MSc Physics and Astronomy

GRAPPA Track

MASTER THESIS

A Generative Model of Galaxies for the Analysis of Strongly Lensed Systems

---

by

Paul Hofma

10179607

Published July 2019

60 EC

September 2018 - July 2019

Supervisor/Examiner:

dr. Christoph Weniger

Examiner:

dr. Samaya M. Nissanke

Carried out at the GRAPPA Research Institute in the Deep Lensing *group*

"[The universe] cannot be read until we have learnt the language and become familiar with the characters in which it is written. It is written in mathematical language, and the letters are triangles, circles and other geometrical figures, without which means it is humanly impossible to comprehend a single word.

— Galileo Galilei, *Opere Il Saggiatore*, p. 171

Abstract

Research into constraining properties of Dark Matter is an ongoing topic, with many of these uncertainties ranging over many orders of magnitude. The research described herein was performed in light of the work being done by GRAPPA's Deep Lensing research group, who are currently looking into one such area of DM research: the detection of low-mass subhalos through strong gravitational lensing effects, using deep learning algorithms. For this purpose, a generative model of galaxies is required. Such a generative model could in theory be fairly easily extracted from a sufficiently accurate Variational Auto-Encoder. The majority of this research, therefore, is focused on building such a VAE. I explore several different varieties of VAE architectures, employing fully-connected as well as convolutional networks; in addition, I explore the effects of β-VAEs, Conditional Auto-Encoders, and Dense networks. The selected models are then run on comparable settings so that we can make an honest comparison, and results are compared by relative ELBO scores and by visual analysis of the images. I conclude that different architectures will have different advantages, which are rather setting specific. The Densenet-based VAE showed the greatest visually reproductive accuracy, but performed poorly at extrapolating galaxies from noisy data. Unexpectedly, the simplest architectures (both ConvNet and FC based) performed best in extrapolative ability and also displayed remarkable reconstructive accuracy. Conditionality, also, was shown to greatly enhance performance, although at the cost of numerical stability. A slight improvement was found for the β-VAE, although this effect would likely have been greater for longer training times. I conclude that the models presented are likely not yet of a high enough standard to serve as a generative model of galaxies, although this might be achieved in future research by combining the best-performing features presented here, in addition to allowing for longer training time.

Keywords: Dark matter, Strong gravitational lensing, Galaxies, Deep learning, Neural networks, Variational auto-encoder, Generative models, Reconstructive models

Acknowledgements

To begin, I would like to thank my supervisor, Christoph Weniger, without whom this project would never have gotten off the ground in the first place. Thank you for never ceasing to be optimistic about my progress (even when I was not), and for always being available as a font of wisdom and guidance when I programmed and/or mathed myself into a corner. Though I felt like a bit of a slow student at times, you never failed to have me leaving our meetings brimming with optimism and new ideas.

I would also like to extend a massive thank you to the whole *Deep Lensing* research group, specifically to Sydney Otten, Adam Coogan, and Marco Chianese, who were always more than willing to spar with me and provide new ideas when my code (still) wasn't doing terribly well. I think I would still be looking at shapeless blobs if it wasn't for you all, so, again, thank you.

I also thank SURFsara (www.surfsara.nl) for the support in using the Lisa Compute Cluster.

Widening my scope, this thesis marks the end of my years a student, and so I find myself looking back more broadly. In doing so, I find that there are many more people to whom I owe a word of thanks, though I will only be able to highlight a few.

First and foremost I would like to thank the friends I've made along the way, during my toils through the physics program. If it weren't for you, I doubt I would have made it through the first year of physics with my sanity (or self-esteem) even slightly intact. On that note, I would like to thank the teachers and mentors who helped me remain inspired by the beauty and elegance of this wonderful field of science; special thanks in this regard go out to Manus Visser, Sander Breur, and Jasper van Wezel.

Second, I owe an eternal debt of gratitude for my parents' unyielding support. No matter how long I ended up managing to take to complete my degree, you always told me to make the choices I believed were right for me - even (or especially) if that meant taking a little longer. Thank you.

In addition, I would like to thank the sport of rowing in general, for teaching me that perseverance wins out over luck and talent at the end of the day, and for showing me that wild dreams can sometimes come true, if only you give it your all. I would also like to thank it for showing me that, sometimes, you need to realize when you have given enough.

Finally, I'd like to thank Aimée for putting up with me through a large part of all this. I don't know how you do it my love, but please know that I am grateful.

Contents

Chapter 1

Introduction

1.1 Shedding Light: A short history of dark matter

Of the many mysteries still plaguing modern physics, the question of dark matter is probably one of the most appealing, even for a wider public. After all, even the very name of the topic implies a general sense of mystique: we are searching for a kind of matter of which we know extremely little, only detectable by the vague hints and clues that it leaves across the vast emptiness of space for us to find. And while it may feel like a "modern" concept for many, the fundamental idea of 'dark matter' - that is, invisible astrophysical objects which nonetheless exert some influence on the universe around them, through gravity or otherwise - has been around for a rather long time.

As early as 1783, John Michell and (1799) Pierre Laplace already brought up that objects might possibly exist that were so heavy that not even light could escape their pull - objects that we now would describe as black holes [**?**, **?**]. While this is presumably the earliest speculation of any sort of 'dark matter', in the decades that followed many more would ruminate on such mysterious, invisible kinds of matter. Rather more famously, it was probably Lord Kelvin who, in 1904, was the first to make an attempt at a dynamical estimate of the amount of dark matter in the Milky Way [**?**]. At the time, he and his colleagues were still firmly convinced that this 'dark matter' was composed of objects of a more ordinary nature: stars too dim to see with the then-current generation of telescopes, or clouds of gas or dust. Furthermore, his estimates concluded that there must of roughly equal amounts of dark and luminous matter present in our galaxy, while other research of that era concluded that if there were any substantial amount of dark matter present at all, it was negligible compared to the amount of luminous matter [**?**].

This conclusion was disturbed rather brusquely by the (rather famous) results from Zwicky's seminal 1933 article, where he found a suspiciously large range of velocities within the Coma cluster through redshift observations [**?**]. Investigating further, he then used the virial theorem in an attempt to estimate the cluster's mass. He found that, were the mass in the cluster composed only of the observable, luminous

matter, it *should* have displayed an average velocity dispersion of about 80 m/s - a number several orders of magnitude smaller than 1000 km/s which was actually observed. As such, he concluded that to explain these large spreads in velocity, we must allow for the possibility that there is in fact much more dark matter than had been previously assumed. Or, as Erik Holmberg (1940) put it at the time [**?**]:

> *It does not seem to be possible to accept the high velocities [in the Virgo and Coma cluster] as belonging to permanent cluster members, unless we suppose that a great amount of mass – the greater part of the total mass of the cluster – is contributed by dark material distributed among the cluster members – an unlikely assumption.*

As evidenced by the above quote, there was still quite some skepticism regarding this conclusion. However, over the years that followed evidence would continue rolling in, most famously from the observed rotation curves of galaxies. The naive expectation here is that we can simply combine our observations with basic orbital mechanics:

$$\frac{GM(r)}{r^2} = \frac{v(r)^2}{r} \tag{1.1}$$

$$v(r)^2 \sim \frac{M(r)}{r} \tag{1.2}$$

$$v(r) \sim \sqrt{\frac{M(r)}{r}}\,. \tag{1.3}$$

Assuming for the moment that the center of our galaxy, where the luminous mass is typically concentrated, is more or less a uniform sphere, we have a constant density ρ, and so inside the radius of that sphere R ($r \leq R$), in which is enclosed a total mass M_{encl}, we find:

$$M(r) = M_{encl}\frac{r^3}{R^3} \tag{1.4}$$

$$v(r) \sim \sqrt{\frac{r^3}{r}} \sim r\,. \tag{1.5}$$

And therefore in this inner region, as we move outward, we expect v to increase linearly, and our observations do in fact align with this expectation. However, as we move into the outer regions of a galaxy, we observe less and less luminous matter - the galaxy's range of influence is finite, and so the density decreases as we move outward. A simple model for ρ outside R might be $\rho \sim \frac{1}{r^3}$, meaning our galaxy's outer regions are essentially sphere that is uniformly decreasing in density as we go beyond R. This gives us for $r > R$:

$$M(r) \sim \int \rho(r)\, dV \sim \int r^{-1}\, dr \tag{1.6}$$

$$M(r) \sim \ln(r) \tag{1.7}$$

$$v(r) \sim \sqrt{\ln(r)/r}\,. \tag{1.8}$$

Resulting in a rotation curve that slowly drops after an initial peak slightly outside of the galaxy's inner region, and then slowly descends - while this may be a poor approximation for what is *actually* happening, we can at the very least expect the trend to be that velocities will tend to *decrease* as we move outward.

Such a trend is, however, *not* what we actually observe. In his 1939 PhD dissertation, Horace Babcock studied the rotation curve of the Andromeda Nebula, M31, and found instead a rotation curve that flattened as he went further outside of the central region ($r > R$) [**?**]. Since this applies that for $r > R$ $v(r)$ becomes constant, we conclude from equation 1.3 that the mass in the outer reaches of the galaxy in fact scales proportionally to r: $M(r) \sim r$ for $r > R$. Babcock, too, concluded as such in his thesis, stating that this implied the existence of a large amount of invisible (or: dark) mass in the outer reaches of the galaxy - though at the time he was still very conservative in his conclusions, stating that the cause of this phenomenon might more likely be due to some physical effect that he had forgotten to take into account.

Eventually, by the late 1970s, it would be the accumulation of accounts of such flat rotation curves, most famously those by Rubin and colleagues [**?**], in combination with 21cm (hydrogen line) observations showing that this additional matter could not just be made up of gas [**?**], that would convince the scientific community that, indeed, there had to be something wrong with the current understanding of galaxies. By the late 1980s, *all* baryonic candidates for dark matter (historically often named MACHOs: MAssive Compact Halo Objects) had been excluded one by one, leaving two options: either our understanding of gravity itself is wrong, or we are dealing with some sort of Weakly Interacting Massive Particle (WIMP for short)[1].

The first option still tantalizes many a theoretical physicist, and the collective theories have become known as MOND theories: MOdified Newtonian Dynamics. While there have been some predictive success for various varieties of such theories (for example [**?**, **?**, **?**, **?**]), a major blow would appear to have been dealt by the article published by Clowe et al. in 2006 [**?**], tellingly titled: "A direct empirical proof of the existence of dark matter". Quoting the article (emphasis added):

> *By using both wide-field ground based images and HST/ACS images of the cluster cores, we create gravitational lensing maps which show that the gravitational potential does not trace the plasma distribution, the dominant baryonic mass component, but rather approximately traces the distribution of galaxies. An 8-sigma significance spatial offset of the center of the total mass from the center of the baryonic mass peaks* ***cannot be explained with an alteration of the gravitational force law, and thus proves that the majority of the matter in the system is unseen.***

While this certainly did not mean the absolute end for MOND theories (for a relatively recent, influential example, see [**?**]), it firmly re-established the position WIMPS already had as the most likely DM candidate. Even so, WIMPs had been favoured for some time by most (astro)particle physicists, and so a substantial body of work was built up over the years accounting for it, trying to discover its properties, and fitting it into existing (or new) theories and models; perhaps most notably, it was in this time that the ΛCDM model arose as a way to combine the separate observations and predictions made by cosmologists of that era, including inflation and the structures found in the CMB [**?**].

During the late 1970s and early 1980s, astrophysicists got a new toy to play with: it was around this time that it first became realistically possible to perform N-body simulations that were actually useful (for a good selection of examples, see Refs. in [**?**]). These simulations investigated the potential formation of

[1] Note that at this point in time, this category was of course much broader than the modern-day theory of WIMPs.

structure from separate particles, and it turned out to be very sensitive to the initial velocities of particles. This allowed cosmologists at the time to make a distinction between the effects of different 'temperature' classes of dark matter for the first time.

For example, one possible dark matter candidate at the time was the SM neutrino. These would have decoupled from the thermal equilibrium when their temperature (and therefore their thermal velocity) was still very high - 'hot' dark matter. Other proposed candidates (like neutralinos) are predicted to freeze out at a much lower temperature, where they are typically non-relativistic, and so these kinds of particles are described as 'cold' dark matter. While structure formation on the largest scales (those of galaxy clusters and superclusters) is generally insensitive to the DM temperature for most models, there are significant differences with regards to small scale structure for different initial dark matter velocities. The problem with hot dark matter is that small-scale density fluctuations in the early universe would have tended to get washed out by the thermal motions of individual DM particles.

In addition, the way in which structure is formed differs greatly between the two ends of the temperature spectrum. In a scenario with hot dark matter, any small scale structure would initially tend to collapse, evolving large scale structure first, and only developing small scale structures as these massive halos eventually fragmented: structure forms "top-down". In contrast, cold dark matter would allow the small-scale density fluctuation from the early universe to survive much more easily, as the dark matter would not tend to move around as much. This means that structure forms "bottom-up": first the smallest halos form, which eventually grow in size through successive mergers to produce halos of all sizes. As a result, we would expect that in a universe with hot dark matter, small-scale structures would be suppressed. Over time, the evidence of these simulations fairly quickly disfavored hot dark matter, causing cold dark matter to rise up as the prominent theory - which is what the C in ΛCDM stands for.

However, there is still plenty of research being done regarding the likelihood of certain temperatures of dark matter (for example: [**?**]), and the distinction between 'hot' and 'cold' dark matter need not necessarily be a very hard line. Warm dark matter, an intermediate temperature category, is still currently under investigation (for example: [**?**, **?**, **?**]). We will go into some further detail on the effects of dark matter temperature and how it is experimentally constrained in chapter 2.

1.2 *Deep Lensing*: Combining deep probabilistic programming with strong lensing

As we have seen, much of the past decades of dark matter research was dominated by the question of whether dark matter, as we understand it it now . This was mostly done by theorists (in the broadest sense), extrapolating from data such as galaxy surveys or CMB data. These days the existence of dark matter is seen as a given, and so most research in the field is instead focused on narrowing down the properties of dark matter. The parameter space allowed by the current set of viable dark matter models is enormously large, allowing for at least 30 orders of magnitude of freedom for the mass, and 40 orders of magnitude for the interaction cross section with a proton, for example [**?**, **?**].

Most of the current high-profile research focuses in one of three areas: direct searches, indirect

searches, and collider searches [**?**]. Direct searches, such as the XENON100 or LUX experiments, attempt to measure the direct nuclear recoil resulting from the scattering of dark matter particles off of a nucleus [**?**, **?**]. Indirect searches, such as the Fermi-LAT and ANTARES/KM3NeT experiments, instead attempt to detect the particles that would result from the decay or annihilation of dark matter particles [**?**, **?**]. Collider searches, such as those performed at the LHC by ATLAS and CMS, rather attempt to produce dark matter through high-energy particle collisions; large amounts of missing transverse momentum in the final states of such collisions could be evidence of the creation of dark matter.

However, simulation-based dark matter research is once again coming in vogue, as evidenced by recent research (some examples: [**?**, **?**, **?**]). In these papers, the authors mostly employ deep learning techniques in an attempt to efficiently distill information from available datasets. These techniques have in recent years drifted over from 'pure' computer science, where they are mostly used for tasks like object or facial recognition, into other fields like physics, mostly due to their nearly limitless versatility and powerful ability to find structures and patterns.

One example of such patterns that we may be able to find through deep learning methods, are low-mass dark satellites of galaxies. Groundbreaking work has recently been done on detecting such dark satellites by observing their strong lensing effects [**?**, **?**, **?**]. In these studies, the main focus is in trying to find (and accurately describe) the perturbations induced in Einstein rings by such galaxies, which act as small foreground lenses; we shall further discuss this line of research in chapter 2. A major problem here is that there are very few such observations available, and even fewer of those in sharp enough detail to draw any definitive conclusions. Hopefully such research will receive a boost when the ELT becomes operative in 2024, as this should make highly detailed images more readily available.

GRAPPA's *Deep Lensing* research group, in the context of whose work this research was carried out, is currently doing research in which these simulation and lensing based approaches are effectively combined. Their methodology is similar to a recent nature paper by Morningstar et al. (2019) [**?**], who describe a deep learning procedure in which they use a neural network to figure out the lens and shear perturbations working on a system, and then feed this to a second network, which is able to extract an approximate model for the source galaxy; this method produced promising results. The *Deep Lensing* group instead starts from a position of wanting to find the effects of subhalos. For this specific variant of the problem, there are three issues with the methodology employed by Morningstar et al.

First, their networks are basing the shape of the source purely on the effects of the main lens - since the system has no knowledge whatsoever of what sources are actually supposed to look like, it would take an incredibly large amount of training samples to break the degeneracy between source substructure and subhalo perturbations. Second, the network which extracts the source model has to be trained on lensed galaxy images, so that information about the lens and shear becomes baked in. As a result, one would need to retrain this network entirely if we want to change the lens model. Finally, the specific type of network they employ (a Recurrent Inference Machine, or RIM) is only applicable to linear problems of the form $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{n}$ [**?**]; if we are dealing with a scenario of line-of-sight halos, the lensing equation becomes nonlinear, and so this approach would not work at all.

The approach put forward by the *Deep Lensing* group, which aims to solve all of these problems, is intended to be able to take any lensed galaxy image, deduce the lensing and shear parameters of the

system, and from there perform an iterative analysis, comparing the result against a model of sources, combined with a model for subhalo perturbation. As a final result they would then be able to distill from this both the shape of the source galaxy as well as the location and size of any foreground subhalos.

1.3 Goals and structure of this thesis

In this research, the main goal was to build a Variational Auto-Encoder (VAE, [**?**]) that would be capable of reconstructing images of galaxies. From such a VAE a generative model of galaxies for the *Deep Lensing* pipeline could then in principle be extracted.

Our requirements for such a reconstruction model are simple: we want it to be able to always reproduce large-scale structures, while also at least reproducing smaller-scale substructures a large majority of the time. If it fails to reproduce substructures, this might not translate into a very large reconstruction error since these tend to be much less bright than the main structure; however, such levels of accuracy would certainly be necessary for any kind of real application, as it is precisely in the details that deviations caused by potential foreground lenses would be found.

To this end, I have structured this thesis as follows. To start, we will look at the relevant theoretical parts of the research in more depth in chapter 2, going over some details regarding constraints on dark matter and details on gravitational lensing, both of which are major components of the *Deep Lensing* pipeline. After this, we will go into detail on the workings of neural networks, different layers and activation functions, and variational auto-encoders in specific.

In chapter 3, we will focus on the methodology: I will discuss the dataset used for training the different network architectures, as well as the different VAE architectures that were tried out and compared over the course of this research. Then, in chapter 4, I will show and discuss the results of testing these different VAE architectures, again going over each architecture in turn. In the final chapter, I draw my conclusions regarding which architecture works best, and make several recommendations for further improvements for future research.

Chapter 2

Theory & Literature Review

2.1 Dark matter constraints

As we have already discussed some of the basics regarding recent and historical dark matter research, this section will be limited to three subjects that are of most interest to the research at hand. First, we will discuss Warm dark matter and Cold dark matter models, and specifically where their predictions differ most noticeably. Second, we will go into some detail regarding Lyman-α-Forest observations, and how these affect the current state of dark matter research and temperature constraints specifically. Finally, we will discuss the most prevalent models for modeling dark matter, being the NFW and Einasto profiles.

2.1.1 The effects of dark matter temperature

In 2002, Bode, Ostriker and Turok [**?**] noted a two important ways in which the then-current CDM models appeared to fail when compared against the actual data:

1. The theory predicts a large number of small halos, which was not observed. For example, within a halo as large as the Milky Way, the theory predicted on the order of 500 low-mass satellite galaxies with mass $> 10^8 M_\odot$, where only 11 were observed at the time.

2. CDM simulations resulted in halos with more concentrated cores than what was inferred from galaxy rotation curves.

The authors do note, however, the remarkable predictive success of the ΛCDM theory with $\Omega_M \sim 0.3, \Omega_\Lambda \sim 0.7$, and so conclude that it is only sensible to seek a (minor) modification of the model that addresses the above concerns. They propose ΛWDM: a model that includes Warm dark matter or WDM (which had been investigated and swiftly rejected before, as discussed in the introduction) as a replacement of the CDM in the ΛCDM model. For clarity's sake, we will note again that WDM is simply HDM, cooled down - we are left with dark matter particles that do have some non-negligible streaming speed,

thereby damping damping the formation of small scale structures. Bode, Ostriker and Turok found that replacing CDM with WDM had the following 7 effects, quoting:

1. Smoothing of massive halo cores, lowering core densities and increasing core radii.
2. Lowering greatly the characteristic density of low-mass halos.
3. Reduction of the overall number of low-mass halos.
4. Suppression of the number of low-mass satellite halos in high-mass halos.
5. Formation of low-mass halos almost solely within caustic pancakes or ribbons connecting larger halos in a "cosmic web." Voids in this web are almost empty of small halos, in contrast to the situation in CDM theory.
6. Late formation ($Z < 4$) of low-mass halos, in a top-down process.
7. Suppression of halo formation at high redshift ($Z > 5$) and increased evolution of halos at lower redshifts relative to CDM.

The authors note several ways in which we might test the viability of such a ΛWDM versus a ΛCDM model; an important constraint could be found through the observations of Lyman-α forest distributions at high redshifts.

2.1.2 Lyman-α constraints

The idea of the Lyman-α-forest, or LAF, was first discussed in Lynds' 1971 article [**?**], in which he discussed the absorption-line spectrum of quasar 4C 05.34. At the time, this quasar had the largest redshift of any object ever observed, and Lynds noted the presence of an oddly large number of very narrowly spaced absorption lines in the quasar's spectrum. Lynds then concluded that these narrowly-spaced absorption lines were all due to the same Lyman-α transition. It has since been well established that the blueward lines in these kinds of spectra is produced by an inhomogeneously distributed, warm, intergalactic medium consisting mostly of hydrogen gas (hence the Lyman-α lines) along the line of sight [**?**]. The LAF can therefore be used as probe of the matter power spectrum up to very high redshifts, by analyzing the opacity fluctuations in these spectra.

This matter power spectrum is highly sensitive to the 'temperature' of dark matter. This was, for example, nicely shown by Bode, Ostriker and Turak (see figure 2.1) [**?**]. Other authors, too, have continued to put constraints on the mass (and other properties) of potential WDM particles through LAF observations; for some examples from throughout the past decade, see [**?**, **?**, **?**, **?**].

Intuitively, LAF observations allow us to deduce (for a given line of sight) how many hydrogen clouds are in the line of sight, how far apart they are, and how dense they are - which in turn allows us to make deductions regarding the density and distribution of (dark) matter along those lines of sight. This, in turn, tells us something about the temperature of dark matter. After all, for a WDM model, we would expect the smallest-scale structures to get washed out, and so we expect intergalactic voids that are truly empty,

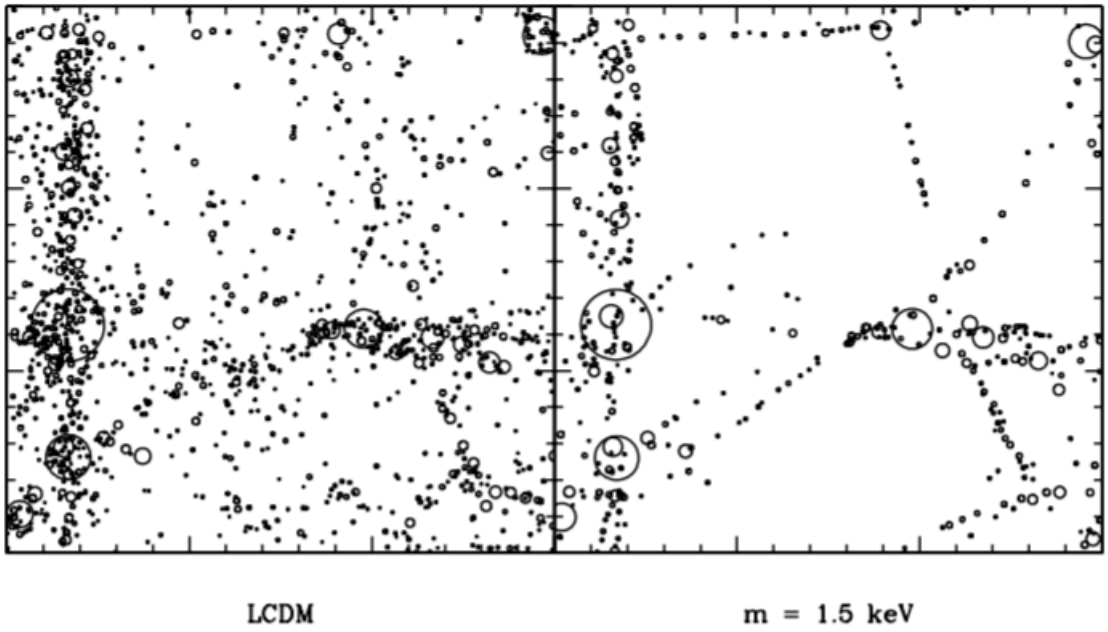


Figure 2.1: From [**?**], showing simulation results for ΛCDM compared to ΛWDM. Original caption: "Position of gravitationally bound halos at redshift $Z = 0$ in a 3 h^{-1} Mpc box, for ΛCDM *(left)* and ΛWDM with $m_x = 1.5$ keV *(right)*. The radius of each circle is r_{200}. The smallest halo mass is $2.7 \times 10^7 h^{-1} M_\odot$. Finite box size effects are visible in the orientation of the largest pancake, but it is clear that the environment of small halos is distinct between the two scenarios." Most relevantly, note how the intergalactic voids are much more sharply defined for the ΛWDM model, with far fewer 'stray' halos. Image ©AAS, reproduced with permission.

rather than interspersed with small clouds of gas; along these lines of sight, we should expect fewer LAF observations. For a CDM model, however, we expect a larger number of small-scale halos, and we expect them to be more dispersed; this should be reflected in the LAF observations, given that our equipment is sensitive enough.

2.1.3 Gravitational detection of low-mass subhalos

As mentioned, the temperature of dark matter candidates is constrained (in part) by the density of low-mass halos. In interesting recent line of research is exemplified by a 2012 nature paper by Vegetti et al. [**?**]. The authors discovered a dark satellite galaxy in the gravitational lens system JVAS B19381+666 by observing a slight perturbation in the Einstein ring caused by the main galaxy; using strong lensing analysis, they were able to determine not only the presence of a dark, low-mass galaxy, but also were able to put constraints on its mass. This, and similar research, is in part what inspired the research as done by the *Deep Lensing* group - constructing an algorithm that can analyse large datasets and search for these kinds efficiently will greatly benefit this line of research, allowing us to put much greater constraints on the mass function beyond the local universe.

2.1.4 Dark matter profiles

When modeling dark matter on smaller (i.e. galactic or stellar) scales, it is important to consider the *shape* of dark matter halos; this is commonly described using density profiles. The most ubiquitous model in use today is the NFW profile, first put forth by Navarro, Frenk and White [**?**]. Using a suite of N-body simulations, they found that regardless of the specific cosmology used, this density profile was a good approximation to the density of the dark matter halos found in their simulations (see also figure 2.2). The profile gives the density of dark matter halos, as a function of radius, as:

$$\rho(r) = \frac{\rho_s}{\frac{r}{r_s}\left(1+\frac{r}{r_s}\right)^2}\,, \tag{2.1}$$

where the scale density ρ_s and the scale radius r_s are constants specific to the particular halo we are studying.

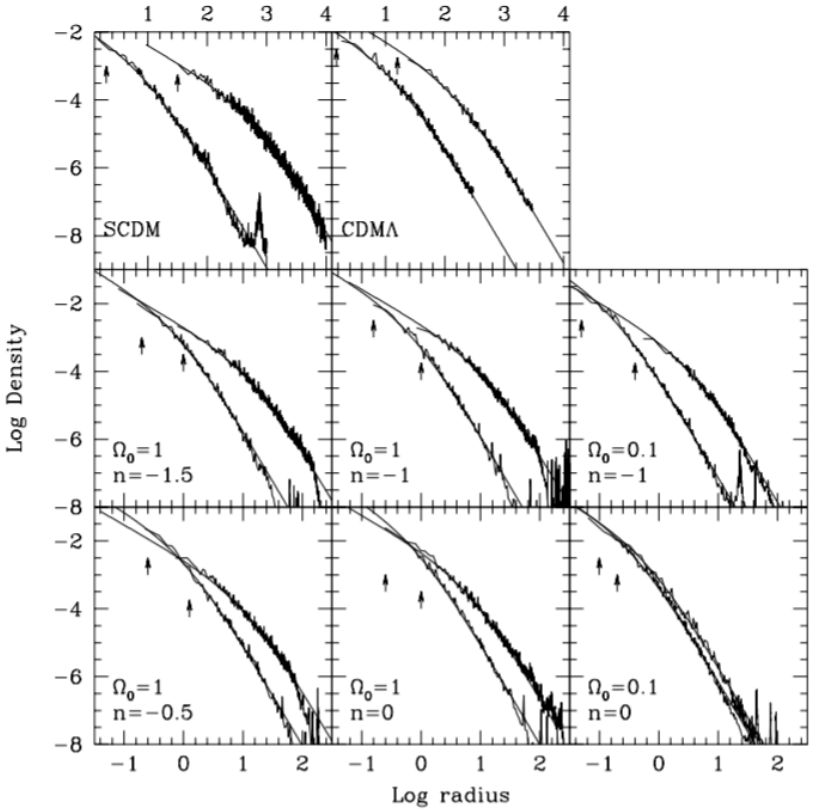


Figure 2.2: From [**?**]. Fitted NFW profiles for various cosmologies, plotted against the results from the simulations carried out by the authors, showing remarkably good agreement. Image ©AAS, reproduced with permission.

There is, however, one major problem with the NFW profile (and, as it turned out, with N-body simulation of dark matter in general), which has become known as the *core-cusp problem*. Initially, the problem appeared to be that the central density cusp of the NFW profile, which goes as $\rho \sim r^{-1}$, was not steep enough when compared to state of the art N-body simulations. However, observational evidence showing that dark matter core density slopes are actually typically much shallower than the proposed r^{-1} has also been mounting since the early 2000s (see, for example, [**?**]). While spatial resolution for

both simulations and observations continue to improve, this apparent contradiction has of yet not been resolved. However, in the meantime a 'new' density profile that may resolve some of these issues has risen to significant prominence.

