non-linear transformations of our 5-band pixel.

Unfortunately there are a few downsides to this model. The largest is that this increases the computational costs significantly. By choosing a fixed (rather than learned) transformation (i.e. a projection) and by fixing the convolutional layers, we only need to run each image through the convolutional layers once. We can cache these results, greatly speeding up the run-time of future training epochs. By adding a learnable layer *before* the fixed convolutional layers, we still need to pass every image through the full CNN during every epoch. (It also means we need to propagate derivatives through the entire pretrained network in order to reach the early 1×1 convolutional transformation, even if the pretrained network is being held fixed). In practice this means using 1×1 convolutions are probably not practical for very training data-starved regimes, but might still be useful in more intermediary regimes where we do not have enough data to fully train a `VGG-19`-like architecture from scratch (and thus want to start with pretrained weights) but do have enough data so that we already plan to leave the `VGG-19` trainable (in order to best adapt to our astronomical images, which likely look different from the

ImageNet images). In that case, the cost to add a 1×1 convolutional layer is trivial compared to the potential gain by not throwing out 40% of our data. Finally, this is not a contrived example; in order to infer metallicity from SDSS images, Wu & Boada (2019) chose to start with a pretrained CNN and allowed it to be trainable, but had to throw out 2 of the SDSS imaging bands since the pretrained network would only accept 3 input bands.

While we do not present an in depth study here, we have run some initial tests on a different problem[1], and can share some intuition of our finds. In that test, we tried to extend the analysis of Wu & Boada (2019); they simply dropped 2 SDSS image bands and passed in the remaining 3-band image to a CNN trained to predict galaxy metallicity. When we instead prepend a 1×1 convolutional layer and pass a 5-band image into that layer, we found a slight but significant improvement in performance. The reason why we never published this result is because even our improved result never actually matched the performance claimed by Wu & Boada (2019). The biggest difference I can think of is that my method use a more standard, less complex optimization approach, whereas theirs appears much more hand-tuned (allowing certain layers to only be trainable during training epochs, using complicated functions for changing learning rate as training progresses, etc.). This seems to suggest that while 1×1 convolutions might be an easy way to get some improvements, focusing instead on optimization schemes might lead to better final results (even if it potentially takes more time / experimentation).

[1] https://github.com/egentry/one_by_one

Chapter 10

Conclusion

Ultimately using CNNs has been less successful than we would have hoped. In almost every application, just using a Random Forest trained on reduced photometry was more powerful, easier to set up, and easier to get the relevant features (querying and downloading a single table, rather than downloading millions of fits files). When we started we did not know that a CNN would be significantly helpful (since there was no guarantee that a 5 band, broad band image contains much information about redshift or stellar mass of a galaxy), but we had at least hoped it would provide *some* benefit above using simple photometric features.

General suggestions for applying machine learning in astronomy Overall, if the goal is to get good results, I would definitely recommend first starting with non-deep machine learning techniques. As I have shown, Random Forests can be one powerful method, but there are plenty others that might work just as well. It helps

being able to have some non-linearity in the decision boundaries (as opposed to, e.g., logistic regression), but there are plenty of other approaches not explored in this part such as support vector machines.

However, if you *are* going to use a deep-learning model, before spending a lot of time setting up a deep learning model, I would recommend testing if you would be able to make the desired inference/prediction by eye. It does not have to be perfect; but deep learning will often do best when it is a task that a human eye/brain *could* do, but would take too long (e.g. Huertas-Company et al. 2015 predicting visual-like morphologies for galaxy images). There are certainly times when a CNN might be able to learn something your eye would not, but if your eye cannot do it, it might be a warning that training a CNN will be a challenge.

Relatedly, people often ask "how many training images do I need in order to use a CNN?". If the desired target is strongly imprinted on the morphology of the target, just a few example images might be enough (using a pretrained, fixed network as a feature extractor, and then passing those features to a traditional, non-deep machine learning model). In more typical cases, it appears that $\sim 10^3$ images is not enough to learn a significant amount (subsection 7.4.2, but by $\sim 10^3$ we do start to see much more noticeable amounts of "learning" (subsection 9.3.2).

Since some astronomy training sets are relatively limited in size, it might be valuable to have a broader set of CNNs trained on large astronomical datasets and publicly released for use as pretrained networks. This would not be perfect; astronomical data can be pretty heterogeneous, especially when it comes from different observational

facilities and is intended for different scientific applications. Still, many astronomical studies using deep learning release neither source code nor trained network weights (for example, in this thesis we have mentioned the works of Huertas-Company et al. 2015, Hoyle 2016 and Wu & Boada 2019, and none of them release either source code or final networks). Rather than always having to start from scratch or always trying to "transfer" learning from networks trained on ImageNet, it would be useful if the community did a better job of being open with source code and results.

However, that is not yet the case; at the moment, one of the best options for training a CNN on astronomical images is still to download a *large number* of astronomical images ($> 10^5$). In my experience, some surveys do a much better job of making this possible. For example, SDSS, even though it is a relatively older survey, makes it relatively fast and reliable to download large amounts of data via Globus. (I do not remember exactly how long it took, but within a few days I could download 5-band fits images for $\sim 2 \times 10^5$ galaxies.) On the other hand, that can take weeks for the HSC survey which requires that you download each band of each image individually using HTTP requests. Compared to datasets like ImageNet, it is not unreasonable to want to gather half a million training images, but the data engineers for current surveys have not all caught up to these modern needs.

