# Mapping Galaxy Images Across Ultraviolet, Visible and Infrared Bands Using Generative Deep Learning

by

© **Youssef Zaazou**

A thesis submitted to the School of Graduate Studies in partial fulfillment of the requirements for the degree of Master of Science.

Scientific Computing

Memorial University

August 2024

St. John's, Newfoundland and Labrador, Canada

# Abstract

We demonstrate that generative deep learning can be used to map astronomical observations of galaxies across various photometric bands including ultraviolet visible and infrared bands. We also demonstrate that this mapping can be learned without the need to use existing multi wavelength observations which are sparse and nontrivial to obtain and preprocess. We do so by using mock observations from the Illustris cosmological simulations to train our models. To date, this is a novel use of Illustris' mock observations. We demonstrate that models trained on mock observations can generalize and extend the learned mapping to real observations using general image comparison metrics such as the structural similarity index (SSIM) and more specific astronomical metrics such as the GINI coefficient and the M20 measurement. This could allow astronomers to significantly augment existing observations, specifically in areas where multiwavelength observations do not exist, with minimal inference cost.

If you're reading this and you struggle to find peace and contentment within yourself, there's not much I can say beyond...

you are not alone.

# Acknowledgements

I want to thank my supervisors, professors Alex Bihlo and Terrence Tricco for their support, insight, and all-around good-natured guidance without which this work would not have been possible. Also, I would like to thank the Center for Analytics and Informatics Research (CAIR) for their resources and support.

# Table of contents

# List of tables

# List of figures

# List of abbreviations

**Neural Network Components**

| | |
|---|---|
| Ck | Convolutional layer with $k$ filters |
| CTk | Transposed convolutional layer with $k$ filters |
| Dk | Dense (fully-connected) layer with $k$ units |
| DBk | Downsampling block with $k$ filters/units |
| UBk | Upsampling block with $k$ filters/units |
| RESk | Residual block with $k$ filters/units |
| GNorm | Group normalization layer |

# Chapter 1

# Introduction

## 1.1 Generative modelling

Studying galaxies is essential for astronomers seeking to gain an understanding of the universe's history and future. By looking at how galaxies form, change, and interact, we can learn about the origins of the universe and its expansion. This knowledge helps us piece together a timeline of the universe and predict what might happen billions of years from now.

By analyzing the light from galaxies, astronomers can gather information about the celestial bodies that inhabit them including what they are made of, their temperatures, and how they move, see [1]. A diverse collection of celestial objects and events, such as stars, black holes, supernovae, and planetary systems are located in galaxies. Studying each of these components offers unique opportunities to learn about different aspects of the universe.

For instance, stars in various stages of their life cycles, from birth to their final days as white dwarfs or supernovae, provide insights into stellar evolution and the processes that power them. Supermassive black holes, often located at the centers of galaxies, are another key area of study. These extreme objects allow scientists to explore the limits of physical laws. Observing how matter behaves near black holes helps us understand conditions that cannot be replicated on Earth. Supernovae, the explosive deaths of massive stars, are also critical for our understanding of the universe. By observing supernovae, astronomers can learn about the life cycles of

stars and the role these explosions play in shaping galaxies. Additionally, supernovae can serve as cosmic distance markers, helping us measure the expansion rate of the universe and refine our models of its overall structure and evolution. Through the study of these and other phenomena within galaxies, we gain a deeper understanding of the intricate and dynamic processes that govern the cosmos. For a deeper dive on the above, see [2].

Traditional methods of data analysis in astronomy, particularly for studying galaxies, have been vastly outpaced by the enormous volume of data generated by modern telescopes and surveys, see [3]. These traditional approaches often involved painstaking manual inspection and interpretation, making them time-consuming and less efficient. An example of this would be the Galaxy Zoo project, see [4], where crowdsourcing was used to assign morphological labels to images of galaxies. However, deep learning algorithms have revolutionized this process by automating many tasks that were once performed manually. For example, deep learning has proven instrumental in classifying galaxies by identifying their various types and structures as in [5]. This automation has significantly accelerated the analysis process and increased reliability, allowing astronomers to handle larger datasets and uncover new insights more effectively.

Initially, discriminative learning was favored over generative learning because it was more straightforward and computationally feasible given the technology and resources available. Discriminative models, which focus on classifying data and predicting labels by learning the boundaries between different classes, were simpler to implement and required less computational power compared to generative models. The complexity and computational demands of generative models, which aim to model the underlying distribution of data to generate new samples, were beyond the reach of earlier computing capabilities. For a detailed survey into deep learning algorithms and techniques, see [6].

Recently however, generative methods have become popular in astronomy due to significant advancements in computational power and the development of sophisticated algorithms. These methods can create realistic synthetic data, enhance image resolution, and fill in gaps where observational data is incomplete. This capability is invaluable for astronomers, as it allows for more comprehensive and detailed analysis

of celestial objects and phenomena. As a result, the integration of generative methods into astronomical research has opened new avenues for exploration. Examples of generative modelling applications in astronomy can be found in [7], [8] and [9].

The aim of this thesis is to further explore and expand the use of generative modelling in astronomy. This work is split into two distinct segments. The first segment, Chapter 2 , involves unconditioned image generation where we investigate a number of popular generative learning algorithms for the task of creating realistic color images of galaxies. This chapter begins with an introduction to generative modelling followed by an outline of the dataset used to train our models. We proceed to discuss, for each individual model, the theory, implementation details and results.

The second segment, Chapter 3, deals with conditioned generation, specifically image-to-image translation. This chapter begins with an introduction to image-to-image tasks followed by an introducion to the CycleGAN architecture which is capable of unpaired image-to-image translation. We demonstrate CylceGAN's performance by training it to map images of galaxies between different morphologies (such as round or cigar-shaped).

The rest of Chapter 3 deals with the task of translating observations of galaxies taken in a specific wavelength to corresponding observations in different wavelengths. Astronomers require observations taken in different wavelengths since each section of the electromagnetic spectrum offers a unique and complementary perspective on a given galaxy. A successful mapping of observations across different wavelengths can be used to significantly augment existing observations with minimal computational cost.

To date, image-to-image models have been used to translate images across different telescopes but not to different wavelengths. We believe this is the first work that attempts such a translation. Furthermore, we learn this desired mapping using training data collected from the Illustris cosmological simulations which is, to date, a novel use of the Illustris's mock observation catalogue. We demonstrate that image-to-image models are capable of learning these mappings and that future efforts to train such models is highly likely to yield very promising results.

Finally, Chapter 4 contains a discussion of the limitations of our wavelength mapping models as well as possible future work to further our current attempts.

# Chapter 2

# Unconditioned Generative Approaches

This chapter is concerned with the problem of unconditioned image generation. These models are unconditioned since there is no direct way of influencing the output of the model.

## 2.1 Background

### 2.1.1 Generative Modelling

Two main pillars make up the current landscape of machine learning: discriminative and generative learning [6]. Another machine learning paradigm is reinforcement learning which is unrelated to the tasks we attempt in this work.

Discriminative learning, or modeling, aims to predict or output a certain value given specific data. For example, given the weather forecast for the last 10 days, a well-trained discriminative model can output an accurate prediction of tomorrow's weather as in [10] (an example of regression). Another example would be a network that, given an image of an animal, produces a probability of the image being of a certain species (an example of classification as in [11]). Generative modeling, however, is concerned with the problem of generating high-quality original data. Such a model is trained by learning to copy the underlying data distribution of a dataset. For

example, say we want to generate images of cats, we would collect a dataset that is representative of what cats tend to look like and then feed this dataset to a generative model which would learn all the salient features of what makes a cat look like a cat. Once the network has done so, we simply ask it to draw an original cat (original here refers to a sample not observed in the training dataset). If the network has learned the underlying distribution of the dataset, then it should be able to generate an original image of a believable cat.

Until recently, most developments in deep learning have been in the discriminative domain, this is because for a given discriminative problem the corresponding generative problem is much more difficult. Consider for example the task of classifying images of cats or dogs compared to the task of creating original and believable images of cats and dogs. The difficulty arises in part due to the vast number of ways in which the search space, comprised of pixels in this example, can be arranged and the relatively tiny number of such arrangements that would constitute an image in the subspace, pixel arrangements resembling cats and dogs, we are attempting to learn and sample from. For more details, see [12].

Additionally, discriminative modeling is usually supervised while most generative models fall under the unsupervised learning paradigm [6]. Supervised learning involves training a model on labeled data, where each input data point is paired with a corresponding output label. The model learns to map inputs to outputs by minimizing the difference between its predictions and the true labels. This paradigm is akin to a teacher guiding a student by providing correct answers during the learning process. One of the primary advantages of supervised learning is its ability to achieve low errors, given a sufficient quantity of labeled data. Additionally, it allows for explicit control over what the model should learn, making it suitable for tasks where high quantitative accuracy is desired.

Unsupervised learning, on the other hand, deals with unlabeled data, where the goal is to extract meaningful patterns or structures without explicit guidance. It is akin to exploring a dataset without predefined labels or categories, allowing the model to uncover complex hidden patterns autonomously. It is particularly useful in scenarios where labeled data is scarce or expensive to acquire. The ability to discover complex underlying structures in data without the need for labeled examples makes unsupervised learning the preferred learning paradigm for generative modeling. For

a deeper dive into unsupervised learning, see [13].

Both learning paradigms come with their own set of restrictions and limitations. Supervised learning relies on the availability of labeled data, which can be expensive and time-consuming to obtain, especially for complex tasks. Moreover, the model's performance may degrade significantly when faced with inputs that deviate from the distribution of the training data. Also, a phenomenon known as overfitting, see [14], can plague discriminative models. Overfitting occurs when a machine learning model learns to capture noise or random fluctuations in the training data, rather than generalizing well to unseen data. This phenomenon often arises when a model becomes overly complex relative to the amount of training data available, effectively memorizing the training examples instead of learning underlying patterns. An example can be seen in Figure (2.1). If overfitting is present, the resulting model would perform well on the training data but would fail to generalize to new, unseen data.

While unsupervised learning is free of such restrictions, evaluating the performance of unsupervised learning algorithms can be challenging since there are no explicit ground truth labels to compare against. For this reason, generative models are usually assessed on qualitative grounds rather than quantitative ones.

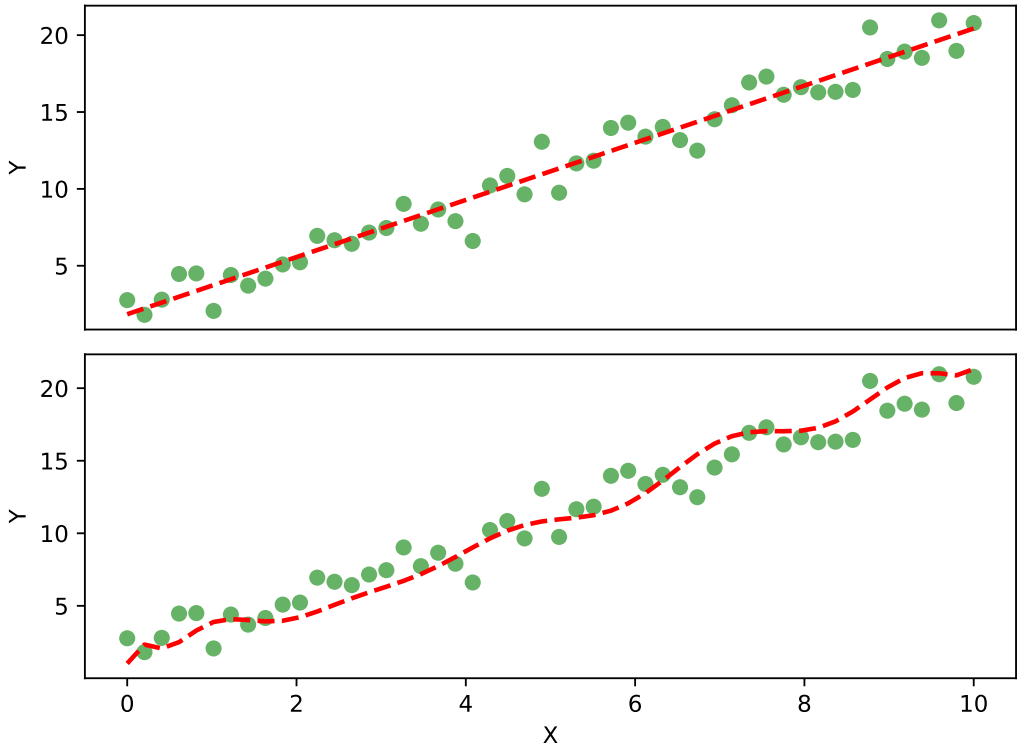



Figure 2.1: A demonstration of overfitting for the task of performing polynomial regression on the plotted data (green). In the top section, we model (red) the data with a first-order polynomial which succeeds in capturing the upwards linear trend in the data. The bottom section of the plot contains the same data but was fit with a polynomial of order 16 which contains too many variables relative to the complexity of the data. The resulting model is too sensitive to the noise in the data and is not well suited to generate any predictions.

### 2.1.2 Approaches

We will look at three distinct generative learning models, each with it's own advantages and disadvantages. The goal of each of the different models presented is to create original images of galaxies that are indistinguishable from real images.

We will investigate the different models and compare their outputs on qualitative grounds and will demonstrate that generating a high quality set of synthetic Galaxy images is possible but that results differ substantially with the type of architecture selected. First however, we must select a suitable training dataset.

## 2.2 Training Dataset

Selecting a suitable training dataset is a matter of vital importance. The criteria we are looking for are fourfold. Firstly, the dataset should be large enough to capture the underlying distribution of the data adequately. A larger dataset allows the generative model to learn more diverse and representative patterns, leading to better sample generation. Secondly, the selected sample must be representative of the total observed population of galaxies. This requires the sampled galaxies to represent all diverse morphological classifications, [15], such that our model is not biased in producing galaxies of a particular morphology. Thirdly, the images must be of sufficient quality to allow our model to capture high-frequency features. If the resolution of the images is insufficient, then the model may learn the overall shapes and orientations of the sampled galaxies but may fail to resolve features such as spirals and bars which would make generating spiral galaxies unfeasible. Finally, it is also preferable that the images are obtained from a single source rather than multiple sources to ensure consistency across the samples in the data set.

There are numerous astronomical surveys that we may resort to obtain such data. Astronomical surveys are systematic observations of the sky conducted to study various celestial objects and phenomena across different wavelengths of light, from radio waves to gamma rays. Examples include the Two Micron All-Sky Survey (2MASS) [16], the Sloan Digital Sky Survey (SDSS) [17], the Dark Energy Survey (DES) [18], The Galaxy Evolution Explorer (GALEX) [19], and the The Panoramic Survey Telescope and Rapid Response System (Pan-STARRS) [20]. These surveys play a crucial role in advancing our understanding of the universe by providing vast amounts of data that astronomers analyze to uncover new insights about cosmic phenomena. Astronomical surveys can vary in scope and objectives, ranging from comprehensive sky surveys that map the entire observable universe to targeted surveys focused on specific types of objects or phenomena, such as galaxies, stars, exoplanets, or transient events like supernovae or gamma-ray bursts. Of the many types of surveys, sky surveys are the most suitable type of astronomical survey for finding isolated images of galaxies since they are designed to cover large areas of the sky and create comprehensive maps of celestial objects, including galaxies. SDSS, DES, 2MASS and Pan-STARRS are examples of sky surveys with SDSS, DES and Pan-STARRS collecting data in

photometric bands centered on visible light and 2MASS collecting data in the infrared range. GALAX on the other hand is a more specialized survey that targeted galaxies and other astronomical objects that emit strongly in the ultraviolet. Most astronomical surveys have made their data publicly accessible however, navigating the databases of surveys that cover vast swaths of the sky to find isolated, well-resolved images of galaxies is no trivial task. We have found SDSS data to be the most accessible through its image cutout tool and for this reason have ellected to construct all training datasets in this chapter from SDSS.