The Einasto profile, which was already proposed by Jaan Einasto in 1965 [**?**], was put forward again by Navarro et al. in 2004 [**?**] as a solution to the discrepancies between the latest simulations and the NFW profile, while at the same time being much easier to reconcile with the observational data. The Einasto profile is of the form:

$$\ln\left(\frac{\rho_\alpha(r)}{\rho_{-2}}\right) = \frac{-2}{\alpha}\left[\left(\frac{r}{r_{-2}}\right)^\alpha - 1\right], \tag{2.2}$$

where ρ_{-2} and r_{-2} are again constants to be fitted to the specific halo, taking again the role of 'scale density' and 'scale radius'; in addition, the shape parameter α is also a free parameter, and it allows for the model to have a freedom in the 'cuspy-ness' of the halo's core.

2.2 Gravitational lensing

Generally, the deflection of light rays by the gravitational pull of massive bodies is referred to as *gravitational lensing*. We typically speak of a background object being lensed by a foreground object - in this case, the light originating from the source object is distorted by the gravitational pull of the lens object. Three kinds of lensing are typically distinguished:

1. Weak lensing. In this case, the deflection of the light from the source is significantly minimal (weak) that we cannot say with any certainty whether a foreground lens is actually present from a single image. Foreground lenses therefore can only be discovered through statistical methods, combining results from a multitude of images, attempting to deduce the presence of a foreground object from the collective shear and magnification of background objects.

2. Strong lensing. In this case, the effects of the lens are strong enough to produce multiple images, arcs, or even the famous Einstein rings. The presence of a foreground object is much more obvious, but these cases are also much rarer.

3. Microlensing. In this case, the background and foreground images are much smaller - of stellar or planetary scale. We can detect such lensing effects by monitoring the apparent brightness of the lensed source for the length of time it takes the lens to pass by it - as the lens is passing, it will bend the light sent out by the source object, changing the brightness over time, which can be detected by sufficiently sensitive telescopes. Note that detection by microlensing is only possible if the pass-by happens on a sufficiently short timescale (hours to months, rather than years), as otherwise the change in brightness is typically too small to be observed.

Strong lensing is the type of lensing relevant to our research here, as we will only be looking at (and, eventually, reconstructing) single images of galaxies. In the following sections, we will give a brief overview of some of the most relevant details of gravitational lensing for modeling purposes. First, we

will shortly discuss the lensing equation and the gravitational displacement field generated by gravitational potentials. Then we will discuss a few different lensing profiles, and shortly discuss how to model them.

2.2.1 Lensing formalism

The gravitational lensing formalism is, in broad strokes, as follows [**?**, **?**]. We start with a situation as shown in figure 2.4, where we have a massive object (the lens) in between a source and an observer. Since we are dealing with astronomical scales, we can (nearly) always apply the thin lens approximation; in this approximation, we assume that the distance between the source and the lens as well as the distance between the observer and the lens is much greater than the size of the lens. As a result, we approximate the lens as being 'flat'.

Light rays originating from the source traveling near enough to the lens will be deflected so that they end up at the observer; these are deflected by the deflection angle $\hat{\boldsymbol{\alpha}}$, which equals, for an incoming photon with impact parameter $\boldsymbol{\xi}$:

$$\hat{\boldsymbol{\alpha}} = \frac{4GM}{c^2 \boldsymbol{\xi}} \,. \tag{2.3}$$

We further define the angular separations $\boldsymbol{\beta}$ and $\boldsymbol{\theta}$, which are the angle between the optic plane and the source, and the optic plane and the image, respectively. We define the distances D_s, D_d, and D_{ds}, which are the distances between the observer and the source, the observer and the lens, and the lens and the source, respectively. We then introduce the reduced reflection angle:

$$\boldsymbol{\alpha} = \frac{D_{ds}}{D_s} \hat{\boldsymbol{\alpha}} \,. \tag{2.4}$$

From simple geometry, one can then deduce

$$\boldsymbol{\theta} D_s = \boldsymbol{\beta} D_s - \hat{\boldsymbol{\alpha}} D_s \,. \tag{2.5}$$

allowing us to arrive at a deceptively simple equation for the source and image separation, which is generally known as the *lens equation*:

$$\boldsymbol{\beta} = \boldsymbol{\theta} - \boldsymbol{\alpha}(\boldsymbol{\theta}) \,. \tag{2.6}$$

An important note for the lens equation is that it typically is non-linear for line-of-sight lensing systems: as a result, we obtain multiple projections $\boldsymbol{\theta}$ for every source point $\boldsymbol{\beta}$, resulting in *multiple images* of lensed galaxies. An excellent example of this is figure 2.3, which shows Einstein's cross. The famous Einstein rings are observed when our source and lens align in such a way that the image is stretched along an arc or - in the case of a near-perfect alignment - a ring. In this case the solution to the non-linear lens equation has a degenerate solution, projecting not to one or more specific locations, but rather stretching the image across an arc or ring at a certain radius.

Note that, in order for this equation to also hold in curved spacetimes, $D_{s,d,ds}$ are defined so that the relation

$$\text{separation} = \text{angle} \times \text{distance}$$

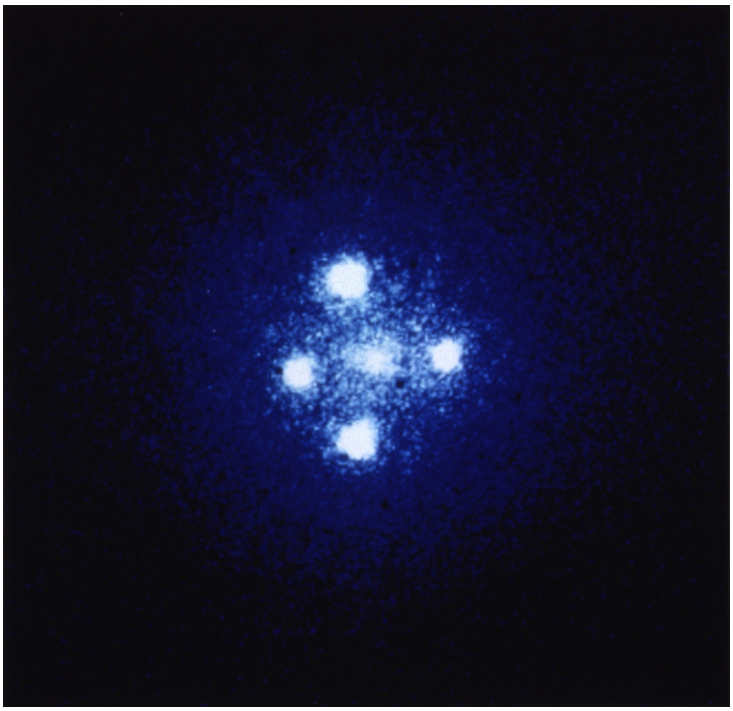


Figure 2.3: An image of Einstein's Cross from NASA's Hubble Space Telescope. This is a particularly famous example of a multiply-lensed source - the background quasar has four distinct projections around the foreground galaxy.

holds regardless of curvature[1]; such distances are called angular diameter distances. Note that this also implies that, generally, $D_s \neq D_d + D_{ds}$.

2.2.2 The deflection field

With the lens equation in hand, one can then proceed to look at the deflection field; or, to look at it differently: we consider how all points from the source space ($\boldsymbol{\beta}$) are remapped to the image space ($\boldsymbol{\theta}$). To properly model this for non-point mass lenses (i.e. lenses that are spread out in space, such as galaxies), the effective lensing potential $\psi(\boldsymbol{\theta})$, obtained by projecting the Newtonian potential on the lens plane and properly rescaling it, is a particularly useful quantity:

$$\psi(\boldsymbol{\theta}) = \frac{D_{ds}}{D_d D_s} \frac{2}{c^2} \int \Phi(D_l \boldsymbol{\theta}, z)\, dz\,. \tag{2.7}$$

The lensing potential has two important properties; first, its gradient gives the reduced deflection angle $\boldsymbol{\alpha}$:

$$\boldsymbol{\nabla}_\theta \psi = D_d \boldsymbol{\nabla}_\xi \psi \tag{2.8}$$

$$= \frac{D_{ds}}{D_s} \frac{2}{c^2} \int \boldsymbol{\nabla}_\perp \Phi\, dz \tag{2.9}$$

$$= \boldsymbol{\alpha}\,. \tag{2.10}$$

[1] though only as long, of course, as distances are very large (i.e. as long as the thin lens approximation holds), so that $\sin(x) \approx x$

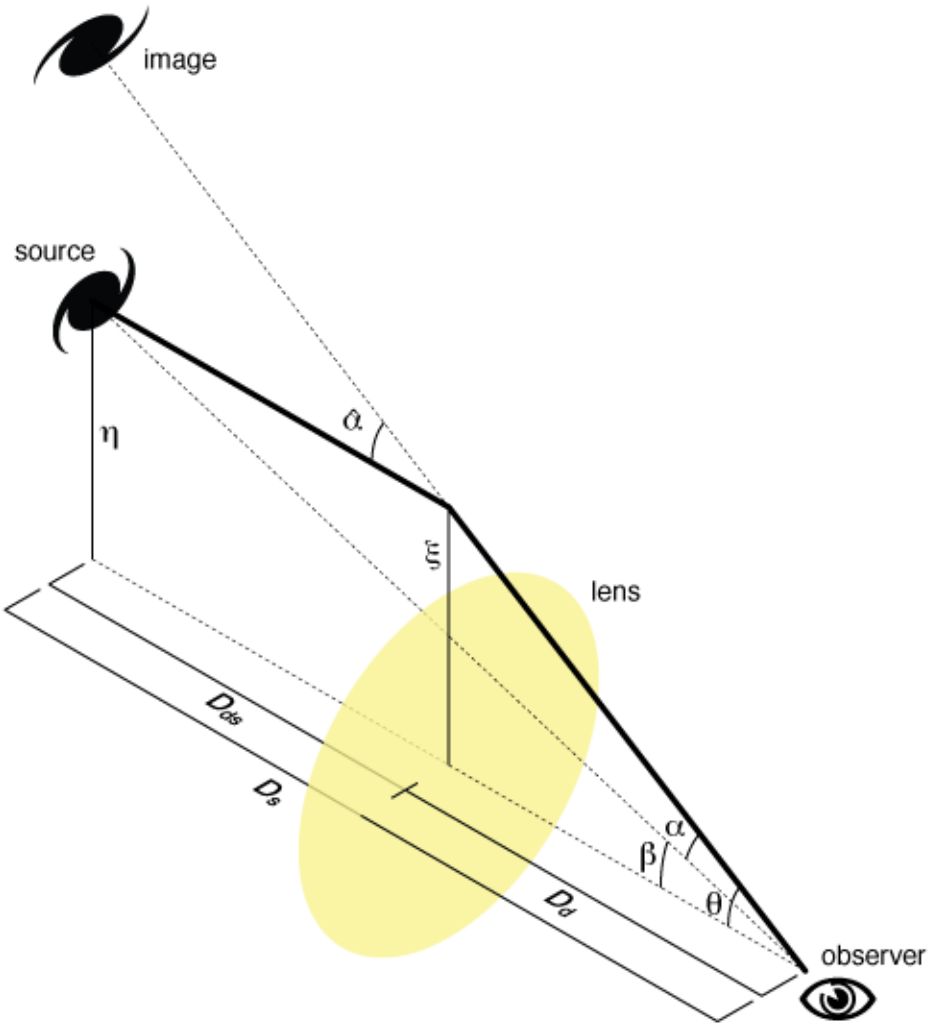


Figure 2.4: A typical gravitational lens scenario. From Wikimedia Commons, courtesy of Michael Sachs.

Second, the divergence of ψ equals twice a quantity that has been dubbed the convergence κ:

$$\Delta_\theta \psi = \frac{D_d D_{ds}}{D_s} \frac{2}{c^2} \int \nabla_\xi^2 \Phi \, dz \tag{2.11}$$

$$= \frac{D_d D_{ds}}{D_s} \frac{2}{c^2} 4\pi G \Sigma(\boldsymbol{\theta}) \tag{2.12}$$

$$= 2 \frac{\Sigma(\boldsymbol{\theta})}{\Sigma_{cr}} \equiv 2\kappa(\boldsymbol{\theta}) \, . \tag{2.13}$$

Here Σ is the surface mass density, and we have condensed (nearly) all constant factors into Σ_{cr}, the critical surface mass density. Note that we only actually get strong lensing effects for $\Sigma > \Sigma_{cr}$ [**?**].

With the lensing potential defined, we now consider the transformation between source and image space as a Taylor expansion:

$$\beta_i \approx \frac{\partial \beta_i}{\partial \theta_j} \theta_j + \frac{1}{2} \frac{\partial \beta_i}{\partial \theta_j \partial \theta_k} \theta_j \theta_k + ... \tag{2.14}$$

Taking the first-order approximation, we can define the (Jacobian) transformation matrix $\mathcal{A}$:

$$\beta_i \approx \frac{\partial \beta_i}{\partial \theta_j}\theta_j = \mathcal{A}_{ij}\theta_j \,, \tag{2.15}$$

$$\mathcal{A}_{ij} \equiv \frac{\partial \beta_i}{\partial \theta_j} = \left(\delta_{ij} - \frac{\partial \alpha_i}{\partial \theta_j}\right) = \left(\delta_{ij} - \frac{\partial^2 \psi(\boldsymbol{\theta})}{\partial \theta_i \partial \theta_j}\right) \,. \tag{2.16}$$

Then, defining:

$$\psi_{ij} = \frac{\partial^2 \psi(\boldsymbol{\theta})}{\partial \theta_i \partial \theta_j} \tag{2.17}$$

we can shorten this to:

$$\mathcal{A}_{ij} = \delta_{ij} - \psi_{ij} \,. \tag{2.18}$$

If we now split off an anisotropic part, we are left with with an antisymmetric, trace-free matrix, called the *shear matrix*:

$$\mathcal{A} - \frac{1}{2}\mathrm{tr}\mathcal{A} \cdot I = \begin{pmatrix} -\frac{1}{2}(\psi_{11} - \psi_{22}) & -\psi_{21} \\ -\psi_{12} & \frac{1}{2}(\psi_{11} - \psi_{22}) \end{pmatrix} \,. \tag{2.19}$$

As the name would suggest, this part of $\mathcal{A}$ describes the shear, or the physical distortion of the source (up to first order). We define the shear as the pseudo-vector $\boldsymbol{\gamma} = (\gamma_1, \gamma_2)$, with components:

$$\gamma_1 = \frac{1}{2}(\psi_{11} - \psi_{22}) \tag{2.20}$$

$$\gamma_2 = \psi_{12} = \psi_{21} \tag{2.21}$$

and eigenvalues:

$$\pm\sqrt{\gamma_1^2 + \gamma_2^2} = \pm\gamma \,. \tag{2.22}$$

As such, we can define a coordinate rotation with angle ϕ so that:

$$\begin{pmatrix} \gamma_1 & \gamma_2 \\ \gamma_2 & -\gamma_1 \end{pmatrix} = \gamma \begin{pmatrix} \cos 2\phi & \sin 2\phi \\ \sin 2\phi & -\cos 2\phi \end{pmatrix} \,. \tag{2.23}$$

The remainder of $\mathcal{A}$, in the meantime, gives us an expression in terms of the convergence, which is obvious from equation 2.11:

$$+\frac{1}{2}\mathrm{tr}\mathcal{A} \cdot I = \left(1 - \frac{1}{2}[\psi_{11} - \psi_{22}]\right)\delta_{ij} \tag{2.24}$$

$$= \left(1 - \frac{1}{2}\Delta\psi\right)\delta_{ij} \tag{2.25}$$

$$= (1 - \kappa)\delta_{ij} \,. \tag{2.26}$$

Our transformation matrix, $\mathcal{A}$, now can be written out in the the following form:

$$\mathcal{A} = (1 - \kappa)\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} - \gamma \begin{pmatrix} \cos 2\phi & \sin 2\phi \\ \sin 2\phi & -\cos 2\phi \end{pmatrix} \,. \tag{2.27}$$

Through this form, the effects of the separate shear and convergence parts become easily visible: the convergence part rescales the image by a constant factor in all directions, while the shear part stretches the original shape along one particular direction.

A final important consequence of our deflection transformation is that the images are amplified; as the source surface brightness is not changed through the transformation, a decrease of apparent surface through the transformation should naturally result in an increased flux - vice versa, if the surface area after the transformation is increased, we get a decrease in flux. It is then clear that the magnification is quantified by the inverse determinant of our transformation matrix $\mathcal{A}$:

$$\mu \equiv \det M = \det \mathcal{A}^{-1} = \frac{1}{1 - \kappa^2 - \gamma^2} \,. \tag{2.28}$$

Note that this final step is not actually necessary for modelling; we get this effect 'for free' by applying the transformation. However, an important note here is that higher-order transformation terms are typically necessary to get any appreciable accuracy, as these, too, are very capable of introducing significant changes in source projection. As the derivation is fairly similar to the steps above, we will not again replicate it here; we would point any interested readers to section 2.5 of Massimo Meneghetti's (rather excellent) lecture notes [**?**].

2.2.3 Lens profiles

A final ingredient necessary for modeling lensing systems, then, becomes the exact lens potential associated with the lenses in our system; for this, what is essentially required is a properly accurate mass distribution of the lens. For the *Deep Lensing* pipeline, we need to consider two different types of lenses: the main galaxy, which provides the major component of the deflection field, and any number of small subhalos, which contribute the perturbations that we are interested in finding.

Lens Profile: Singular Isothermal Sphere (SIS)

The simplest, at least somewhat realistic representation of a gravitational lens's mass distribution is a Singular Isothermal Sphere (SIS). This is one of the few analytically solvable lens mass distributions, and is therefore still used somewhat frequently for modelling purposes. For this model, we assume that the stars and any other mass in our lens galaxy behave like the particles in an ideal mass, with a flat rotation curve. the surface mass density, projected onto the lens plane, is:

$$\Sigma(\xi) = \frac{v^2}{2G} \frac{1}{\xi} \,, \tag{2.29}$$

where v is the velocity dispersion along the line of sight, and ξ is defined in polar coordinates, so that $\boldsymbol{\xi} \equiv (\xi \cos(\phi), \xi \sin(\phi))$ in the lens plane. Note that this density is divergent at the center, and so the galaxy's total mass is also divergent. This is of course non-physical, and so the equation is usually truncated at some radius to prevent this for simulation purposes.

From this surface mass density, we obtain the constant deflection angle:

$$\hat{\alpha} = 4\pi \frac{v^2}{c^2} \,, \tag{2.30}$$

which gives a likewise constant reduced deflection angle, which we name the Einstein angle:

$$\alpha = \hat{\alpha}\frac{D_{ds}}{D_s} \tag{2.31}$$

$$= 4\pi\frac{v^2}{c^2}\frac{D_{ds}}{D_s} \tag{2.32}$$

$$= \theta_E\,, \tag{2.33}$$

and correspondingly, the lens equation becomes:

$$\boldsymbol{\beta} = \boldsymbol{\theta} - \theta_E\frac{\boldsymbol{\theta}}{|\boldsymbol{\theta}|} \tag{2.34}$$

so that we generally get two images for every source point: $\theta_\pm = \beta \pm \theta_E$.

A relevant note here is that magnification can become very large for sources aligned with the lens axis:

$$\mu_\pm = \frac{\theta_\pm}{\beta} = 1 \pm \frac{\theta_E}{\beta} = \left(1 \mp \frac{\theta_E}{\theta_\pm}\right)^{-1}\,. \tag{2.35}$$

As we can see, as the angle β goes to 0, μ approaches infinity. This is due to the divergent total mass in the model, as noted earlier; in reality the mass is not divergent, so while the magnification is maximized for an aligned source-lens-observer system, magnification does not actually go to infinity.

Lens Profile: Singular Isothermal Ellipsoid

If we wish to have a more generally applicable lens representation, a significant improvement comes in the form of the Singular Isothermal Ellipsoid (SIE), as first outlined in detail independently by both Kassiola and Kovner (1993) [**?**], and Kormann, Schneider, and Bartelmann (1994) [**?**]; the description in this section takes mostly after the treatment by Kormann, Schneider and Bartelmann. The SIE is a two-parameter family of lens models, where one the parameters (the velocity dispersion) can be scaled out, leaving us with a single non-trivial parameter: the axis ratio f.

Defining f':

$$f' = \sqrt{1 - f^2}\,, \tag{2.36}$$

our deflection potential becomes, after some algebra:

$$\psi(\theta,\phi) = \theta\frac{\sqrt{f}}{f'}\left[\sin\phi\arcsin(f'\sin\phi) + \cos\phi\text{arsinh}\left(\frac{f'}{f}\cos\phi\right)\right]\,, \tag{2.37}$$

and, likewise, the lensing equation becomes

$$\boldsymbol{\beta} = \boldsymbol{\theta} - \frac{\sqrt{f}}{f'}\left[\text{arsinh}\left(\frac{f'}{f}\cos\phi\right)\boldsymbol{e}_1 + \arcsin(f'\sin\phi)\boldsymbol{e}_2\right]\,, \tag{2.38}$$

with $\boldsymbol{e}_i$ the unit vector in direction x_i. Note that these equations indeed converge to the SIS model for $f \to 1$. Finally, differentiating the lens equation we get our transformation matrix $\mathcal{A}$:

$$\mathcal{A} = \begin{pmatrix} 1 - 2\kappa\sin^2\phi & \kappa\sin 2\phi \\ \kappa\sin 2\phi & 1 - 2\kappa\cos^2\phi \end{pmatrix}\,. \tag{2.39}$$

Notably, this $\mathcal{A}$ is actually not changed from the SIS model, though we should note that it is actually through the convergence k that the addition of the axis ratio f becomes visible:

$$\nabla\psi \equiv 2\kappa \tag{2.40}$$

$$= \frac{\sqrt{f}}{x\nabla(\phi)} \tag{2.41}$$

$$\kappa = \frac{1}{2}\frac{\sqrt{f}}{x\nabla(\phi)}\,. \tag{2.42}$$

As a result, our magnification μ becomes, in analogy to the SIS:

$$\mu = \frac{1}{1-2\kappa} = \frac{1}{1-\frac{\sqrt{f}}{x\nabla(\phi)}}\,. \tag{2.43}$$

Like for the SIS, the problem of the singularity at the core of the lens is typically solved (in simulations) by truncating the mass distribution at some small inner radius, within which the density is kept constant. Note that this treatment means that the equations are no longer exactly solvable, and so the treatment from here on out is, technically, only an approximation to such a scenario.

Interestingly, we should note the number of images produced by such a lens: sources in the 'outer region' have a single, slightly magnified image; a very faint second image is produced very close to the lens center as it moves closer. Moving further in, we get a second set of images (again: one faint, one magnified) as it crosses the caustic[2]. For the rare case of a source perfectly aligned with the center of such a lens, we find four images on the axes of the lens plane - which is what would appear to be happening for Einstein's cross (figure 2.2), for example, lending at least some credence to the model.

Lens Profile: Singular Power Law Ellipsoid (SPLE)

A further 'generalization' is then proposed by Tessore and Metcalf (2015) [**?**], in the form of a lens following an elliptical power law mass model - the Singular Power Law Ellipsoid (SPLE). Their work was based on some of the works mentioned above, as well as research on (mostly numerical) methods for solving non-singular varieties of this profile (specifically, see: [**?**, **?**]). They propose a surface density Σ as:

$$\Sigma(R) = \frac{2-t}{2}\left(\frac{b}{R}\right)^t\,. \tag{2.44}$$

with $0 < t < 2$ the slope of the profile, $b > 0$ the scale length and $R > 0$ the elliptical radius equal to the semi-minor axis of the ellipse passing through (x, y), defined as:

$$R = \sqrt{f^2x^2 + y^2}\,, \tag{2.45}$$

with f the axis ratio as before. The authors define $\Sigma_{cr} = 1$ so that $\Sigma = \kappa$. After some algebra[3], they arrive at the following formula for the (complex) deflection angle:

$$\alpha(R,\phi) = \frac{2b}{1+f}\left(\frac{b}{R}\right)^{t-1}\exp^{i\phi}\ {}_2F_1\left(1, \frac{t}{2}; 2-\frac{t}{2}; -\frac{1-f}{1+f}\exp^{i2\phi}\right)\,, \tag{2.46}$$

[2] In short: lines θ along which the magnification becomes maximized (in ideal scenarios: infinite) we call *critical lens lines*; the corresponding source lines we call *caustics*. For a more detailed explanation, see [**?**] or [**?**]

[3] For this profile, as well as for the NFW in the next section, the full derivation is rather cumbersome, and so it will not be reproduced here. We refer any interested readers to the original articles.

with ${}_2F_1(a, b; c; d)$ the Gaussian hypergeometric function, and ϕ defined as the angle in the lens plane: $\phi = \arctan(fx, y)$. They then find a potential that solves for equation 2.8 through an educated guess, inspired by the methodology employed by Kormann, Schneider and Bartelmann:

$$\psi(x, y) = \frac{x\alpha_x + y\alpha_y}{2 - t}\,. \tag{2.47}$$

Notably, this potential is very cheaply calculable if the deflection angle α has already been found. This potential results in the following shear:

$$\gamma = -\exp^{i2\phi}\kappa + (1 - t)\exp^{i\theta}\frac{\alpha}{r}\,, \tag{2.48}$$

with $r = \sqrt{x^2 + y^2}$ and $\theta = r\cos\phi$ the regular polar coordinates. Note how this gives an identical shear to the SIE for $t = 1$, as expected. For the magnification we now obtain:

$$\mu = \left[1 - 2\kappa\left(1 - (1 - t)\frac{x\alpha_x + y\alpha_y}{r^2}\right) - (1 - t)^2\frac{|\alpha|^2}{r^2}\right]\,. \tag{2.49}$$

Lens Profile: NFW

Following, again, the algebraic process of Kormann et al., Hurtado, Castañada and Tejeiro [**?**] derive[4] a lensing profile for the NFW mass distribution; this is of particular interest for modelling the subhalos for the *Deep Lensing* research, as the expected mass distribution for the subhalos is the NFW mass profile[5]

We start from the mass density as per equation 2.1, obtaining for the surface density/convergence:

$$\kappa(x) = -\frac{1}{2C(1 - x^2)}\left(1 - \frac{2}{\sqrt{(1 - x^2)}}\,\text{artanh}\left[\sqrt{\frac{1 - x}{1 + x}}\right]\right) \tag{2.50}$$

with

$$x = \xi/r_s \tag{2.51}$$

$$C = \frac{\Sigma_{cr}}{4\rho_0 r_s} \tag{2.52}$$

For the deflection angle they then find:

$$\alpha(x) = \frac{1}{Cx}\left(\ln\left(\frac{x}{2}\right) - \frac{2}{\sqrt{(1 - x^2)}}\,\text{arctanh}\left[\sqrt{\frac{1 - x}{1 + x}}\right]\right) \tag{2.53}$$

and for an image location y, the lens equation then becomes

$$y = \left|x - \frac{1}{Cx}\left(\ln\left(\frac{x}{2}\right) - \frac{2}{\sqrt{(1 - x^2)}}\,\text{arctanh}\left[\sqrt{\frac{1 - x}{1 + x}}\right]\right)\right| \tag{2.54}$$

the shear becomes

$$\gamma(x) = \frac{1}{2Cx^2}\left(2\ln\left(\frac{x}{2}\right) + \frac{x^2}{1 - x^2} + \frac{4 - 6x^2}{(1 - x^2)^{3/2}}\,\text{arctanh}\left[\sqrt{\frac{1 - x}{1 + x}}\right]\right) \tag{2.55}$$

and the magnification then follows, as usual, from equation 2.28.

[4] Again, I opted to summarize the findings here, rather than reproducing the derivation fully or in part, as it is somewhat cumbersome. For details, see the original paper [**?**].

[5] Or, often more realistically, a truncated NFW profile - see for example [**?**].

2.3 The Variational Auto-Encoder: VAE

At the most basic level, the concept of a Variational Auto-Encoder, or VAE, is simple: we have a trainable encoder, or recognition model, which turns an image into a sequence of N numbers, or latent variables. Likewise, we have a trainable decoder, which turns such a sequence of N numbers back into an image. If our VAE is 'good', the difference between our input and output images is minimal. In addition, because of the learning criterion used (the ELBO, or variations of it - as will be discussed in section 2.3.2), the distribution of variables in latent space will tend to have some logical structure to it. An example is shown in figure 2.5, where we see that all numbers of 0-9 are (with differing clarity) all represented in the 2D plane of the latent space of an autoencoder trained to reproduce handwritten digits.

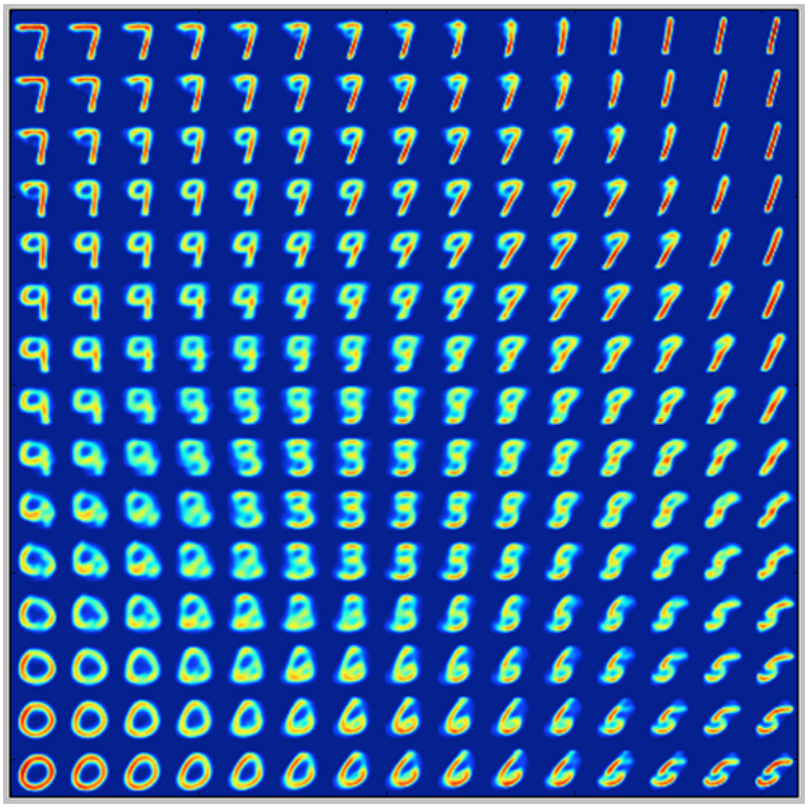


Figure 2.5: Image from [**?**] (reproduced with permission), showing an example of how a multitude of different features are encoded in the latent space. In this case, we have an autoencoder that was trained on a dataset of handwritten numbers 0-9 (MNIST). We can see that all numbers have been encoded with at least some success, though we might also surmise that this particular model would have trouble distinguishing (and hence reproducing clearly) the number 2. Note also some transitional parts of the latent space, for example on the mid-bottom left; here we see a transitional state between a zero and a nine, which makes little sense for a categorical distribution as we have here, but could be potentially interesting in a system without distinct classes.