Project-specific recommendations Here are some more complicated machine learning techniques that I would have been interested in trying if I had more time:

1. Improving the GAN approach of chapter 8. In particular, others have had more

success when you *also* condition on/feed in a lower-dimensional representation of the image such as the latent state from an image auto-encoder (e.g., Antoniou et al. 2017). In chapter 9 we showed how to use `VGG-19` as a morphological feature extractor—perhaps conditioning on something like those features, thus provide a lower-dimensional representation of the image, would prevent the GAN from learning the incorrect scalings that we observed?

2. Replacing the RF pre-filter of chapter 7 with a fully-connected neural network. Train both simultaneous, so that an example galaxy always starts with the fully-connected network with access just to the photometric features. If the network is too uncertain, pass in the image data to a CNN, but add a penalty to the overall loss function (so that the network does not try to ask for images from *every* galaxy). Perhaps by training both at the same time, the first "pre-"network will learn when it is or is not useful for the convolutional network to see the full image, unlike our RF which has no knowledge of the "post-"filtering network.

More generally, I think it would be useful to try to pretrain the convolutional layers of our network on a large astronomical dataset, even if this pretraining was done with a different target and slightly different population of galaxies compared to the "real" application we have in mind. The motivation for this is twofold. First, it is a direct example of "transfer learning" but now starting with a network trained specifically on astronomical images, rather than pictures of cats and dogs. Secondly, it is partially inspired by the ImageNet dataset and the results of networks trained on

that dataset. In particular, although that dataset contains millions of images, there are thousands of categories, meaning that an average category can only have roughly a thousand training examples, yet still CNNs are able to learn remarkably well. If our goal was to only predict between 2 categories, we would expect the $\sim$ 2000 training images would be far too few to train a useful CNN, but by adding $\sim$ 1000 classes to the training process (which might seem like nuisance, uninteresting classes), the network tends to become *better* at differentiating between the 2 original, "interesting" classes. So in subsection 7.3.2, by only training on the pre-filtered images we might be limiting the network's possible performance. If we instead tried to first train our convolutional layers on general astronomical relationships (such as photo-z estimation or morphology classification), or trained our classifier on the full COSMOS field (even though we would only apply it to a pre-filtered population in production), maybe our network would be able to learn more about the differences between galaxy images, even though the additional images or training are not directly relevant to our primary task.

Recommendations for students interested in data science and machine learning jobs in industry And finally, even if our deep learning models were not as powerful as we would have liked, it was still extremely useful for my own career development to have gained this deep neural network experience.

In order to get my first internship, I needed

- A basic understanding of common (non-deep) machine learing models and metrics. Basically you should be able to have someone give you a dataset (structure as a

table with the features as columns and the observations as rows) and you should be able to do something "interesting" with it (e.g., use a few columns to predict another, and give a quantitative metric of how good your model was). For this, learning to use `scikit-learn` is a good start; taking CMPS 242 will help you understand the models better, but was not strictly necessary.

- A basic understanding of statistics. Being able to code with `scikit-learn` learn is not enough if you do not understand how probability works. For example, one interview question simply asked "If I give you the result of N coin flips, how would you predict the result of the next flip?" and wanted to see if you could reason about it in a structured way. Just coming up with "intuitive" heuristics (like using the mean) is an okay start, but they often want to see if you have a good enough understanding to then figure out how to estimate uncertainties on your prediction. For this, I found AMS 206 very useful; it will teach you how to approach the problem the way a statistician does. I think further classes could be useful in being able to come up with statistically-motivated extensions to existing machine learning models, but is probably not necessary for passing the inverviews

- Connections. I cannot stress this enough: connections at these companies are incredibly useful for getting your first job. Most recruiters have never met an astronomer, and some do not know UC Santa Cruz exists. This means that you will often look like a very risky hire, unless you have someone on the inside willing to recommend you. Reach out to the network of UCSC alumni and do not be

shy about asking for those alumni to put you in touch with other people in their network.

After you get your first job or internship in industry things become much easier. It is pretty common to recieve a job offer from the company of your internship. The biggest question then shifts to what kind of a job you want. There are so many different paths (non-deep data scientist, data engineer, machine learning engineer, data analyst, etc.); I will not get into a comparison of them here. Instead, I will add a few thoughts on how to get the best-possible deep learning scientist job given my experience so far.

In order to really stand out as a machine learning scientist, I would suggest:

1. Get experience using both deep learning and traditional machine learning. Very few places want someone who can only do deep learning; on the other hand, very few "cutting edge" jobs avoid deep learning currently.

2. Do something creative with deep learning. Taking CMPS 290C is a good way to get started with this. For example, come up with some statistical extension to existing deep learning technique and apply it to some astronomical data. (Ideally you would get a scientifically interesting result, but if not, the practice experience is the most directly useful component anyway.) Make a well-designed github repo highlighting your work; maybe even a simple webpage as part of your portfolio. Get down a 5 minute pitch, aimed at computer scientists; the goal is for you to be able to go into an interview and have the interviewer feel like they came out

having learned something new.

3. Publish a conference paper at a machine learning conference. Example conferences: Neural Information Processing [NeurIPS], Computer Vision and Pattern Recognition [CVPR], or Knowledge Discovery in Databases [KDD]). Preferably do this *before* you go on the full-time job. While it is still good if (2) gets published in an astronomical journal (even as a Research Note) it can be far more valuable to publish at a machine learning-specific venue for at least 2 reasons. First, some job postings now use this as a criterion to help filter/sort for "promising" candidates; only publishing in ApJ might not cut it. Second, these conferences can be valuable for networking, both with recruiters and prospective coworkers.