SDSS, [17], is one of the most significant and comprehensive sky surveys ever conducted and has covered nearly a third of the sky, mostly in the northern hemisphere, over the last 20 years or so. To collect a sample of galaxies from SDSS that meets our criteria, we will resort to Galaxy Zoo 2 (GZ2), [4]. GZ2 is a citizen science project with morphological classifications of 304,122 galaxies drawn from SDSS. The primary goal of GZ2 is to classify galaxies based on their morphological features, such as their shapes, structures, and other visual characteristics. Participants in the GZ2 survey were presented with a hierarchical sorting questionnaire, see Figure (2.2), which started with the most obvious features, like the overall shape and smoothness, and proceeded through to more specific questions, such as irregular features or the number of spirals, with the final galaxy classification reached at the end of the questionnaire. Due to this hierarchical design, initial questions received a greater number of voters which makes their answers have a higher level of confidence compared to classifications made further down the questionnaire.

We selected a sample of galaxies from GZ2 to form our dataset. Our sample, initially unbalanced, was stratified by morphology such that there was roughly an equal number of spiral and elliptical galaxies. We also selected galaxies based on their field of view, since we wanted galaxies that would take up most of the area within the image rather than ones that would constitute a minority of the total image area. This is due to high-frequency features such as spirals being better resolved in galaxies that have a higher field of view. Also, images of galaxies with a small field of view were subject to background influences that are considered a hindrance to the generative model during training. This selection was done by sorting the galaxies by their Petrosian 90% angular radius (PetroR90) which measures the angular radius which encompasses an annulus within which 90% of local surface brightness is found. The higher the PetroR90 measurement, the larger the galaxy's field of view. Note

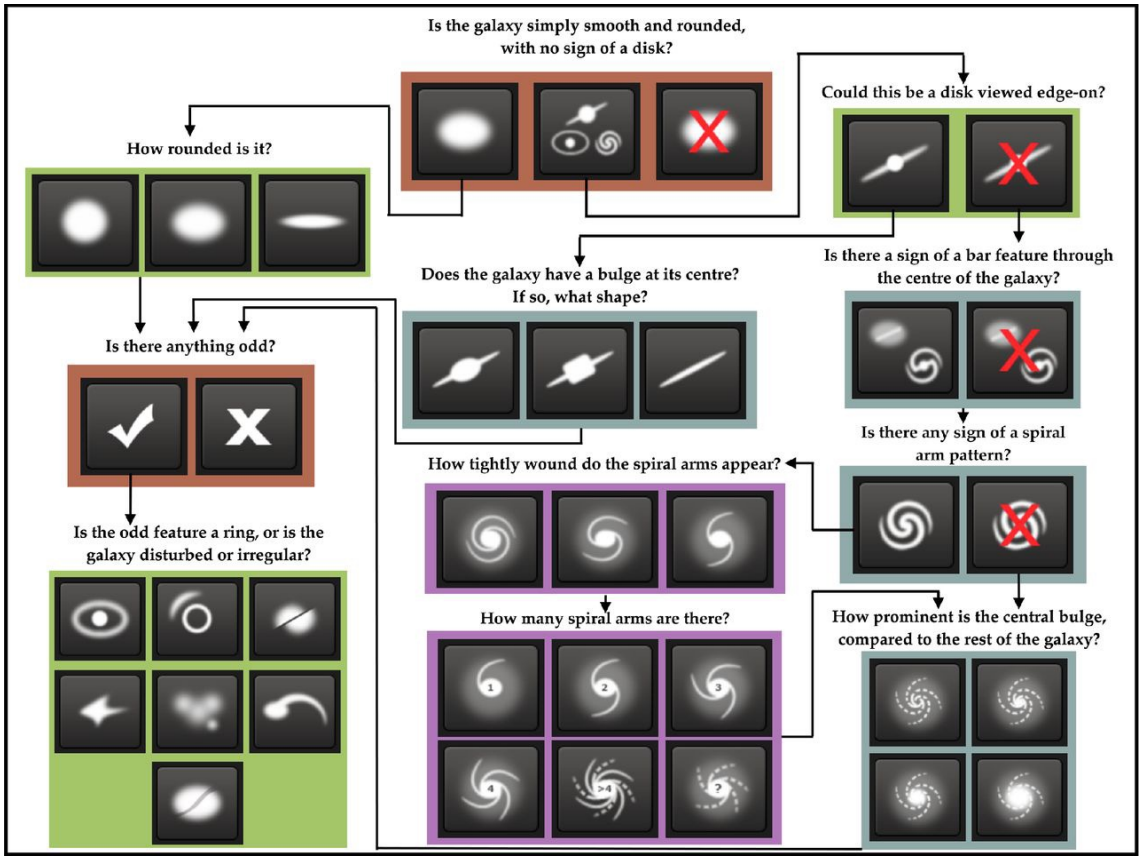



Figure 2.2: The GZ2 hierarchical questionnaire. For a detailed explanation of this figure see [4]

that the PetroR90 measurement was obtained from the metadata included in the GZ2 data.

The galaxies' coordinates were obtained from GZ2, and a Python script was run to download the data from the SDSS image cutout service. Example galaxies can be seen in Figure (2.3).

## 2.3 Variational Autoencoder

The first unconditioned generative archicture we will investigate is the variational autoencoder (VAE). First we will begin with a quick revision of the traditional autoencoder framework.

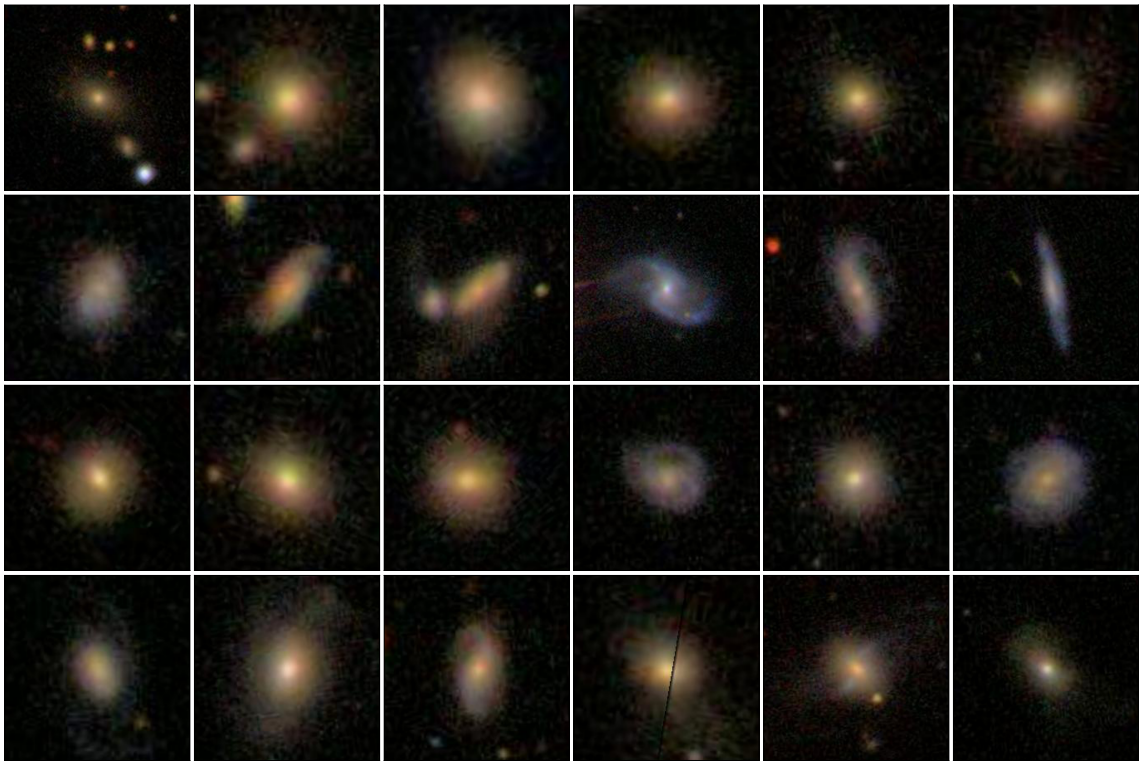


Figure 2.3: Random sample of images taken from our downloaded dataset. The images are not free of noise and artifacts but will serve as our training data. The galaxies in our dataset were filtered by selecting the ones with the highest field of view.

### 2.3.1 Autoencoder

An autoencoder, [21], is a neural network that accepts input data, in our case images, compresses it to a latent representation, and then attempts to reconstruct the original image from this latent representation. The autoencoder (AE), shown in Figure (2.4), can be decomposed into two distinct networks, an encoder and a decoder, that are joined at a bottleneck layer which contains the latent variables $z$.

It is the encoder's job to find the best possible way to represent the given input image in terms of the latent variables. Essentially, the encoder applies a compression on the original image to produce a lower dimensional representation. If the encoder is successful, it should be able to detect the most salient features in the input and do away with more superfluous features and noise. The decoder's job is that of attempting to reconstruct the original image from the latent representation. Since the decoder does not have any access to the original image data except through the

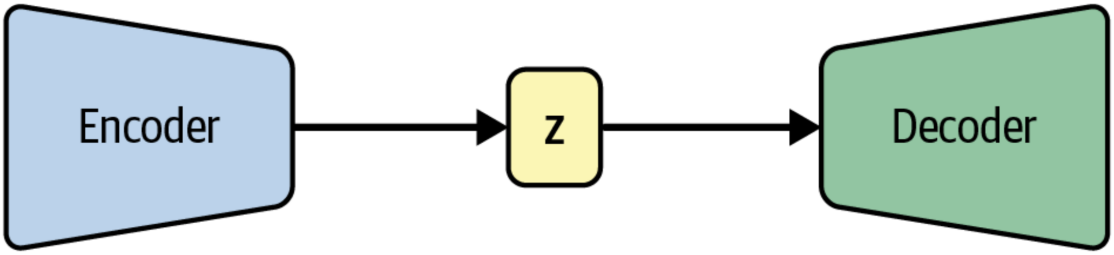



Figure 2.4: The encoder compresses the input to its latent representation, $z$, which can also referred to as the bottleneck. The decoder then attempts to reconstruct the original image using only input from $z$.

latent variables, it is forced to attempt to reconstruct the original image with access to what are only the most important features (according to the encoder of course) in the given input image.

The loss function of the AE is some measure of distance between the input and reconstructed images; examples of such metrics include the mean squared error (MSE), binary cross-entropy, or structural similarity index. This measure of distance between the input image and the reconstructed image is referred to as the reconstruction loss. The reconstruction loss for a single input using the MSE as a metric is given by:

$$(\mathbf{x} - f_\theta(g_\phi(\mathbf{x})))^2, \tag{2.1}$$

where $\mathbf{x}$ is the input, $f$ is the decoder parameterized by $\theta$ and $g$ is the encoder parameterized by $\phi$.

For a batch of inputs the reconstruction loss becomes

$$\mathcal{L}_{AE} = \frac{1}{n}\sum_{i=1}^{n}(\mathbf{x}_{(i)} - f_\theta(g_\phi(\mathbf{x}_{(i)})))^2, \tag{2.2}$$

where $n$ is the batch size.

During training, the AE is penalized for outputs that differ significantly from the input image and is encouraged to output an image that is as similar as possible to the original input image, such that

$$\mathbf{x} \approx f_\theta(g_\phi(\mathbf{x})). \tag{2.3}$$

Applications for this architecture include noise reduction as in [22], and anomaly detection as in [23]. Noise reduction is a task well suited for an AE since a well-trained encoder will not transmit the random noise to the latent variables; this produces relatively noise-free images that retain the structure of the input images. Anomaly detection is a suitable task for an AE since training is conducted on images drawn from a dataset that has a specific underlying distribution. If the trained AE is presented with an image that lies outside the underlying distribution of the original training set, then the reconstruction loss, Equation (2.2), will be significantly higher since the AE was not trained to reconstruct images that lie outside the distribution of the training set.

In theory, we now have a model that can generate original images (original in the sense that the output images differ from the input images) once it is trained. All we would have to do to get original images is discard the encoder after training, sample a random point from the latent space, and feed it to the decoder. If we sample points of the latent space that the decoder has not seen during training, then the decoder should output a novel image. However, there are a few issues with this architecture, such as it is, that renders it rather unhelpful as a generative model. It is these inadequacies that the VAE attempts to fix which we will discuss in Section 2.3.2.

### 2.3.2 Latent Space Regularity

Any dimensionality reduction algorithm, such as an AE, that perfectly reconstructs its input tends to come with the price of lack of a regularity. Lack of regularity means that the latent space does not have exploitable and interpretable structures which we can take advantage of in the data generation process. Regularity of the latent space can be summarized by two properties. The first is completeness where the latent space is structured in a meaningful way that allows us to manipulate the output image by modifying the latent variables. Figure (2.5) shows an example of a an incomplete latent space while (2.6) shows an example of a a complete latent space. In those figures are depicted the output of two encoders for the task of compressing images of cylinders into a 2-dimensional representation. The optimal encoder may utilize the fact that a cylinder is defined by its height and width to assign one latent dimension to the height of the cylinder and the second latent dimension to the width of the cylinder. Such an encoder has produced a complete latent space. This allows

us to interpret the latent variables and sample specific cylinders with a specific width and height by manipulating the latent variables.

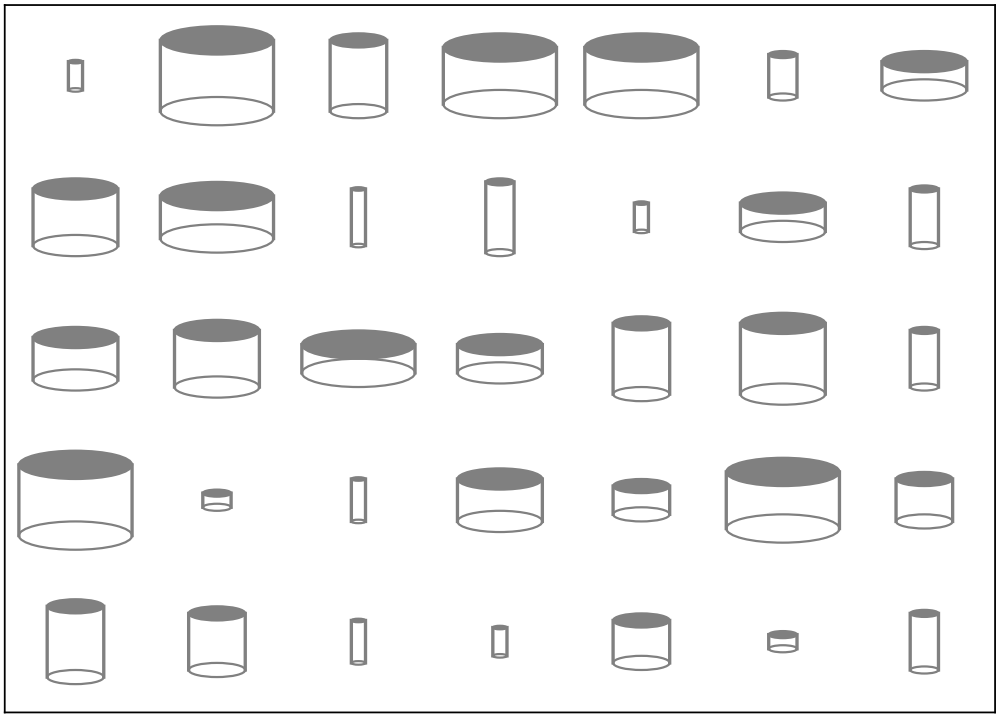


Figure 2.5: An example of an incomplete two-dimensional latent space. Note that there exists no interpretable structure in the space since the latent variables lack semantic meaning.