2.3.1 Auto-Encoding Variational Bayes: probabilistic reconstruction

In their seminal 2014 article, Kingma and Welling [**?**] introduce the concept of the Variational Auto-Encoder (VAE). In this paper, they start from a desire to perform efficient approximate inference for

probabilistic models with continuous latent variables with intractable[6] posterior distributions, which eventually leads them to the Auto-Encoding Variational Bayes (AEVB) algorithm, and by extension the Variational Auto-Encoder (VAE).

The exact scenario they sketch is as follows. Consider some large dataset $\boldsymbol{X} = \{\boldsymbol{x}^{(i)}\}_{i=1}^{N}$, consisting of N independent and identically distributed samples of some continuous or discrete variable $\boldsymbol{x}$. In our case, this would be a dataset containing N images of galaxies, which we assume are governed by a set of physical relations, which are in turn dependent on some set of variables $\boldsymbol{x}$.

The process of 'generating' data, then, has two steps: a value $\boldsymbol{z}^{(i)}$ is first drawn from a prior distribution $p_{\theta^*}(\boldsymbol{z})$. Then, a value of $\boldsymbol{x}^{(i)}$ is generated from the conditional distribution $p_{\theta^*}(\boldsymbol{x}|\boldsymbol{z})$. Our only requirements here are that the prior $p_{\theta^*}(\boldsymbol{z})$ and likelihood $p_{\theta^*}(\boldsymbol{x}|\boldsymbol{z})$ come from the parametric families $p_\theta(\boldsymbol{z})$ and likelihood $p_\theta(\boldsymbol{x}|\boldsymbol{z})$, so that $\boldsymbol{\theta}^*$ are the set of specific, 'true' parameters for θ, and that the PDFs of the prior and likelihood are both differentiable almost everywhere with respect to both $\boldsymbol{\theta}$ as well as $\boldsymbol{z}^{(i)}$. Importantly, they do not make any simplifying assumptions regarding the marginal or posterior probabilities. Having made the problem declaration in this way, our goals are threefold. We want to be able to:

1. Efficiently obtain an approximate maximum-likelihood or maximum a posteriori estimation of the true set of parameters $\boldsymbol{\theta}^*$ for the model.

2. Efficiently obtain an approximate posterior inference of the latent variables $\boldsymbol{z}$ given an observed value $\boldsymbol{x}$ (in our case: given a specific image) for a given $\boldsymbol{\theta}$.

3. Efficiently obtain an approximate marginal inference of the variable $\boldsymbol{x}$.

In order to solve these problems, they then introduce a few important concepts. First is a recognition model, or *probabilistic encoder* $q_\phi(\boldsymbol{z}|\boldsymbol{x})$, which is an approximation to the intractable true posterior, $p_\theta(\boldsymbol{z}|\boldsymbol{x})$. In a similar vein, we call $p_\theta(\boldsymbol{x}|\boldsymbol{z})$ a *probabilistic decoder*. These terms originate from computer science, where the unobserved variable $\boldsymbol{z}$ can be interpreted as a latent representation, or *code*; essentially, the code is a 'compressed' representation of the data $\boldsymbol{x}$, and we can compress $\boldsymbol{x}$ to $\boldsymbol{z}$ using the encoder, or vice versa decompress from $\boldsymbol{z}$ to $\boldsymbol{x}$ using the decoder.

The reason they are called probabilistic is because these do not return a singular value, but rather a distribution of values for a given input. Given a value of $\boldsymbol{x}$, the encoder $q_\phi(\boldsymbol{z}|\boldsymbol{x})$ will produce a distribution over the possible values of the code $\boldsymbol{z}$ from which the datapoint $\boldsymbol{x}$ could have been generated. By the same token, the decoder $p_\theta(\boldsymbol{x}|\boldsymbol{z})$ will produce a distribution over the possible corresponding values of $\boldsymbol{x}$ for a given $\boldsymbol{z}$.

Second, they introduce an estimator that allows them to solve the problem defined above: the Stochastic Gradient Variational Bayes (SGVB) estimator, which is a differentiable, unbiased estimator of the variational lower bound for models with continuous latent variables. This variational lower bound is commonly known as the the Evidence Lower BOund (ELBO); I will go through the full derivation of the ELBO, as well as some contributions to the concept by other authors, in the next section (section 2.3.2).

[6]Meaning they cannot be exactly evaluated or differentiated either numerically or analytically through 'normal' methods, and so usually require us to use costly iterative inference schemes, such as MCMC

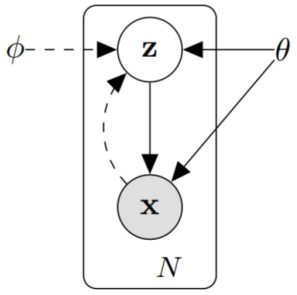


Figure 2.6: Image from [**?**] (reproduced with permission). As per the original description: "The type of directed graphical model under consideration [in AEVB]. Solid lines denote the generative model $p_\theta(\boldsymbol{z})p_\theta(\boldsymbol{x}|\boldsymbol{z})$, dashed lines denote the variational approximation $q_\phi(\boldsymbol{z}|\boldsymbol{x})$ to the intractable posterior $p_\theta(\boldsymbol{x}|\boldsymbol{z})$. The variational parameters ϕ are learned jointly with the generative model parameters θ."

The SGVB estimator is then simply a slight modification to the ELBO (the derivation of which we go through in section 2.3.3), since the naive Monte Carlo gradient estimator for minimization of this lower bound tends to exhibit very high variance.

With the necessary estimator defined, they can then go on to describe the Auto-Encoding Variational Bayes (AEVB) algorithm (1), which can be used to find the parameters $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$ for any general scenario that meets the conditions and assumptions outlined above. If we then proceed to use a neural network for the recognition model $q_\phi(\boldsymbol{z}|\boldsymbol{x})$, we have arrived at what Kingma and Welling dubbed the Variational Auto-Encoder, which is the inference model we use throughout this research.

Algorithm 1 Minibatch version of the Auto-Encoding Variational Bayes (AEVB) algorithm; original from [**?**].

$\boldsymbol{\theta}, \boldsymbol{\phi} \leftarrow$ Initialize parameters
repeat
  $\boldsymbol{X}^M \leftarrow$ Random minibatch of M datapoints (drawn from full dataset)
  $\boldsymbol{\epsilon} \leftarrow$ Random samples from noise distribution $p(\boldsymbol{\epsilon})$
  $\boldsymbol{g} \leftarrow \nabla_{\boldsymbol{\theta},\boldsymbol{\phi}} \tilde{\mathcal{L}}^M(\boldsymbol{\theta}, \boldsymbol{\phi}; \boldsymbol{X}^M, \boldsymbol{\epsilon})$ (Gradients of minibatch estimator $\tilde{\mathcal{L}}^M$) ▷ see also section 2.3.3
  $\boldsymbol{\theta}, \boldsymbol{\phi} \leftarrow$ Update parameters using gradients $\boldsymbol{g}$ through an optimizer, e.g. SGD, Adam, etc. [**?**]
until convergence of parameters θ, ϕ
return $\boldsymbol{\theta}, \boldsymbol{\phi}$

2.3.2 Deriving the ELBO: variational bound

There are two different methods to derive the ELBO. We will first go through the derivation starting from the model evidence and employing Jensen's Inequality; in the section after, we will start from the Kullback-Leibler Divergence between p and q, instead.

Deriving the ELBO from Jensen's Inequality

In this case, we start with an attempt to rewrite a basic expression for the log of the evidence: $\log p_\theta(x)$.

$$\log p_\theta(x) = \log \int_z p_\theta(x,z) dz \tag{2.56}$$

$$= \log \int_z p_\theta(x,z) \frac{q_\phi(z|x)}{q_\phi(z|x)} dz \tag{2.57}$$

$$= \log \int_z q_\phi(z|x) \frac{p_\theta(x,z)}{q_\phi(z|x)} dz\,, \tag{2.58}$$

which we recognize as an expression for an expected value over $q_\phi(z|x)$:

$$\log p_\theta(x) = \log \mathbb{E}_{q_\phi(z|x)} \left[\frac{p_\theta(x,z)}{q_\phi(z|x)} \right] . \tag{2.59}$$

By Jensen's inequality:

$$\log \mathbb{E}_{q_\phi(z|x)} \left[\frac{p_\theta(x,z)}{q_\phi(z|x)} \right] \geq \mathbb{E}_{q_\phi(z|x)} \left[\log \left(\frac{p_\theta(x,z)}{q_\phi(z|x)} \right) \right] , \tag{2.60}$$

and thus:

$$\log p_\theta(x) \geq \mathbb{E}_{q_\phi(z|x)} \left[\log \left(\frac{p_\theta(x,z)}{q_\phi(z|x)} \right) \right] \tag{2.61}$$

$$\geq \mathbb{E}_{q_\phi(z|x)} \left[-\log q_\phi(z|x) + \log p_\theta(x,z) \right] . \tag{2.62}$$

Then, defining:

$$\mathcal{L}(\theta, \phi; x) = \mathbb{E}_{q_\phi(z|x)} \left[-\log q_\phi(z|x) + \log p_\theta(x,z) \right] , \tag{2.63}$$

we have just defined a tractable expression for a *lower bound* on the evidence $\log p_\theta(x)$:

$$\log p_\theta(x) \geq \mathcal{L}(\theta, \phi; x) \tag{2.64}$$

Another interesting note on equation 2.63, is that this version (with a very minor rewrite) allows for an interpretation in an almost physical sense.

$$\mathcal{L}(\theta, \phi; x) = \mathbb{E}_{q_\phi(z|x)} \left[\log p_\theta(x,z) \right] + \mathbb{H}[q_\phi(z|x)] \tag{2.65}$$

The first term can now be interpreted as the negative energy in a Boltzmann distribution, while the second term can even explicitly be written as the entropy of q [**?**].

Deriving the ELBO from the Kullback-Leibler Divergence

In this second case, we start instead from the Kullback-Leibler Divergence D_{KL} - also sometimes known as the Kullback-Leibler Distance or relative entropy, and often shortened to KLD. The KLD is a non-symmetric, information theoretic measure of similarity between two probability distributions [**?**]. Its magnitude is a measure of how different two distributions are. Furthermore, it is inherently always non-negative, and $D_{KL}(P||Q) = 0$ only if $P = Q$ almost everywhere.

For distributions P and Q of a continuous random variable x, with $p(x), q(x)$ the respective probability density functions (PDFs), the KLD is defined as:

$$D_{KL}(Q||P) = -\int_{-\infty}^{\infty} q(x) \log\left(\frac{p(x)}{q(x)}\right) dx\,, \tag{2.66}$$

which we recognize, again, as an expected value over q(x). Plugging in the various expressions for the recognition model and true posterior, we can therefore write:

$$\begin{aligned} D_{KL}(q_\phi(z|x)\,||\,p_\theta(z|x)) &= -\int_{-\infty}^{\infty} q_\phi(z|x) \log\left(\frac{p_\theta(z|x)}{q_\phi(z|x)}\right) dz && (2.67)\\ &= -\,\mathbb{E}_{q_\phi(z|x)}\left[\log\left(\frac{p_\theta(z|x)}{q_\phi(z|x)}\right)\right] && (2.68)\\ &= -\,\mathbb{E}_{q_\phi(z|x)}\left[\log p_\theta(z|x) - \log q_\phi(z|x)\right]\,. && (2.69) \end{aligned}$$

Then, using Bayes' theorem:

$$\begin{aligned} D_{KL}(q_\phi(z|x)\,||\,p_\theta(z|x)) &= -\,\mathbb{E}_{q_\phi(z|x)}\left[\log p_\theta(z|x) - \log q_\phi(z|x)\right] && (2.70)\\ &= -\,\mathbb{E}_{q_\phi(z|x)}\left[\log\left(\frac{p_\theta(z,x)}{p_\theta(x)}\right) - \log q_\phi(z|x)\right] && (2.71)\\ &= -\,\mathbb{E}_{q_\phi(z|x)}\left[\log p_\theta(z,x) - \log p_\theta(x) - \log q_\phi(z|x)\right]\,. && (2.72) \end{aligned}$$

Then, recognizing that $p_\theta(x)$ is independent of z and therefore $\mathbb{E}_q[p_\theta(x)] = p_\theta(x)$:

$$D_{KL}(q_\phi(z|x)\,||\,p_\theta(z|x)) = -\,\mathbb{E}_{q_\phi(z|x)}\left[\log p_\theta(z,x) - \log q_\phi(z|x)\right] + \log p_\theta(x)\,. \tag{2.73}$$

Reordering:

$$\log p_\theta(x) = D_{KL}(q_\phi(z|x)\,||\,p_\theta(z|x)) + \mathbb{E}_{q_\phi(z|x)}\left[-\log q_\phi(z|x) + \log p_\theta(z,x)\right]\,. \tag{2.74}$$

And so we find, again, by using the same definition for the estimator $\mathcal{L}$ (equation 2.63):

$$\log p_\theta(x) = D_{KL}(q_\phi(z|x)\,||\,p_\theta(z|x)) + \mathcal{L}(\theta,\phi;x) \tag{2.75}$$

where:

$$\mathcal{L}(\theta,\phi;x) = \mathbb{E}_{q_\phi(z|x)}\left[-\log q_\phi(z|x) + \log p_\theta(z,x)\right] \tag{2.76}$$

Note that, since $D_{KL}(q_\phi(z|x)\,||\,p_\theta(z|x)) \geq 0$ we again find that $\mathcal{L}$ is a lower bound on the evidence, $\log p_\theta(x)$.

$$\log p_\theta(x) \geq \mathcal{L}(\theta,\phi;x)\,. \tag{2.77}$$

But, no less importantly, we can now also write this as:

$$D_{KL}(q_\phi(z|x)\,||\,p_\theta(z|x)) = -\mathcal{L}(\theta,\phi;x) + \log p_\theta(x)\,, \tag{2.78}$$

which serves nicely to bring us back to the problem at hand: the optimization problem where we want to optimize our recognition model $q_\phi(z|x)$ in such a way that it looks as much as possible like the true posterior $p_\theta(z|x)$. As we discussed previously, the KLD is a measure of how similar two distributions are; the smaller the value of D_{KL} in this case, therefore, the closer we are to our declared goal. A way to achieve this, then is to *minimize* the LHS of equation 2.78. Since the model evidence is independent of q, it is essentially a constant - meaning that minimizing the LHS means minimizing $-\mathcal{L}(\theta,\phi;x)$. That is, minimizing the KLD means that our optimization goal is to maximize the ELBO.

2.3.3 From ELBO to SGVB

Kingma and Welling [**?**], as noted previously, do have some qualms regarding using the 'pure' ELBO as an optimization objective, since the gradient of the lower bound with respect to ϕ tends to exhibit very high variance. This is because, as pointed out by Paisley et al. (2012) [**?**], given S samples of a random vector X, the covariance of the unbiased sample mean $\overline{X}$ is given by:

$$Cov(\overline{X}) = Cov(X)/S\,. \tag{2.79}$$

When the diagonal values of $Cov(X)$ are then large, an extremely large number of samples would then be required to bring down the variance down to an appreciable level, which significantly slows down the approximation procedure. To this end, we must further manipulate the ELBO, which will provide us with some additional insights along the way.

Starting from equation 2.76, we can again use Bayes' Theorem to rewrite the ELBO:

$$\begin{aligned}
\mathcal{L}(\theta,\phi;x) &= \mathbb{E}_{q_\phi(z|x)}\left[-\log q_\phi(z|x) + \log p_\theta(z,x)\right] && (2.80)\\
&= \mathbb{E}_{q_\phi(z|x)}\left[-\log q_\phi(z|x) + \log\left(p_\theta(z)\frac{p_\theta(z,x)}{p_\theta(z)}\right)\right] && (2.81)\\
&= \mathbb{E}_{q_\phi(z|x)}\left[-\log q_\phi(z|x) + \log p_\theta(z) + \log\left(\frac{p_\theta(z,x)}{p_\theta(z)}\right)\right] && (2.82)\\
&= \mathbb{E}_{q_\phi(z|x)}\left[-\log q_\phi(z|x) + \log p_\theta(z) + \log p_\theta(x|z)\right] && (2.83)\\
&= \mathbb{E}_{q_\phi(z|x)}\left[-\log\left(\frac{q_\phi(z|x)}{p_\theta(z)}\right) + \log p_\theta(x|z)\right] && (2.84)\\
&= -\,\mathbb{E}_{q_\phi(z|x)}\left[\log\left(\frac{q_\phi(z|x)}{p_\theta(z)}\right)\right] + \mathbb{E}_{q_\phi(z|x)}\left[\log p_\theta(x|z)\right] && (2.85)
\end{aligned}$$

And so, by the opposite process from the one by which we transformed equation 2.67 into equation 2.68:

$$\begin{aligned}
\mathcal{L}(\theta,\phi;x) &= -\,\mathbb{E}_{q_\phi(z|x)}\left[\log\left(\frac{q_\phi(z|x)}{p_\theta(z)}\right)\right] + \mathbb{E}_{q_\phi(z|x)}\left[\log p_\theta(x|z)\right] && (2.86)\\
&= -D_{KL}\left(q_\phi(z|x)\,||\,p_\theta(z)\right) + \mathbb{E}_{q_\phi(z|x)}\left[\log p_\theta(x|z)\right] && (2.87)
\end{aligned}$$

Or, writing for specific datapoints $\boldsymbol{x}^{(i)}$:

$$\mathcal{L}(\theta,\phi;\boldsymbol{x}^{(i)}) = -D_{KL}\left(q_\phi(z|\boldsymbol{x}^{(i)})\,||\,p_\theta(z)\right) + \mathbb{E}_{q_\phi(z|\boldsymbol{x}^{(i)})}\left[\log p_\theta(\boldsymbol{x}^{(i)}|z)\right] \tag{2.88}$$

Stepping back for a moment, we find an interesting interpretation for this rendition of the ELBO. Noting that our optimization goal is to maximize the ELBO, we therefore wish to either a) minimize the KLD term or b) maximize $\mathbb{E}_{q_\phi(z|\boldsymbol{x}^{(i)})}\left[\log p_\theta(\boldsymbol{x}^{(i)}|z)\right]$. The KLD term, then, acts simply as a *regularizer*: it encourages the recognition model $q(z|x)$ to look like the prior for the latent variables $p(z)$. In addition, it is usually fairly straightforward to calculate this analytically (see [**?**]). The second term we can interpret as a *negative reconstruction error*; this term is the one that causes significant problems for optimization.

In order to optimize the ELBO, we need to take derivatives with respect to both ϕ and θ. The usual Monte Carlo estimator of the gradient of the reconstruction error term w.r.t. ϕ introduces the problem of

high variance as discussed before:

$$\nabla_\phi \mathbb{E}_{q_\phi(\boldsymbol{z})}[f(\boldsymbol{z})] = \mathbb{E}_{q_{\phi(\boldsymbol{z})}}[f(\boldsymbol{z})\nabla_{q_\phi(\boldsymbol{z})} \log q_\phi(\boldsymbol{z})] \tag{2.89}$$

$$\simeq \frac{1}{L}\sum_{l=1}^{L} f(\boldsymbol{z})\nabla_{q_\phi(\boldsymbol{z}^{(l)})} \log q_\phi(\boldsymbol{z}^{(l)})\,. \tag{2.90}$$

The solution put forward by Kingma and Welling (and, similarly, by Paisley et al.) is to introduce a "parametrization trick" to allow us to approximate this gradient *without* having to worry about massive variances. That is, we can reparametrize our random variable $\tilde{\boldsymbol{z}} \sim q_\phi(\boldsymbol{z}|\boldsymbol{x})$:

$$\tilde{\boldsymbol{z}} = g_\phi(\boldsymbol{\epsilon}, \boldsymbol{x}) \quad \text{with} \quad \boldsymbol{\epsilon} \sim p(\epsilon)\,, \tag{2.91}$$

with $g_\phi(\boldsymbol{\epsilon}, \boldsymbol{x})$ a differentiable transformation of the noise variable $\boldsymbol{\epsilon}$. Our MC gradient estimates of the expectation of some function $f(\boldsymbol{z})$ w.r.t. $q_\phi(\boldsymbol{z}|\boldsymbol{x})$ then become:

$$\mathbb{E}_{q_\phi(\boldsymbol{z})}[f(\boldsymbol{z})] = \mathbb{E}_{p(\boldsymbol{\epsilon})}\left[f(g_\phi(\boldsymbol{\epsilon}, \boldsymbol{x}^{(i)}))\right] \tag{2.92}$$

$$\simeq \frac{1}{L}\sum_{l=1}^{L} f(g_\phi(\boldsymbol{\epsilon}^{(l)}, \boldsymbol{x}^{(i)}))\,. \tag{2.93}$$

Applying this to equation 2.88, we obtain the SGVB estimator, $\tilde{\mathcal{L}}^B(\boldsymbol{\theta}, \boldsymbol{\phi}; \boldsymbol{x}^{(i)}) \simeq \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}; \boldsymbol{x}^{(i)})$:

$$\tilde{\mathcal{L}}^B(\boldsymbol{\theta}, \boldsymbol{\phi}; \boldsymbol{x}^{(i)}) = -D_{KL}\left(q_\phi(z|\boldsymbol{x}^{(i)}) \,||\, p_\theta(z)\right) + \frac{1}{L}\sum_{l=1}^{L}\left(\log p_\theta(\boldsymbol{x}^{(i)}|z^{(i,l)})\right) \tag{2.94}$$

where

$$z^{(i,l)} = g_\phi(\boldsymbol{\epsilon}^{(i,l)}, \boldsymbol{x}^{(i)}) \tag{2.95}$$

with, as before

$$\boldsymbol{\epsilon}^{(l)} \sim p(\epsilon) \tag{2.96}$$

To put it explicitly: instead of a an expectation value over all z, we instead take L samples $z^{(i,l)}$ per datapoint x^i. These samples are taken from the predefined distribution g_ϕ (which we are free to choose ourselves, see [**?**]), where noise is introduced to allow multiple values of z^i per x^i over which we can then take an average. Kingma and Welling also note, however, that as long as we take minibatch size M sufficiently large, the number of samples L can be set to 1 - making this a very efficient way to estimate the expected reconstruction error.

2.4 Neural networks

As mentioned, a VAE is essentially the result of applying the AEVB algorithm and using a neural network for the recognition model q_ϕ. For completeness, therefore, I will summarily go over the concept of a neural network here.

Neural networks (or neural nets) were, as the name implies, originally inspired by the way in which animal brains function. A real human brain is made up of about 100 billion neurons (hence the 'neural'

part), each of which is connected to some number of other neurons, forming one massive, interlinked network (hence: 'network'). A (basic) understanding of how biological neurons work was the basis for the work on the *artificial neuron* - which is in essence nothing more than a mathematical abstraction of the basic way in which biological neurons function.

A simplified model of a real neuron is as follows: every neuron has some number of inputs (from other neurons or sensory organs), and one output, which may be connected to any number of other neurons in the network. Along the input, it receives (electrical) signals, and depending on whether those incoming signals surpass a certain threshold, the neuron itself may also fire, sending a signal of some strength along its output. This output is then again connected to any number of other neurons, where it serves as input.

A more mathematical description of such a neuron would go roughly like this: for a given artificial neuron, let there be m inputs with signals x_1 through x_m and weights w_1 through w_m. Let us also add a term b_k, to these inputs, which can serve as the neuron's bias. These inputs are summed, and then passed to the *activation function* ϕ. This is typically a threshold function of some kind, and determines the final output from the sum of the inputs. The output of the kth neuron is then:

$$y_k = \phi\left(b_k + \sum_{j=1}^{m} w_{kj} x_j\right) . \tag{2.97}$$

For a visual representation, see image 2.7.

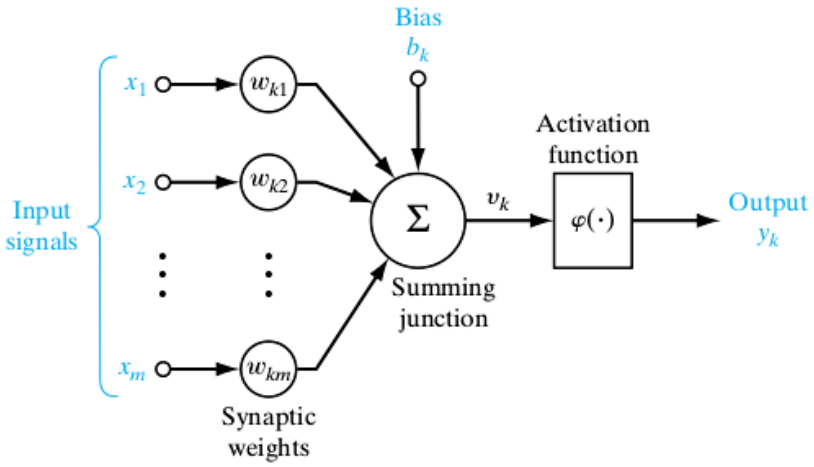


Figure 2.7: Diagram of a neuron as described in equation 2.97.

This is essentially the *threshold logic* as proposed by McCulloch and Pitts in 1943 [**?**], which is at the basis of all varieties of neural net; hence, the basic model of the neuron as proposed above is also often referred to as the McCulloch-Pitts model (MCP). When first proposed, the activation function was always a step function - outputs and inputs were therefore purely binary. More modern implementations tend to favor more gradual activation functions, as we shall see in section 2.5. Besides that, however, the main principle has remained roughly the same.

Neural networks are typically constructed as a series of *layers*, which is the situation where neurons in layer n receive inputs only from neurons in layer $n-1$ and only output to neurons in layer $n+1$. In such a scenario, the very first layer is called the *input layer*, whereas the final layer is called the *output*

layer - all layers in between are called *hidden layers*, since we never directly have access to either their in- or output. For a visual representation, see image 2.8.

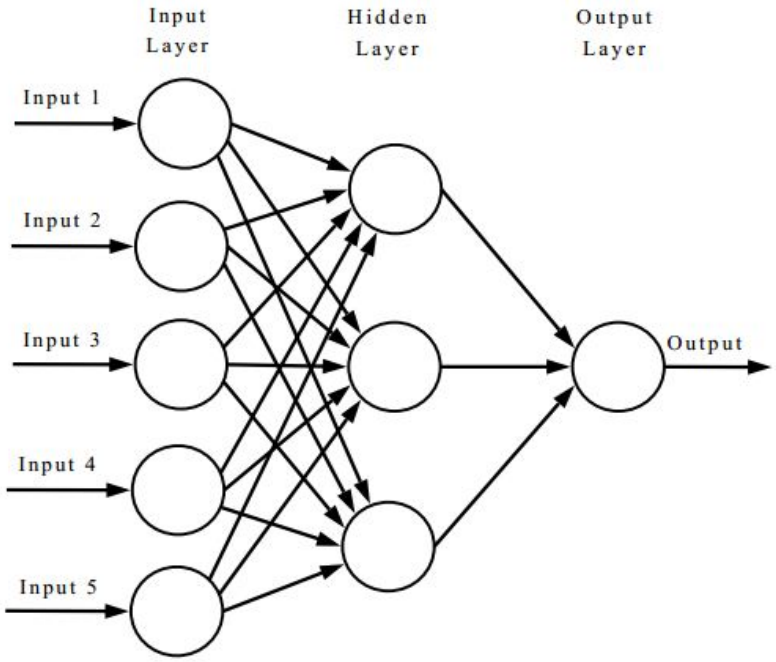


Figure 2.8: Diagram of a (basic) neural network with one hidden layer.

2.4.1 Backpropagation

A final, important element of artificial neural networks is the concept of backpropagation [**?**], which is essentially what allows these models to 'learn' at all.

Consider a neural network with some loss (or error) function E. This error function is some way of scoring the output against the expected output (for example the MSE comparing an input image against an output image). To determine how we want to change the weights in the network to achieve a lower loss (ideally eventually arriving at a local or global loss minimum), we can use any gradient descent method. For this, however, we require the gradient of the loss with respect to the weights w_{kj}: $\frac{\partial E}{\partial w_{kj}}$. While we don't know the gradient of the loss w.r.t the weights directly, we do know the gradients of the layers of neurons w.r.t. the weight from equation 2.97. Considering the interaction between the final layer and the output, and rewriting slightly:

$$y_k = \phi\left(b_k + \sum_{j=1}^{m} w_{kj}x_j\right) = \phi(N_k)\,. \tag{2.98}$$

Then, applying the chain rule twice:

$$\frac{\partial E}{\partial w_{ij}} = \frac{\partial E}{\partial y_k}\frac{\partial y_k}{\partial N_k}\frac{\partial N_k}{\partial w_k j}\,. \tag{2.99}$$

All of these are fairly straightforward terms. $\frac{\partial E}{\partial y_k}$ is the derivative of E with respect to the output y_k. Note that this should be trivial, as the error function is always defined as a function of the output. The second

term

$$\frac{\partial y_k}{\partial N_k} = \frac{\partial \phi(N_k)}{\partial N_k} \tag{2.100}$$

is clearly the derivative of the activation function, which is, again, typically trivial. Note that this means that activation functions that are non-differentiable (in some point) may therefore cause problems during training. Finally, the third term:

$$\frac{\partial N_k}{\partial w_k j} = \frac{\partial}{\partial w_k j}\left(b_k + \sum_{i=1}^{m} w_{ki} x_i\right) \tag{2.101}$$

$$= \frac{\partial}{\partial w_k j} w_{kj} x_j \tag{2.102}$$

$$= x_j \tag{2.103}$$

is simply equal to the input. So, in the end, $\frac{\partial E}{\partial w_{ij}}$ equals the derivative of the error function, multiplied by the derivative of the activation function, multiplied by the layer's input.