As a reminder, you can still get a good job without doing these things, so do not worry too much if you have not done these things. (I certainty did not do all of them and still feel that I got a good job.) I simply offer these ideas as retrospective advice if anyone is thinking about this pathway early in their grad career and wants to optimize their prospects for getting into big name labs or cutting edge research positions (for which I was not ideally positioned).

And finally, of course, talk to others and get their advice too; I only have a limited amount of experience so far. Others might be able to provide different experiences and different angles.

Appendix A

Clustered SNe I appendix

A.1 Code Verification

A.1.1 Sedov Verification

The mass and energy of each supernova is injected into the innermost zone, with all the energy injected as thermal energy. This is not a realistic configuration; at no stage do we expect a uniformly mixed sphere, on the order of 0.1 parsecs in radius, which is over-pressured but not yet expanding. Given these convenient but unphysical initial conditions, we need to verify that our system will evolve into a realistic configuration.

We can look at an early time snapshot of a single SN simulation to verify that the system accurately relaxes into a physical configuration. At early times, cooling losses should still be negligible, so we expect our system to be in the Sedov phase. Figure A.1 shows a snapshot of our numerical results, compared with the analytic Sedov prediction. There is a noticeable overdensity at inner radii, but this is to be expected: the analytic

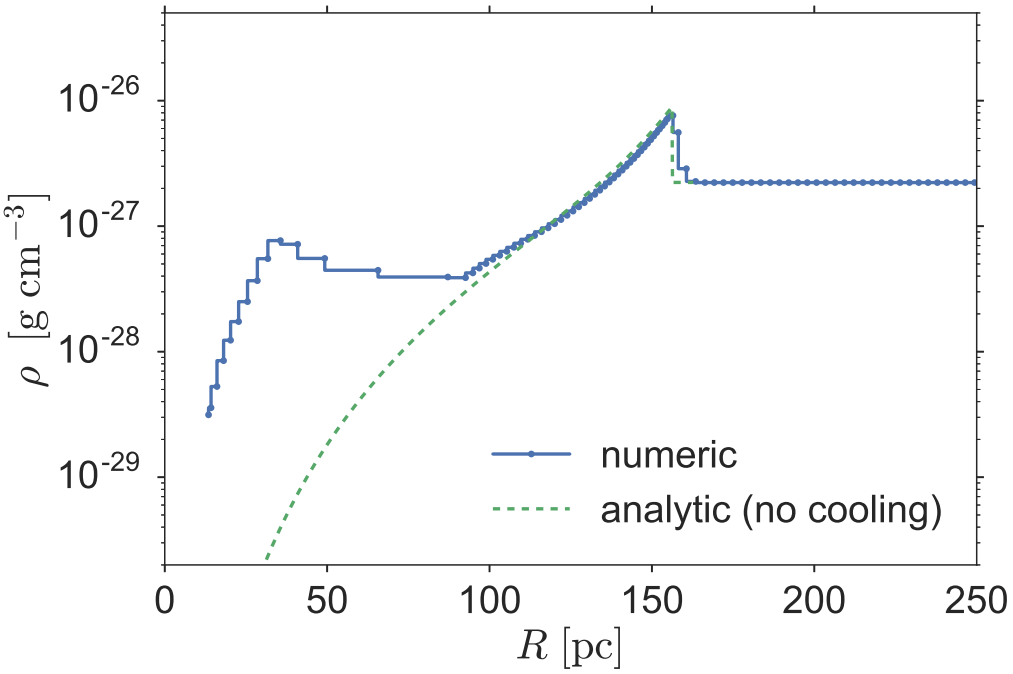

Figure A.1: Comparison of our numeric results (solid line) against the analytic Sedov solution (dashed line) for a $\rho = 1.33 \times 10^{-3}$ $m_{\rm H}$ cm^{-3}, $Z = Z_{\odot}$, $M_{\rm cluster} = 10^2$ $M_{\odot}$ ($N_{\rm SNe} = 1$) cluster, at $t = .17$ Myr.

Sedov solution assumes no ejecta mass, whereas our simulation includes ejecta mass. The extra ejecta mass appears as an overdensity at inner radii. Excepting that, our simulation is in good agreement with the Sedov prediction so we consider our injection scheme valid.

A.1.2 Thornton et al. Verification

We also verified our code against the results of Thornton et al. (1998), who measured the total energy from single SNe. We ran single SN simulations at the same background conditions as Thornton et al., fixing the SN ejecta mass to be 3 $M_{\odot}$ with

an ejecta metallicity equal to the background metallicity, and extracting results at the same time as Thornton et al.. We compare our simulations to the model provided by Thornton et al. in Figure A.2 and Figure A.3.

We judge that our residuals are comparable to the residuals present in the data of Thornton et al., and we are not surprised that there are discrepancies. We use different initial conditions: our simulation injects all of the SNe energy into the innermost zone as thermal energy, while Thornton et al. spreads the energy across 150 zones, and adds some of it as kinetic energy. We were able to use different initial conditions because we used different hydrodynamic solvers: Thornton et al. uses a finite-difference method which cannot handle the strong shock that occurs by injecting all the energy into one zone, while our finite-volume method is much more robust to these strong shock conditions. Finally, we use a cooling package that differs from the cooling function used by Thornton et al.. All these differences lead us to expect the minor discrepancies between our results and the results of Thornton et al..