The second property of a regular latent space is continuity where points close to each other in the latent space should give similar outputs when decoded. A continuous latent space has a smooth gradient over the information defined within it. A discontinuous latent space is challenging to sample from for several reasons. For instance, there may be undefined regions in the latent space that the AE is not trained to map to and from which means that only a subspace of the latent space is defined during training. So, the decoder may yield unsatisfactory results if it is given latent data points that lie outside the well-defined subspace. Also, it is not guaranteed that a random point sampled from the latent space, even if it lies within the well-defined subspace in the latent space, will yield a decent image. For example, the AE may learn to map the point $(1.0, 1.0)$ to a decent image but nothing in our training guarantees that the point $(1.01, 1.01)$ maps equally well.

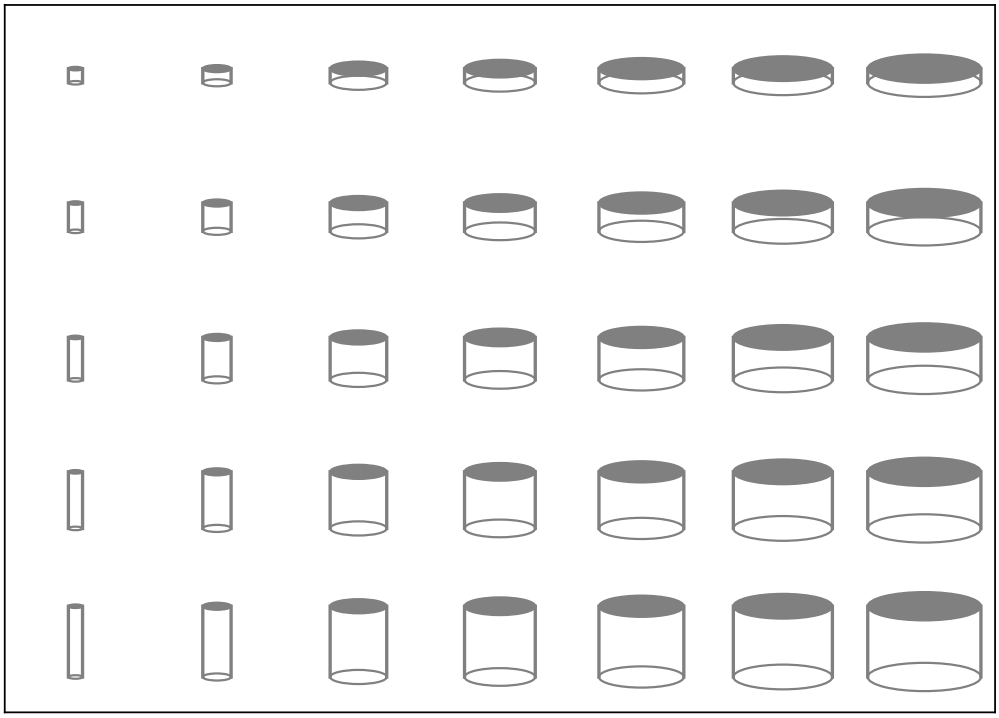


Figure 2.6: An example of a complete two-dimensional latent space. Note the inherent structure and organization of the space. Each latent variable represents a meaningful aspect which can be interpreted and manipulated for generative purposes. The latent variable taking up the $x$-axis represents the cylinder width while the latent variable plotted on the $y$-axis represents the cylinder height.

An irregular latent space that lacks the well suited to the task of novel data generation since sampling from it is fraught with challenges. To enforce regularity, we must modify both the encoding process and the loss function.

### 2.3.3 From AE to VAE

The encoding process in an AE maps a single input image to a discrete point in the latent space. To ensure the latent space is continuous, we would have to map every single point in the latent space to an image which is an infeasible task. The solution that the VAE presents, as can be found in [24], is to have the encoder map from the input space to a distribution over the latent dimensions as can be seen in Figure (2.7). Since we are now sampling not from a point in the latent space, but from a region defined by a distribution, the decoder must ensure that all points sampled from that

region produce very similar images when decoded such that the reconstruction loss remains small. This means that even when the decoder samples a point that it has not seen before, it will produce a meaningful output. The modified encoder now maps its inputs to a multivariate Gaussian distribution which means that it is required to map each input to a mean vector and a variance vector as visualized in Figure (2.8). We assume a covariance of zero between latent variables to ensure minimal entanglement in the latent dimensions which means we do not have to worry about a covariance matrix in our encoder outputs.

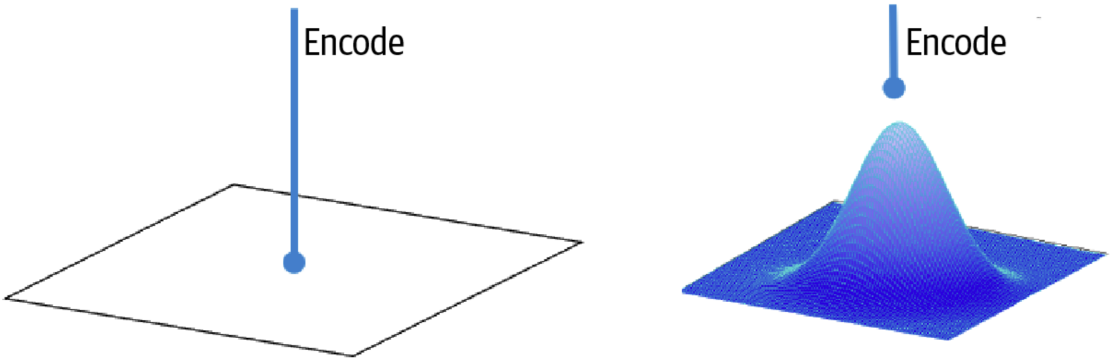



Figure 2.7: On the right, we have the encoding process of an AE and on the left, we have the encoding process of a VAE for a single latent variable. The AE maps each input to a single point in the latent space while the VAE maps each input to a distribution. To clarify, for an 8-dimensional latent space the AE would output an 8-dimensional vector while the VAE would output 8 distributions, one for each latent variable.

However, this modification of the encoding process does not ensure continuity or completeness on its own, see [25]. If left to its own devices, the model can learn to ignore the fact that the encoder returns distributions and behaves much like a traditional AE. It can do this by either making the variances of the returned distributions very small, or by making the means very far apart. The former would make the distributions very narrow, making them behave punctually, while the latter would produce undefined gaps in the latent space. Both issues would produce a latent space that is not regular.

To remedy these potential issues, we must regularize the mean and variance vectors returned by the encoder. This is done by enforcing that the returned distributions be close to unit Gaussians. Thus, we penalize the means from deviating from 0 and the variances from deviating from the identity. The variances being close to identity forces

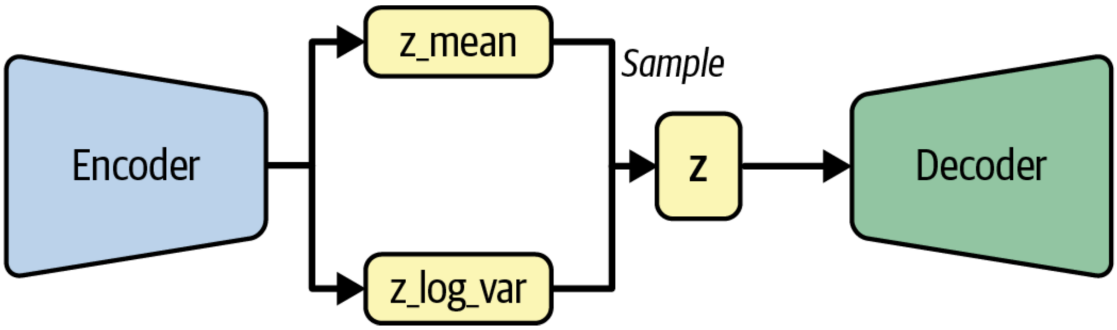



Figure 2.8: Representation of a VAE. The encoder now produces the mean and variance of the Gaussian distribution(s) that makes up the latent space. $Z$ is then sampled from these distributions such that the decoder can function just like the AE decoder.

regularity by ensuring that the input images are mapped to a well-defined region in the latent space rather than a point which ensures continuity. Forcing the means to be close to zero ensures that distributions for different image classes are centered around zero which forces the distributions to cluster together as close as possible which leads to a latent space with no undefined gaps in it and a smooth gradient across distribution borders. This in turn leads to the ability to sample from regions where distributions overlap and get well-formed images that may even combine features from both classes yielding a smooth transition from one class to another.

Thus, in addition to the reconstruction loss that a traditional AE comes with, the authors of the VAE, [24], add a regularization term that ensures the latent space is regular. This regularization term is taken to be the Kullback-Leibler (KL) divergence between the latent space variables and a standard unit Gaussian. The new loss is derived below.

### Derivation of VAE Loss

The derivation of the VAE loss was obtained from [26].

The main goal of a VAE is to maximize the probability of the observed data under the model, $p(\mathbf{x})$. However, directly computing this probability is intractable because it requires integrating over all possible latent variables $\mathbf{z}$. To address this, we introduce a variational distribution $q_\phi(\mathbf{z}|\mathbf{x})$. This is the encoder of the VAE, parametrized by $\phi$, and approximates the true posterior distribution $p(\mathbf{z}|\mathbf{x})$.

Instead of maximizing the log-likelihood of the observed data, $\log p(\mathbf{x})$, directly, we maximize a lower bound on this quantity, known as the Evidence Lower Bound (ELBO). This derivation is done in the equations below.

$$\log p(\mathbf{x}) = \log \int p(\mathbf{x}, \mathbf{z})\, dz = \log \int \frac{p(\mathbf{x}, \mathbf{z})}{q_\phi(\mathbf{z}|\mathbf{x})} q_\phi(\mathbf{z}|\mathbf{x})\, dz \tag{2.4}$$

$$\log \int \frac{p(\mathbf{x}, \mathbf{z})}{q_\phi(\mathbf{z}|\mathbf{x})} q_\phi(\mathbf{z}|\mathbf{x})\, dz = \log \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} \frac{p(\mathbf{x}, \mathbf{z})}{q_\phi(\mathbf{z}|\mathbf{x})} \tag{2.5}$$

$$\log p(\mathbf{x}) \geq \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} \log \frac{p(\mathbf{x}, \mathbf{z})}{q_\phi(\mathbf{z}|\mathbf{x})} \tag{2.6}$$

Where in Equations (2.4) and (2.5), we introduced $q_\phi(\mathbf{z}|\mathbf{x})$ such that we can express the log likelihood in a form that allows us to apply Jensen's inequality. In Equation (2.6), we used Jensen's inequality to obtain a lower bound on the log likelihood. This lower bound, known as the ELBO, can be decomposed into two terms:

$$\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} \log \frac{p(\mathbf{x}, \mathbf{z})}{q_\phi(\mathbf{z}|\mathbf{x})} = \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} \left[\log p_\theta(\mathbf{x}|\mathbf{z})\right] - \mathrm{KL}\left[q_\phi(\mathbf{z}|\mathbf{x}) \| p(\mathbf{z})\right] \tag{2.7}$$

Where $p_\theta(\mathbf{x}|\mathbf{z})$ is the deterministic decoder, parameterized by $\theta$, that attempts to reconstruct the input observation, $\mathbf{x}$, from the latent representation $\mathbf{z}$. For further details on the derivation of Equation (2.7), see page 4 in [26].

The two terms in Equation (2.7) can be interpreted as follows: $\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} \left[\log p_\theta(\mathbf{x}|\mathbf{z})\right]$ is the expected log likelihood of the data which is equivalent to the reconstruction loss of the AE. This term encourages the model to reconstruct the input data $\mathbf{x}$ from the latent variables $\mathbf{z}$. While the second term, $\mathrm{KL}\left[q_\phi(\mathbf{z}|\mathbf{x}) \| p(\mathbf{z})\right]$, is the Kullback-Leibler (KL) divergence between the approximate posterior $q_\phi(\mathbf{z}|\mathbf{x})$ and the prior $p(\mathbf{z})$. This term acts as a regularizer, ensuring that the learned latent space $q_\phi(\mathbf{z}|\mathbf{x})$ remains close to the prior distribution $p(\mathbf{z})$. As we have mentioned previously, $p(\mathbf{z})$ will be taken to be a unit Gaussian. This means that more the latent variables $\mathbf{z}$ diverge from the unit Gaussian, the higher the KL divergence. This term ensures a regular latent space.

By maximizing the ELBO, we simultaneously ensure that the reconstructed data is close to the original data and that the approximate posterior is close to the Gaussian prior. This results in a generative model with a regular latent space that allows for

targeted sampling from the learned latent space. The final ELBO objective function to be maximized during training is:

$$\mathcal{L}_{VAE}(\mathbf{x}) = \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\log p_\theta(\mathbf{x}|\mathbf{z})\right] - \mathrm{KL}\left[q_\phi(\mathbf{z}|\mathbf{x})\|p(\mathbf{z})\right] \tag{2.8}$$

**Reparameterization Trick**

We have delayed the discussion of a core feature of the VAE's modified encoding process, namely the fact that the process is now probabilistic as compared to the deterministic encoder in the AE. As we have discussed, the fact that the encoder returns a distribution of the latent variables is a key feature that distinguishes the VAE from its deterministic predecessor. It does however pose a problem that we will address by going over the single training iteration for the VAE.

First, the encoder maps an input image, $\mathbf{x}$, to a multivariate normal distribution. The decoder accepts as inputs a given latent variable $\mathbf{z}$ which is a random sample from the distribution returned by the encoder $q_\phi(\mathbf{z}|\mathbf{x})$. In this way, we can provide the decoder with a single point from the latent space such that it can do its job just like it did within an AE.

The problem arises due to the stochasticity of the sampling process; because the sampling is stochastic, we cannot perform backpropagation through the latent layer. The authors of the original VAE, [24], came up with a neat reparameterization trick to circumvent this issue. In short, the random latent variable $\mathbf{z}$ is expressed as a deterministic variable by introducing a random auxiliary variable epsilon and rewriting the latent $\mathbf{z}$ as

$$\mathbf{z} = \boldsymbol{\mu} + \boldsymbol{\sigma} \odot \boldsymbol{\epsilon} \tag{2.9}$$

where $\boldsymbol{\epsilon} \sim \mathcal{N}(0, \boldsymbol{I})$ is an auxiliary independent random variable and $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$ are the mean and the variance of the Guassian distribution returned by the encoder

The partial derivative of the layer output with respect to its input is independent of the random $\boldsymbol{\epsilon}$. This provides us with the ability to backpropagate the error and train the model as we would any supervised neural network.

### 2.3.4 Implementation

#### Training Dataset

We selected 22,000 images of the SDSS images mentioned in Section (2.2). The images were scaled to a range of [0,1] and the dataset was divided into batches of size 128 which we found to be ideal as far as performance and computational efficiency is concerned.

#### Architecture and Training

The architecture of the encoder is given by:

C32-C64-C128-C256-C512-D512.

While the decoder's architecture is given by:

D256-CT512-CT256-CT128-CT64-CT32-C3.

Where Ck defines a convolutional layer with k filters, CTk defines a transposed convolutional layer with k filters and Dk defines a dense (fully connected) layer with k units. Please refer to the abbreviations page for a full accounting of architecture layer acronyms. The encoder and decoder are connected by a latent layer with a dimension of 128 corresponding to 128 latent variables. There are plenty of hyperparameters that could be investigated to improve the performance of the model such as batch size, optimizer selection and learning rate. We did not investigate these hyperparameters due to resource constraints but instead opted with the default settings as is found in the original VAE implementation from [24].

All convolutional layers have a stride of 2 effectively downsampling the image by a factor of 2 in the encoder and upsampling by the same amount in the decoder; standard practice for this kind of architecture. The ReLU activation function was used all throughout both networks with the exception being the decoder's output later which uses a sigmoid activation to ensure the outputs are in the [0,1] range.