Note that we only considered the interaction between the final layer and the output here; however, we can recursively perform these steps, working backwards from the output - hence the name backpropagation - by taking the x_j in the third term of equation 2.99 and realizing that it is again the output of the previous layer, to which we can apply a similar procedure. In this way, propagating backwards from the output, we can find an N-dimensional gradient function, with N the total number of weights in the model, for which we can then take a gradient descent step.

2.5 Layers

Now that we have introduced the concept of a neural net with different layers, it is relevant to go over the different types of layers, activation functions, and optimizers that were employed (or considered) for actually building the models for the encoder and decoder.

2.5.1 Fully connected layers

The first, most basic type of layer to consider is the fully connected or linear layer, or FC. This is the type of layer depicted in figure 2.8. As the name suggests, for a vector of inputs $\boldsymbol{x}$ of size N, each of the M nodes (neurons) k_m in the layer have access to all N inputs, so that they have weights w_{mn}. The result is an output vector $\boldsymbol{y}$ of size M. It is plain to see that this can be computed as an Affine transformation, using matrix multiplication (where the matrix W is the $M \times N$ matrix of weights) plus a bias vector $\boldsymbol{b}$ (with b_m the bias of node k_m):

$$\boldsymbol{y} = W\boldsymbol{x} + \boldsymbol{b} \tag{2.104}$$

$$\text{or:} \quad y_m = b_m + \sum_{i=1}^{N} w_{mi} x_i \tag{2.105}$$

Which we recognize from equation 2.97 (confirming that this is indeed the basic neuron structure we discussed before). Stacking these layers on top of each other, then is simply applying a series of these matrix transformation. The learnable parameters in such layers are the values w_{ij} of the matrix W, which are the weights of the neurons in the layer.

2.5.2 Convolutional layers

The second type of layer that needs to be discussed is the Convolutional Layer. This type of layer has become increasingly popular in recent years, as Convolutional Neural Nets (often called CNNs or ConvNets), which are nets built primarily out of such layers, have shown an impressive performance for image recognition tasks compared to nets using just traditional FC layers. As the name would imply, these layers work through the process of performing a convoluting between a matrix and a kernel. This works as follows, assuming 2D convolution[7].

We start with a 2D, black and white[8] image, with a height of H_1 and width of W_1. We can see this image as an $H_1 \times W_1$ matrix I, where each value I_{hw} is the value in the image at pixel $[h, w]$. We then define our kernel (or filter) as a matrix G of size $K \times K$[9] (denoted as simply size K). As a simple example, we take a size 3 kernel, and we initialize it as:

$$G = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 0 \end{bmatrix} . \tag{2.106}$$

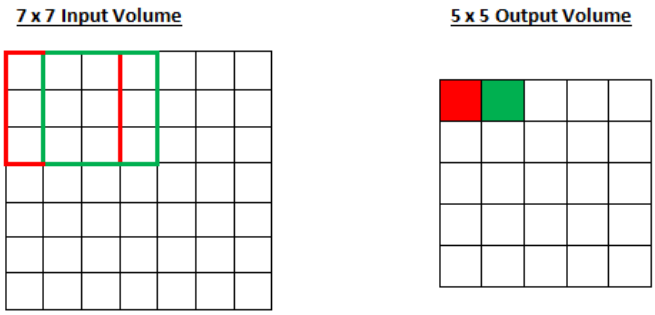
A 'run' of this kernel over I now means that we look at every pixel I_{hw} of I, and apply a Frobenius inner product between the kernel and the $K \times K$ subset of the image formed by I_{hw} and its immediate neighbors on each side. Afterwards, to normalize the values, we divide this value by the grand sum of the kernel. So, in our example case, for every pixel I_{hw} we convolute with, we perform the inner product and divide by 5:

$$\frac{1}{5}\left\langle \begin{bmatrix} I_{h+1,w-1} & I_{h+1,w} & I_{h+1,w+1} \\ I_{h,w-1} & I_{h,w} & I_{h,w+1} \\ I_{h-1,w-1} & I_{h-1,w} & I_{h-1,w+1} \end{bmatrix}, \begin{bmatrix} 0 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 0 \end{bmatrix} \right\rangle_F \tag{2.107}$$

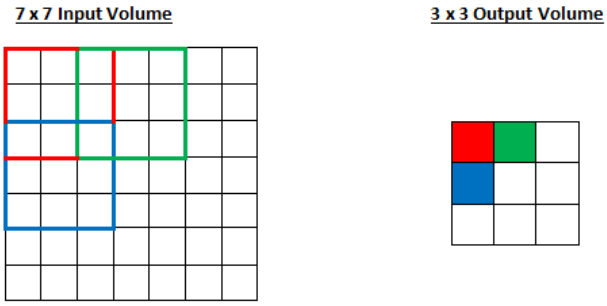
$$= \frac{1}{5}(I_{h+1,w} + I_{h,w-1} + I_{h,w} + I_{h,w+1} + I_{h-1,w}) . \tag{2.108}$$

To put it explicitly: using this kernel our output pixel value is simply the average over the original pixel and its immediate neighbors. Now, if we apply this operation to every pixel in I, we have performed a convolution with a stride S of 1 - the stride is simply how many pixels we move away from our previous pixel on every next iteration. $K = 3$ and $S = 1$, as per our example, means that we take the 3×3 neighborhood of every pixel. A stride $S > 1$ means that not every pixel in the image is taken as the origin of a convolution - whether this is desirable depends entirely on the setup. One might imagine, for example,

[7] The process is essentially the same for any number of dimensions, but is less intuitive to explain.
[8] An RGB image would essentially by a 3D image, where the third dimension simply contains the RGB values in each pixel.
[9] Non-square kernels are possible, but rarely used.



(a) Using a kernel with $[K = 3, S = 1]$ on a 7×7 input volume



(b) Using a kernel with $[K = 3, S = 2]$ on a 7×7 input volume

Figure 2.9: Visualization of a 3×3 kernel working on an input image, and how the final output image is produced. Images from [**?**] (reproduced with permission).

that a very large image is more efficiently analyzed using a larger kernel size, say $K = 7$. If we expect to be able to pick up on relevant structures and patterns with a 7×7 kernel, we probably will be better off moving the kernel slightly farther (say, $S = 3$), as while we discard some amount of information, the end goal is usually more focused on efficiency rather than perfect accuracy. In fact, collecting too much 'needless' information will more than likely slow down training significantly. In practice, it is very rare to use $S > 3$.

A relevant note is that for this basic configuration, our output 'images' will always be smaller than our input images for $K > 1$ and/or $S > 1$, with the final width/height being:

$$W_{out} = \frac{W_{in} - K}{S} + 1\,. \tag{2.109}$$

Oftentimes, it is convenient to have input and output sizes be equal. It is for this reason that the input image is often zero-padded before starting the convolution, with P rows/columns of zeroes on each side. This causes the final image size to be:

$$W_{out} = \frac{W_{in} - K + 2P}{S} + 1\,. \tag{2.110}$$

Note that if using this equation we find a non-integer value for W_{out}, this means that we have chosen an illegal combination of $[K, S, P]$ - with the chosen combination of kernel size, stride, and padding, it is simply impossible to pass over the image an integer number of times. In practice, this usually means the algorithm in question will do an incomplete run over the image, causing it to discard some amount of information on one of the edges of the image.

The final output image of a single kernel is often called a *feature map*; while above we discussed how a single kernel 'runs' over an image, usually a convolutional layer will consist of a set of kernels of size N, each of which will produce a different, distinctive feature map. It should then be obvious that the learnable parameters of a convolutional layer are in fact the values G_{ij} - it is in the choice of the values of the kernel that it becomes sensitive to different features (hence the name). For an insightful visualization of such feature maps and convolutional layers and networks in general, Zeiler and Fergus' 2014 article "Visualizing and Understanding Convolutional Neural Networks" [**?**] is highly recommended.

2.5.3 Deconvolutional layers

Convolutional layers are intuitively useful in reducing dimensionality, as they are designed to extract features from the input. For a VAE, this means that convolutional layers are a natural fit for the encoder. The decoder, however, would in essence require an opposite transformation: we have a low-dimensional encoding (of features), from which we want to extract a higher-dimensional image. In this context, deconvolutional layers become an obvious choice.

As the name suggests, deconvolutional layers (also known as upsampling convolutional layers, fractionally strided convolutional layers, or transpose convolutional layers), essentially perform the opposite operation from a convolutional layer. Whereas convolutional layers apply a many-to-one operation where they take the input from multiple pixels and condense this into a single pixel, the deconvolutional layer instead takes the value of a singular pixel, multiplies it by the filter, and then outputs the (normalized) output into a range of pixels, as visualized is figure 2.10. The output size then becomes:

$$W_{out} = S \times (W_{in} - 1) + K - 2P\,. \tag{2.111}$$

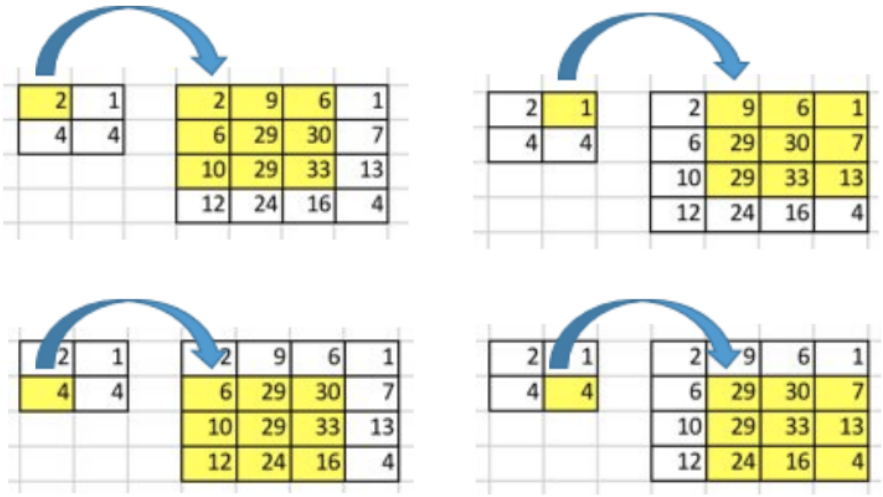


Figure 2.10: Image from [**?**] (reproduced with permission). Visualization of deconvoluting a 2×2 image into a 4×4 image, using a 3×3 kernel with $S = 1$.

2.5.4 Pooling layers

Pooling layers (also sometimes known as downsampling layers) are used to reduce the dimensionality of an image. Their functioning is fairly straightforward: a filter moves over the data with some stride, in the same fashion as we saw for the convolutional layer. In this case, however, the stride is typically equal to the filter size (in which case we speak of a Pool layer with size N), so that there is no overlap between blocks. It then performs a 'pooling' operation in an attempt to condense all information in the block into a single value. The most widely used variant is a MaxPool operation: we simply take the maximum value in the block, which is taken as the output value. Other, less popular options are taking an average (AveragePool) or taking the L2-norm. As a result of the operation, the image is reduced in size by a factor equal to the filter/stride size (as per equation 2.109). Important to note is that Pooling layers are static; they don't have any learnable parameters.

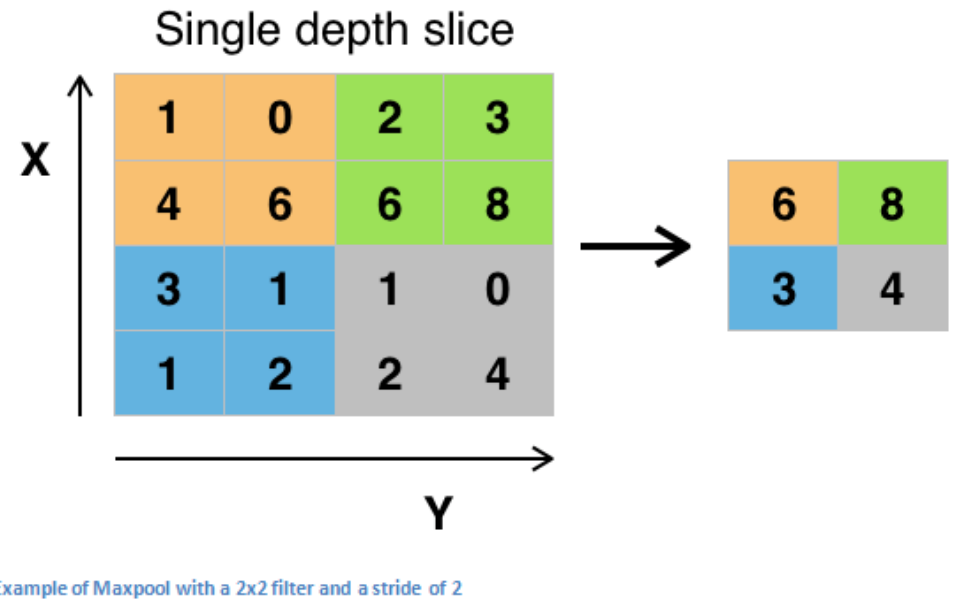


Figure 2.11: Visualisation of a MaxPool layer. As shown, the filter picks out the maximum value in each 'block'.

The primary use of this is to simply reduce the dimensionality of the input data; it is fairly typical to apply a pooling operation of size $N = 2$. This then reduces the total image volume by by 75%, reducing the computational cost by the same - a significant gain. Contrary to what one might expect (we are, after all, essentially discarding information), it has also been found time and time again that use of pooling layers *improves* performance of networks, both over networks without subsampling methods and over other such subsampling methods [**?**]. This is presumed to be because of two reasons: the first is that once we know a specific feature exists in the input volume, it's exact location is then less important compared to its location relative to other features - by taking a pooling operation, the model tends to become less sensitive to changes in scale or orientation. The second is that it's been found that pooling helps prevent overfitting - the scenario where a network becomes so conditioned on the training set that it actually starts to perform worse on the test set.

An important side note here is that the necessity of pooling layers has recently been questioned [**?**]. In

this study, the authors replaced the pooling layers in their ConvNet by convolutional layers with increased stride (thereby still achieving the desired reduction in dimensionality), and found no significant loss in model accuracy. A side-effect of this, however, is that such models are more computationally expensive, while apparently providing no direct benefit other than being 'more simplistic'. It is likely because of this that this technique is not (yet) very widely employed.

2.5.5 Activation functions

While the choice of the activation function ϕ for our neurons is, in theory, completely free, there are several options that have been empirically found to produce good results. While the step-function was favoured originally by McCulloch and Pitts, this has largely fallen out of fashion; understandably so, since it is impossible to perform gradient descent on these functions. However, the **sigmoid** (eq. 2.112 and 2.113 (eq. 2.113) functions have been popular in the past, as they are a natural smooth approximation to step functions, which have the benefit of being in a clear 'on' or 'off' state as we move away from 0, while being differentiable around and in 0.

$$f(x) = \frac{1}{1 + e^{-x}} \tag{2.112}$$

$$f(x) = \tanh(x) \tag{2.113}$$

However, the current gold standard of the field is the **ReLU** (Rectified Linear Unit) and variations of it. The original ReLU function (eq. 2.114) was first conceived in 2010 [**?**], and has been the standard of the field ever since (for a few high-profile examples, see: [**?**, **?**, **?**]).

$$f(x) = \begin{cases} 0, & \text{if } x < 0 \\ x, & \text{if } x \geq 0 \end{cases} \tag{2.114}$$

Some relevant modifications have been made over the years, which has given us a whole family of activation functions: LReLU, PReLU, RReLU, and ELU foremost among them.

LReLU, or Leaky ReLU, (eq. 2.115) has very quickly asserted a new dominance as the most-used activation function, as it is found to outperform ReLU in almost all cases. The problem with ReLU nodes, namely, is that they can 'die'. If their input ever goes below zero, not only does the output become zero, but also the gradient - meaning that that specific node is now out of commission, as the network is unable to move that input node back out of the negative region, therefore meaning the node is 'stuck in the off position'. By making the function's output in the negative region very small but non-zero, Leaky ReLUs prevent the 'death' of our ReLU nodes [**?**], which can now (albeit slowly) escape the nearly-flat negative region.

$$f(x) = \begin{cases} 0.01x, & \text{if } x < 0 \\ x, & \text{if } x \geq 0 \end{cases} \tag{2.115}$$

$$\tag{2.116}$$

A popular variation of Leaky ReLU is **PReLU**, or Parametric ReLU (eq. 2.117) [**?**]. Here, rather than having a predetermined small gradient as per the Leaky ReLU, we instead have the gradient of the

function's negative half as a learnable parameter a. This not only prevents the problem of nodes dying, but can also achieve even further model accuracy for a relatively low computational cost. How much exactly it improves model accuracy is still up for debate however; Xu et al. [**?**] found that, at least for small datasets, PReLU typically only slightly outperformed the static LReLU (though only for very leaky LReLUs - meaning their gradient in the negative half was much greater than the 0.01 mentioned above). Xu et al. also proposed yet another variant of ReLU: Random ReLU, or RReLU, which used a randomized, small slope in the negative region - this was shown then to significantly reduce overfitting for cases where only little training data is available (in their case: 25.000 images). It has, however, not found much mainstream success yet.

$$f(a,x) = \begin{cases} ax, & \text{if } x < 0 \\ x, & \text{if } x \geq 0 \end{cases} \tag{2.117}$$

$$\tag{2.118}$$

A final ReLU variant that I will discuss here is **ELU** (eq. 2.119): the Exponential Linear Unit [**?**]. In contrast to ReLUs, they, too allow for negative values without irreversibly dying. The authors of [**?**], demonstrate that ELUs offer yet another improvement in model accuracy for models with more than 5 layers. This is because ELUs better ensure a noise-robust deactivation state once they are below zero, unlike LReLUs, while still never going entirely 'dead' like ReLUs. In addition, they remove the need for batch normalization layers (which represents a relevant reduction of computational cost much computationally expensive) as they naturally drive the normal gradient to the unit natural gradient. In their experiments, ELUs resulted in faster learning and better generalization performance without batch normalization compared to ReLU-based networks *with* batch normalization.

$$f(a,x) = \begin{cases} a(e^x - 1), & \text{if } x < 0 \\ x, & \text{if } x \geq 0 \end{cases} \tag{2.119}$$

One last activation function worthy of mention, though not directly relevant to the research here, is the **SoftMax** function (eq. 2.120). This function is nearly always used as the final layer in a class-detection architecture, which is the most wide-spread use for ConvNets generally. The SoftMax is a function that takes an input vector of size K, and normalizes it into a probability distribution consisting of K probabilities, in the range $[0, 1]$. This translates very well into taking the output of a network, and turning it into the probabilities of the K relevant classes.

$$\sigma(\boldsymbol{z})_i = \frac{e_i^z}{\sum_{j=1}^{K} e_j^z} \qquad \text{for } i = 1, ..., K \text{ and } \boldsymbol{z} = (z_1, ..., z_k) \in \mathbb{R}^K \tag{2.120}$$

2.6 VAE depth, skip connections and DenseNets

One last detail of VAEs that we will discuss here, is the problem of depth. The consensus for CNNs in general is that as we increase a model's depth (i.e. add more layers), we will generally achieve better performance. However, as information about the input and the gradient is passed along the chain, some of it may get washed out along the way. It is for this exact reason that CNN architectures like ResNet [**?**] and

DenseNet [**?**], which employ *Skip Connections* in order to pass information along through the network, have become popular.

The idea of a Skip Connection (SC) in ResNet is simple: we start with an input $\boldsymbol{x}$, and apply to it the nonlinear transformation $\mathcal{F}$ representing the layer $\boldsymbol{x}$ is passed through, so that we obtain an output $\boldsymbol{y} = \mathcal{F}(\boldsymbol{x})$. Applying a SC is then the simple process of taking the sum of $\boldsymbol{x}$ and $\boldsymbol{y}$, so that our final output becomes $\boldsymbol{y}^* = \mathcal{F}(\boldsymbol{x})$. The next layer in line then applies it's own perturbation in the same fashion, where we now add our perturbed $\boldsymbol{x}$ again at the end, etc. We can then interpret the network as applying a series of perturbation on the original input $\boldsymbol{x}$, ensuring that information regarding the original input data is conserved throughout the entirety of the network. This process is visualized nicely in image 2.12.

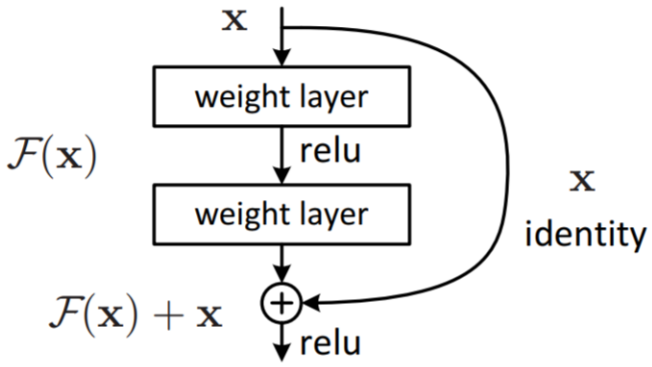


Figure 2.12: Visualization of a Skip Connection (SC); image from [**?**] (reproduced with permission).

The DenseNet architecture further builds on the ideas of ResNet [**?**]. In an effort to further improve information flow, the authors constructed an architecture where they did not just connected one convolutional layer to the next, but rather built networks out of a few blocks, within which every layer was connected to every future layer. Their implementation of 'information pass-through' was also slightly different from the traditional SC; rather than simply summing, so that we can treat the operation of the neurons as a perturbation, they concatenated the feature maps.

To elaborate: let us say we have one of these 'dense blocks'. The first node in the block takes in the input data, of size N, which consists of k_0 feature maps. It outputs k feature maps of size N, which it passes along to the second node. This second node then receives a concatenation of the original k_0 feature maps (which were the input of the first unit), along with the k output feature maps of the first unit. In total, it therefore takes $k_0 + k$ feature maps as input, and it, too, outputs k feature maps. The third node has $k_0 + 2k$ input feature maps and k output feature maps, etc. As such, it is plain to see that every node gets to build on the work done by previous nodes, while still having access to feature maps from nodes before the one directly preceding it. A visualization of such a dense block is shown in figure 2.13.

Not only is this approach very successful, rivaling or beating state-of-the-art performance on multiple object recognition benchmark tasks, it does so whilst typically requiring fewer training parameters. This is because there is no need to re-learn redundant feature maps once they are present, since all feature maps are retained within a dense block; this redundancy of feature maps cannot be addressed simply by applying SCs [**?**].

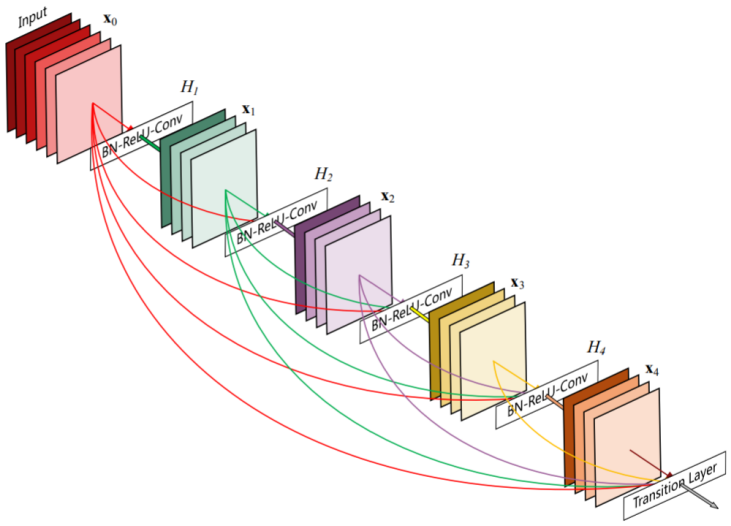


Figure 2.13: Image from [**?**] (reproduced with permission). Visualization of a 'Dense Block' - note how each layer takes all preceding feature maps as input.

For VAEs specifically, the problem of information loss along chains is especially severe, as outlined by Zheng et al. [**?**]. In fact, the typical result for VAEs is that a shallow network (say, with a single hidden layer in both encoder and decoder) tends to do much *better* than a deeper network (with, for example, 10 hidden layers in de-/encoder) - this is in stark contrast to typical results from CNNs, which tend to perform better with increasing model depth. The authors show, both visually (see image 2.14) and through a theoretical exploration of the Fisher Information, that for deeper VAEs there is a strong degeneracy in the latent space, and as a result we suffer a loss of correlation between input and latent code - and thus, by extension, between input and output.

The authors therefore propose Skip Connections as a simple but effective way to preserve information throughout the layers of a deep VAE in order to avoid such degeneracies; they name this architecture SCVAE, and show that it performs comparable to state-of-the-art VAE architectures such as PixelVAE [**?**] or a VAE with enhanced priors such as VampPrior [**?**]. In addition, it is actually fairly easily added to such architectures, and is shown in all cases to yield improvement on MNIST classification.

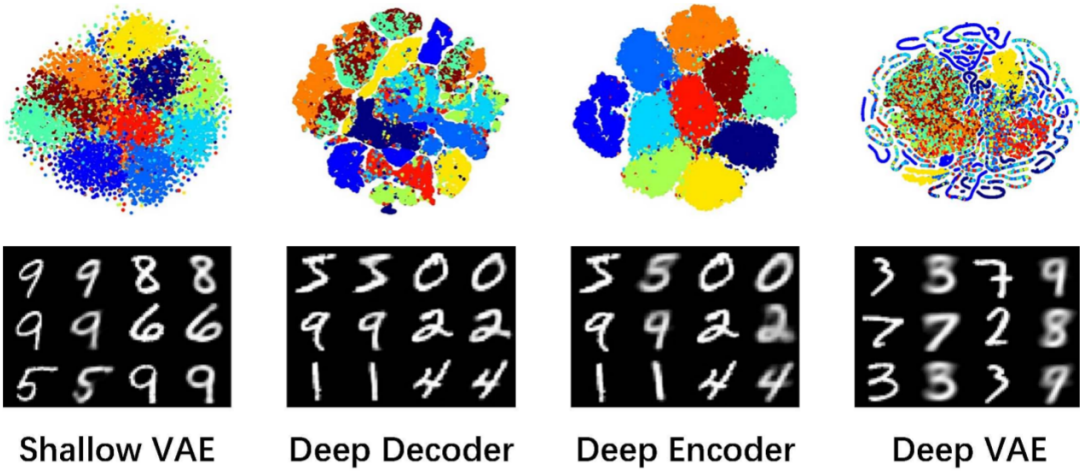


Figure 2.14: Image from [**?**] (reproduced with permission), visualizing the performance of several different architectures of VAE trained on the MNIST dataset, which consists of a large set of handwritten numbers. Original description: "Deeper networks in VAE models. **Upper**: representation of latent variables; **Lower**: ground truth (odd columns)." Note how the shallow VAE has much better performance than the deep VAE, both in terms of results as well as latent space organization. Also note how a model with only a deep decoder results in better reconstruction but worse latent space organisation, while we see the opposite effect for a model with only a deep encoder.

Chapter 3

Methodology

3.1 The dataset

Reconstructing high-quality images of galaxies is, of course, highly dependent on the *original* quality of the images the VAE is trained on. However, obtaining a large quantity of such high-quality originals is, as of yet, somewhat difficult. The largest current survey of galaxy images that is publicly available for use for image simulation purposes is the COSMOS survey, which is carried out using the Hubble Space Telescope (HST) [**?**]. For this research, I used a subsection of the COSMOS dataset as provided for the Great3 Challenge, which has been pre-processed by the challenge organisers through GalSim for ease of use, resulting in a dataset containing 56.062 square galaxy images [**?**]. In addition, this dataset came with two catalog files, which together contained the following information for each galaxy:

1. IDENT: COSMOS identifier.
2. RA, DEC: Right ascension and declination (J2000, degrees). See also appendix B.
3. MAG/MAG_AUTO: F814W magnitude. See also appendix B.
4. BAND: Passband for the observed image (F814W in all cases).
5. WEIGHT: A weight factor to use to account for the fact that larger galaxies are preferentially excluded from this catalog due to the proximity to CCD edges.
6. PIXEL_SCALE: pixel scale for the galaxy and PSF images in arcsec (0.03 in all cases).
7. NOISE_MEAN: mean value of the pixels in the image that do not contain the galaxy. See also appendix B.
8. NOISE_VARIANCE: variance of the pixel values for pixels in the image that do not contain the galaxy. See also appendix B.

9. FLUX_RADIUS: half-light radius for the PSF-convolved object (flux_radius from the COSMOS catalog), in units of pixels.

10. Z_PHOT: Photometric redshift (z_{phot}) from the COSMOS catalog.

11. SERSICFIT: An array of 8 numbers which are the results of fitting the galaxy image to a single Sérsic profile [**?**] with free n. The array entries are:

    a) I: the intensity at the half-light radius

    b) $R_{1/2}$: the half-light radius measured along the major axis, in pixels.

    c) n: Sérsic n.

    d) q: the ratio of minor axis to major axis length.

    e) x_0: the central x position in pixels.

    f) y_0: the central x position in pixels.

    g) ϕ: the position angle in radians (if phi=0, the major axis is lined up with the x axis of the image).

12. Several quantities relating to fitting status: BULGEFIT, FIT_STATUS, FIT_MAD_S, FIT_MAD_B, FIT_DVC_BTT, USE_BULGEFIT, VIABLE_SERSIC. Most notably, the variable VIABLE_SERSIC reflects whether the Sérsic fit is a viable way of representing the galaxy; according to the instructions included with the dataset, this is "generally [...] True, except for very rare cases." As such, it was opted to simplify the problem by simply removing all entries for which this entry was False, meaning that all included galaxies can, according to the authors, be viably represented with a Sérsic profile with the parameters from the SERSICFIT entry.