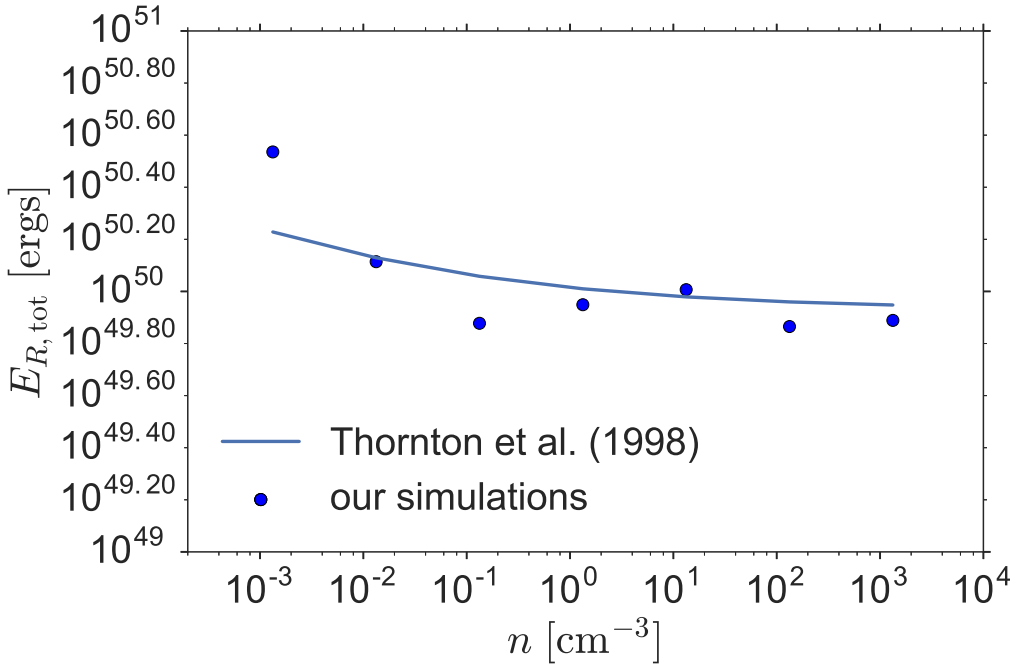

Figure A.2: Verification that our code can reproduce the results of Thornton et al. (1998), for total energy contained within the SNR ($E_{\mathrm{R,tot}}$) at the completion time defined by Thornton et al. (1998).

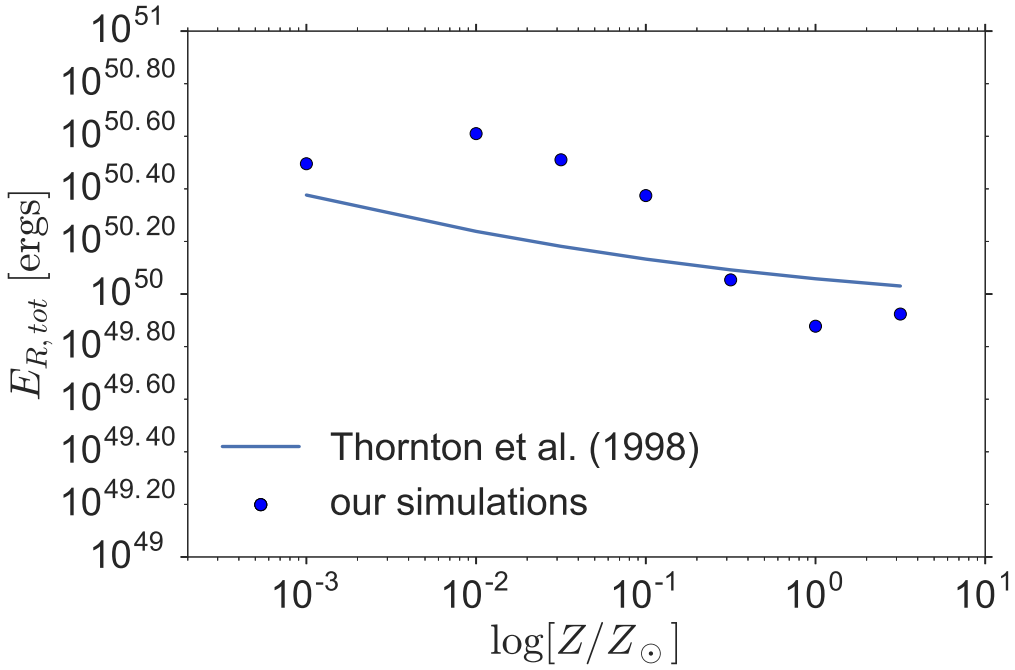

Figure A.3: Same as Figure A.2, but now with total energy as a function of metallicity.

Appendix B

Clustered SNe II appendices

B.1 Sensitivity to Initial Perturbations

In the fiducial 3D simulations presented in the main text, we set up the initial GIZMO particle positions by placing them in a uniform grid and then randomly perturbing each particle position using a Gaussian kernel with a dispersion of 10^{-3} times the initial spatial resolution. This results in an uncorrelated artificial density perturbation with a standard deviation of about 2×10^{-4} times the mean density as inferred by GIZMO's density solver, regardless of resolution.