The network was trained for 30 epochs on the training dataset with the Adam optimizer being used for gradient descent Checkpointing was used to save intermediate

model weights during training to verify the progression of the models' outputs during training. We found that there were little to no improvements to the loss after about 20 epochs of training. We then investigated the outputs from checkpointed models and found no observable differences in the models' outputs after 25 epochs so we concluded that 30 epochs was enough to train the model.

The metric used for the reconstruction loss was binary cross entropy which was calculated per image and averaged across the whole batch.

### 2.3.5 Results

The progression of the losses during training is shown in Figure (2.9). As we can see, the reconstruction loss levels out after only a couple of epochs while the KL loss slowly increases. This trend is understandable as the KL loss is dominated by the reconstruction loss. One thing to keep in mind is that for generative tasks such as this one, tracking the loss can be far less informative than it is with traditional supervised tasks. This is because in discriminative tasks, the loss function is a direct measure of the model's performance and can often be used, alongside other metrics, to measure the accuracy of the model. However, with generative modelling, the loss function acts as a proxy for model performance and is not in of itself a measure of the quality of the model's generative capability.

Before we take a look at the results, we must discuss the sampling process. After the VAE is trained, we discard the encoder which is not helpful for the generative process. If we are working with a labeled dataset, the encoder can be used to study the latent space by learning the regions in the latent space that correspond to particular classes but this is not a task we are interested in right now. We generate a batch of latent vectors by sampling from a Gaussian distribution. These vectors are then passed to the isolated decoder which generates our desired images.

As can be seen in Figure (2.10), the VAE is able to produce a variety of galaxy-like objects with variations in color, orientation, roundness, and dispersion. The images lack the detail present in the original dataset. This is probably due to the bottlenecking behavior of the latent layer which makes it difficult for high-frequency features, such as spirals, to be transmitted through the network. On the flip side, the images are very crisp with a noticeable reduction in noise compared to the training

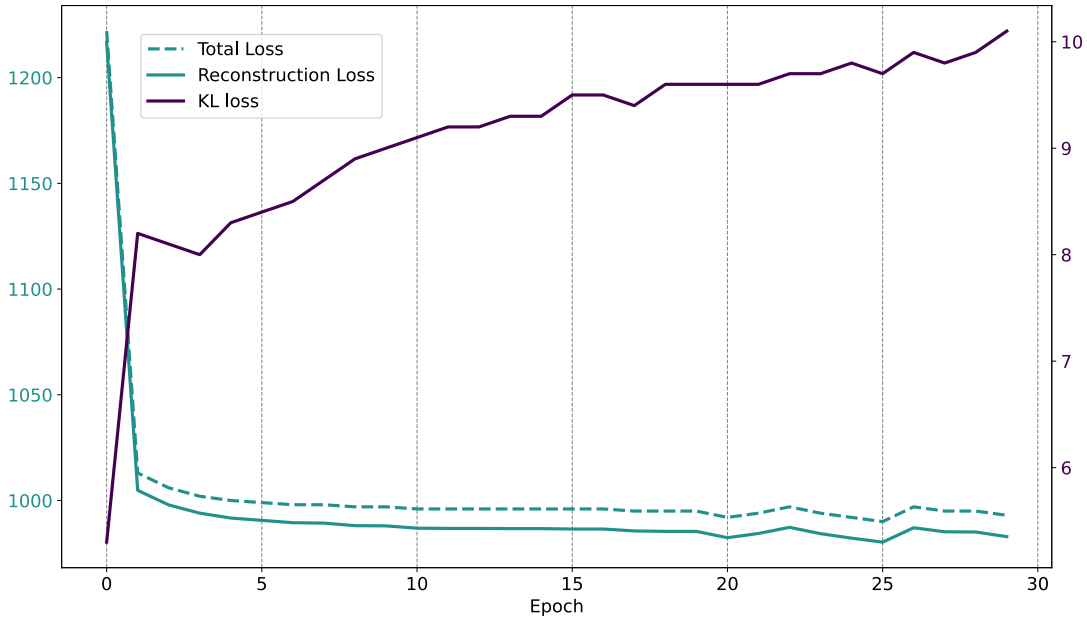



Figure 2.9: Training losses progression. The reconstruction loss dominates the KL loss. The right side of the $y-$axis is for the KL loss while the left side is for the reconstruction loss and total loss.

data; after all, like AEs, VAEs do compress the original image and are excellent denoisers due to the bottlenecking of the latent layer.

Finally, Figure (2.11) shows that the desirable property of regularity is present in the VAE's latent space; note how there is a smooth progression of Galaxy orientation as well as overall size and dispersion along the space. We cannot locate a one-to-one correspondence between a given latent variable and an aspect of the outputs. This is probably due to the high dimensionality of the latent space; perhaps benefits could be seen from increasing the weight of the KL loss to enforce regularity more strictly.

Figure 2.10: Sampled Galaxies from the trained VAE decoder; note the variety in orientation, shape and color. A lack of structure such as spiral arms is unfortunately also noted.

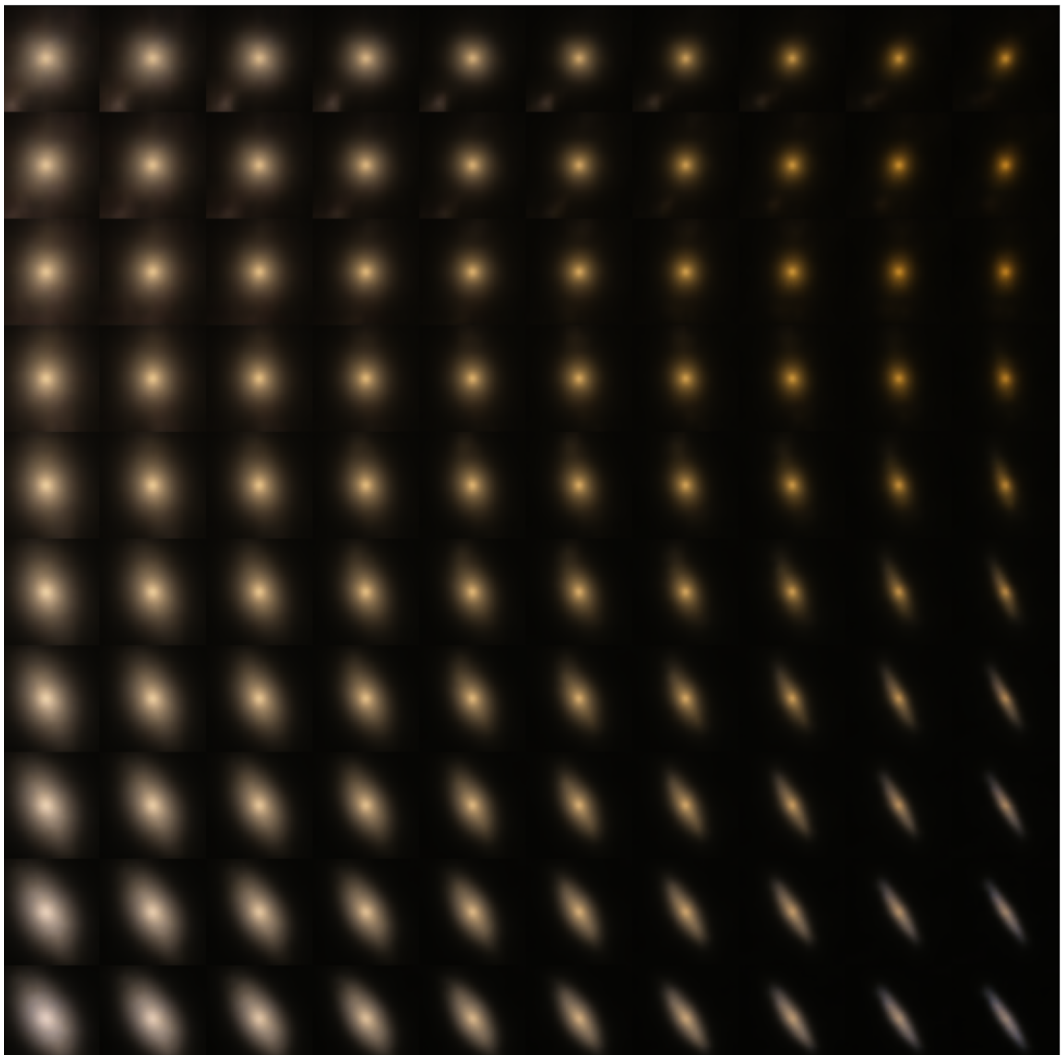


Figure 2.11: In this plot, we randomly selected two latent variables and plotted representations of sampled decoder outputs as these two latent variables were changed across their range; all other latent variables were fixed at 0. The $x-$axis corresponds to one latent variable and the $y-$axis corresponds to the second latent variable. We observe a fair bit of structure indicating that the latent space observes the desirable quality of being regular.

## 2.4 Diffusion Probabilistic Model

### 2.4.1 Derivation from VAE

While the VAE architecture, see Section (2.3.3), is structurally very dissimilar to the diffusion probabilistic model (DPM), we can use it as a starting point from which we formulate the DPM. This idea of using the VAE as a starting point to derive the DPM comes from [26] which is a much recommended resource for those who seek a deep dive into diffusion models and will be referenced in this section multiple times.

If we stack VAEs on top of one another, such that the outputs of one become the inputs of the other, we end up with an architecture that resembles Figure (2.12).

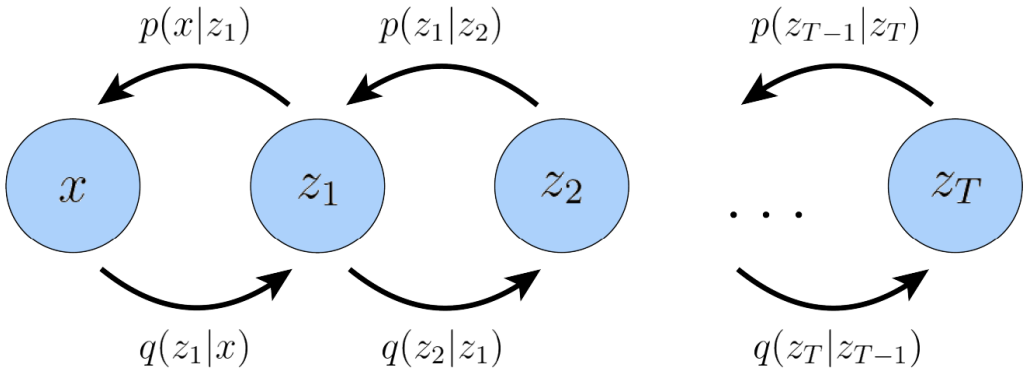



Figure 2.12: HVAE architecture with latents $\{z_t|0 < t \leq T\}$. See page 6 in [26].

The architecture shown in Figure (2.12) is a hierarchical VAE or an HVAE with $T$ hierarchical levels and $T-1$ latents; in other words, an HVAE is a generalization of a VAE with multiple hierarchies over multiple latent spaces. In a regular HVAE, the latents in level t are conditioned on all the previous latents; the plot shown above is a Markovian HVAE in which the latents in level $t$ are conditioned only by the latents in level $t-1$.

The DPM is a specific type of HVAE with constraints set upon its architecture. The first constraint is that the latent dimensions are constant across equal to the input dimensions. Secondly, unlike the VAE, the DPM's encoder's parameters are not learned but are defined prior to training. Lastly, the encoder's parameters are set up such that the latent at the final timestep $T$ is a standard Gaussian distribution.

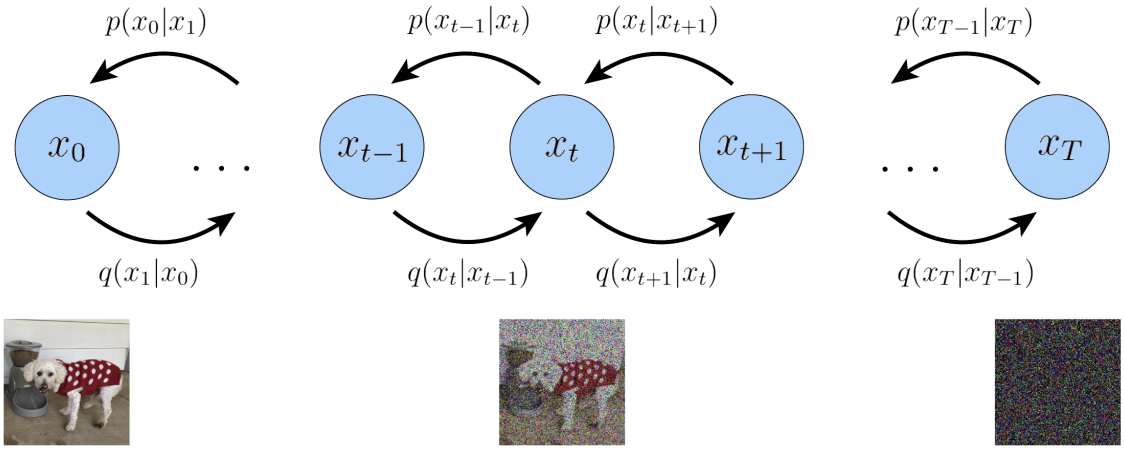



Figure 2.13: Diffusion Model adding noise in forward step and denoising in backwards step. Page 7 in [26].

The result is an architecture, seen in Figure (2.13), that gradually adds noise to its input data as it is passed along the levels of the model. The input image $\mathbf{x}_0$ is diffused with noise along $T$ timesteps with $\mathbf{x}_t$ denoting an intermediate noisy version of $\mathbf{x}_0$. Since the forward diffusion process $q(\mathbf{x}_{t+1}|\mathbf{x}_t)$ is defined before training, the learned parameters are the ones of the reverse diffusion process $p(\mathbf{x}_t|\mathbf{x}_{t+1})$. We will unpack each of these processes in detail in Section (2.4.2).

### 2.4.2 Inner workings

#### Forward Diffusion

Our diffusion model starts with the forward diffusion process where we gradually add noise to the input image until the pixel distribution of the image follows a Gaussian distribution at the final timestep $T$. At each step $t$ in the Markov chain, we add Gaussian noise with variance $\{\beta_t \in (0,1)\}_{t=1}^{T}$ making a latent at timestep $t$:

$$q(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x}_t; \sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t\mathbf{I}) \tag{2.10}$$

$$= \sqrt{1-\beta_t}\mathbf{x}_{t-1} + \sqrt{\beta_t}\boldsymbol{\epsilon}, \tag{2.11}$$

where $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$.

The variance parameter $\beta_t$ can be fixed or follow a schedule that adjusts over the $T$ timesteps. The authors of the original paper used a linear diffusion schedule where $\beta_t$ increases linearly over time; the increase over time is due to being able to afford a larger update step as the images get noisier. We used the same linear diffusion schedule.

Note that, unlike the VAE, the encoder distributions are not parameterized since they are not learned during training but are predefined by the variance parameter. This means that we only have to learn the conditionals $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$ to generate original data.