13. HLR: This is an Nx3 array of the circularly-averaged half-light radii in arcsec.

14. FLUX: This is an Nx4 array of the flux.

These images were by no means uniform in size; the distribution of the sizes of these images is shown in figure 3.1. Training a network on images of unequal size slows down the training process significantly (and needlessly), and so the first step in preparing the data was to make sure that all images were of equal size. To further ease training, we would also like to rotate each image so that its major axis is aligned with the x-axis (alignment angle = 0°). In addition, as the images tended to be a) rather well centered and b) contain mostly empty space, often with imaging artifacts, near the edges of the images (see also figure 3.2), it was desirable to limit the training to the central part of the image containing only the galaxy. Because of this, I opted to crop the images to the central portion.

Correcting the alignment was fairly trivial, as the rotation angle ϕ from the SERSICFIT array, as described earlier, gives us the exact angle for which we want to rotate for each image. Rotation was performed using `Scipy`'s *ndimage.rotate* function [**?**].

Cropping the images requires us to make an educated guess on how much to crop; cropping too little yields little benefit but is safer, while cropping too much would mean losing essential information about

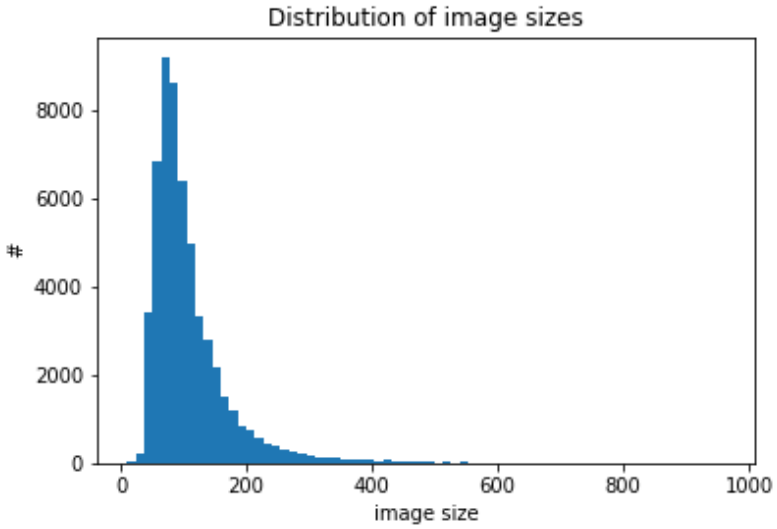


Figure 3.1: Original distribution of image sizes (NxN) in the dataset used. Mean size is 110 pixels, with $\sigma = 70$ pixels. Median size is 90 pixels.

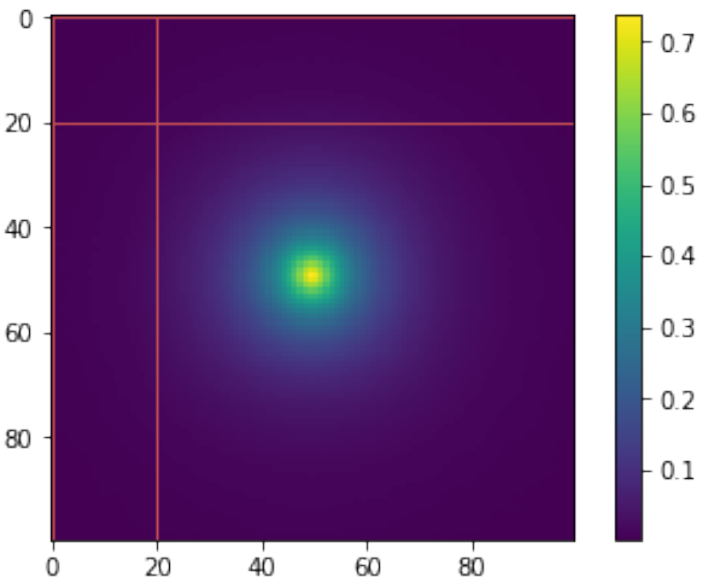
the galaxy's structure. Eventually, a crop of 10% on all sides for every image was chosen, resulting in a total crop of 20% on each axis; this meant that 94.1% of the total average luminosity density was retained, which seemed like an acceptable loss.

Finally, resizing was done using cubic interpolation, through `scikit-image`'s transform.resize function [**?**]. After some consideration, I chose 64x64 pixels as the final size for all images; this was based on the median size of 90 pixels, which, after cropping, left a size of 72 pixels. However, it is worth considering that upscaling generally has a much greater effect on image quality than downscaling. Therefore a slightly smaller image size seemed prudent in order to accommodate the smaller images in the dataset. In addition, using a power of 2 for the final image size seemed like a decent choice, potentially making the images slightly easier to neatly fit into blocks of computer memory.

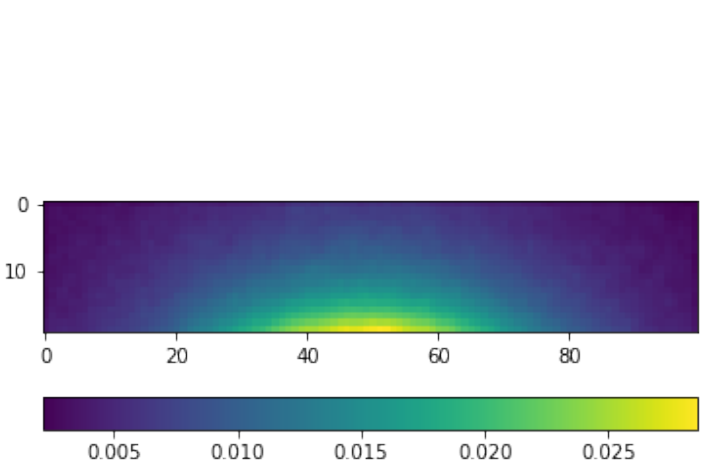
Pre-processing was then done in the order:

$$\text{Crop} \rightarrow \text{Rotate} \rightarrow \text{Resize}$$

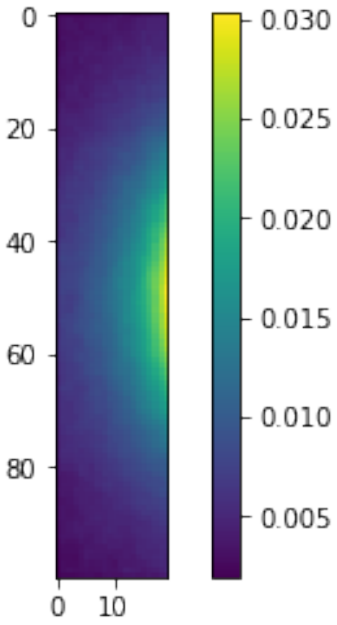
On visual inspection, it quickly become clear that nearly all very small images in the dataset were extremely blurry, with structures often being indistinguishable from background noise. A second problem was that scaling up these small images through interpolation introduced even more artifacts. In the end, I therefore chose to exclude all 87 images with size<32 pixels from the dataset. In the end (including skipping over images which could not be viable approximated with a Sérsic fit, $N = 6$), this left us with 55.969 64x64 images for training. Of these, we set aside 10% beforehand to use for testing, leaving us with 50.373 images for training and 5596 images for testing.



(a) Average luminosity density for all images in the dataset (scaled). Outlined in red are the top and bottom 20% that are shown for greater detail below.



(b) Cutout; top 20% of average luminosity density.



(c) Cutout; left 20% of average luminosity density.

Figure 3.2: Average luminosity density in the dataset. Obtained by rescaling images to 100x100 through interpolation first, and then normalizing by dividing by the maximum (yielding a range of [-1.01, -1.00] over all images). After this, each pixel was taken to be the average in that pixel over all images. Images (b) and (c) depict enlarged cutouts (with adjusted coloring to make details easier to see) from (a).

There are two further steps to prepare an image for training, both of which were performed on load rather than during initial dataset preparation. Firstly, we subtract the value in every pixel by the image's (estimated) mean noise; this is taken from catalog NOISE_MEAN value as described above. Secondly, we

perform a Tanh-normalization so that our data is rescaled to the interval [-1,1]. These operations were all relatively light on terms of computing cost, and allowed for some flexibility during experimentation (e.g. changing the way in which we normalize, choosing not to subtract the noise for comparison's sake, etc.).

Further details which were not directly relevant to the research presented here can be found in appendix B.

3.2 Galaxy VAE

In light of the theory discussed in section 2.3 onward, a VAE seems like a good choice of reconstruction (and, eventually, descriptive or generative) model of galaxy images, or of complex physical systems in general. After all, one would expect the most efficient encoding of a physical system to at least in some way relate to the physical laws and parameters that actually make up that system.

Note that, to enable optimal learning, we don't wish to constrict the system in terms of latent dimensionality too greatly. If the system has N true parameters, and we only give it an N-dimensional latent space, it would have to immediately guess the perfect organization of the latent space, or get stuck in a 'wrong' orientation. It would then most likely be unable to optimize out of it because there is too little room for the latent space distributions to 'shift'. As such, we take a fairly generous estimate for the number of parameters necessary to encode galaxies, and use a 64-dimensional latent space, or $Z_{dim} = 64$, for all cases. This likely means that the final latent space distribution will be highly cross-correlated, and that several of its dimensions will be redundant; we can imagine this as a N-D manifold existing in a Z_{dim}-D space. In theory, it should be possible to simplify this manifold to remove redundant dimensions and project it onto an N-D space, which we might expect to have a much closer resemblance to some ensemble of physical parameters, though we will not go further into this in this research.

In these next sections I will outline the five main VAE architectures that were tested, as well as a few notable variants. I will first outline some constant factors across all models.

ELU activation functions were used after all hidden and input layers. For the likelihood and priors, we use normal distributions, since we are dealing with continuous, physical systems, and so expect to be dealing with continuous distributions for all true parameters. For the noise $\epsilon(z)$ we similarly also use a normal distribution; for each image we obtain a value μ_{img} from the network, and we use $\sigma_{img} = 0.02$ (so that the noise σ is $\frac{1}{50}$ of the maximum). Finally, we used the Adam optimizer ([**?**]) as implemented in `Pyro` for all training scenarios, with a base learning rate of 10^{-5}. The stochastic variational inference procedure was implemented using `Pyro`'s *svi* function [**?**]. The implementation of various specific operations is detailed in table 3.1. The full code for all architectures can be found at https://github.com/PaulHofma/VAE-thesis.

.

3.2.1 VAE 1: Basic FC

To start, I will elaborate on what is, in essence, the most simple variant of VAE one can build. A schematic layout of the model is shown in figure 3.3. For our encoder, starting from a 64×64 black and white (and therefore 2D) image, we first reshape it so that our data is 1D. We then feed it to a $[4096 \times H_D]$ FC

Table 3.1: List of operations and their specific implementation.

Operation/Layer/etc.	Function	Library
Reshaping	*reshape*	`Numpy` [**?**]
Sampling	*sample*	`Pyro` [**?**]
Concatenation	*cat*	`Pyro`
Normal Distribution	*distributions.Normal*	`Pyro`
Fully Connected Layer	*nn.Linear*	`PyTorch` [**?**]
Convolutional Layer	*nn.Conv2D*	`PyTorch`
Deconvolutional Layer	*nn.ConvTranspose2D*	`PyTorch`
MaxPool	*nn.MaxPool2d*	`PyTorch`
Nearest-Neighbour interpolation	*nn.UpsamplingNearest2d*	`PyTorch`
Batch Normalization (2D)	*nn.BatchNorm2d*	`Pytorch`

layer with an ELU activation, and then to two separate $[H_D \times Z_D]$ FC layers which encode μ_z and σ_z, respectively. These are the μ and σ for our normal distribution in latent space for that particular image, $\mathcal{N}_z(\mu_z, \sigma_z)$. Sampling from this distribution, we get the code for the image, z. Here, as mentioned, I used $F_C = 64$; in addition, I used $H_D = 512$ across the model.

For the decoder, We then feed it back through a mostly identical but inverted setup ($[Z_D \times H_D]$ FC, ELU, $[H_D \times 4096]$ FC) to obtain μ_{img}, which we feed into $\mathcal{N}_{img}(\mu_{img}, \sigma_{img})$, with $\mu_\sigma = 0.02$ as indicated earlier. Sampling from this, we get a set of 4096 numbers; reshaping to 64×64 pixels gives us our output image.

3.2.2 VAE 2: Deeper FC

From our previoust, simplest variant, we now simply 'upscale' slightly by adding some additional layers. That is, we now use four FCs rather than just two in both the encoder and decoder. For the encoder, we use with FC layers with sizes $[4096 \times 8H_D]$, $[8H_D \times 4H_D]$ and $[4H_D \times H_D]$, which then feed into the $[H_D \times Z_D]$ size layers that generate μ_z, σ_z, as before. Similarly, for the decoder we use FC layers with consecutive sizes $[Z_D \times H_D]$, $[H_D \times 4H_D]$, $[4H_D \times 8H_D]$, $[8H_D \times 4096]$, with $H_D = 128$ across the model.

3.2.3 VAE 3: Basic ConvNets

Moving on from the simplest possible network, we introduce convolutional layers; the layout is shown in figure 3.5. Note that at this point we still require some few-to-many FC layers to make the dimensions work out at several different stages. In an attempt to reduce the aliasing that interpolation would introduce at such an early stage, I opted not to include upscaling layers (i.e. interpolation) in the decoder model. We apply our single deconvolution with S=2, after which we again plug the resulting feature maps into the final FC layer for μ_{img}. Across the model, we use $Z_d = 64, F_d = 32$.

3.2.4 VAE 4: Deeper ConvNet

In this next step, we start looking at a slightly deeper convolutional network in order to improve performance; the layout of this model is shown in figure 3.6. We employ four 'blocks' of Conv-ELU-MaxPool

in the encoder, followed by the usual reshape-FC-Normal to encode the input image. For the decoder, we again use an initial FC layer to 'rescale' to 8×8, then apply three blocks of DeConv-ELU-MaxPool, using $S = 2$ so that our deconvolutions again also serve as upscaling layers. After these three blocks, we have a fourth block with $S = 1$ to prevent a further growth in image size. We then again feed the results to the final FC layer, obtaining μ_{img}. Across the model (as well as the two variants), we use $Z_d = 64$ and $F_d = 16$; attempts to train using $F_d = 32$ were hindered by the available RAM.

VAE 4, variant 1: β-VAE

For this architecture, we also explored two variants. The first is the β-VAE, as proposed by Higgins et al. [**?**]. In this case we change nothing at all about the general design of the model, but rather use a loss function that is slightly different from the standard ELBO. Recalling equation 2.78, and shuffling terms:

$$\mathcal{L}(\theta, \phi; x) = \log p_\theta(x) - D_{KL}(q_\phi(z|x) \,||\, p_\theta(z|x))$$

Higgins et al. suggest the addition of a hyperparameter β (with $\beta > 1$) to amplify the regularizing KLD term. This increased regularizing pressure on the ELBO means the model is encouraged to learn a disentangled version of the latent space organization, which they show to work well in their research.

$$\mathcal{L}(\theta, \phi; x) = \log p_\theta(x) - \beta D_{KL}(q_\phi(z|x) \,||\, p_\theta(z|x)) \tag{3.1}$$

For this, we explore a few different values for β (being $\beta = 2, 5, 10, 20, 50, 100$) to see whether the β-enhanced loss has any effect on our final images, and whether the latent space ends up being better organised - the latter would make sampling from the latent distribution for a potential generative model much simpler.

VAE 4, variant 2: Conditional VAE

Next, we explore a conditional setup, or CVAE, for the deeper ConvNet model. For this variant, we employ a setup as proposed by Kihyuk et al. [**?**], and we would expect it to perform (slightly) better than the basic Mid-Deep ConvNet model. In practice, the simplest implementation of such a conditional setup is to concatenate some known variables about the network to both the input (before it is fed to the encoder) as well as the latent variables (before it is fed to the decoder). Our 'known variables', in this case, are those from the list in section 3.1. Since we normalize our images to the range [0,1], the intensity I becomes meaningless. Likewise, since we subtract the noise mean μ_{noise} from every image before processing, including this only adds needless parameters; the same reasoning applies to ϕ, since we rotate all images to set $\phi = 0$. Finally, from both a visual inspection as well from inspection of image 3.2 it is clear that all images are centered fairly well, and so the x_0, y_0 parameters would not add much meaningful information. This leaves us with five relevant parameters.

1. Noise variance σ^2_{noise}
2. Z_{phot}
3. $R_{1/2}$

4. n

5. q

Concatenating these information variables to the image requires us to do add an additional reshape and FC layer, as we cannot simply concatenate the (5×1) 'information vector' to the (64×64) image. As such, we first reshape the image to a (4096×1) vector before appending the information vector, so that we end up with a (4100×1) vector, which we then plug into the added FC layer (size 4100×4096) to end up with a 1D vector that we can reshape back to our original image size.

Initial testing showed that these models were, unfortunately, numerically rather unstable, typically diverging after approximately 10 epochs. To solve this problem I was forced to modify the architecture slightly, as this appears to be a normalization problem that is not adressed by simply using or adding additional ELU layers. The minimal functional modification turned out to be replacing all ELU layers with LeakyReLUs, and adding batch-normalization layers (see [**?**]) after the first and third and layer for both the encoder and decoder. unfortunately, this appears to be a normalization problem that is not adressed by simply adding ELU activations, as testing showed this did not have the desired stabilizing effect. Other than these changes, the general layout of the model was fortunately mostly preserved. However, it is still important to note that not all improvements in quality are necessarily purely attributable to the conditional nature of the network.

3.2.5 VAE 5: DenseVAE

For constructing our DenseNet-based VAE, which we shall call DenseVAE, we adapt the structures of DenseNet-B as proposed by Huang et al. in their original paper [**?**] for both the encoder and decoder. We use (nearly) identical setups for both the encoder and decoder, as shown in figure 3.7.

We start the encoder with a Conv-ELU-MaxPool 'initial' set of layers, with $[K = 5, S = 1, P = 2, F = k]$ for the convolution, and $[K = 2, S = 2]$ for the MaxPool. We follow this up with three sets of dense blocks, with transition layers in between. The transition layers (shown in figure 3.9) consist of a Conv-MaxPool block with $[K = 1, S = 1, P = 0, F = F_{in}]$ for the convolution and $[K = 2, S = 2]$ for the pooling layer. The dense blocks, in turn, each consist of six consecutive 'dense units' (visualized in figure 3.8), each of which takes the output feature maps of all previous layers, as well as the original input, as input. These dense units are composed of ELU-Conv(1x1)-ELU-Conv(3x3) layers. We apply an ELU layer, then feed into a $[K = 1, S = 1, F = 2k]$ convolutional layer (the *Bottleneck layer*), then another ELU layer, then into a $[K = 3, S = 1, P = 1, F = k]$ output convolution. We opted for $2k$ output layers for the bottleneck, as we use a smaller number of dense units per block compared to Huang et al. After the last dense block, we apply a final ELU layer, followed by a final MaxPool ($[K = 2, S = 2]$). We then reshape the $(4 \times 4 \times k)$ data to be 1D, and feed this to our usual FC layers to transform into μ_z, σ_z, from which we create the normal distribution for z. A final note here is that we opted to skip the Batch Normalization layers present in the DenseNet paper, as we use ELU layers rather than ReLU, which should make batch normalization redundant. For the decoder, the process is much the same, where we mostly replace every instance of a convolutional layer with an instance of a deconvolutional layer, since

we are trying to extract a high-dimensional representation from a low-dimensional representation. An additional change is that we use nearest-neighbor interpolation (NNI) as the inverse-MaxPool operation. For clarity's sake, we will refer to these types of dense blocks, dense units, and transition layers as *inverse* dense blocks, dense units, and transition layers.

To start the decoder, we reshape back to $(8 \times 8 \times 1)$. We then use a DeConv layer to upscale slightly, with $[K = 3, S = 2, P = 1, F = k]$. We then follow up with our three inverse dense blocks, with the two inverse transitions in between scaling up, so that in the end we are left with k sets (60×60) of feature maps. These are then, as usual, reshaped to 1D arrays and fed into the FC layer that outputs μ_{img}.

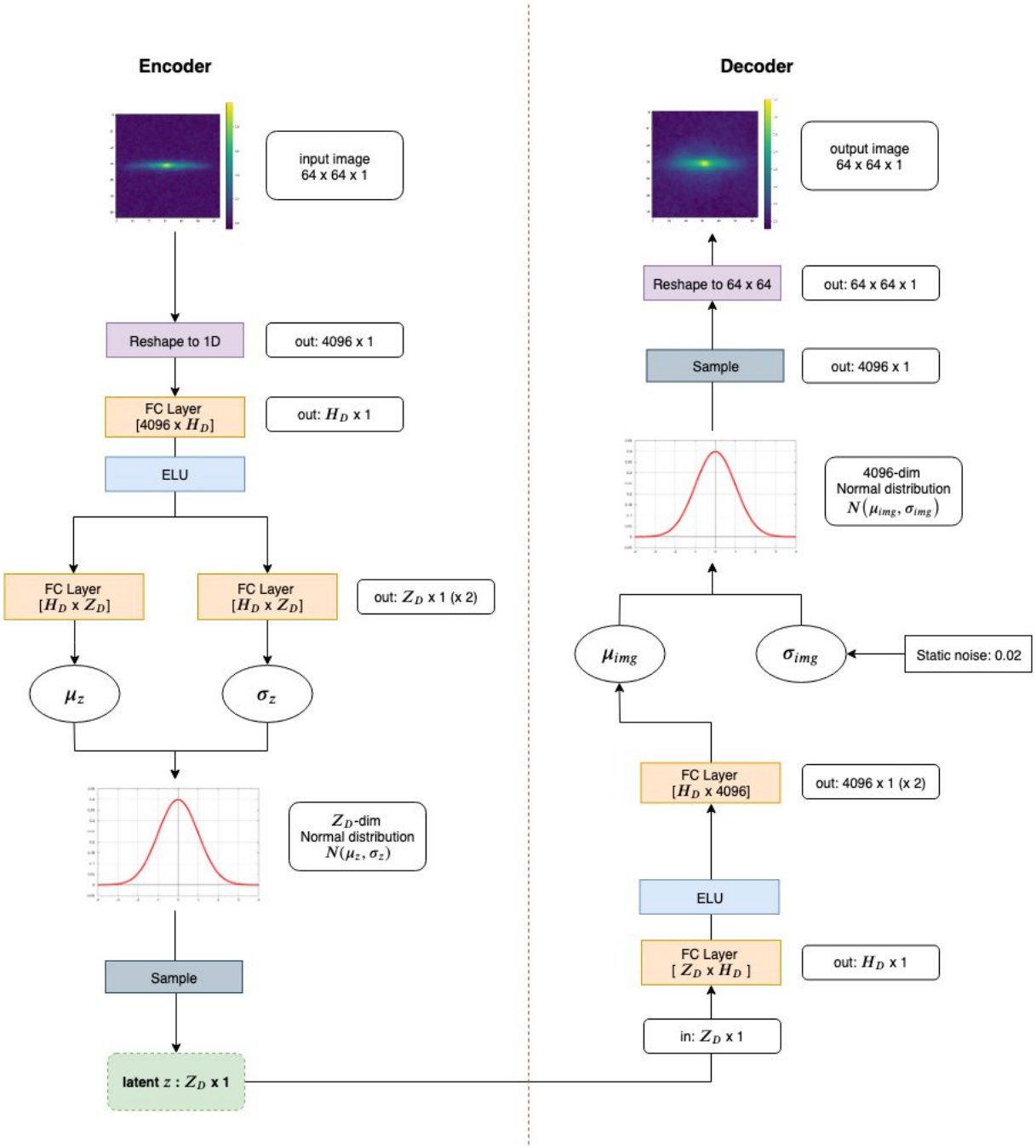


Figure 3.3: Schematic layout of the basic FC VAE model, as described in section 3.2.3, for both the encoder and decoder networks. Z_D is the size of the latent space, H_D is the size of the hidden layer. In this research, we used $Z_D = 64$ and $H_D = 1024$.

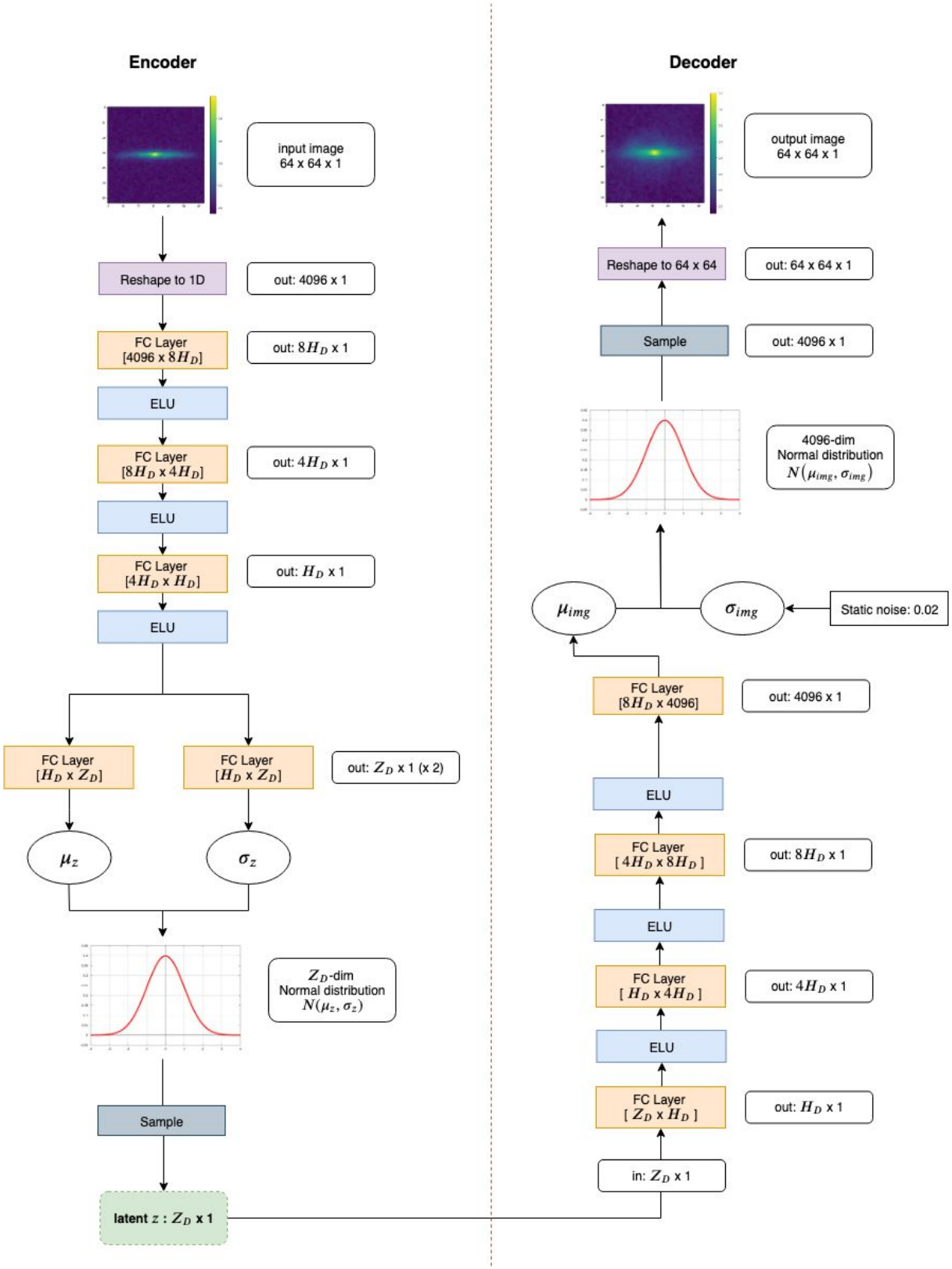


Figure 3.4: Schematic layout of the deeper FC VAE model, as described in section 3.2.3, for both the encoder and decoder networks. Z_D is the size of the latent space, H_D is a factor determining the size of the hidden layers. In this research, we used $Z_D = 64$ and $H_D = 128$.

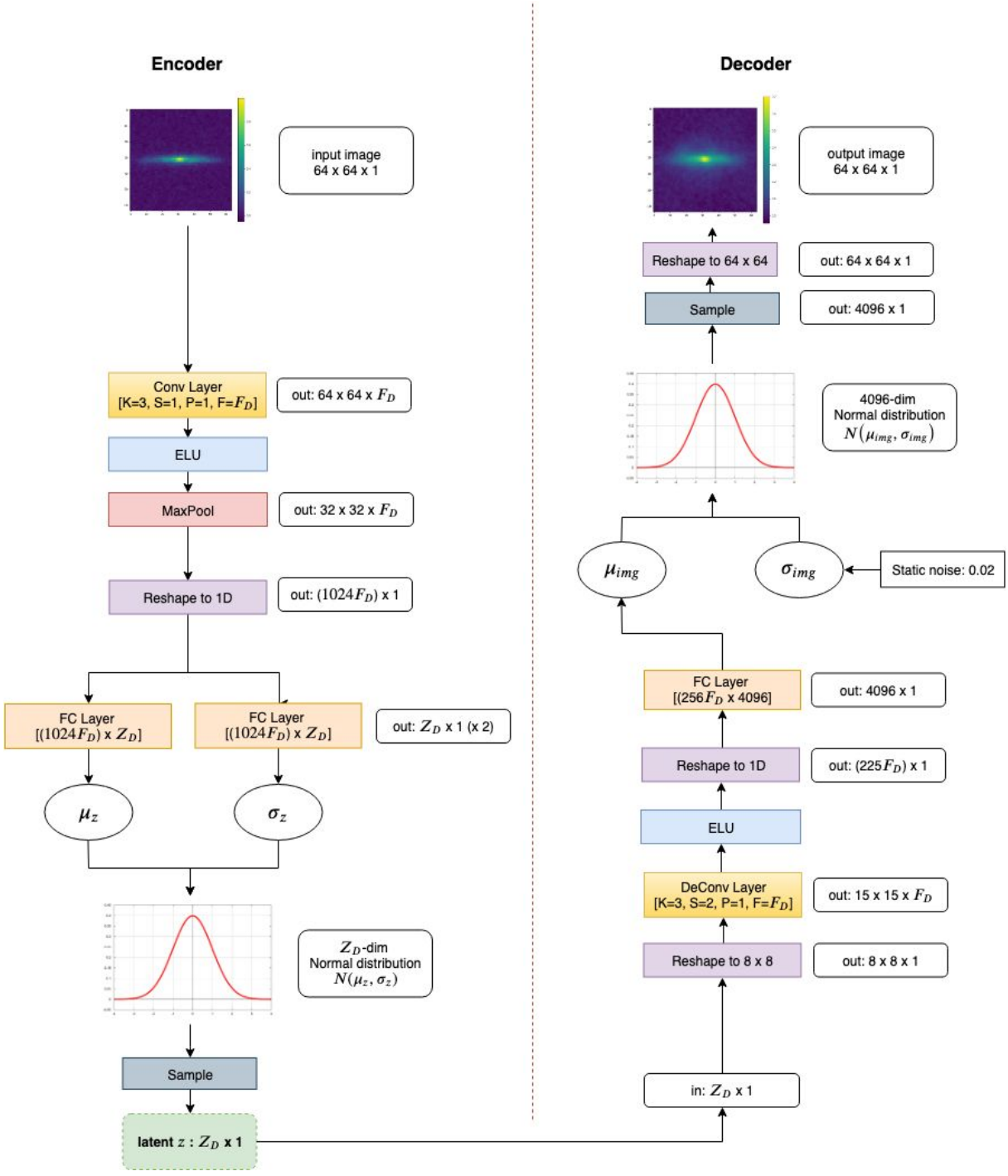


Figure 3.5: Schematic layout of the Basic ConvNet model, as described in section 3.2.3, showing both the encoder and decoder networks. Z_D is the size of the latent space, F_D is the number of feature maps used for the convolutional layers. In this research, we used $Z_D = 64$ and $F_D = 32$.