In order to understand the effect of this perturbation, and how our results depend on its magnitude and whether that magnitude scales with resolution, we rerun a subset of our simulations with an additional perturbation. In addition to the artificial coordinate-based perturbation, we apply a "physical-like" perturbation field directly to the particle masses and densities. To realize this, we generate a white (uncorrelated)

Gaussian perturbation field with a magnitude of 5% of the mean density sampled on the grid of our highest resolution completed simulation (3D_10_HD). For our lower resolution runs, we average the perturbation over appropriately larger apertures, matching what should happen if this were a physical perturbation. This averaging results in a decreasing perturbation magnitude at worsening resolution, but the magnitude is always at least a factor of 25 larger than the standard coordinate-based perturbation, and the power spectral density of the perturbation is the same at all resolved scales in all simulations. This resolution-dependence is a key difference from the artificial perturbation in our primary runs which has a magnitude that does not change with resolution. This process also introduces minor spatial correlations, as some higher resolution particles are equidistant between lower resolution particles, and their perturbation must be shared between multiple lower resolution particles.

In Figure B.1, we show the results of rerunning our three completed 3D HD simulations (3D_10_HD, 3D_20_HD, and 3D_40_HD) with these alternative initial conditions.[1] We find that the details of the initial perturbation has very little effect compared to changing the resolution; increasing the perturbation magnitude by a factor of more than 25 has a smaller effect than increasing the spatial resolution by a factor of 2.

[1] The variant of 3D_10_HD with the additional perturbation has only been run for about 15 Myr due to its computational cost, but we do not expect our conclusions would change if it were run to completion.

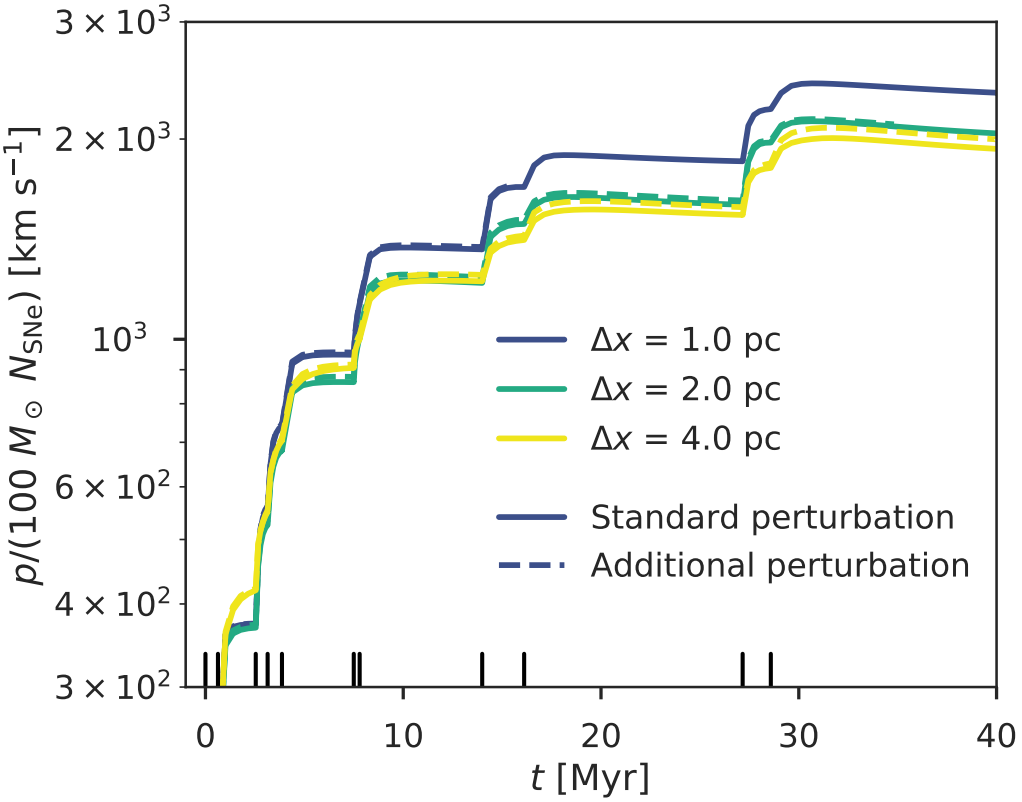

Figure B.1: Comparison of the momentum evolution of our completed 3D simulations (`3D_10_HD`, `3D_20_HD`, `3D_40_HD`), and similar simulations with an additional, stronger perturbation with magnitudes that correctly scale with resolution.

B.2 Simulation 3D_40_HD as an outlier at early times

In the resolution study (e.g. Figure 3.2) we see that the momenta of our simulations are well-ordered with respect to resolution at late times but that between the second and third SNe our lowest resolution simulation (3D_40_HD) has more momentum than our highest resolution simulation (3D_07_HD). We conjecture that this anomalous behaviour of 3D_40_HD is related to our SN injection method. As noted in subsection 3.3.1, a typical SN is added using only ~ 5 new particles in 3D_40_HD, leading to an undersampled injection kernel. While it is not clear precisely why undersampling would lead to a systematic increase in momentum, it is strongly suggestive that our simulations start behaving differently right as we hit the resolution limit of one of our methods.

Fortunately, this does not appear to affect our late-time results or our major conclusions. At early times, we recommend treating 3D_40_HD as an outlier, in which case the momentum will be monotonic with respect to resolution at effectively all times.