### Reparameterization Trick

An unfortunate shortcoming of the forward diffusion process, as we have currently defined it, is that to obtain the noised image $x_t$, we would have to jump through $t-1$ applications of $q$. However, there is a nifty reparameterization trick that allows us to circumvent this shortcoming. First we take $\alpha_t = 1 - \beta_t$ and $\bar{\alpha}_t = \prod_{i=1}^{t} \alpha_i$. Then we substitute into Equation (2.11) and obtain for an arbitrary sample $\mathbf{x}_t \sim q(\mathbf{x}_t|\mathbf{x}_0)$:

$$\begin{aligned}
\mathbf{x}_t &= \sqrt{\alpha_t}\mathbf{x}_{t-1} + \sqrt{1-\alpha_t}\boldsymbol{\epsilon}_{t-1} && (2.12)\\
&= \sqrt{\alpha_t\alpha_{t-1}}\mathbf{x}_{t-2} + \sqrt{1-\alpha_t}\boldsymbol{\epsilon}_{t-1} + \sqrt{1-\alpha_{t-1}}\boldsymbol{\epsilon}_{t-2} && (2.13)\\
&= \sqrt{\alpha_t\alpha_{t-1}}\mathbf{x}_{t-2} + \sqrt{1-\alpha_t\alpha_{t-1}}\bar{\boldsymbol{\epsilon}}_{t-2} && (2.14)\\
&= \sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon} && (2.15)\\
&\sim \mathcal{N}(\mathbf{x}_t; \sqrt{\bar{\alpha}_t}\mathbf{x}_0, (1-\bar{\alpha}_t)\mathbf{I}), && (2.16)
\end{aligned}$$

where $\boldsymbol{\epsilon}_{t-1}, \boldsymbol{\epsilon}_{t-2}, \cdots \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ and where $\bar{\boldsymbol{\epsilon}}_{t-2}$ merges two Gaussians using the fact that the merger of two Gaussians with different variances, $\mathcal{N}(\mathbf{0}, \sigma_1^2\mathbf{I})$ and $\mathcal{N}(\mathbf{0}, \sigma_2^2\mathbf{I})$, results in a Gaussian $\mathcal{N}(\mathbf{0}, (\sigma_1^2 + \sigma_2^2)\mathbf{I})$. We have applied this recursively to arrive at a closed form of $q$, see Equation (2.16), that allows us to sample an arbitrary $x_t$ using only knowledge of $x_0$ and the $\alpha_i$ parameters. For more details, see page 11 in [26].

**Reverse Diffusion**

For the reverse diffusion process, we are looking to model the ground truth denoising step $q(\mathbf{x}_{t-1}|\mathbf{x}_t)$. For a small enough variance $\beta_t$, $q(\mathbf{x}_{t-1}|\mathbf{x}_t)$ is a Gaussian, however, we cannot estimate it directly since we need access to the entire dataset. Therefore, we seek to model $q(\mathbf{x}_{t-1}|\mathbf{x}_t)$ using a neural network $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$. If we can optimize the network parameters $\theta$, we can generate novel images by passing random Gaussian noise, $\mathcal{N}(\mathbf{0}, \mathbf{I})$, to $p_\theta$ and iterate backward $T$ timesteps all the way to a novel $\mathbf{x}_0$.

To train this model, we will be using the same ELBO loss we used in the VAE section. To derive the ELBO, we rewrite the encoder transitions $q(\mathbf{x}_t|\mathbf{x}_{t-1}) = q(\mathbf{x}_t|\mathbf{x}_{t-1}, x_0)$. The added conditioning term $\mathbf{x}_0$ does not affect the calculations due to the Markov property. The resulting ELBO takes the form:

$$\log p(\mathbf{x}) = \log \int p(\mathbf{x}_{0:T})\, d\mathbf{x}_{1:T} \tag{2.17}$$

$$= \log \int \frac{p(\mathbf{x}_{0:T}) q(\mathbf{x}_{1:T}|\mathbf{x}_0)}{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}\, d\mathbf{x}_{1:T} \tag{2.18}$$

$$= \log \mathbb{E}_{q(\mathbf{x}_{1:T}|\mathbf{x}_0)} \left[ \frac{p(\mathbf{x}_{0:T})}{q(\mathbf{x}_{1:T}|\mathbf{x}_0)} \right] \tag{2.19}$$

$$\geq \mathbb{E}_{q(\mathbf{x}_{1:T}|\mathbf{x}_0)} \left[ \log \frac{p(\mathbf{x}_{0:T})}{q(\mathbf{x}_{1:T}|\mathbf{x}_0)} \right] \tag{2.20}$$

$$= \underbrace{D_{\mathrm{KL}}\left[q(\mathbf{x}_T|\mathbf{x}_0) \parallel p_\theta(\mathbf{x}_T)\right]}_{L_T} - \underbrace{\log p_\theta(\mathbf{x}_0|\mathbf{x}_1)}_{L_0} + \underbrace{\sum_{t=2}^{T} D_{\mathrm{KL}}\left[q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) \parallel p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)\right]}_{L_{t-1}} \tag{2.21}$$

Equation (2.21) gives the loss function of the DPM. $L_0$ is the reconstruction term which is identical to the reconstruction term in the VAE's loss function. $L_T$ is the prior matching term which indicates how close the final latent $\mathbf{x}_T$ is to a standard Gaussian; this term has no trainable parameters and is effectively zero as we assume a large enough $T$ such that $\mathbf{x}_T$ is Gaussian. $L_{t-1}$ is a denoising matching term that measures how far off the model's estimate of the noise is from the noise added during forward diffusion. We learn desired denoising transition step $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$ as an approximation to tractable, ground-truth denoising transition step $q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)$; $L_{t-1}$ is minimized

when our predicted denoising step matches the ground-truth denoising step. The optimization cost is primarily dominated by $L_{t-1}$ since we must optimize over all timesteps $t$. For more details on the derivation of Equation (2.21), see pages 9 and 10 of [26].

For an arbitrary Markovian HVAE, the $L_{t-1}$ term would be difficult to minimize due to the complexity of having to simultaneously learn the encoder but we can utilize the fact that for our Diffusion model, the encoder transitions are predefined as Gaussians. Thus, we rewrite $q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)$ using Bayes rule:

$$q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) = \frac{q(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{x}_0)q(\mathbf{x}_{t-1}|\mathbf{x}_0)}{q(\mathbf{x}_t|\mathbf{x}_0)} \tag{2.22}$$

Due to the Markovian property, $q(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{x}_0)$ is equivalent to our encoder transitions $q(\mathbf{x}_t|\mathbf{x}_{t-1})$ given by $\mathcal{N}(\mathbf{x}_t; \sqrt{\bar{\alpha}_t}\mathbf{x}_0, (1-\bar{\alpha}_t)\mathbb{I})$. What remains is to derive the forms of $q(\mathbf{x}_{t-1}|\mathbf{x}_0)$ and $q(\mathbf{x}_t|\mathbf{x}_0)$ so that we can arrive at a closed form expression of $q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)$. We leverage the parametrization trick given above to rewrite

$$q(\mathbf{x}_{t-1}|\mathbf{x}_0) = \mathcal{N}(\mathbf{x}_{t-1}; \sqrt{\bar{\alpha}_{t-1}}\mathbf{x}_0, (1-\bar{\alpha}_{t-1})\mathbb{I}) \tag{2.23}$$

$$q(\mathbf{x}_t|\mathbf{x}_0) = \mathcal{N}(\mathbf{x}_t; \sqrt{\bar{\alpha}_t}\mathbf{x}_0, (1-\bar{\alpha}_t)\mathbb{I}). \tag{2.24}$$

Plugging these forms into Equation (2.22) gives:

$$q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) = \frac{\mathcal{N}(\mathbf{x}_t; \sqrt{\alpha_t}\mathbf{x}_{t-1}, (1-\alpha_t)\mathbb{I})\mathcal{N}(\mathbf{x}_{t-1}; \sqrt{\bar{\alpha}_{t-1}}\mathbf{x}_0, (1-\bar{\alpha}_{t-1})\mathbb{I})}{\mathcal{N}\mathbf{x}_t; \sqrt{\bar{\alpha}_t}\mathbf{x}_0, (1-\bar{\alpha}_t)\mathbb{I})} \tag{2.25}$$

$$\propto \mathcal{N}\left(\mathbf{x}_{t-1}; \boldsymbol{\mu}_q(x_t, x_0), \boldsymbol{\Sigma}_q(t)\mathbb{I}\right) \tag{2.26}$$

where

$$\boldsymbol{\mu}_q(x_t, x_0) = \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})\mathbf{x}_t + \sqrt{\bar{\alpha}_{t-1}}(1-\alpha_t)\mathbf{x}_0}{1-\bar{\alpha}_t}, \tag{2.27}$$

and

$$\boldsymbol{\Sigma}_q(t) = \frac{(1-\alpha_t)(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}. \tag{2.28}$$

For details on the derivation of Equation (2.26), see page 12 in [26].

We now have a form for the ground truth denoising step that is a Gaussian with mean $\boldsymbol{\mu}_q(x_t, x_0)$ as a function of $x_t$ and $x_0$ and variance $\boldsymbol{\Sigma}_q(t)$ as a function of the $\alpha$ parameters. Additionally, we can use

$$\mathbf{x}_0 = \frac{\mathbf{x}_t - \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}_0}{\sqrt{\bar{\alpha}_t}}, \tag{2.29}$$

to substitute into Equation (2.27) and arrive at

$$\boldsymbol{\mu}_q(\mathbf{x}_t, \boldsymbol{\epsilon}_0) = \frac{1}{\sqrt{\alpha_t}}\Big(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}_0\Big), \tag{2.30}$$

where we have parameterized $\boldsymbol{\mu}_q$ by $\mathbf{x}_t$ and the source noise $\boldsymbol{\epsilon}_0$ that determines $\mathbf{x}_t$ from $\mathbf{x}_0$.

Now, we can set our learned denoising step $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$ to be a Gaussian; since we want $p_\theta$ to model the ground truth denoising step as closely as possible, we can construct the variance of $p_\theta$ to be equal to Equation (2.28) which, as mentioned previously, is only dependant on the $\alpha$ parameters. However, unlike the mean of $q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)$, the mean of $p_\theta(\mathbf{x}_{t1}|\mathbf{x}_t)$ cannot be parameterized on $\mathbf{x}_0$ so we must parameterize it as a function of $\mathbf{x}_t$ only. We can take inspiration from the form of $\boldsymbol{\mu}_q$, Equation (2.30), and set the mean as

$$\boldsymbol{\mu}_\theta(\mathbf{x}_t, t) = \frac{1}{\sqrt{\alpha_t}}\Big(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\hat{\boldsymbol{\epsilon}}_{\boldsymbol{\theta}}(\mathbf{x}_t, t)\Big), \tag{2.31}$$

where $\hat{\boldsymbol{\epsilon}}_{\boldsymbol{\theta}}(x_t, t)$ is a nueral network that seeks to predict the source noise $\boldsymbol{\epsilon}_{\mathbf{0}}$ which differentiates $\boldsymbol{x}_t$ from $\boldsymbol{x}_0$.

Our optimization problem is then:

$$\arg\min_\theta \parallel \boldsymbol{\epsilon}_{\mathbf{0}} - \hat{\boldsymbol{\epsilon}}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t) \parallel_2^2 . \tag{2.32}$$

That is we seek to minimize the difference between the ground truth noise $\boldsymbol{\epsilon}_{\mathbf{0}}$ and the noise predicted by the network $\hat{\boldsymbol{\epsilon}}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t)$.

### 2.4.3 Implementation

#### Network Architecture

The architecture of the DPM is known as a U-net. First used in [27], the U-net design is similar to the VAE design in that it is comprised of two halves; a downsampling half which decreases spatial resolution and increases channels, and an up sampling half which increases spatial resolution and decreases channels. These two halves surround the latent layers of the network; these are the layers where no down or up sampling occurs and that have the lowest spatial resolution across the whole network.

Like the VAE, the U-net architecture's output has the same shape as the input dimension but the U-net is distinguished by its skip connections between down-blocks and up-blocks (each block is a collection of layers that either downsample or upsample their input) with the corresponding spatial resolution. This makes the U-net a non-sequential network since it allows data to take shortcuts through the network. The U-net architecture is very well suited for applications where the desired output has the same size as the input as is the case with the predicted and the input image respectively.

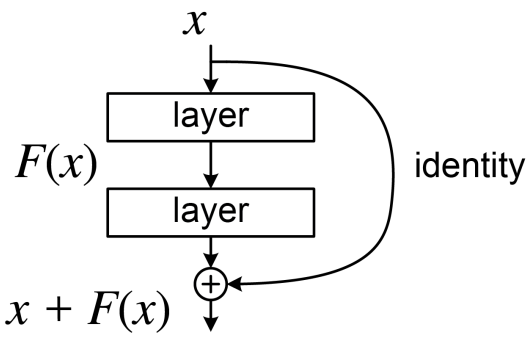



Figure 2.14: Implementation of a residual block. The input $x$ is added to the output of the block $F(x)$ such that the final output of the block is $F(x) + x$

The down-blocks and up-blocks that comprise the U-net are made up of residual blocks first introduced in [28] and visualized in Figure (2.14). The residual block is comprised of a group of layers with a skip connection between the input of the block and the final layer in the block; note that now we have two types of skip connections in

the network: skip connection between down-blocks and up-blocks, courtesy of the U-net architecture, and also skip connections within the blocks, courtesy of the residual block design. The advantage of skip connections is that the network does not have to reconstruct the images from scratch, rather, it can learn the transformation between the input image and the desired output. Another advantage of the skip connections is that they alleviate the issues associated with vanishing gradients which allows us to build much deeper networks.

The residual block itself is made up of convolutional layers, dense layers to handle the time embedding inputs and normalization layers; regardless of the layers' combination, the output of the residual block has the same size as the input and a skip connection is implemented between input and output layers.

No downsampling or upsampling is performed within the residual blocks; to perform downsampling and upsampling, average pooling layers and upsampling layers accompany the residual blocks in down-blocks and up-blocks respectively. The down-blocks' outputs are saved such that they may be used by the up-blocks; the up-blocks take the down-blocks' output and concatenate it to their own thus implementing the skips as seen in Figure (2.15).

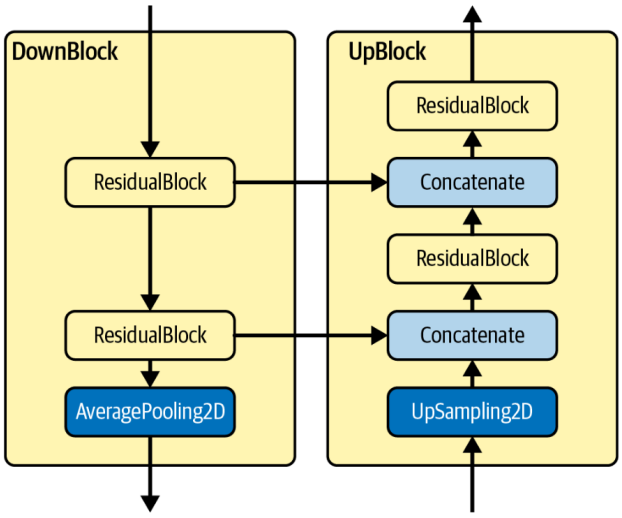



Figure 2.15: This diagram, page 223 in [12], demonstrates how the down-blocks are connected to corresponding up-blocks.