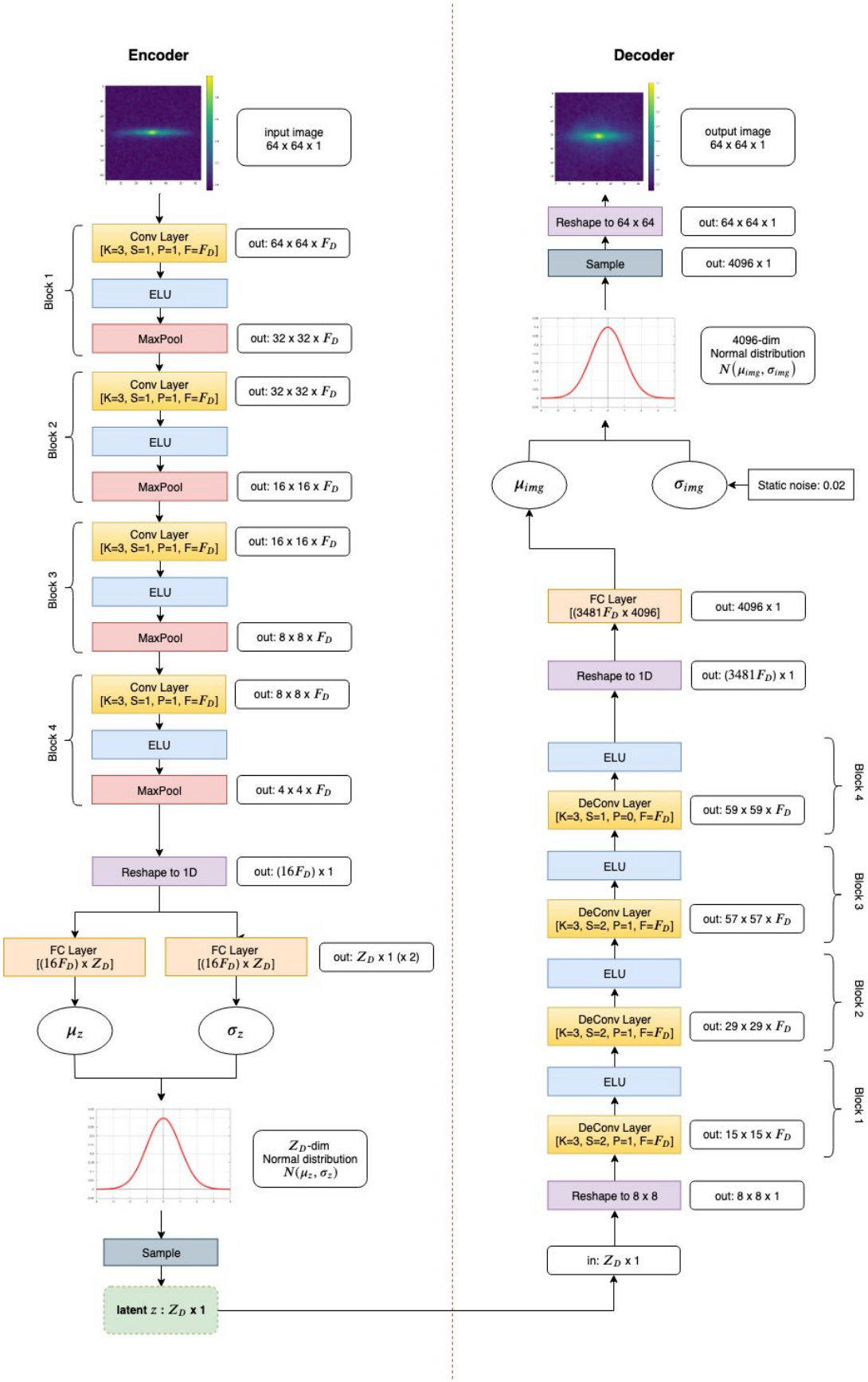


Figure 3.6: Schematic layout of the deeper ConvNet model, as described in section 3.6, showing both the encoder and decoder networks. Z_D is the size of the latent space, F_D is the number of feature maps used for the convolutional layers. In this research, we used $Z_D = 64$ and $F_D = 16$.

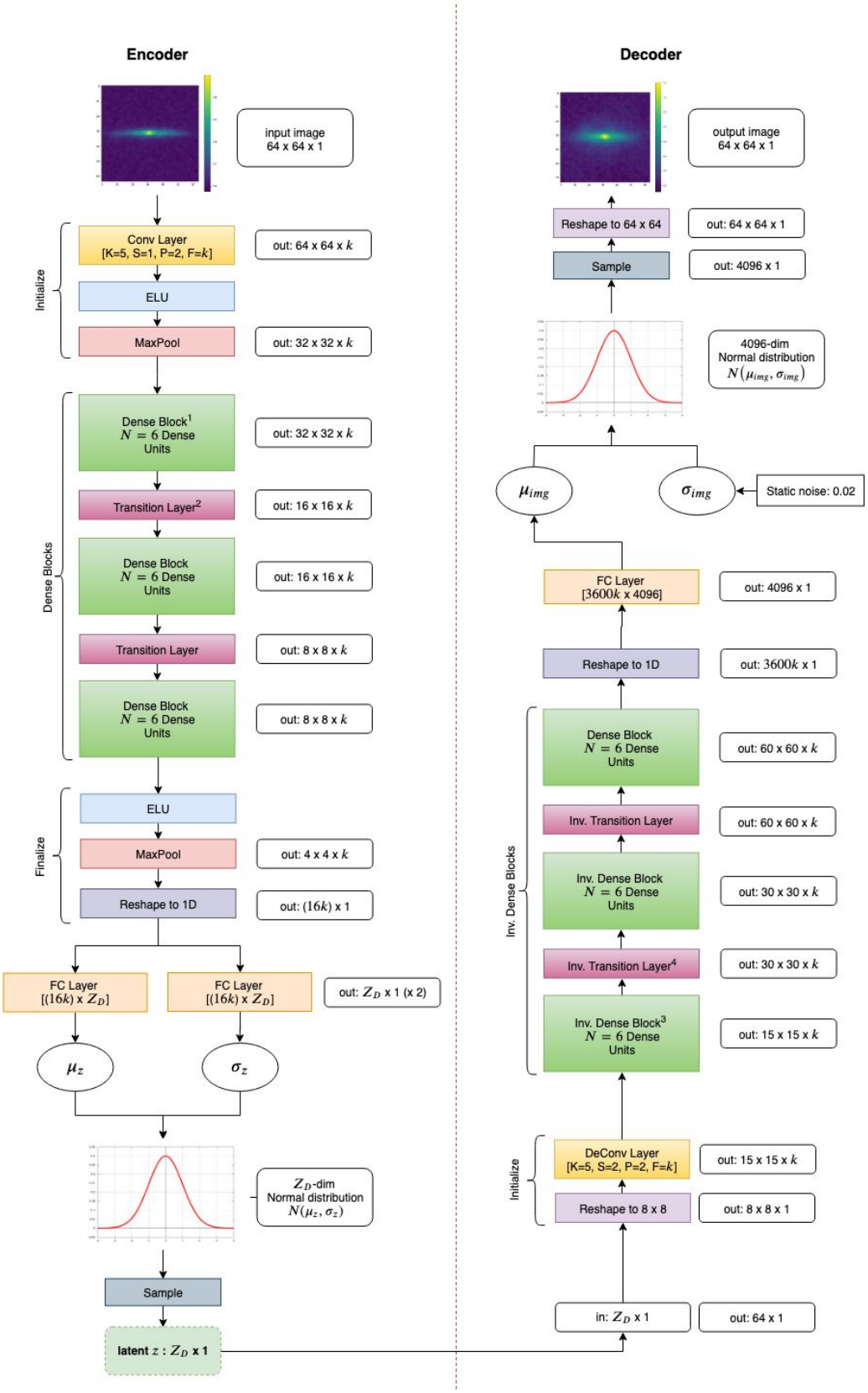


Figure 3.7: Schematic layout of the DenseNet model, as described in section 3.2.5, showing both the encoder and decoder networks. Z_D is the size of the latent space, k is the growth rate. In this research, we used $Z_D = 64$ and $k = 16$. Layouts for the (inverse) dense units and (inverse) transition layers are shown in figures 3.8 and 3.9, respectively.

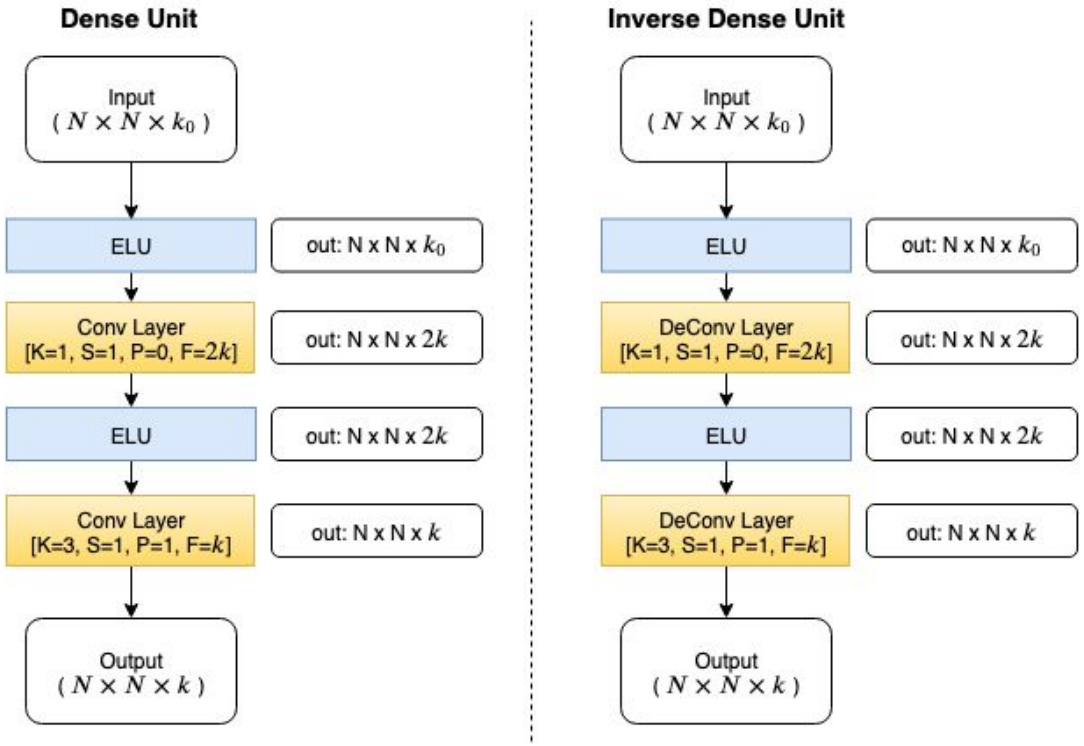


Figure 3.8: Schematic layout of the Dense Units used, which are the building blocks of the Dense Blocks in the DenseNet model as describe in section 3.2.5.

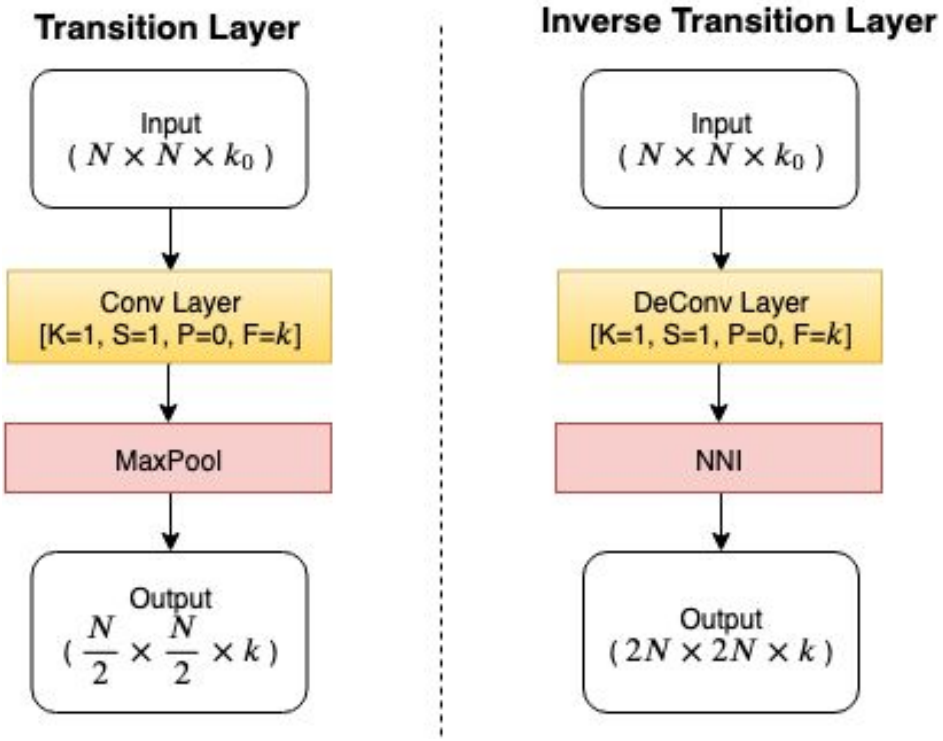


Figure 3.9: Schematic layout of the Transition Layers used in the DenseNet model as describe in section 3.2.5.

Chapter 4

Results & Discussion

"Reconstruction sheets" were created for all architectures; for these, a selection of 32 images from the test set (outlined in green) is shown side-by-side vertically with their reconstructed counterpart (outlined in red, and always directly below the original). Note that the same set of original images is used across all reconstruction sheets. These reconstructions sheets are unfortunately best viewed at larger sizes so that substructure reconstruction is better visible. As a result, it was opted to include these in appendix A rather than as part of the main text for legibility's sake. As I shall be going over several of these in some detail, we shall from here on out refer to specific subfigures on a sheet as by their row and column numbers, in the format R#C#. For example, for a given figure, the subfigure in row 1, column 5, shall be referred to as R1C5. Note that, by design, all odd rows are original images, and all even rows are reconstructions. When referencing a single original-reconstruction pair, I will refer to these by the indicator for the original image.

In the scenario's presented here, all models were trained for 101 epochs, the ELBO scores of which are shown in figures 4.1 through 4.5. At this point most models were still trending downward (slightly) in their losses, and had not yet reached a point where they were overfitting (see fig. 4.2). This means that there quite likely was still some further improvement to be gained through further training - regardless, the relative differences between the various models' training and test performance is at this point clearly visible and not expected to change much.

Interestingly enough, if we are looking only at the ELBO, it would appear that the Basic FC VAE (and thus our most basic possible model) appears to show the *best* performance. Taking in mind what was discussed in section 2.6 regarding VAE depth and information degeneration, this does not come as all that big of a surprise: we might expect a shallower model to do better purely on the basis of information conservation. What is interesting that it also clearly beat out the basic convnet (which has effectively the same depth), while convnets generally perform (much) better at image recognition tasks. However, looking at the slope of both trend lines (best visible in fig. 4.3), one might speculate that performance of

the basic ConvNet model would approach that of the basic FC model for some number of epochs greater than the 101 used here.

Another unexpected result is that the basic FC model *also* beat out the DenseVAE, which was constructed specifically to combat this information loss. From figure 4.3, it is clear to see that the DenseVAE performs appreciably worse (in terms of ELBO) compared to the basic FC model, and the trend appears to have stagnated for the most part as well. This can perhaps be explained by the fact that while the DenseVAE is designed to optimize information flow, this apparently need not necessarily result in better ELBO scores directly. Rather, we might expect to see the improved information conservation in the form of (visually) better reconstructions. We are left to conclude that while the ELBO is a good training objective for VAEs, perhaps after-the-fact quantification of the quality of the reconstructions might be better served by a different estimator (perhaps something as simple as an MSE). We shall have to leave this objective quantification to some future research; for now, I will limit myself to (an admittedly inherently subjective) visual inspection of the results. In doing so, the Basic FC model (being the one with the highest ELBO score) shall serve as the benchmark.

4.1 Comparing reconstructions

Comparing the reconstructions for the basic FC (fig. A.1) against those for the deeper FC (fig. A.2), we can make some immediate observations. Generally, as expected, smaller-scale features are more smeared and less-well defined, which is especially obvious in (for example) R1C6 and R7C5. Another interesting 'feature' is that the deeper FC model tends to create circular 'noise' features; these artifacts are especially visible in the reconstructions R4C5-8.

Comparing the basic FC against the basic ConvNet model (fig. A.3), we can appreciate already that the latter does a better job of reconstructing galaxies than the deeper FC model. At this point, we note that both models have trouble reconstructing the brightest spots in otherwise faint images (see, for example, the various point sources as well as the rather odd galaxy R5C3), but do at least a satisfactory job of reconstructing relatively faint spiral arms (present in both R3C3 and R5C6). Most visually significant differences are found in the noisier images; for example, the reconstruction of R5C7 has a much more distinct 'flattened' core in the basic FC model compared to the basic ConvNet model, where this core has become more of a point source. Another interesting example here is R7C5, where the ConvNet-based model presents an arguably more 'physical' scenario (though the original image is so noisy it is hard to be certain).

Moving on to the deeper ConvNet model (fig. A.4) we see again a clear decrease in quality compared to both of the simpler models. Not only do we see the return of the circular noise patterns, but the model also appears to have developed a strong preference for flattened disk-shaped galaxies (especially visible in R3C4 and R5C4).

For the variants of the deeper ConvNet, we shall start by discussing the β-VAE. Here, we shall start our discussion with a look at figures 4.4 and especially 4.5, which show the ELBO evolution of these variants. Perhaps the most interesting feature is that these generally follow the same evolution as the regular deeper ConvNet model: a plateau phase until about the 20th epoch, after which we get a swift

drop, followed by another near-plateau, after which more gradual descent sets in. It is, of course, fairly difficult to judge this set of models on the basis of pure ELBO, since they by their very nature should be expected to have higher ELBO scores than the models with a 'regular' ELBO.

Comparing them on grounds of ELBO, then, becomes a matter of estimating how large the β-enhanced KLD is compared to the log prob. part. From a quick numerical inspection, it became clear that at early times the KLD part tends to be only a small component of the ELBO ($< 1\%$). This means that at such early times the reconstruction error in the ELBO is prioritized, which in turn explains why the early evolution of all variants is so similar. By the same token, this means that at late time, the relative contribution of the KLD term is likely relatively much larger compared to early times, and so is presumably responsible for the divergence of the different β variants from around epoch 20 onward (and then further divergence from around epoch 40 onward). Observing that the divergence of the various models at end time is still fairly small, one might than also suspect that simply training for longer would allow one to reap more benefits from the added β parameter.

However, there are still some appreciable visual differences in the reconstructions (shown in figures A.5 through A.10). From a quick visual inspection, it would appear that $\beta = 20$ ends up giving the best results (which was perhaps to be expected, as this was also the value that was found by Higgens et al. [**?**] to result in optimal performance). Most notably, both $\beta = 50$ and $\beta = 100$ introduce a strange artifact where all non-point source galaxies appear to have their brightest point inverted (becoming a darker rather than brighter pixel after reconstruction). Two details that are highly noticeable for the $\beta = 20$ compared to the regular $\beta = 1$ case, is that the preference for flattened galaxies is still there; however, central, high-luminosity pixels are now better reconstructed (closer to the original brightness, while for $\beta = 1$ these were typically too dim in reconstructions), and the circular noise patterns seem to be at least suppressed compared to $\beta = 1$. However, despite all this, I would still be hard-pressed to describe these as 'good reconstructions', especially compared to the basic FC or basic ConvNet model.

Moving on to the Conditional variant, we see a very clear improvement in the ELBO compared to the regular deeper ConvNet model. I should preface this, however, by saying that one must be conservative in drawing conclusions from this, as the model layout is not insignificantly changed from the original deeper ConvNet model. From visual inspection, the first thing that catches the eye is that the conditional model recreates the noise in much more granular fashion compared to the previous models discussed. In addition, overall performance seems very good, matching or beating the basic FC model generally speaking. Point sources and bright spots within large structures are generally reproduced very swell, though it still has problems reconstructing R5C3 (as has been the trend), where the basic FC model still does a slightly better job reconstructing the brightest part. Perhaps the most important trade-of between it and the basic FC model is that is tends to do better at filtering out heavier noise (compare R1C6, R5C8) while slightly losing out on faint substructures (compare R1C3, R1C4).

Then, finally, we compare the results from the DenseVAE to those from the basic FC. Interestingly, the part that the DenseVAE does an especially good job replicating is the noise (in both intensity and granularity); while not exactly the result we would hope for, it is interesting to remark nonetheless. Many reconstructions are on par (or arguably slightly better) than those from the basic FC; point sources, both with and without dim substructures, typically are reproduced very well (examples: R5C5, R5C6, R7C3,

R7C4). An interesting comparison can be gotten from R3C5-8; these have similar noise levels, and especially the middle two galaxy images are oddly shaped. What we can clearly see here is the increased granularity that DenseVAE allows, where much more of the images substructure is conserved in the reconstruction. However, this comes at the cost of extrapolating power; a good set of examples is R5C7-8, where the basic FC model clearly is attempting to recreate something that at least looks like a galaxy, while the DenseVAE achieves a more 'granularly accurate' reconstruction, which is a result overly dominated by the high noise in the image.

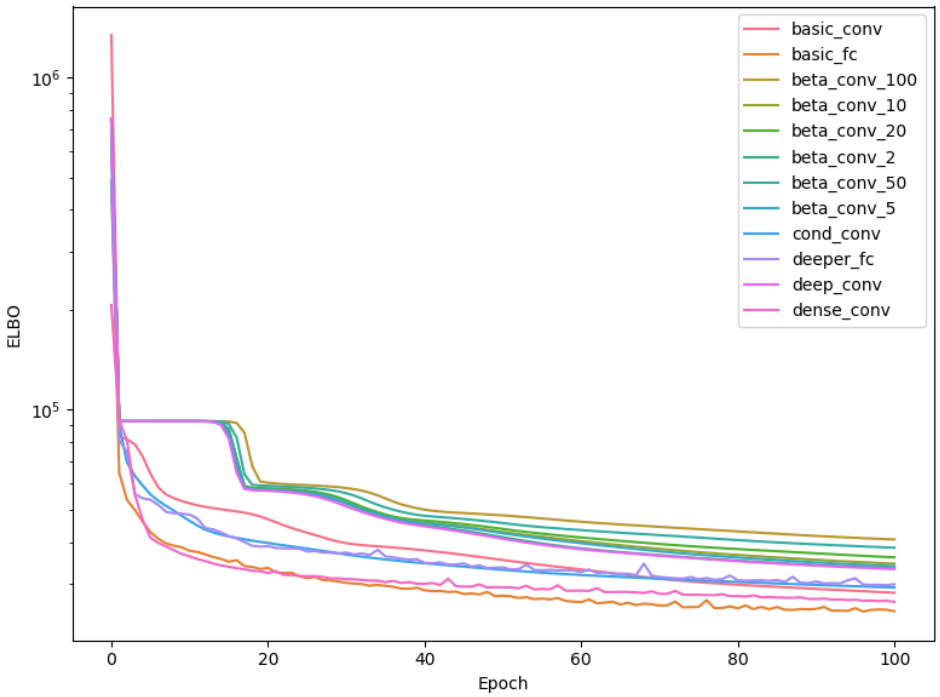


Figure 4.1: ELBO over the training set for all models.

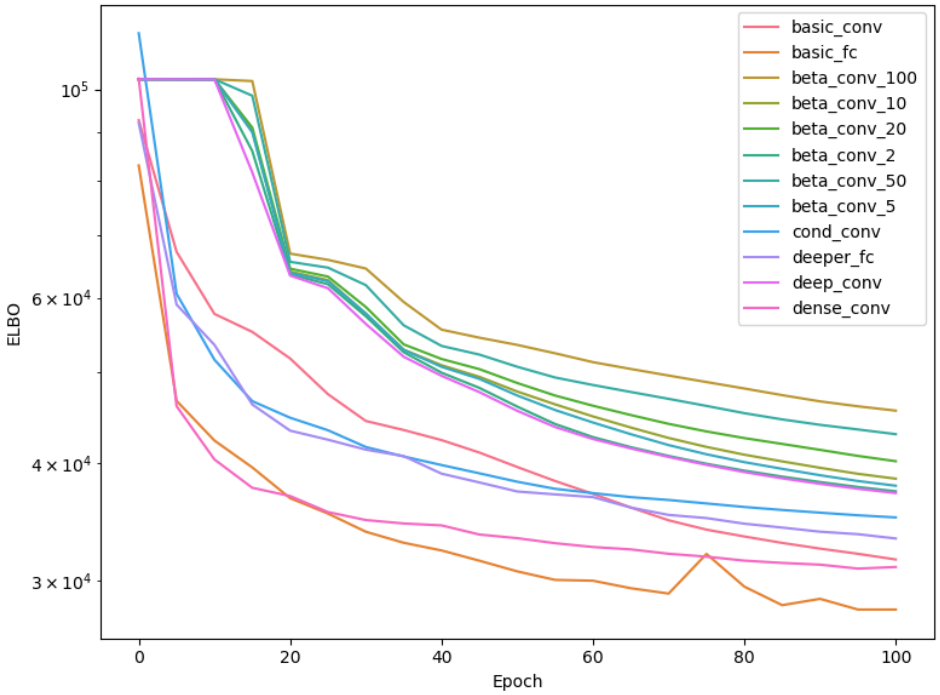


Figure 4.2: ELBO over the test set for all models.

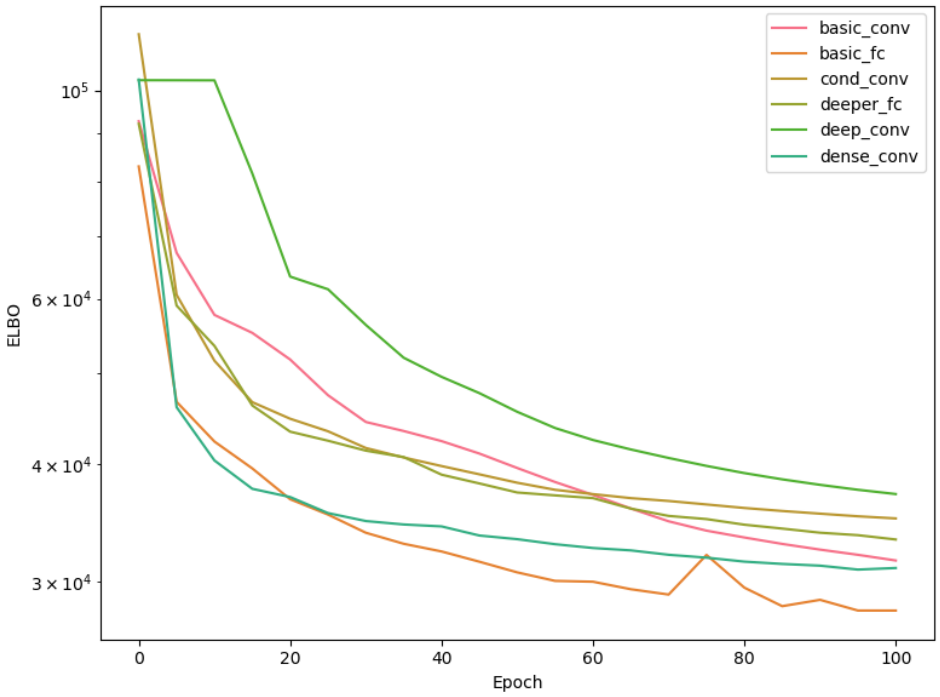


Figure 4.3: ELBO over the test set for the 6 best performing models.

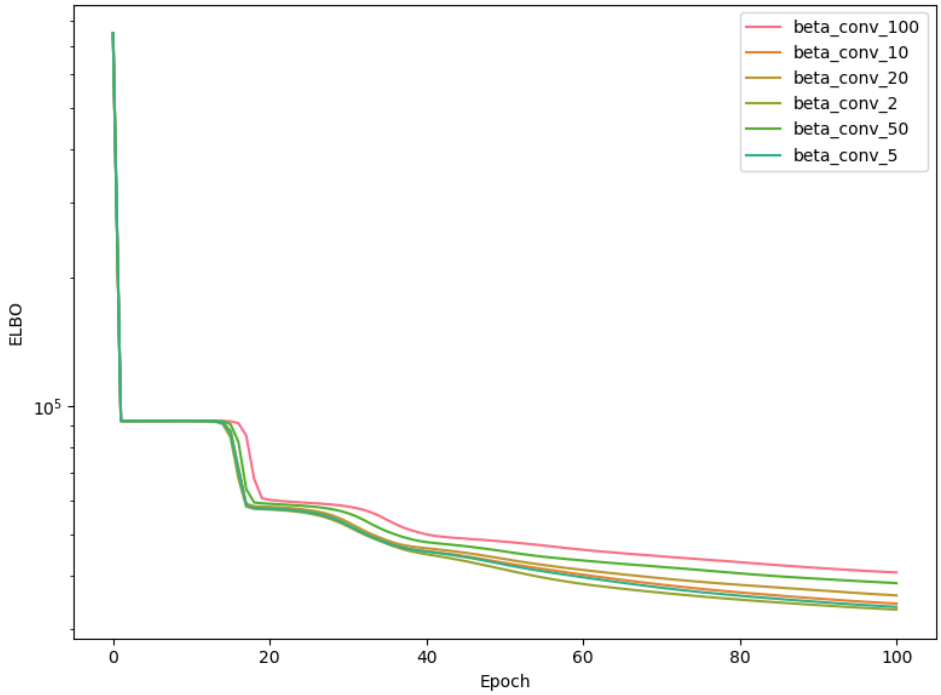


Figure 4.4: ELBO over the training set for the 6 different values of β tested for the β-VAE.