Appendix C

Clustered SNe III appendix

C.1 Convergence in the simultaneous energy injection method

As seen in Figures 4.5 and 4.6, even at high resolution the simultaneous energy injection method does not necessarily converge towards a deterministic result. In this section we will discuss the source of this behaviour, which touches on the limiting properties of the model with respect to several key parameters: $m_{\rm kernel}$, $N_{\rm ngb}$, $\Delta\epsilon$. And while there are many ways we could look at the convergence of this method, we will focus on the distribution of the injected energy.

First, let's look at the mean of the distribution. By design, if $m_{\rm kernel}\Delta\epsilon \geq N_{\rm SNe}E_{\rm blast}$, then the method will be unbiased. (If that condition is not true, then the method becomes deterministic and will always inject too little energy, so we will assume an appropriate value of $\Delta\epsilon$ has been chosen for the remainder of this section.)

The more interesting quantity is the variance of the injected energy. For this

method, the fractional variance is:

$$\mathrm{var}\left(\frac{\Delta E}{N_{\mathrm{SNe}} E_{\mathrm{blast}}}\right) = \left(\frac{m_{\mathrm{kernel}}\Delta\epsilon}{N_{\mathrm{SNe}} E_{\mathrm{blast}}} - 1\right) \sum_i \left(\frac{m_i^2}{m_{\mathrm{kernel}}^2}\right) \tag{C.1}$$

$$\mathrm{var}\left(\frac{\Delta E}{N_{\mathrm{SNe}} E_{\mathrm{blast}}}\right) \approx \left(\frac{m_{\mathrm{kernel}}\Delta\epsilon}{N_{\mathrm{SNe}} E_{\mathrm{blast}}} - 1\right) N_{\mathrm{ngb}}^{-1} \tag{C.2}$$

where the second form holds for resolution elements of similar mass.

This shows that so long as $m_{\mathrm{kernel}}\Delta\epsilon > N_{\mathrm{SNe}} E_{\mathrm{blast}}$, the method remains stochastic. The only way to ensure true determinism while remaining unbiased is to chose a $\Delta\epsilon$ such that $m_{\mathrm{kernel}}\Delta\epsilon = N_{\mathrm{SNe}} E_{\mathrm{blast}}$. While this might work for some methods (those with a fixed kernel mass and with an identical N_{SNe} per cluster) it does not work for all (such as those with varying m_{kernel}, stochastic number of SNe per clusters, or those with m_{kernel} so large that $\Delta\epsilon$ does not correspond to heating the gas beyond the peak of the cooling curve).

But there are still *some* cases in which this method converges towards a deterministic result. In particular, for fixed m_{kernel}, as N_{ngb} increases, the variance will approach 0. But in practice, it is typically m_{kernel} that is improved, while N_{ngb} is held fixed. We can see that that will also lead to a decrease in variance, but will eventually hit the limit $m_{\mathrm{kernel}}\Delta\epsilon = N_{\mathrm{SNe}} E_{\mathrm{blast}}$ if $\Delta\epsilon$ is not raised (which increases the variance).

When designing this method to apply across a large range of scales, the best we can do is *prescribe* how the variance should depend on resolution and total blast energy. For instance, if we set a minimum variance at high resolution (i.e. fixed the ratio of $m_{\mathrm{kernel}}\Delta\epsilon / N_{\mathrm{SNe}} E_{\mathrm{blast}}$) then the injected energy would converge to a distribution

and always remain non-trivially stochastic (e.g. the right panel of Figure 4.5 shows how energy is distributed as a scaled binomial for our implementation). If instead we adopted a different function for $\Delta\epsilon(m_{\text{kernel}}, N_{\text{SNe}})$, we could force the variance to shrink to 0 as resolution increases, resulting in a delta function for the distribution of injected energy. Dalla Vecchia & Schaye (2012) do not specify a recommended form for $\Delta\epsilon(m_{\text{kernel}}, N_{\text{SNe}})$ at high resolution, so the the exact behaviour will depend on the particular implementation.

Bibliography

Agertz O., Kravtsov A. V., Leitner S. N., Gnedin N. Y., 2013, ApJ, 770, 25

Aihara H., et al., 2018a, Publications of the Astronomical Society of Japan, 70, S4

Aihara H., et al., 2018b, Publications of the Astronomical Society of Japan, 70, S8

Antoniou A., Storkey A., Edwards H., 2017, arXiv e-prints, p. arXiv:1711.04340

Begelman M. C., McKee C. F., 1990, ApJ, 358, 375

Breiman L., 2001, Machine Learning, 45, 5

Bryan G. L., et al., 2014, ApJS, 211, 19

Castor J., McCray R., Weaver R., 1975, ApJ, 200, L107

Chevalier R. A., 1974, ApJ, 188, 501

Churazov E., Inogamov N., 2004, MNRAS, 350, L52

Cioffi D. F., McKee C. F., Bertschinger E., 1988, ApJ, 334, 252

Cowie L. L., McKee C. F., 1977, ApJ, 211, 135

Crain R. A., et al., 2015, MNRAS, 450, 1937

Creasey P., Theuns T., Bower R. G., 2013, MNRAS, 429, 1922

Dalla Vecchia C., Schaye J., 2012, MNRAS, 426, 140

Dekel A., Krumholz M. R., 2013, MNRAS, 432, 455

Dekel A., Silk J., 1986, ApJ, 303, 39

Dekel A., Sarkar K. C., Jiang F., Bournaud F., Krumholz M. R., Ceverino D., Primack J. R., 2019, arXiv e-prints, p. arXiv:1903.00962