Additional to the original U-net architecture are sinusoidal embedding and attention layers. Sinosudial embedding's idea here, in short, is that we want to transform

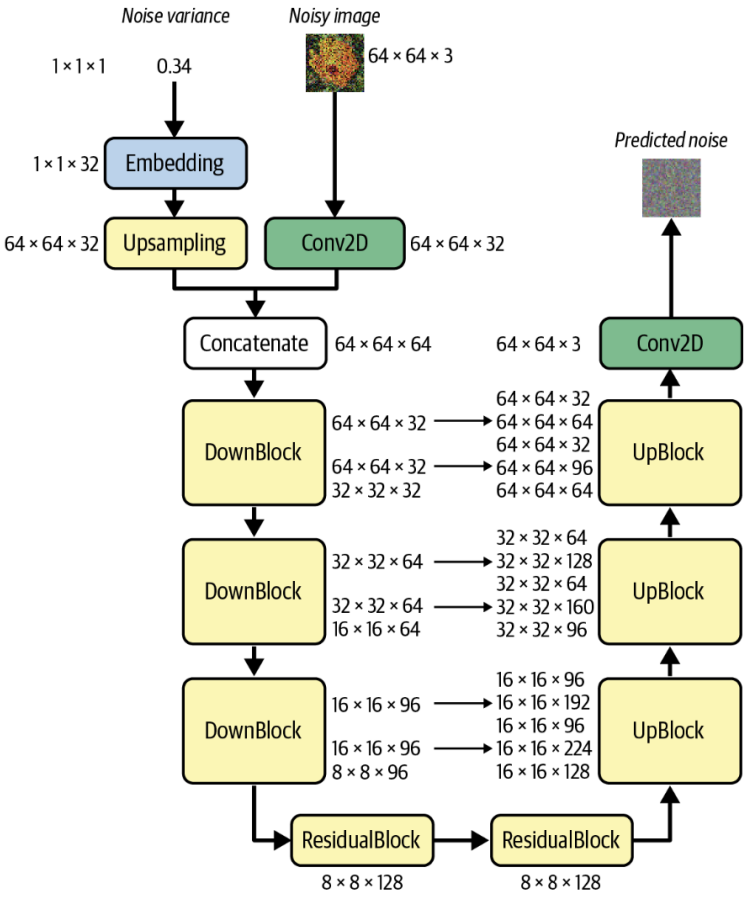



Figure 2.16: The U-net architecture of the DPM (page 217 in [12]). The network accepts 2 inputs on the right hand side: the timestep parameter t is accepted by the sinusoidal embedding layer and is then up sampled before being concatenated to the noised image input. The output of the network is the predicted noise.

a scalar value into a higher dimensional tensor capable of a more complex representation that will be used along the network. The idea is analogous to the encoder of a VAE but in reverse since the encoder attempts to find a lower dimensional representation while sinusoidal embedding seeks to find a higher dimensional representation. We pass $t$ to the sinusoidal embedding layer at the start of the network to produce a higher dimensional vector that will be carried all along the network.

Attention layers are implemented as well in our U-net. In a nutshell, attention layers allow the network to assign higher values, and thus pay more "attention", to areas of the image that contain the most useful information. An example of that would be assigning higher values to areas that denote edges that define objects and

lower values to areas that could be considered the background of an image. Note that attention layers are an architectural choice and are not necessarily required for the DPM to function as intended. For a deep dive into the inner workings of the attention mechanism, see [29].

The body of the U-net architecture follows the following layout:

DB32-DB64-DB96-RES128-RES128-UB96-UB64-UB32.

where DBk is a down-block with k filters, UBk is an up-block with k filters and RESk is a residual block with k filter. The complete U-net can be seen in Figure (2.16).

#### Training details

We employ the same dataset as we did to train the VAE but we randomly selected around 8000 images from the 22000 used to train the VAE. The DPM's training dataset size was arrived at independently of the VAE's training dataset size; the only factors we considered were that an increase in dataset size beyond 8000 did not yield any noticeable difference in the generated samples of the training DPM. The relatively small size of the training data compared with the VAE's training data is a testament to the power of the DPM, as compared to the VAE since the network is able to learn from a much smaller dataset. This ability is courtesy of all those features we discussed in the previous section as well as the more sophisticated training algorithm. We do however run the DPM for 800 epochs which despite the smaller dataset does take considerably longer compared to the VAE; the results are worth the added waiting time as we will see in the next section.

Finally, we use a batch size of 32 and set $T = 1000$ corresponding to 1000 timesteps along which Gaussian noise is diffused to the initial image $\mathbf{x}_0$ to transform it to Gaussian noise $\mathbf{x}_T$. The batch size number, timestep number and epoch number selected were used in the original diffusion paper, [30].

### 2.4.4 Results

Due to the complexity of the model, as compared to the VAE, the sampling process is significantly more involved. To sample from our trained DPM, we first generate

random Gaussian noise $\mathbf{x}_T$. Note that we are not constrained to using the same batch size used during training; we can use a custom batch size to generate any number of images in one go. This batch of Gaussian noise is fed into our trained network along with the timestep $t$, initially set to $T = 1000$, to obtain the predicted noise. This predicted noise is removed from the noisy batch to give $\mathbf{x}_{T-1}$, these outputs are fed again into the model along with the modified timestep, and the whole process is repeated $T$ times until we arrive at a batch of original generated images $\mathbf{x}_0$.

Figure (2.17) shows the progression of losses during training. If we were to take the training loss at face value, we would perhaps conclude that the DPM's performance does not change much across its 800 epochs of training and that perhaps training it for this many epochs is a waste of resources. We have even plotted a uniform distribution centered on the mean loss, across all epochs, and it exhibits almost as much structure and behaviour as the actual training loss.

This is the downside with interpreting a generative model's performance through the loss function; the loss function while providing adequate feedback to the network during training is not a measure in itself of the model's ability to generate novel images. Furthermore, Figure (2.18) shows a clear progression of improved performance from the outputs of later epochs' checkpoints compared to the outputs from earlier epochs.

Our investigation of the model's performance is a qualitative one as the generated images have to be visually inspected to assess our model's generative capabilities.

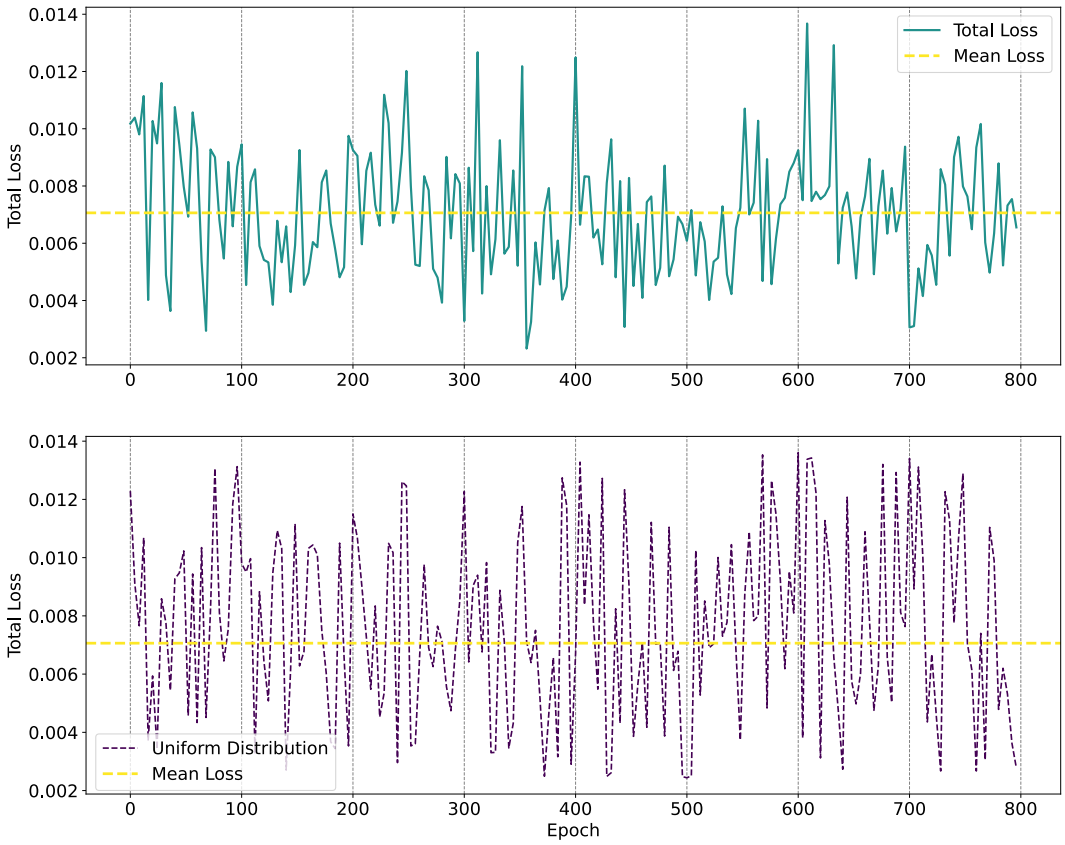



Figure 2.17: Total loss across training (above) and a random uniform distribution (below). If the labels were removed, the total training loss could perhaps be mistaken for the uniform distribution.

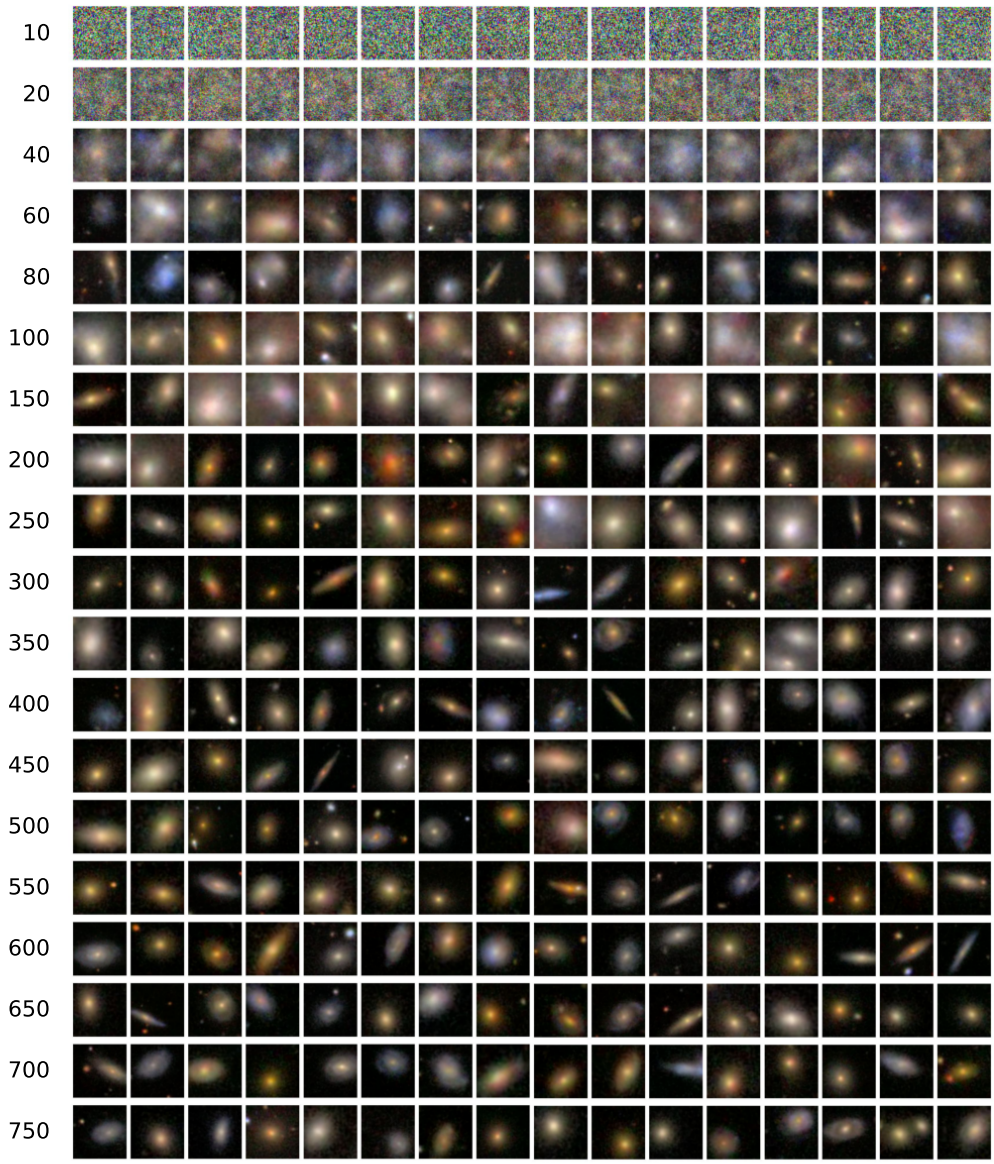



Figure 2.18: The numbers on the left hand side of the plot indicate the epoch of the checkpoint used to generate the row of images opposite to it. This figure shows the clear progression across training in the generated images from noisy outputs to galaxy-like blobs and finally to high quality images.

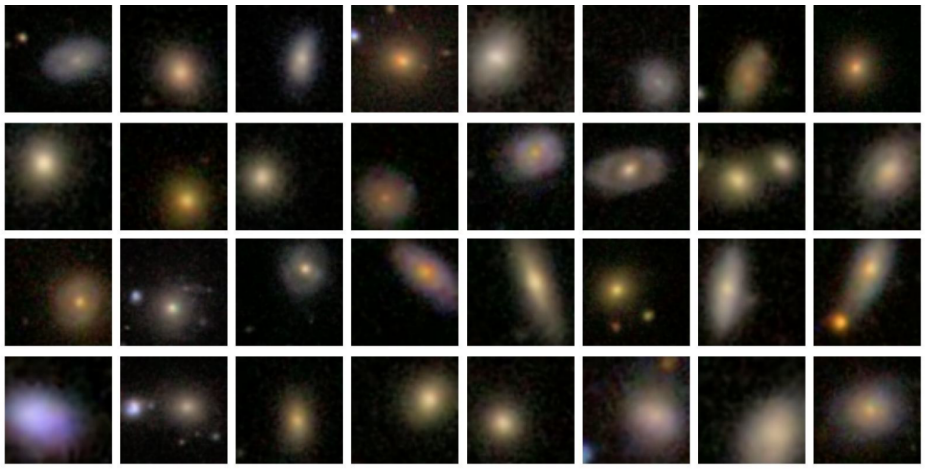


Figure 2.19: Sample of generated images from the trained (800 epochs) DPM.

The results of the DPM, seen in Figure (2.19) are a vast improvement over the results of the VAE from Section (2.3.5). Note the variety of colors, shapes and orientations. The DPM even manages to resolve some spiral arms in a few instances. The DPM's more sophisticated architecture and training routine are more computationally costly compared to the VAE, but are worth the results.

## 2.5 DCGAN

In this section, we present an implementation of a deep convolutional generative adverserial network (DCGAN). We present a much more sophisticated implementation of a generative adverserial network, which builds on the concepts introduced in this section, in Chapter 3.

### 2.5.1 Theory and Background

The VAE architecture could be decomposed into two distinct models (encoder and decoder) joined at the bottleneck layer. Both these models are working together to achieve the goal of recreating the original input image. The generative adversarial

network (GAN), first introduced in [31], similarly has two models defined within it, but rather than working together, they are adversaries competing with each other to achieve opposite ends.

The two networks that make up a GAN are a generator, $G$, and a discriminator, $D$. The purpose of training a GAN is to produce a high quality generator; the generator is the part of the GAN that is retained for generative purposes after training much like the decoder is with the VAE. Specifically, the generator converts random noise $\mathbf{z} \sim p_{\mathbf{z}}(\mathbf{z})$ into images that look like they came from the original training data distribution $p_{data}$. In this sense, we can think of the generator as a forger whose job is to create novel data that is indistinguishable from the original set. The discriminator facilitates the generator's training; its job is to classify images based on how likely they are to have come from the original dataset as compared to the generator. In this way, the discriminator is like a detective whose job is to stop the generator from slipping any fake data past it. This process is visualized in Figure (2.20).