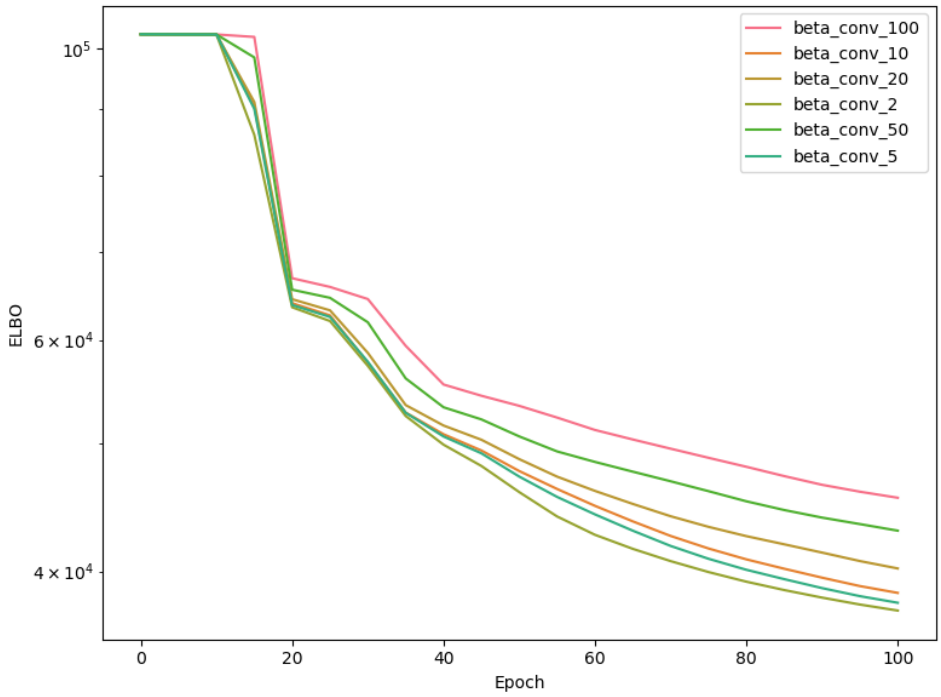


Figure 4.5: ELBO over the test set for the 6 different values of β tested for the β-VAE.

Chapter 5

Conclusion

The initial goal for this research was to build a VAE that is capable of reconstructing images of galaxies with such accuracy that it could be extrapolated into a sufficiently accurate generative model of galaxies. Such a model would potentially be useful for a wide variety of (statistical) research, but in this case the primary interest was creating a model that would be good enough for use in the *Deep Lensing* pipeline.

From the results as presented above, a few clear conclusions stand out. First, as we might have expected, the basic FC and basic ConvNet models are clearly outperforming their deeper counterparts - the degenerative nature of deeper VAEs clearly plays an important role here. As a result, as we might have expected, the DenseVAE does a much better job than its deeper counterparts conserving information - this is unfortunately mostly expressed as a higher-fidelity reconstruction of the noise present in an image. On the surface, this was perhaps to be expected since the noise does, in most cases, make up the majority of the galaxy image by necessity[1].

For the purposes of creating a generative model of galaxies, this seems like a suboptimal result. However, one might expect that training this model on a more noise-free dataset (either by further massaging the data currently available, or by utilizing some future dataset that inherently features less noise) would therefore be able to deliver significantly better results, especially since the model was still improving at the point training was terminated and so presumably has more room for improvement in general (as was the case for all models).

Another interesting result is that both variants of the deeper ConvNet (by itself one of the worse performing models, really only winning out over the deeper FC) achieved some improvement over the original. For β-VAE, even in the apparently optimal neighbourhood of $\beta = 20$, this improvement was fairly minimal as far as actual results are concerned, although from the increasing relevance of the KLD component at late training times, I suspect that this effect might be increased if we allow the model to

[1] After all, the 'average galaxy', as we saw in figure 3.2, is a point source with a half-light radius of a few pixels; as such, for the average image the actual galaxy only makes up a small part of the whole image, whereas noise will be present throughout the entire image

train for longer. In addition, β-VAEs should have a positive effect on the priors, potentially making the actual construction of a generative model easier.

The conditional VAE in contrast, resulted in a significant improvement over its baseline, yielding a significantly lower ELBO at late training times, and yielding substantially better reconstructions overall that generally were a match in quality for the basic FC and basic ConvNet VAEs. However, the introduction of persistent numerical instabilities was rather worrisome, even going so far as to require significant changes to the model in order for them to be resolved. As such, it is not impossible to imagine that at least part of the success is due to the slightly different architecture (with LeakyReLUs instead of ELUs and adding a few BatchNorm layers). Still, the potential offered here seems fruitful ground for further investigation.

To conclude, it would seem that some further research will still be necessary if our desire is to build an expressive enough VAE to accurately reproduce galaxies, as the results as presented here are not quite developed enough to be able to function as a stable basis for a generative model. However, potential has certainly been shown, and there are several different directions for future research that seem potentially fruitful.

5.1 Directions for future research

A first important step would appear to be to procure resources so that models can be trained for longer times. During the research performed here, none of the models ever showed any signs of overfitting, which means that the model still had (potentially significant) room for growth. I would suggest, therefore, that a good first step would be to increase training times to 500 epochs, and perhaps even longer if necessary.

A second problem was the clarity of the available training data. Clearly, the provided data on the noise levels was not enough to nullify the noise in the images - especially for the more 'granularly accurate' models, like the DenseVAE, this posed a significant problem. In addition, there were still many images in the dataset that were still clearly composed of an overwhelming amount of noise - for this, one need only have a look at the reconstruction sheets in appendix A, where several of such images are already present in a selection of 32 images. A cleaner dataset, as remarked before, would likely have resulted in a significantly more accurate DenseVAE, as it would have allowed training to focus more on learning features of galaxies rather than features of noise (at which the model performed quite admirably).

One important direction that this research might take from here, other than addressing these problems, is further combination of the different successful model types. The conditional variant performed admirably when applied to the Deep ConvNet model, which showed lackluster results by itself; as such, it would be very interesting to see if we could get similar improvements by adding conditionality to the basic FC and ConvNet models and the DenseVAE. The same goes, of course, for the β-VAE.

Finally, an important next step in actually constructing a generative model lies in properly approximating the learnt priors. From some quick experimentation, it was already found that plain or multivariate Gaussians typically do a very poor job, and result in a generative model that tends to produce as many 'realistic' output galaxies as it produces evidently unphysical 'garbage' galaxies. Slightly better results are expected from Gaussian mixture models, but this was not tested extensively. A further problem in this

area is that our models, with $Z_D = 64$, likely contain many superfluous dimensions, making the topology of our latent space potentially highly complicated. A next step, then, would be to decrease the size of the latent space and organize it more efficiently. β-VAEs could certainly be of assistence in this regard, as could the concept of normalizing flows, recently put forward by (among others) Rezende and Shakir (2015) and Kingma et al. (2016) [**?**, **?**].

Appendix A

Reconstruction Sheets

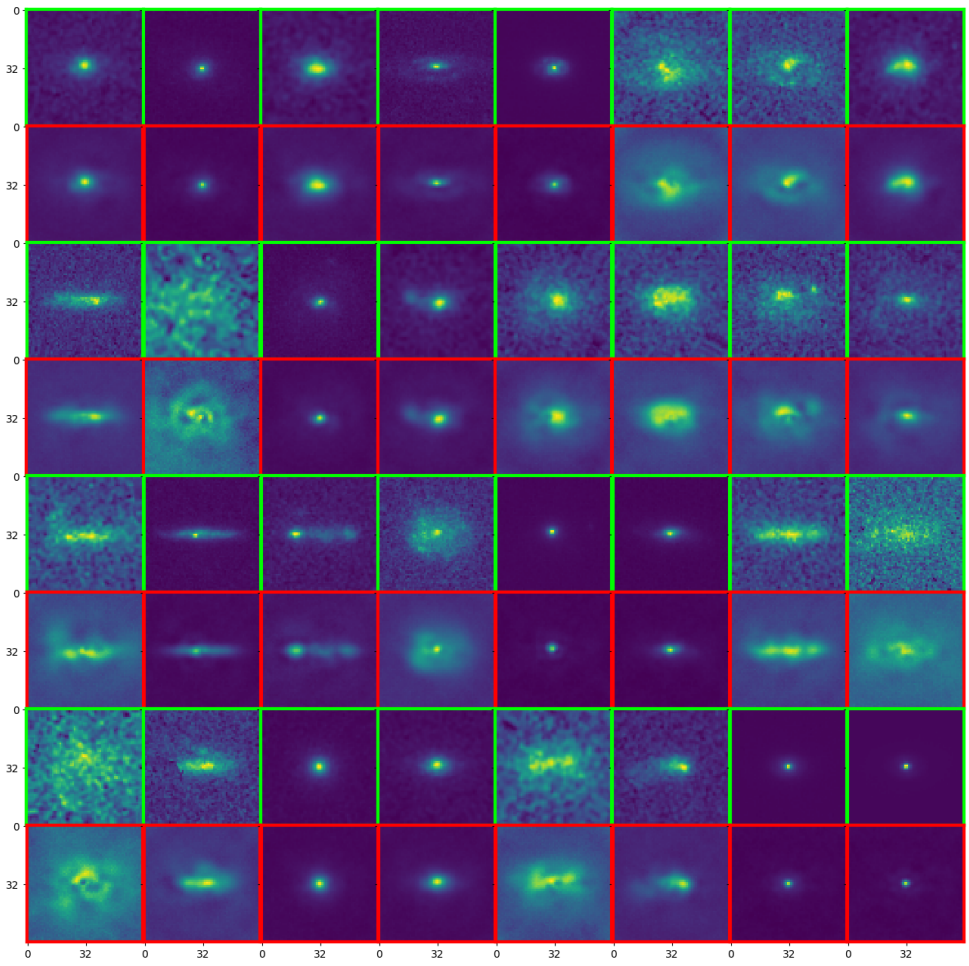


Figure A.1: Reconstruction sheet for the basic FC model (fig. 3.3), showing reconstructions of the first 32 images of the test set. Images are shown in vertical pairs, where the top image (outlined in green) shows the original galaxy image, while the bottom image (outlined in red) shows the corresponding reconstructed image. Image coloring is normalized per image pair, and does not necessarily match across the entire set of images.

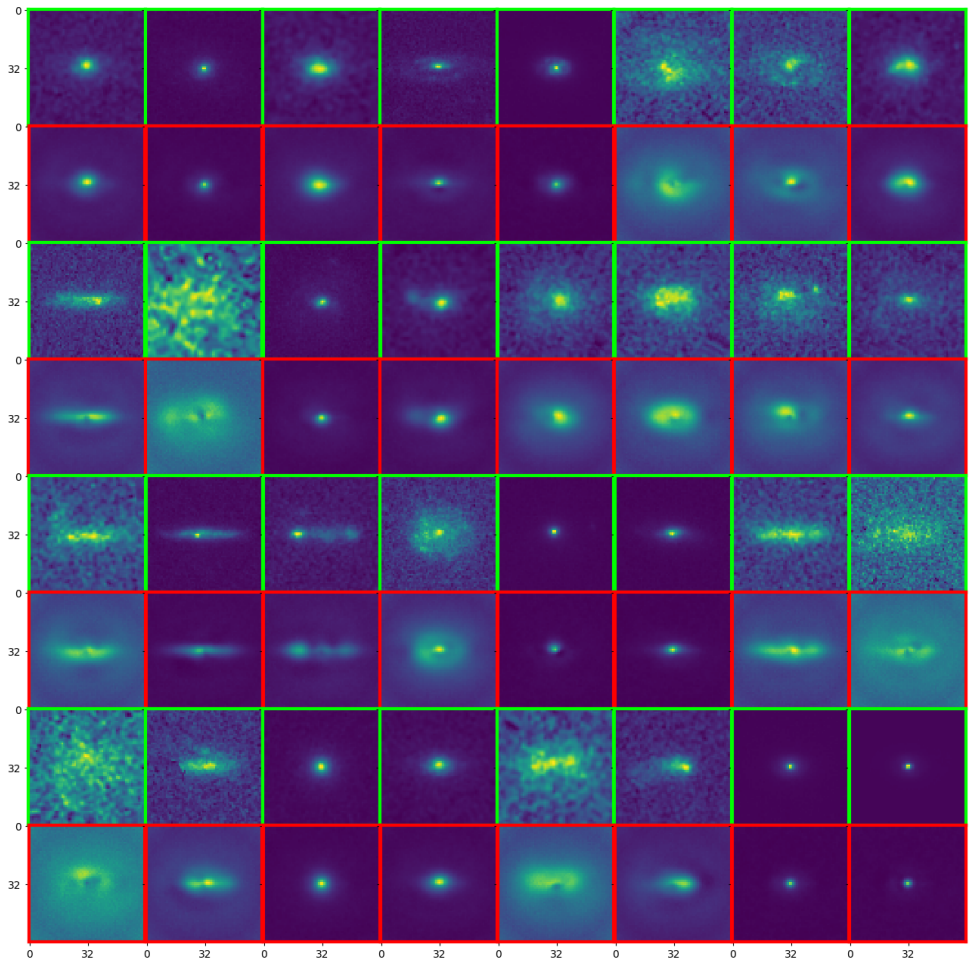


Figure A.2: Reconstruction sheet for the deeper FC model (fig. 3.4), showing reconstructions of the first 32 images of the test set. Images are shown in vertical pairs, where the top image (outlined in green) shows the original galaxy image, while the bottom image (outlined in red) shows the corresponding reconstructed image. Image coloring is normalized per image pair, and does not necessarily match across the entire set of images.

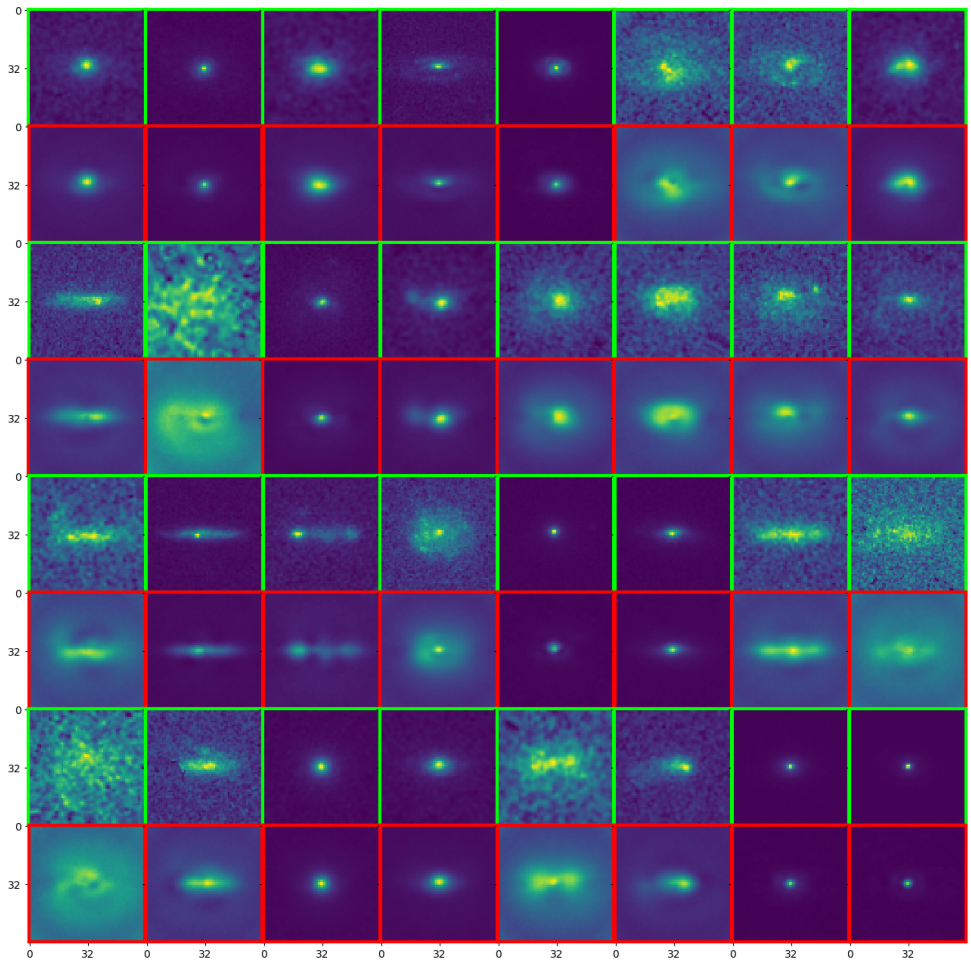


Figure A.3: Reconstruction sheet for the basic ConvNet model (fig. 3.5), showing reconstructions of the first 32 images of the test set. Images are shown in vertical pairs, where the top image (outlined in green) shows the original galaxy image, while the bottom image (outlined in red) shows the corresponding reconstructed image. Image coloring is normalized per image pair, and does not necessarily match across the entire set of images.

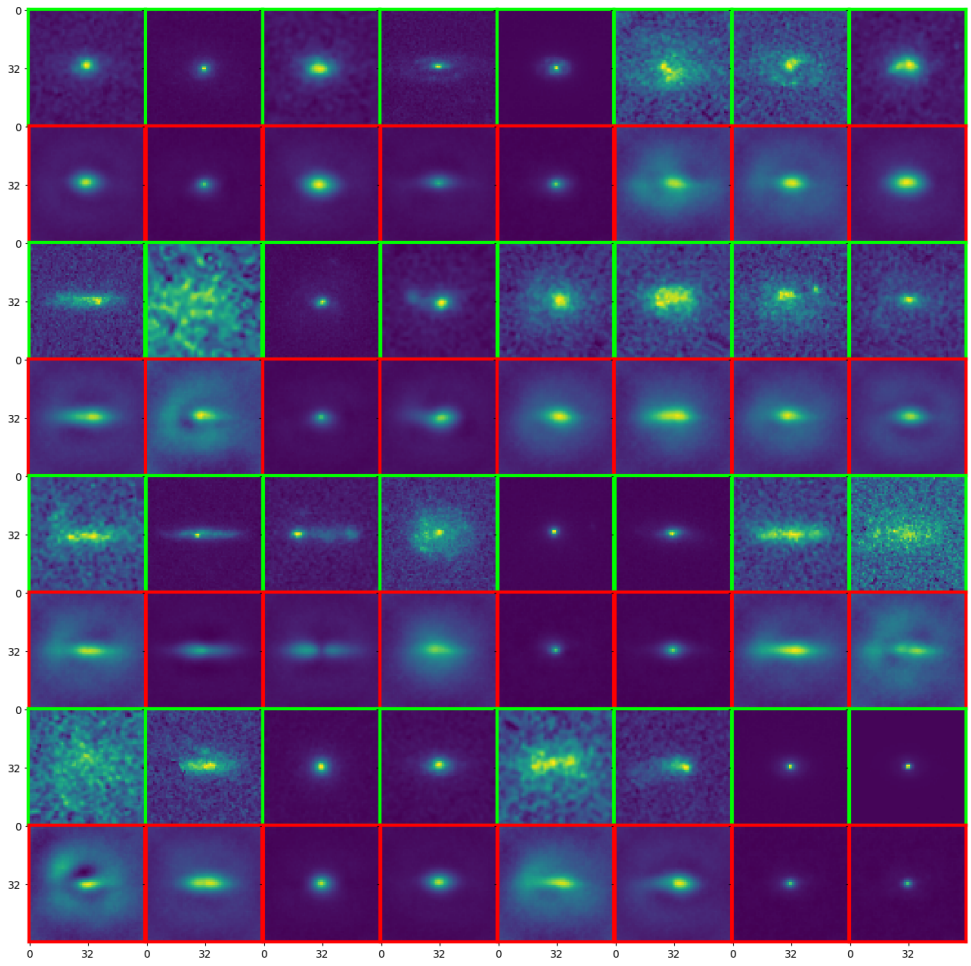


Figure A.4: Reconstruction sheet for the deeper ConvNet model (fig. 3.6), showing reconstructions of the first 32 images of the test set. Images are shown in vertical pairs, where the top image (outlined in green) shows the original galaxy image, while the bottom image (outlined in red) shows the corresponding reconstructed image. Image coloring is normalized per image pair, and does not necessarily match across the entire set of images.

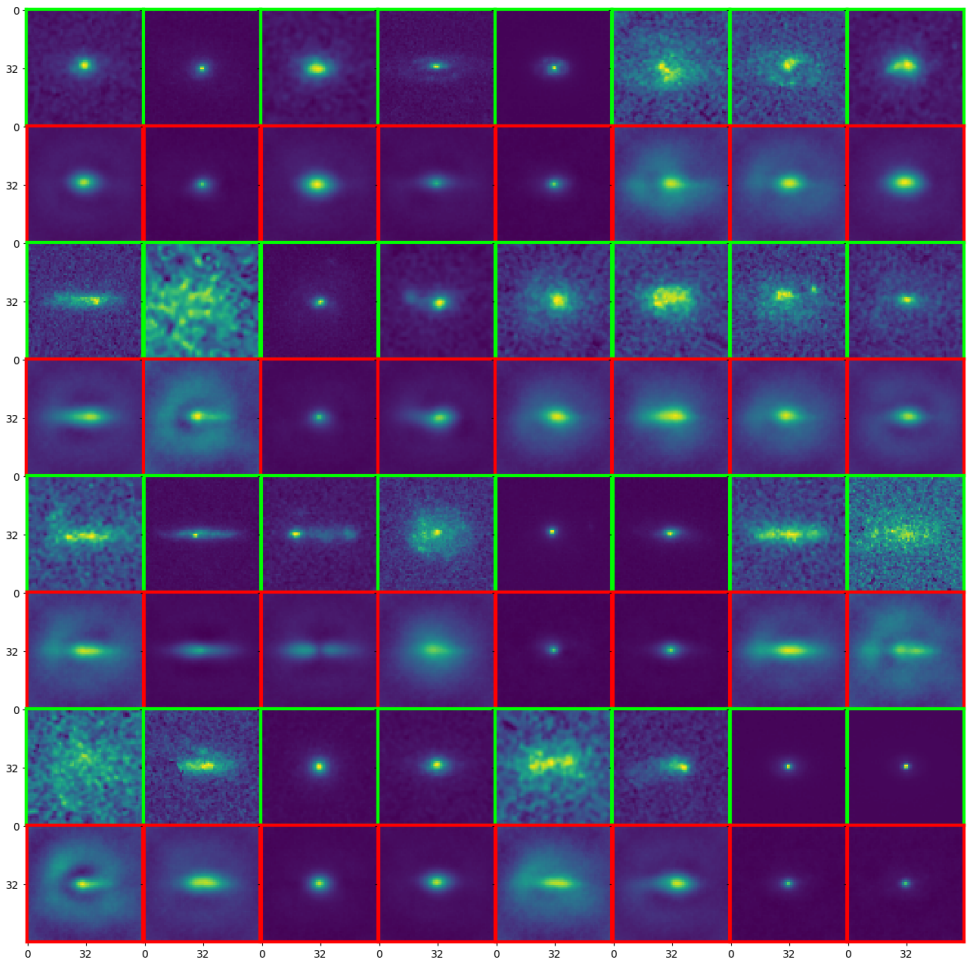


Figure A.5: Reconstruction sheet for the β-VAE variant of the deeper ConvNet model (using $\beta = 2$; for the base model, see fig. 3.6), showing reconstructions of the first 32 images of the test set. Images are shown in vertical pairs, where the top image (outlined in green) shows the original galaxy image, while the bottom image (outlined in red) shows the corresponding reconstructed image. Image coloring is normalized per image pair, and does not necessarily match across the entire set of images.

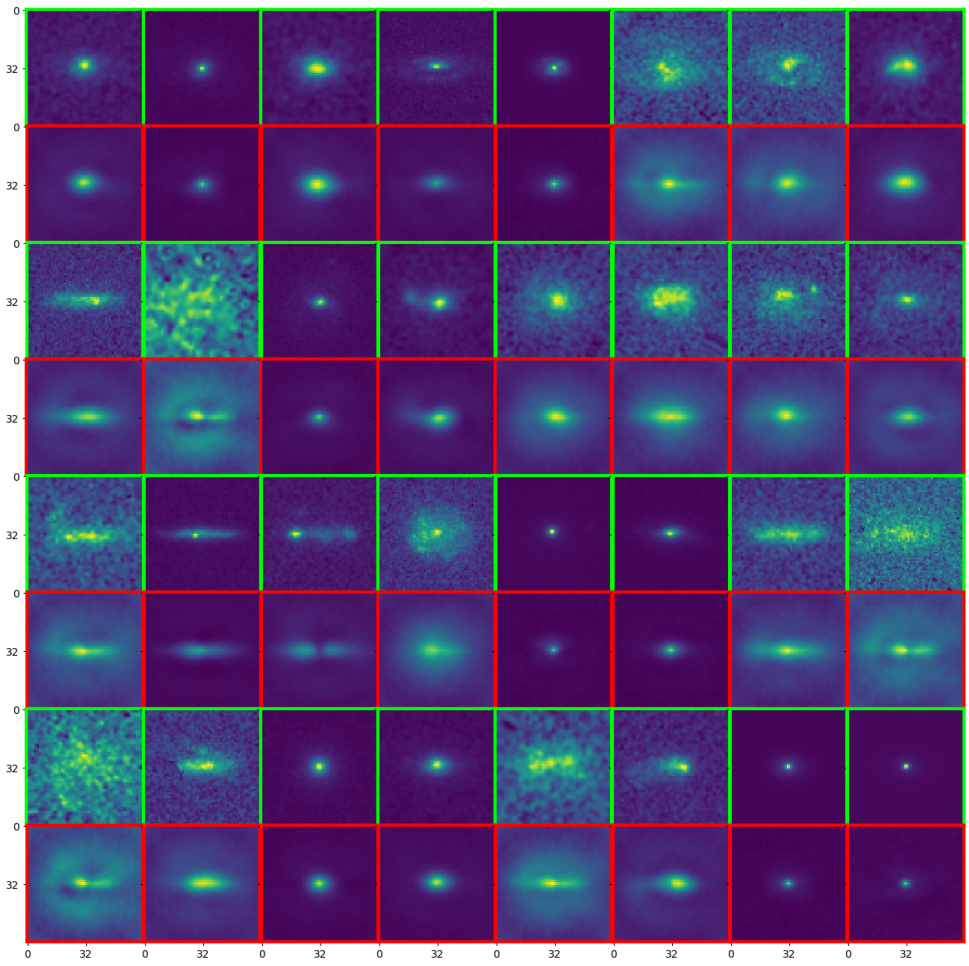


Figure A.6: Reconstruction sheet for the β-VAE variant of the deeper ConvNet model (using $\beta = 5$; for the base model, see fig. 3.6), showing reconstructions of the first 32 images of the test set. Images are shown in vertical pairs, where the top image (outlined in green) shows the original galaxy image, while the bottom image (outlined in red) shows the corresponding reconstructed image. Image coloring is normalized per image pair, and does not necessarily match across the entire set of images.

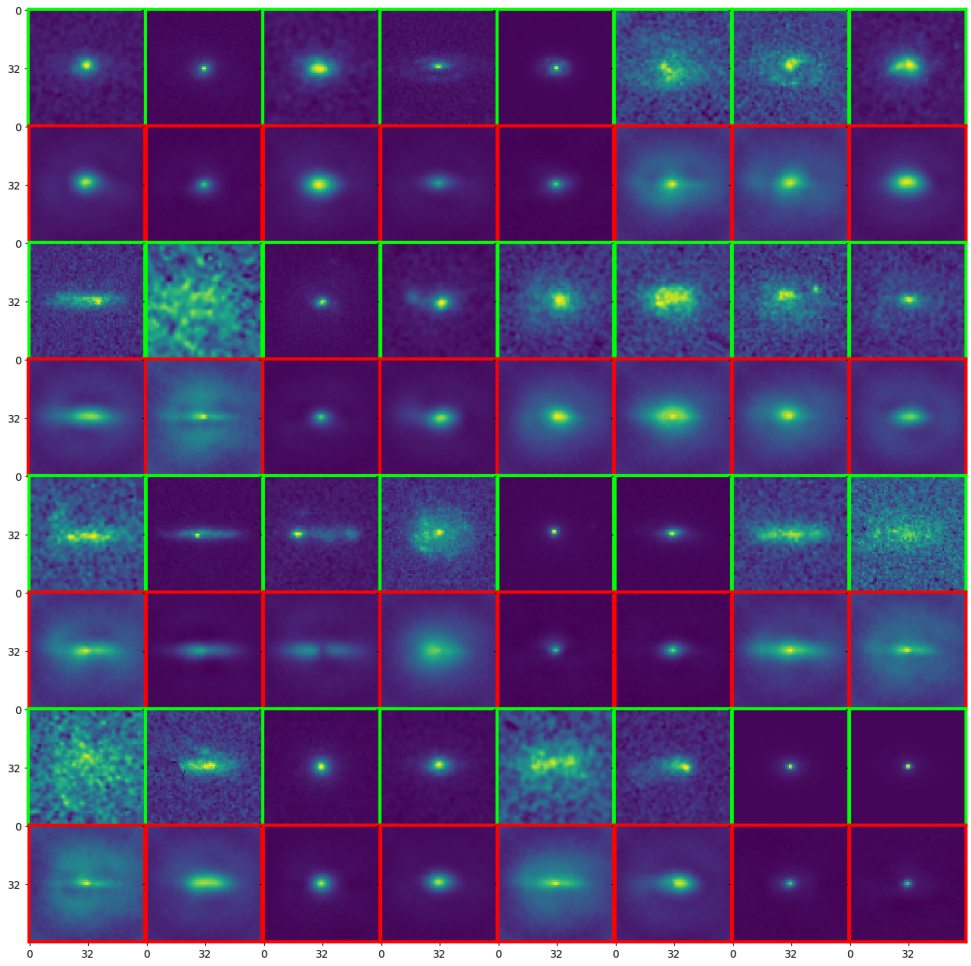


Figure A.7: Reconstruction sheet for the β-VAE variant of the deeper ConvNet model (using $\beta = 10$; for the base model, see fig. 3.6), showing reconstructions of the first 32 images of the test set. Images are shown in vertical pairs, where the top image (outlined in green) shows the original galaxy image, while the bottom image (outlined in red) shows the corresponding reconstructed image. Image coloring is normalized per image pair, and does not necessarily match across the entire set of images.