Draine B. T., 2011, Physics of the Interstellar and Intergalactic Medium. Princeton Univ. Press, Princeton, NJ

Duffell P. C., 2016, ApJ, 821, 76

Ekström S., et al., 2012, A&A, 537, A146

El-Badry K., Ostriker E. C., Kim C.-G., Quataert E., Weisz D. R., 2019, arXiv e-prints, p. arXiv:1902.09547

Ertl T., Janka H.-T., Woosley S. E., Sukhbold T., Ugliano M., 2016, ApJ, 818, 124

Faucher-Giguère C.-A., Quataert E., Hopkins P. F., 2013, MNRAS, 433, 1970

Fawcett T., 2006, Pattern Recognition Letters, 27, 861

Ferland G. J., Korista K. T., Verner D. A., Ferguson J. W., Kingdon J. B., Verner E. M., 1998, PASP, 110, 761

Fielding D., Quataert E., Martizzi D., 2018, MNRAS, 481, 3325

Fierlinger K. M., Burkert A., Ntormousi E., Fierlinger P., Schartmann M., Ballone A., Krause M. G. H., Diehl R., 2016, MNRAS, 456, 710

Forbes J. C., Krumholz M. R., Goldbaum N. J., Dekel A., 2016, Nature, 535, 523

Gentry E. S., Krumholz M. R., Dekel A., Madau P., 2017, MNRAS, 465, 2471

Gentry E. S., Krumholz M. R., Madau P., Lupi A., 2019, MNRAS, 483, 3647

Gerritsen J. P. E., 1997, PhD thesis, , Groningen University, the Netherlands, (1997)

Goldbaum N. J., Krumholz M. R., Forbes J. C., 2016, ApJ, 827, 28

Haardt F., Madau P., 2012, ApJ, 746, 125

Hayward C. C., Hopkins P. F., 2017, MNRAS, 465, 1682

Hennebelle P., Iffrig O., 2014, A&A, 570, A81

Hopkins P. F., 2015, MNRAS, 450, 53

Hopkins P. F., Raives M. J., 2016, MNRAS, 455, 51

Hopkins P. F., Quataert E., Murray N., 2011, MNRAS, 417, 950

Hopkins P. F., Quataert E., Murray N., 2012, MNRAS, 421, 3522

Hopkins P. F., Kereš D., Oñorbe J., Faucher-Giguère C.-A., Quataert E., Murray N., Bullock J. S., 2014, MNRAS, 445, 581

Hopkins P. F., et al., 2018a, MNRAS, 477, 1578

Hopkins P. F., et al., 2018b, MNRAS, 480, 800

Hoyle B., 2016, Astronomy and Computing, 16, 34

Huertas-Company M., et al., 2015, The Astrophysical Journal Supplement Series, 221, 8

Iffrig O., Hennebelle P., 2015, A&A, 576, A95

Jenkins E. B., Tripp T. M., 2011, ApJ, 734, 65

Katz N., 1992, ApJ, 391, 502

Katz N., Gunn J. E., 1991, ApJ, 377, 365

Keller B. W., Wadsley J., Benincasa S. M., Couchman H. M. P., 2014, MNRAS, 442, 3013

Kim C.-G., Ostriker E. C., 2015, ApJ, 802, 99

Kim C.-G., Kim W.-T., Ostriker E. C., 2011, ApJ, 743, 25

Kim J.-h., et al., 2014, ApJS, 210, 14

Kim C.-G., Ostriker E. C., Raileanu R., 2017, ApJ, 834, 25

Kimm T., Cen R., 2014, ApJ, 788, 121

Kimm T., Cen R., Devriendt J., Dubois Y., Slyz A., 2015, MNRAS, 451, 2900

Kingma D. P., Ba J., 2014, arXiv e-prints, p. arXiv:1412.6980

Koyama H., Inutsuka S.-i., 2002, ApJ, 564, L97

Krause M., Fierlinger K., Diehl R., Burkert A., Voss R., Ziegler U., 2013, A&A, 550, A49

Krizhevsky A., Hinton G., 2009, Master's thesis, University of Toronto

Krizhevsky A., Sutskever I., Hinton G. E., 2012, in Pereira F., Burges C. J. C., Bottou L., Weinberger K. Q., eds, , Advances in Neural Information Processing Systems 25. Curran Associates, Inc., pp 1097–1105, http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf

Kroupa P., 2002, Science, 295, 82

Krumholz M. R., 2013, MNRAS, 436, 2747

Krumholz M. R., 2014, Phys. Rep., 539, 49

Krumholz M. R., 2017, Star Formation. World Scientific Publishing Co. Pte. Ltd., doi:10.1142/10091

Krumholz M. R., Federrath C., 2019, Frontiers in Astronomy and Space Sciences, 6, 7

Krumholz M. R., Matzner C. D., 2009, ApJ, 703, 1352

Krumholz M. R., McKee C. F., Tumlinson J., 2009, ApJ, 699, 850

Krumholz M. R., Fumagalli M., da Silva R. L., Rendahl T., Parra J., 2015, MNRAS, 452, 1447

Laigle C., et al., 2016, The Astrophysical Journal Supplement Series, 224, 24

LeCun Y., Boser B., Denker J. S., Henderson D., Howard R. E., Hubbard W., Jackel L. D., 1989, Neural Computation, 1, 541