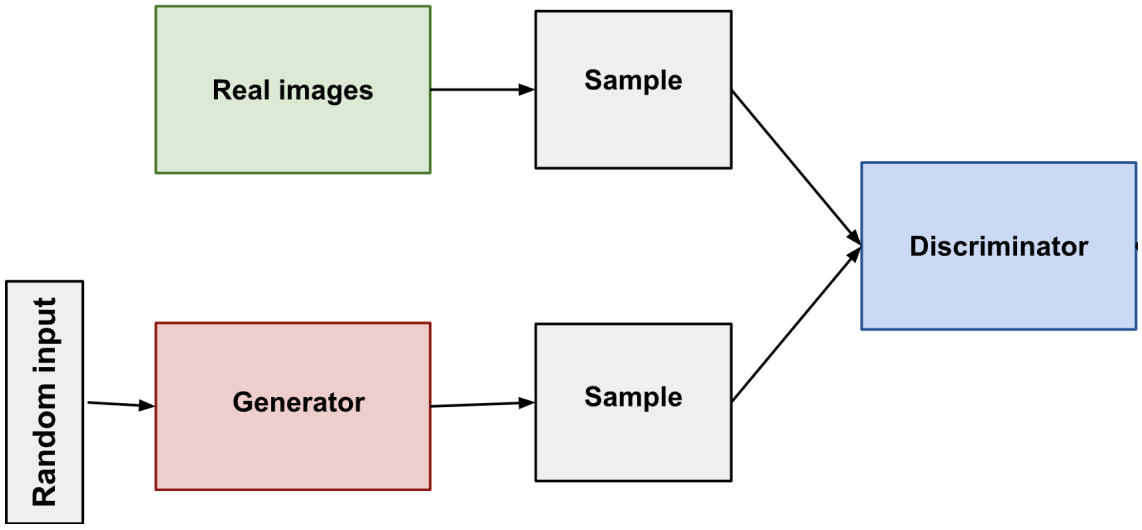



Figure 2.20: This figure shows a flowchart that outlines the how the generator and discriminator of a GAN share data.

Training the discriminator boils down to a standard image classification task where the model takes as input an image and outputs the probability of the image coming from the real dataset. This is then a supervised learning task where we assign real and fake images the labels 1 and 0 respectively. Therefore, the discriminator is trained to classify inputs from the real dataset as 1 and inputs from the generator as 0. The

discriminator's objective is given by:

$$V(D) = \mathbb{E}_{\mathbf{x}\sim p_{\text{data}}(\mathbf{x})}[\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{z}\sim p_{\mathbf{z}}(\mathbf{z})}[\log(1 - D(G(\mathbf{z})))]. \tag{2.33}$$

To train the generator, we rely on the discriminator to score its outputs. An ideal generator will be able to produce results that fool the discriminator; upon being given a batch of fake images, the discriminator would output a vector of 1's indicating that it recognizes the generated images as real. The generator's objective is given by:

$$V(G) = \mathbb{E}_{\mathbf{z}\sim p_{\mathbf{z}}(\mathbf{z})}[\log(1 - D(G(\mathbf{z})))] \tag{2.34}$$

The two networks are trained in an alternating process where feedback is exchanged between the networks such that their progress is interdependent. This interplay between both networks can be modeled by the following two-player minimax game where $G$ wants to minimize $V$ while $D$ wants to maximize it.

$$\min_G \max_D V(D, G) = \mathbb{E}_{\mathbf{x}\sim p_{\text{data}}(\mathbf{x})}[\log D(\mathbf{x})] - \mathbb{E}_{\mathbf{z}\sim p_{\mathbf{z}}(\mathbf{z})}[\log(D(G(\mathbf{z})))] \tag{2.35}$$

Note that we have substituted $[1 - \log(D(G(\mathbf{z}))]$ for $[-\log(D(G(\mathbf{z}))]$ since the former may not provide sufficient gradient for $G$ to perform backpropagation. This modification maintains equilibrium dynamics while providing the generator and discriminator with stronger gradients. The improved signals lead to faster convergence and more efficient training as can be referenced in [31].

### 2.5.2 Implementation

#### Network Architecture

Let's delve into the architecture of both networks, $G$ and $D$, to see how they are implemented. Our generator accepts random noise, $\mathbf{z}$ and proceeds to upsample it through transposed, strided convolutions. The ReLU activation was used throughout the netork. Our generator, $G$, has the following architecture:

D100-CT256-CT128-CT64-C3

This architecture is nearly identical to the decoder of the VAE since (see Section 2.3.4), like the decoder, its function is to map from the a low dimensional latent space to an image. Of course, the VAE's latent space is organized and semantically meaningful while the latent space of $G$ is random Gaussian noise.

The discriminator employs a fairly basic architecture as concerns binary image classification. The network is essentially comprised of convolutional layers that use strided convolutions to reduce the spatial resolution of filters while also increasing the number of channels along the depth of the network. The output is a single neuron with a sigmoid activation to output as is standard for binary classification problems like this, see [6]. The discriminator, $D$, has the following architecture:

C64-C128-C128-D1

**Implementation**

We trained our GAN using the same dataset used to train the VAE in Section (2.3); that is a total of 22,000 images. We used a batch size of 64 images and trained the model for 50 epochs. No further benefits were seen from training beyond 50 eopchs. The Adam optimizer was used to perform gradient descent.

### 2.5.3 Results

The results, seen in Figure (2.21), are far inferior to the results obtained previosly with the DPM and the VAE seen in Figures (2.19) and (2.10) respectively. The outputs of the DCGAN are noisy and exhibit no structure whatsoever as far as spirals or bars are concerned.

As unconvincing as the results displayed in Figure (2.21) are, they were in fact obtained after extensive hyperparameter tuning. We found the DCGAN to be significantly more sensitive, compared to the VAE or DPM, to different hyperparameter configurations such that models whose architecture differed slightly from the one listed in Section (2.5.2) produced only noise after training. These results are not unexpected since these sort of unconditioned GANs are notoriously difficult to train, see [32].

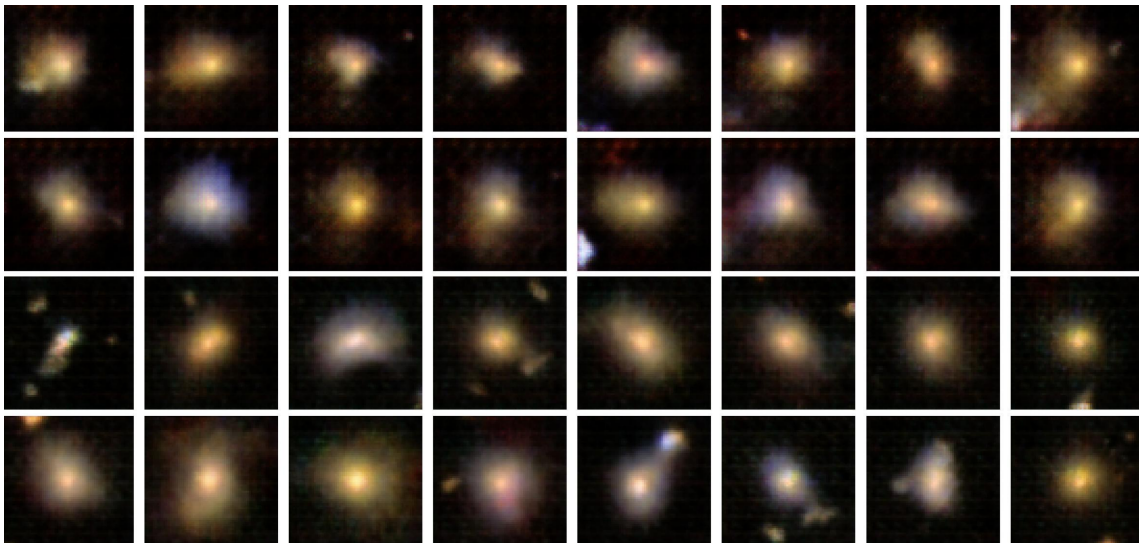


Figure 2.21: Sample of generated images from the trained generator after 50 epochs.

# Chapter 3

# Conditioned Generation

## 3.1 Image-to-image

In Chapter 2, we explored unconditioned generative models which require no explicit conditioning to produce their output. These models learn to generate output images from random noise or latent representations without any specific conditioning restricting their outputs. Unconditioned generative models offer more flexibility in generating diverse outputs however, this lack of conditioning results in less deterministic or interpretable outputs. Conditioned models learn to generate output images that satisfy a given set of conditions supplied to the network. This provides us with fine-grained control over the produced images and is easier to assess qualitatively due to the ability to determine whether or not the output image meets the condition we have set. For more details on conditioned vs unconditioned generative modeling, see [12].

An example of such conditioned models is image-to-image translation where each input image is paired with a corresponding output image which enables targeted image manipulation. Image-to-image models vary in implementation and scope and can be supervised as in [33], where labeled data is available, or unsupervised as in [34]. For this section, we will initially assume image-to-image to refer to supervised tasks but will introduce an unsupervised implementation in Section (3.2).

Image-to-image techniques involve learning a mapping that translates images from

one domain to another, where each domain represents a different visual style or semantic meaning. An example is the task of image colorization where grayscale images serve as input, while colorized versions of these images are provided as output. The grayscale inputs serves as a reference for generating the output images in the target domain. This type of image-to-image translation deals with pairs of images which places a considerable limitation on finding appropriate training data. Despite this, image-to-image techniques have a broad range of applications from image colorization, see [35], to semantic segmentation, see [36].

Another example of conditioned generative modeling is style transfer, see [37]. Style transfer typically operates on individual images, where the content and style images are provided as input, and the output is a single stylized image that imbues the content image with the style characteristic. For this reason, it does not require paired data and is more flexible than image-to-image techniques. However, it is not well suited for approaches where quantitative accuracy is desired but is more suited for artistic applications, such as creating stylized artwork, transforming photographs into paintings, or generating visually appealing images with artistic effects. Image-to-image techniques are limited by requiring paired datasets but do a much better job with tasks where quantitative accuracy is needed. So, it is a trade-off between flexibility and quantitative accuracy.

## 3.2 CycleGAN

CycleGAN, introduced in [34], is an image-to-image technique that belongs to the family of GANs. Unlike previous image-to-image techniques like pix-to-pix, [33], which rely on paired data for training, CycleGAN introduces a novel framework for unpaired image-to-image translation which provides the flexibility of style transfer with the quantitative accuracy of supervised image-to-image models.

Let us first introduce some notation for the problem we are exploring. We are concerned with translating an input image $x$ sampled from the data distribution $X$ to a corresponding image $y$ in the target domain $Y$. Examples of $X$ and $Y$ pairings could be photographs and paintings, horses and zebras, or apples and oranges. As concerns astronomical image data, we could select pairings such as elliptical and spiral galaxies or images taken at different photometric bands; we will discuss such pairings in more

detail further ahead.

As we saw in Section (2.5), an unconditioned GAN is comprised of a generator and discriminator that are trained in tandem. CycleGAN uses two distinct pairs of GANs. The first pair contains the mapping function $G : X \rightarrow \hat{Y}$ along with an associated adversarial discriminator $D_Y$ while the second pair is comprised of the inverse mapping function $F : Y \rightarrow \hat{X}$ along with its associated discriminator $D_X$. $\hat{X}$ and $\hat{Y}$ are distributed identically to $X$ and $Y$ respectively.

The problem is that there may be infinitely many mappings between input image $x$ and the output $\hat{y} = G(x)$, with the same being true for the counterpart, with no guarantee of finding a semantically meaningful mapping. The authors of the original CycleGAN paper, [34], came up with a solution that utilizes the 'cycle consistency' property of these transformations. Cycle consistency implies that if $G$ and $F$ are inverse mappings, then we should observe that $G(F(y)) = y$ and $F(G(x)) = x$.

To enforce this cycle consistency property, CycleGAN introduces a cycle consistency loss term

$$\mathcal{L}_{\text{cycle}}(G, F) = \mathbb{E}_{x \sim p_{\text{data}}(x)} \left[ \|F(G(x)) - x\|_1 \right] + \mathbb{E}_{y \sim p_{\text{data}}(y)} \left[ \|G(F(y)) - y\|_1 \right]. \quad (3.1)$$

$\mathcal{L}_{\text{cycle}}(G, F)$ penalizes the discrepancy between the original input image and the image reconstructed from its translated counterpart, visualized in Figure (3.1), and is added to the adversarial loss which is identical to the objective of the unconditioned GAN we looked at earlier. Since we have two pairs of GANs, we have a pair of adversarial losses, first introduced as Equation (2.35), the loss for $G$ and its corresponding discriminator $D_y$ is

$$\mathcal{L}_{\text{adv}}(G, D_Y, X, Y) = \mathbb{E}_{y \sim p_{\text{data}}(y)}[\log D_Y(y)] + \mathbb{E}_{x \sim p_{\text{data}}(x)}[\log(1 - D_Y(G(x)))], \quad (3.2)$$

while the loss for $F$ and its associated discriminator $D_x$ is

$$\mathcal{L}_{\text{adv}}(F, D_X, X, Y) = \mathbb{E}_{x \sim p_{\text{data}}(x)}[\log D_X(x)] + \mathbb{E}_{y \sim p_{\text{data}}(y)}[\log(1 - D_X(F(y)))]. \quad (3.3)$$

The full objective function of the cycle GAN becomes:

$$\mathcal{L}(G, F, D_X, D_Y, X, Y) = \mathcal{L}_{\text{adv}}(G, D_Y, X, Y) + \mathcal{L}_{\text{adv}}(F, D_X, Y, X) + \mathcal{L}_{\text{cycle}}(G, F) \tag{3.4}$$

To sum up, we are training two GANs in tandem which in turn are comprised of two networks that are trained in tandem. The adversarial loss ensures that each generator learns a mapping from its input domain to the target domain while the cycle consistency loss acts as a regularizing term that constrains the learned mapping to be semantically meaningful.

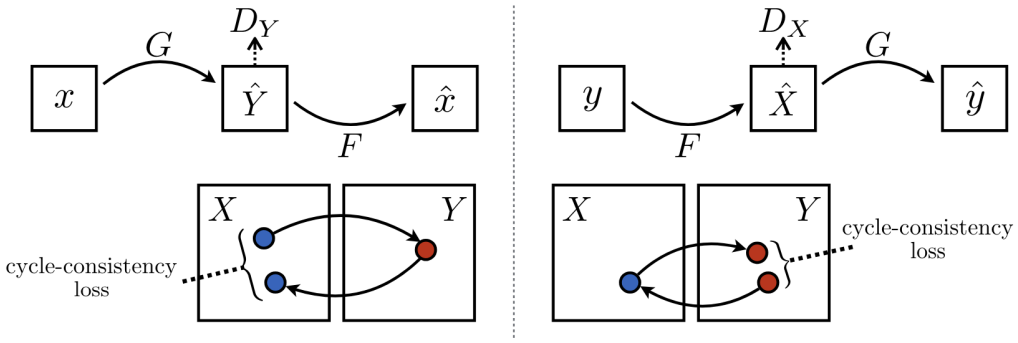



Figure 3.1: Here we see a visual representation, from [34], of the cycle consistency loss. On the left we have the cycle-consistency loss measuring the distance between $x$ and the twice transformed $\hat{x}$ and on the right is the counterpart measuring the distance from $y$ to the twice transformed $\hat{y}$.

### 3.2.1 Network Architecture

The architecture of the generators is the U-net, [27], discussed in Section (2.4.3), along with some slight modifications. The residual blocks that make up the latent space of the U-net are much greater in number compared to the DPM. The number of down-blocks determines the spatial resolution of the latent space with an increase in down-blocks corresponding to a decrease in resolution. Finally, this network does not contain any time embedding layers nor does it include attention blocks.

The exact architecture is given by:

DB64-DB128-
RES256-RES256-RES256-RES256-RES256-RES256-
UB128-UB64-
CT3

The discriminator is based on the PatchGAN architecture, [33], which produces a spatial map of predictions. Each spatial element corresponds to the realism of a local image patch which is in contrast to more traditional discriminators that output a single scalar value representing the probability that the entire image is real or fake. By performing local discrimination at the patch level, PatchGAN can capture fine-grained details and spatial structures in the images. This enables more precise feedback during training, allowing the generator to focus on producing realistic textures, patterns, and structures at a local level. PatchGAN also has fewer parameters than a traditional discriminator and can be used on arbitrarily sized images. The resolution used is $70 \times 70$ which means that the PatchGAN discriminator aims to classify whether $70 \times 70$ overlapping image patches are real or fake.