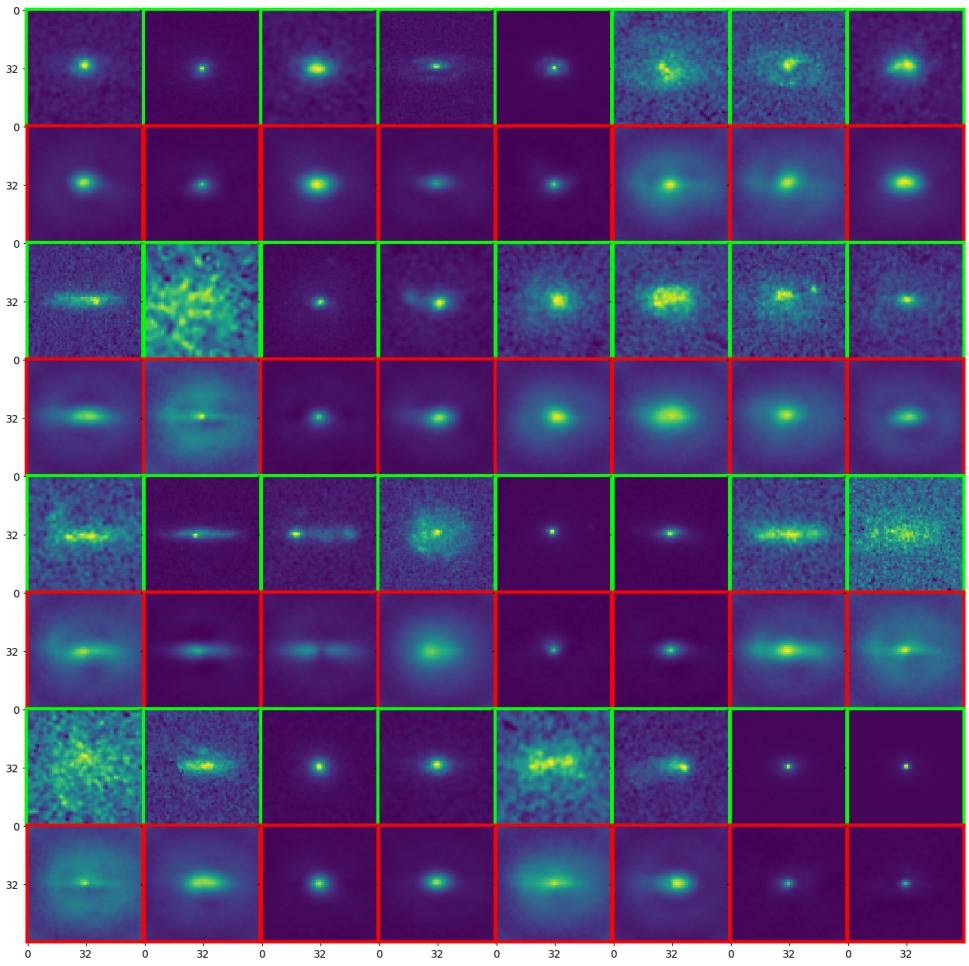


Figure A.8: Reconstruction sheet for the β-VAE variant of the deeper ConvNet model (using $\beta = 20$; for the base model, see fig. 3.6), showing reconstructions of the first 32 images of the test set. Images are shown in vertical pairs, where the top image (outlined in green) shows the original galaxy image, while the bottom image (outlined in red) shows the corresponding reconstructed image. Image coloring is normalized per image pair, and does not necessarily match across the entire set of images.

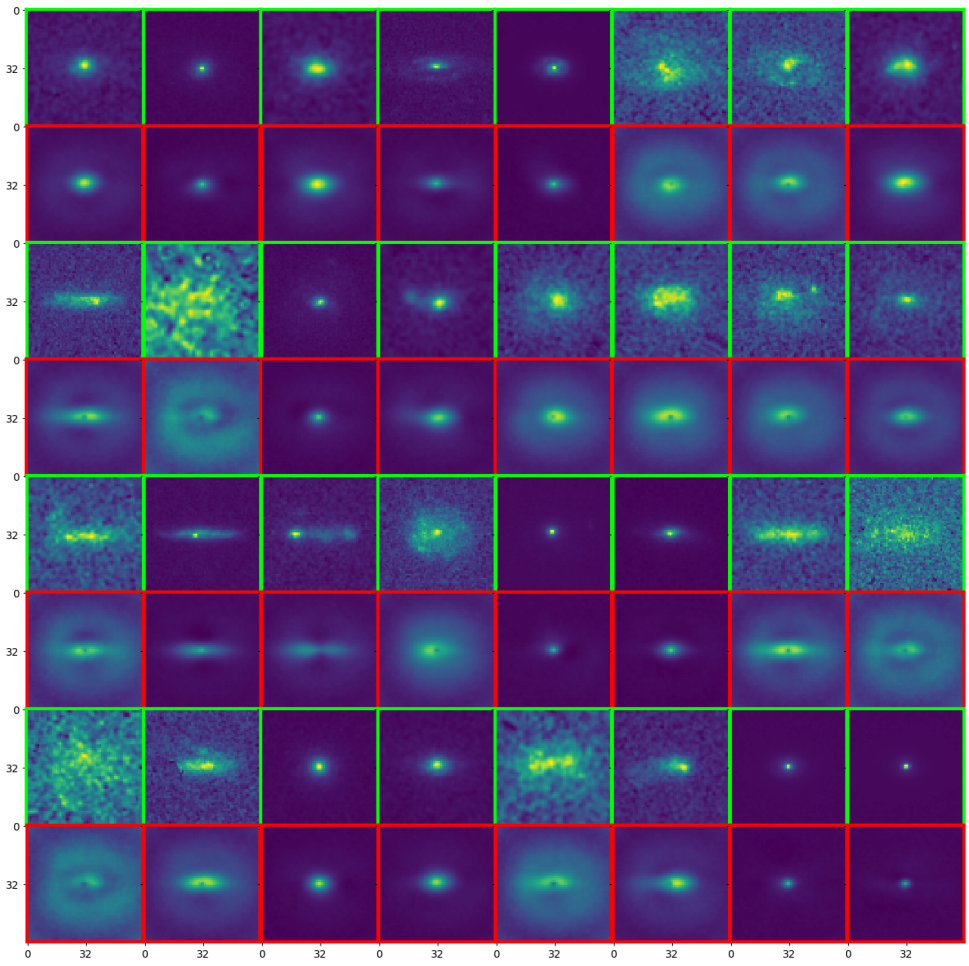


Figure A.9: Reconstruction sheet for the β-VAE variant of the deeper ConvNet model (using $\beta = 50$; for the base model, see fig. 3.6), showing reconstructions of the first 32 images of the test set. Images are shown in vertical pairs, where the top image (outlined in green) shows the original galaxy image, while the bottom image (outlined in red) shows the corresponding reconstructed image. Image coloring is normalized per image pair, and does not necessarily match across the entire set of images.

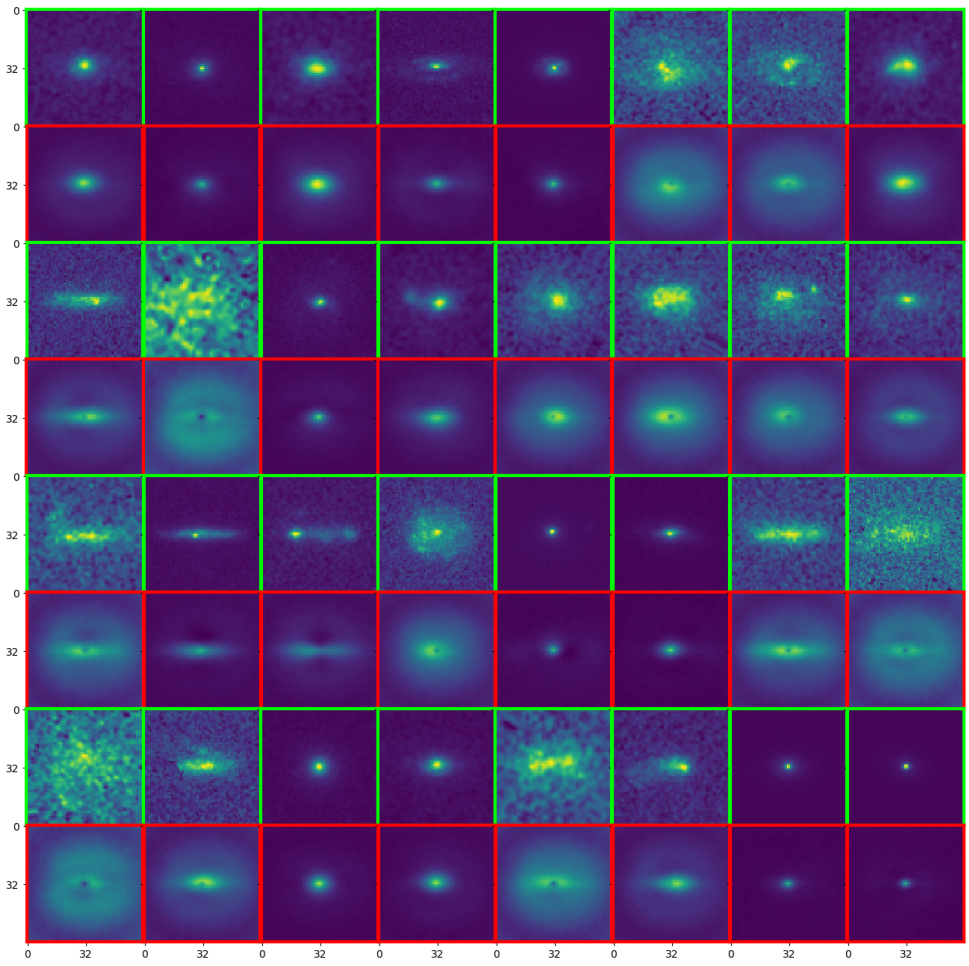


Figure A.10: Reconstruction sheet for the β-VAE variant of the deeper ConvNet model (using $\beta = 100$; for the base model, see fig. 3.6), showing reconstructions of the first 32 images of the test set. Images are shown in vertical pairs, where the top image (outlined in green) shows the original galaxy image, while the bottom image (outlined in red) shows the corresponding reconstructed image. Image coloring is normalized per image pair, and does not necessarily match across the entire set of images.

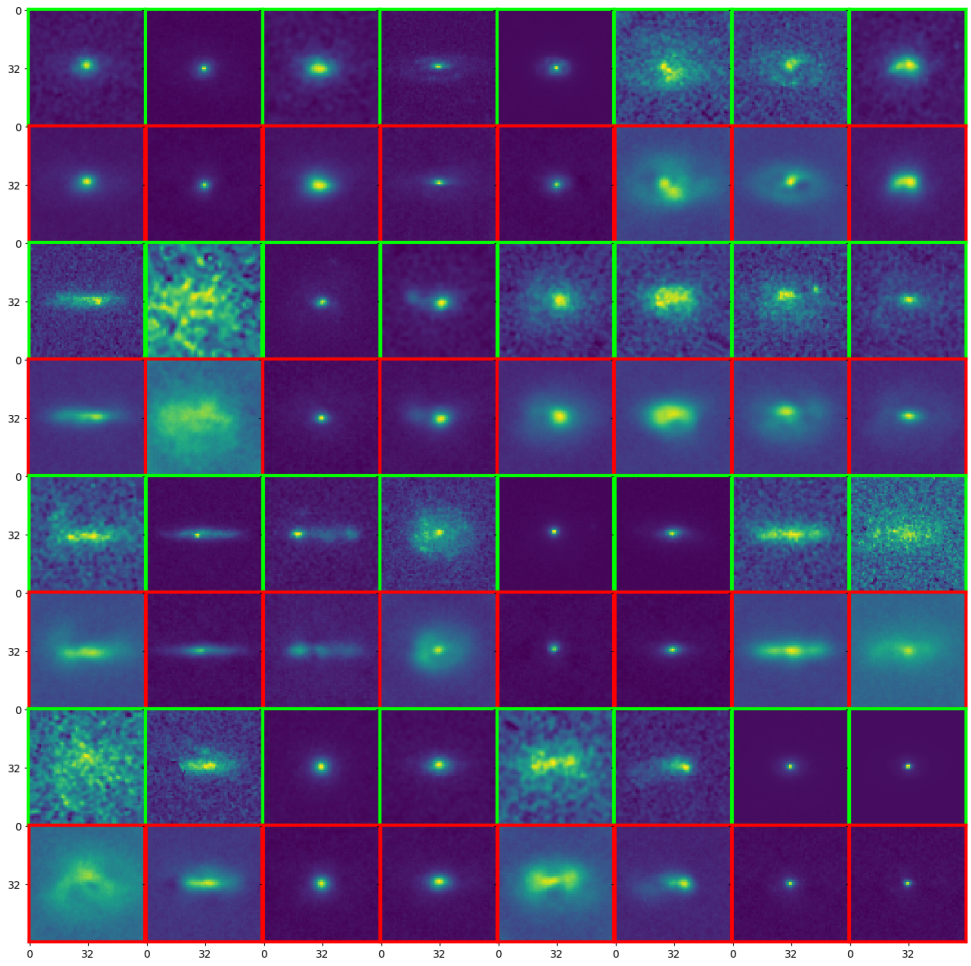


Figure A.11: Reconstruction sheet for the conditional variant of the deeper ConvNet model (for the base model, see fig. 3.6), showing reconstructions of the first 32 images of the test set. Images are shown in vertical pairs, where the top image (outlined in green) shows the original galaxy image, while the bottom image (outlined in red) shows the corresponding reconstructed image. Image coloring is normalized per image pair, and does not necessarily match across the entire set of images.

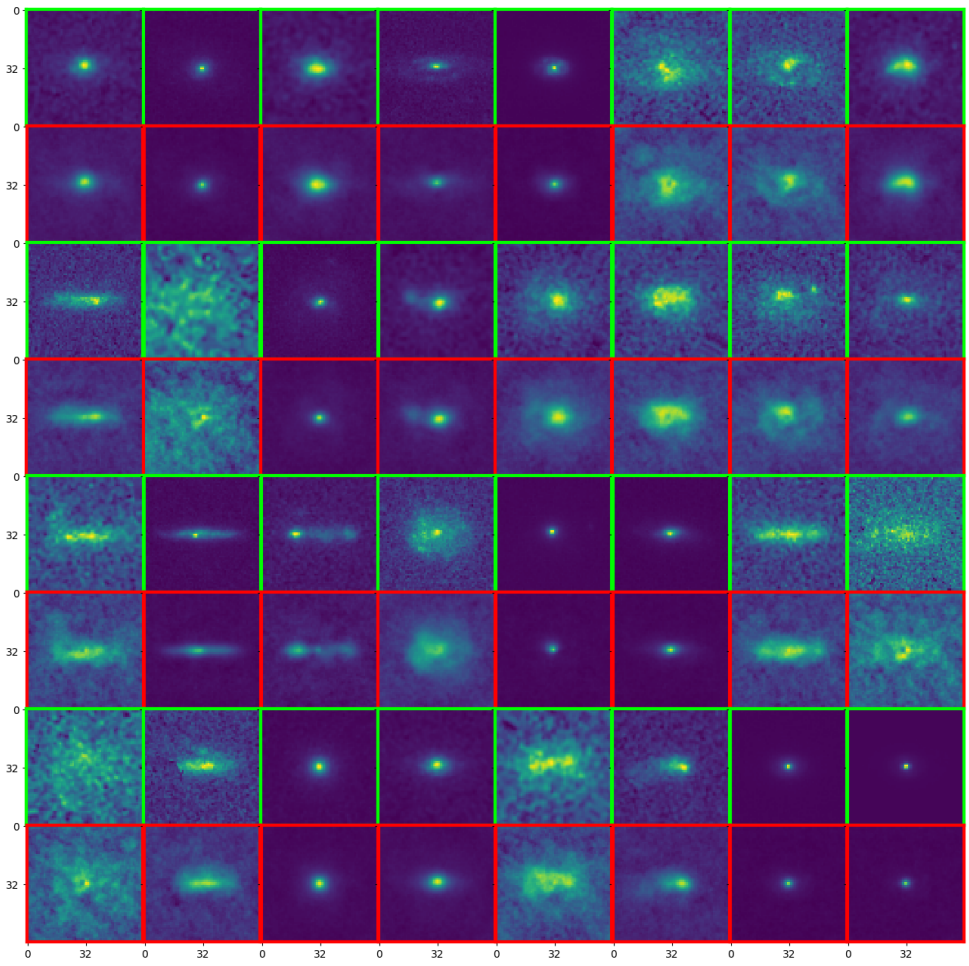


Figure A.12: Reconstruction sheet for the DenseVAE, showing reconstructions of the first 32 images of the test set. Images are shown in vertical pairs, where the top image (outlined in green) shows the original galaxy image, while the bottom image (outlined in red) shows the corresponding reconstructed image. Image coloring is normalized per image pair, and does not necessarily match across the entire set of images.

Appendix B

Further details of G3C dataset

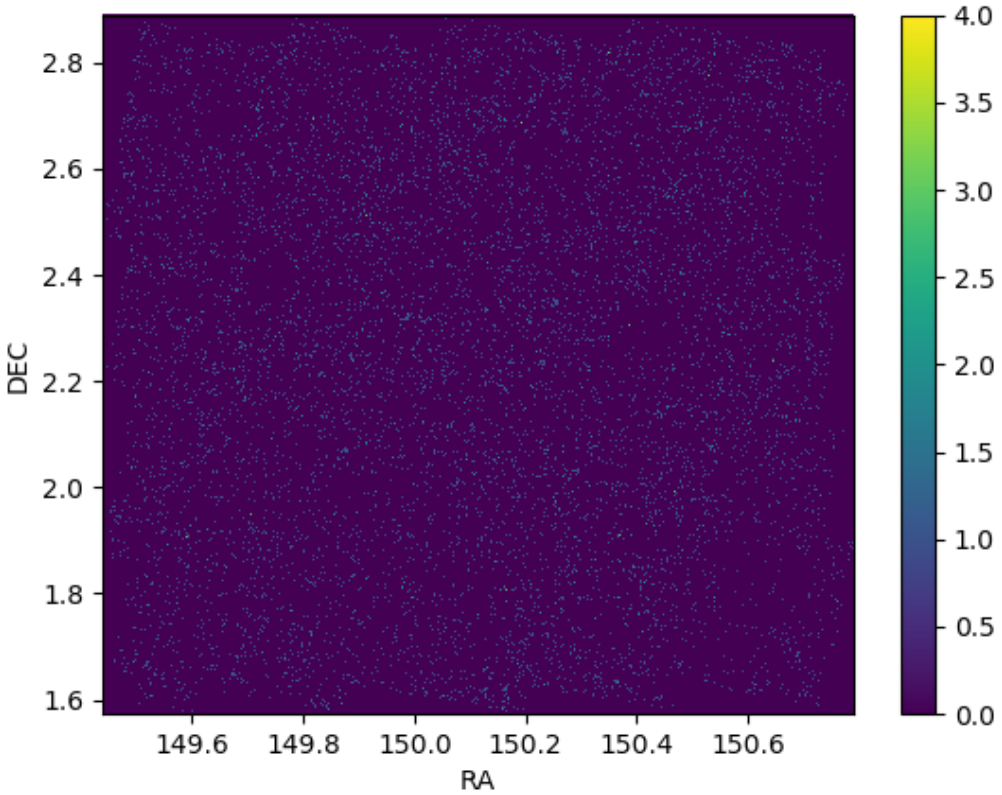


Figure B.1: Positioning of the galaxies included in the dataset, within COSMOS' 2deg^2 area.

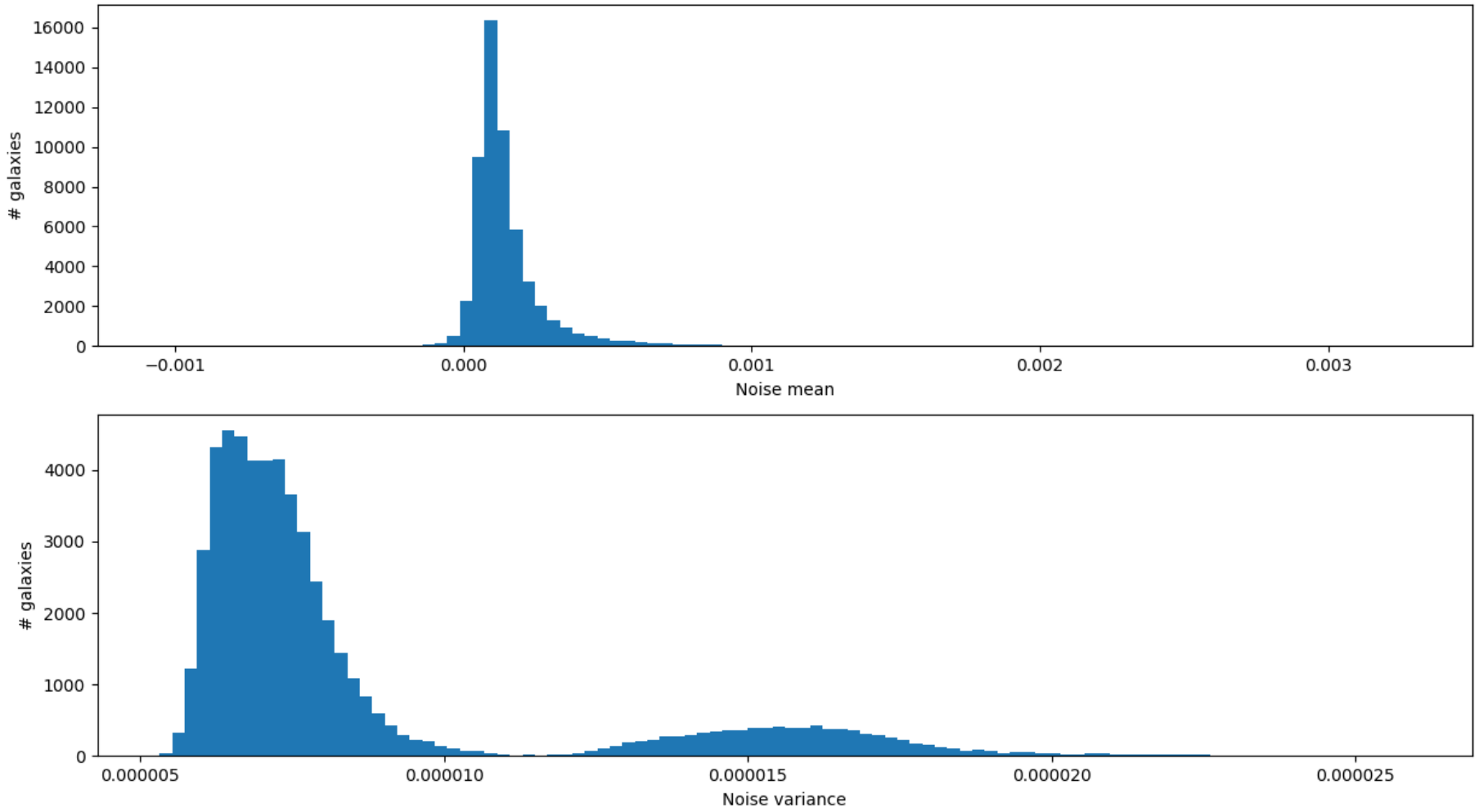


Figure B.2: Histograms of the noise means (top) and variances (bottom), as provided for the COSMOS data set. Note that noise means are typically of order 10^{-4}, with variances of typical order $6 \cdot 10^{-6}$. Note also that both distributions have very long (and especially in the case of the noise mean: thin) tails, with a prominent bulge in the tail for the variance at $1.5 \cdot 10^{-5}$

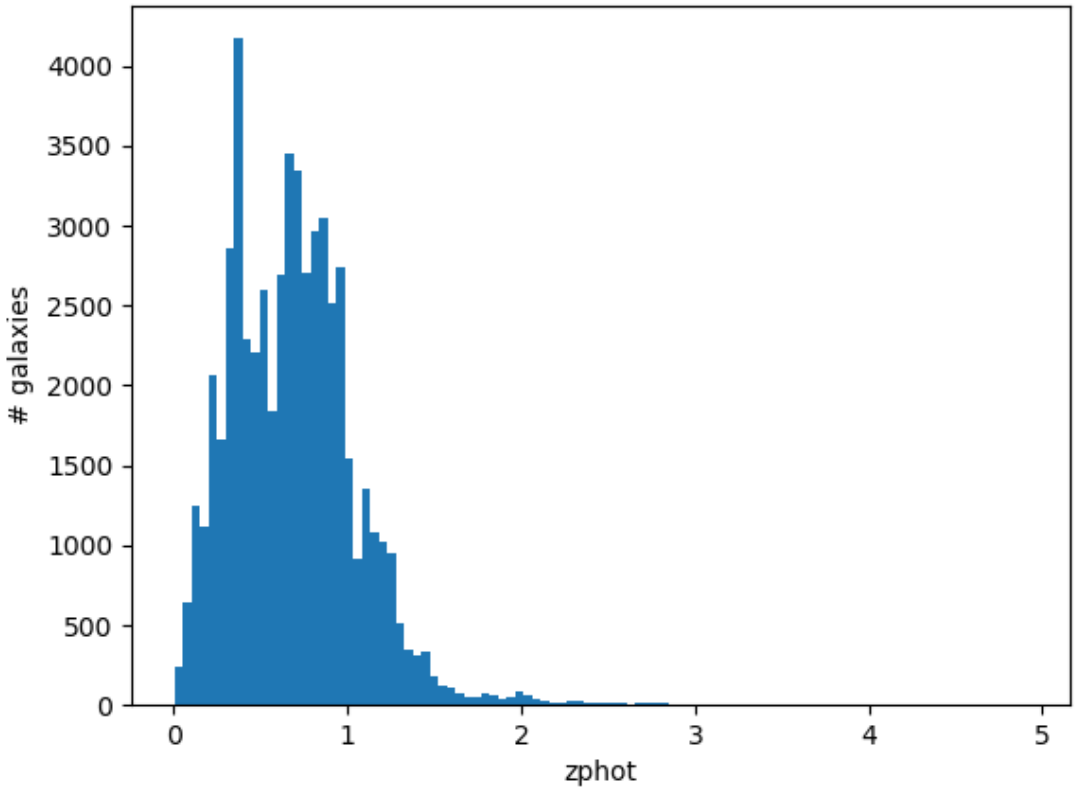


Figure B.3: Histogram of photometric redshift (Z_{phot}) distribution; measurements were originally done by Ilbert et al. [**?**]

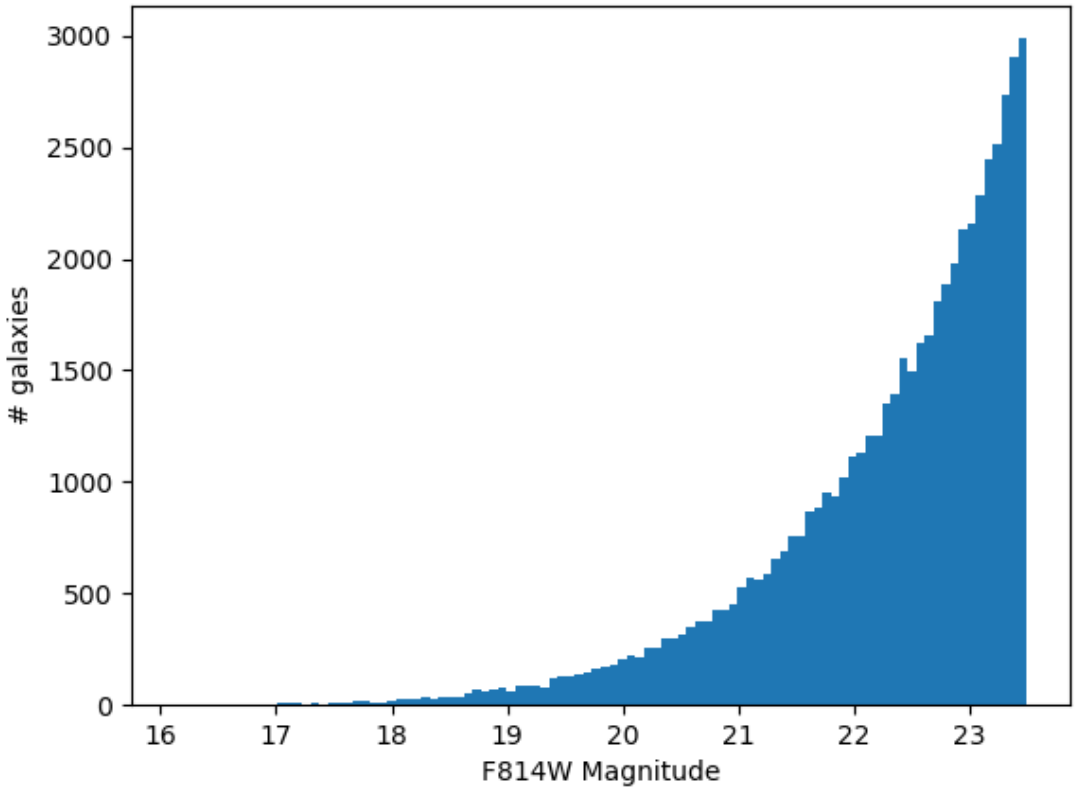


Figure B.4: Magnitudes of images, measured through the F814W bandpass filter.