Leauthaud A., Singh S., Luo Y., Ardila F., Greco J. P., Capak P., Greene J. E., Mayer L., 2019, arXiv e-prints, p. arXiv:1905.01433

Leitherer C., et al., 1999, ApJS, 123, 3

Lupton R. H., Gunn J. E., Szalay A. S., 1999, The Astronomical Journal, 118, 1406

Mac Low M.-M., McCray R., 1988, ApJ, 324, 776

Mac Low M.-M., Norman M. L., 1993, ApJ, 407, 207

Mac Low M.-M., McCray R., Norman M. L., 1989, ApJ, 337, 141

Maoz D., Mannucci F., Brandt T. D., 2012, MNRAS, 426, 3282

Markevitch M., Vikhlinin A., 2007, Phys. Rep., 443, 1

Martizzi D., Faucher-Giguère C.-A., Quataert E., 2015, MNRAS, 450, 504

McKee C. F., Ostriker J. P., 1977, ApJ, 218, 148

Michaut C., Cavet C., Bouquet S. E., Roy F., Nguyen H. C., 2012, ApJ, 759, 78

Mirza M., Osindero S., 2014, arXiv e-prints, p. arXiv:1411.1784

Murray N., Quataert E., Thompson T. A., 2005, ApJ, 618, 569

Noh W. F., 1987, J. Comput. Phys., 72, 78

Odena A., Dumoulin V., Olah C., 2016, Distill

Ostriker E. C., Shetty R., 2011, ApJ, 731, 41

Ostriker E. C., McKee C. F., Leroy A. K., 2010, ApJ, 721, 975

Pejcha O., Thompson T. A., 2015, ApJ, 801, 90

Pillepich A., et al., 2018, MNRAS, 473, 4077

Pontzen A., Governato F., 2014, Nature, 506, 171

Ravanbakhsh S., Lanusse F., Mandelbaum R., Schneider J., Poczos B., 2016, arXiv e-prints, p. arXiv:1609.05796

Richtmyer R. D., 1960, Communications on Pure and Applied Mathematics, 13, 297

Roy A., Nath B. B., Sharma P., Shchekinov Y., 2013, MNRAS, 434, 3572

Russakovsky O., et al., 2015, International Journal of Computer Vision (IJCV), 115, 211

Salimans T., Goodfellow I., Zaremba W., Cheung V., Radford A., Chen X., 2016, arXiv e-prints, p. arXiv:1606.03498

Schaye J., et al., 2015, MNRAS, 446, 521

Schive H.-Y., Liao M.-H., Woo T.-P., Wong S.-K., Chiueh T., Broadhurst T., Hwang W. Y. P., 2014, Phys. Rev. Lett., 113, 261302

Sharma P., Roy A., Nath B. B., Shchekinov Y., 2014, MNRAS, 443, 3463

Shetty R., Ostriker E. C., 2012, ApJ, 754, 2

Shrivastava A., Pfister T., Tuzel O., Susskind J., Wang W., Webb R., 2016, CoRR, abs/1612.07828

Simonyan K., Zisserman A., 2014, arXiv e-prints, p. arXiv:1409.1556

Simpson C. M., Bryan G. L., Hummels C., Ostriker J. P., 2015, ApJ, 809, 69

Smith B. D., et al., 2017, MNRAS, 466, 2217

Smith M. C., Sijacki D., Shen S., 2018, MNRAS, 478, 302

Spitzer Jr. L., 1978, J. R. Astron. Soc. Canada, 72, 349

Springel V., Hernquist L., 2003, MNRAS, 339, 289

Stinson G., Seth A., Katz N., Wadsley J., Governato F., Quinn T., 2006, MNRAS, 373, 1074

Sukhbold T., Ertl T., Woosley S. E., Brown J. M., Janka H.-T., 2016, ApJ, 821, 38

Tanaka M., et al., 2018, Publications of the Astronomical Society of Japan, 70, S9

Taylor G., 1950, Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences, 201, 192

Thompson T. A., Krumholz M. R., 2016, MNRAS, 455, 334

Thompson T. A., Quataert E., Murray N., 2005, ApJ, 630, 167

Thornton K., Gaudlitz M., Janka H.-T., Steinmetz M., 1998, ApJ, 500, 95

Toro E., Spruce M., Speares W., 1994, Shock Waves, 4, 25

Vikhlinin A., Markevitch M., Murray S. S., 2001, ApJ, 549, L47

Vishniac E. T., 1983, ApJ, 274, 152

Vishniac E. T., 1994, ApJ, 428, 186

Vogelsberger M., Genel S., Sijacki D., Torrey P., Springel V., Hernquist L., 2013, MNRAS, 436, 3031

Wadsley J. W., Keller B. W., Quinn T. R., 2017, MNRAS, 471, 2357

Walch S., Naab T., 2015, MNRAS, 451, 2757

Wheeler C., et al., 2018, arXiv e-prints, p. arXiv:1812.02749

Woosley S. E., Heger A., 2007, Phys. Rep., 442, 269

Woosley S. E., Heger A., 2015, ApJ, 810, 34

Wu J. F., Boada S., 2019, MNRAS, 484, 4683

Yadav N., Mukherjee D., Sharma P., Nath B. B., 2017, MNRAS, 465, 1720

da Silva R. L., Fumagalli M., Krumholz M., 2012, ApJ, 745, 145

da Silva R. L., Fumagalli M., Krumholz M. R., 2014, MNRAS, 444, 3275

de Blok W. J. G., 2010, Advances in Astronomy, 2010, 789293