The exact architecture of the PatchGAN is given by:

DB64-DB128-DB256-DB512

## 3.3 Cigar to Round

We will demonstrate the effectiveness of the CycleGAN model by transforming images obtained from the GZ2, [4], questionnaire. We require two classes of galaxies to perform transformations between. We selected two galaxy classifications from question 7 of the GZ2 questionnaire: 'How round is it?'. There were three distinct answers for this question:

1. Completely round
2. In between
3. Cigar-shaped

We selected galaxies which fell in the first or third classifications which can be seen in Figure (3.2). We believe the second category would not be suitable for this problem since its distribution may overlap significantly with the distributions of either of the more concretely defined classifications. This set of classifications arises quite early on in the hierarchical questionnaire which means that there is a high degree of confidence in these classifications since higher numbers of voters exist for classifications higher up in the questionnaire. The dataset is naturally unpaired since there is no possible way of getting a ground truth for what a specific galaxy would look like if it existed in a different morphology. Thus, no quantitative analysis can be conducted; the outputs will merely be investigated on qualitative aspects.

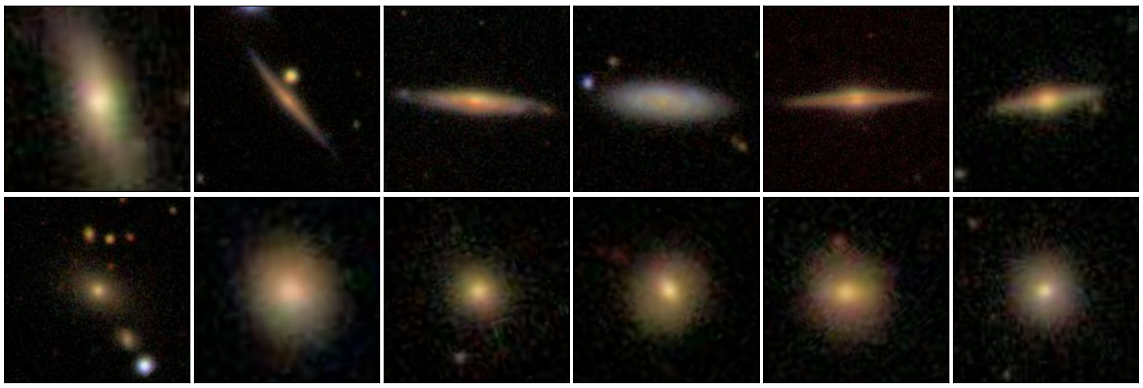


Figure 3.2: Samples from the training data. The top row gives examples of cigar-shaped galaxies while the bottom row shows round galaxies.

The first generator, $G$, will be mapping cigar-shaped galaxies to corresponding round galaxies while $F$ will be learning the inverse mapping from round galaxies to corresponding cigar-shaped galaxies. We selected around 2000 galaxies for each distinct classification and extracted the data using the SDSS image cut out service in the same manner as mentioned in Section (2.2).

As can be seen in Figure (3.3), the model does a fairly decent job of converting round galaxies into cigar-shaped galaxies and vice versa. Other qualities such as orientation, location, and color were preserved indicating that the network successfully isolated the property that distinguished the two domains which is the shape of the galaxy.

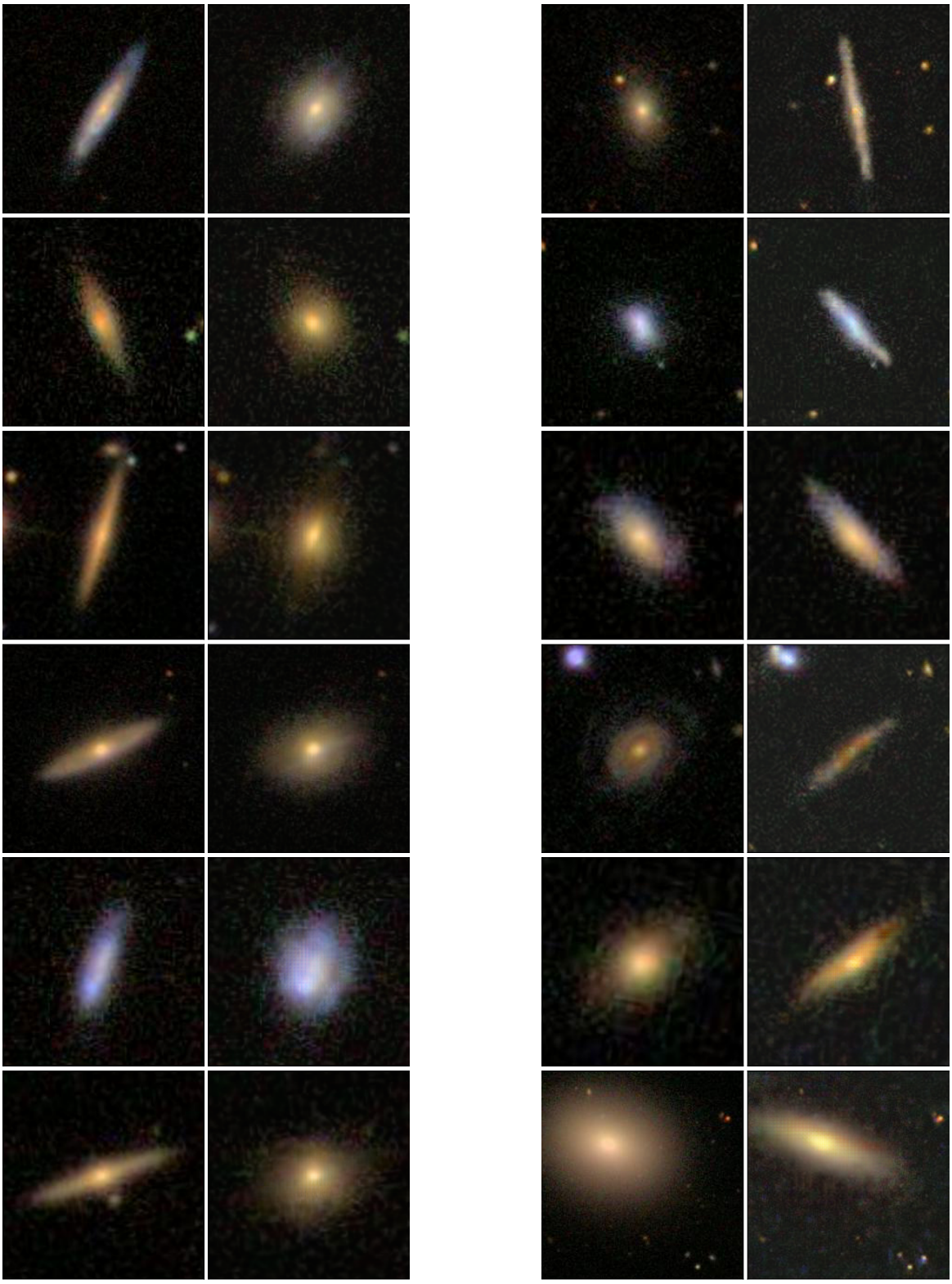


Figure 3.3: The first column contains cigar-images from the validation set while the second column shows the transformation to corresponding round images applied by $G$. Similarly, the third and fourth columns show round images as well as the corresponding cigar-shaped images generated by $F$. Note the networks' ability to preserve all other features besides the degree of roundness; the semantically meaningful mappings are courtesy of the Cycle consistency loss.

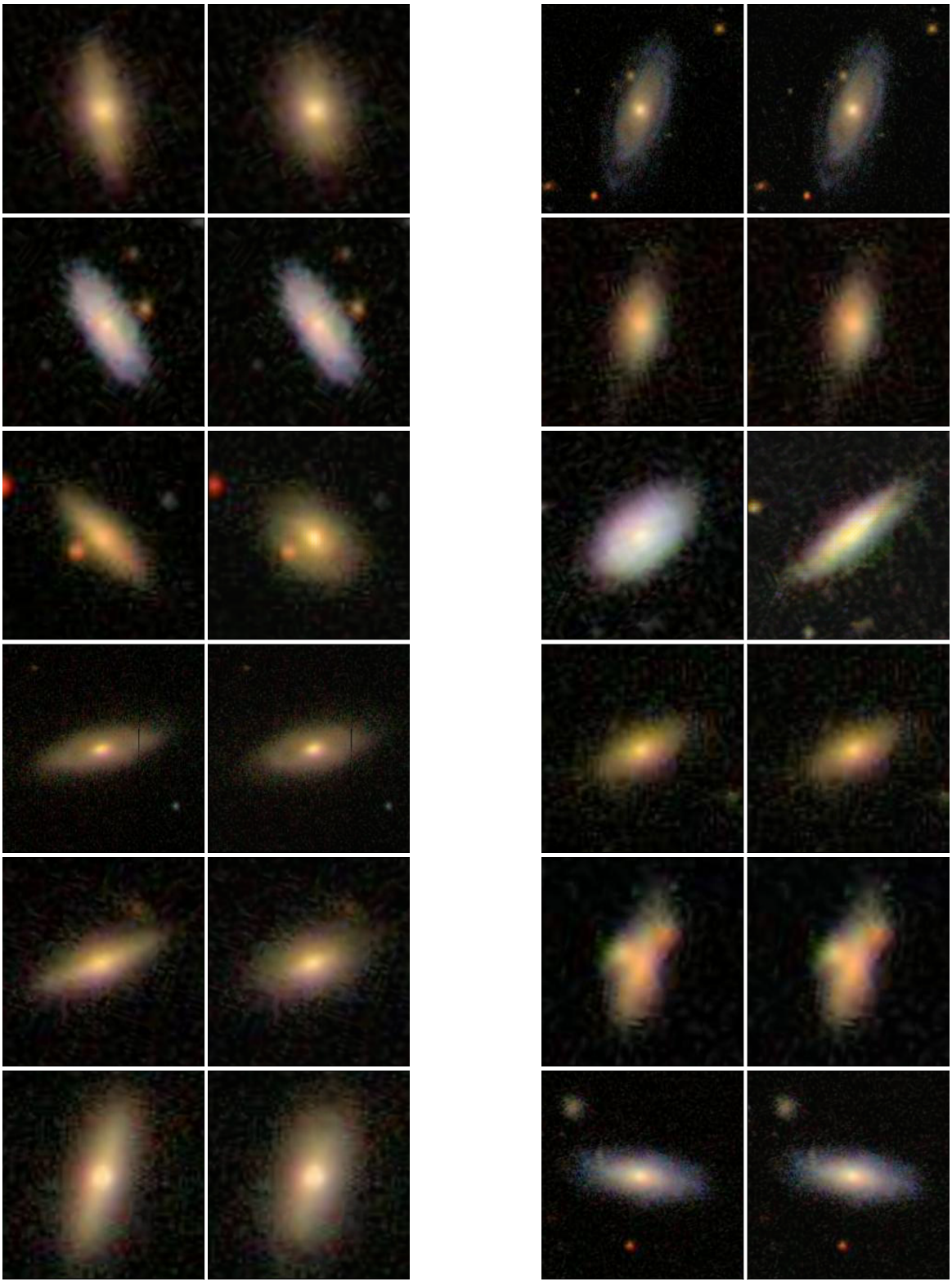


Figure 3.4: The layout of the columns is identical to Figure (3.3) where the first pair of columns are for $G$ and the second pair is for $F$. We observe no transformation applied whatsoever to most of these edge case images. The identity function is doing its job but that means that both networks require input data from regions in the input distributions that do not overlap with each other.

There are however examples, Figure (3.4), of the network failing to translate the images satisfactorily. The thing to note about these instances is that the input images could be considered edge cases in that in their shape is not strictly round or cigar-shaped but can be said to lie somewhere in the middle of these two extremes. In fact, these images were sampled from galaxies that were voted into the 'in between' classification, which was not used during training, and our prediction of this class yielding mixed results is confirmed. The reason our network does not translate these images well is due to the identity loss. The identity loss forces the network to collapse to identity if and when it receives an input that already belongs to the domain of the target distribution. In other words, if $F$, which is meant to produce cigar-shaped galaxies, receives as input a galaxy that it deems to be cigar-shaped then no translation will occur. Therefore, our CycleGAN is limited to galaxies with somewhat strict classifications.

## 3.4 Band Transfer

The task we now turn our attention to is that of cross-photometric band image mapping. A photometric band is an optical filter with a known sensitivity for a particular section of the electromagnetic spectrum. For a deep dive into astronomical photometry, see [38]. Each band has an effective wavelength and is designated by a particular letter; while the letters are not standards, they are recognized by astronomers by agreement and are a matter of convention. SDSS, for example, captures data in the following bands: U, G, R, I, and Z with effective wavelengths of $[360, 470, 620, 800, 900]$ nanometers (nm) respectively, see [17] for more details on SDSS. Table (3.1) gives a comprehensive, though not exhaustive, list of bands sorted in ascending wavelength.

The need for collecting data at different bands arises from the fact that each band provides information about different characteristics of a particular galaxy, see Figure (3.5). For instance, ultraviolet (UV) bands reveal the emission from larger, younger stars which indicates locations of ongoing star formation while longer-wavelength bands in the infrared (IR) range trace the majority of the stellar mass (comprised of older, cooler stars) more efficiently and exhibit less band-to-band variation, see [39]. Specifically, the arms of spiral galaxies are areas of star formation and therefore emit mostly in the UV range while the central bulge of spiral galaxies are made up

| Filter | $\lambda_{\text{eff}}$ (nm) | Description | Peak Thermal Emission (K) |
|---|---|---|---|
| FUV | 150 | Far Ultraviolet(UV) | 30,000 |
| NUV | 230 | Near UV | 12,500 |
| U | 360 | UV | 8,000 |
| B | 445 | Blue | 6,500 |
| G | 470 | Green | 6,150 |
| V | 550 | Visual | 5,200 |
| R | 620 | Red | 4,600 |
| I | 800 | Infrared | 3,600 |
| Z | 900 | - | 3,200 |
| H | 1,630 | - | 1,800 |
| K | 2,200 | - | 1,300 |
| M | 4,750 | Mid-Infrared | 600 |
| N | 8,800 | - | 330 |

Table 3.1: This table presents a selection of photometeric bands. The first columns gives the band name, the second gives the effective wavelength of the band and the third column gives a description of the abbreviated name of the band name (if it exists). The last column gives the peak thermal emission of a body emitting radiation at the given effective wavelength; the reason for including this is to show how sensitive each band is to objects of certain temperatures. For example, UV bands are sensitive to radiation from hotter, younger stars while Infrared bands are more sensitive to older cool stars.

of mostly older stars that emit in the IR range. Elliptical galaxies have little to no structure and are comprised of mostly older stars that emit in the IR range, see [15] for more details.

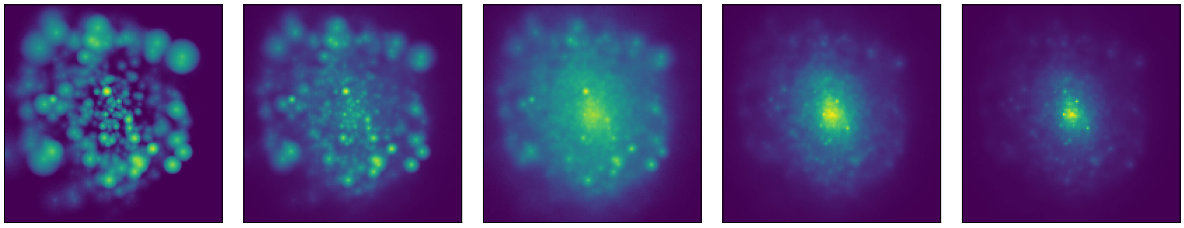


Figure 3.5: From the right to the left, we have images of a single galaxy taken in bands FUV, U, R, K and M. Note the following features: the spiral arms contain most of the star forming and are mostly visible in the UV bands while the central bulge, containing older stars, is mostly visible in the IR bands.

The stellar mass and the star formation rate (SFR) are among the most critical features for understanding the present state of galaxies, their history, and future evolution. IR photometry is widely used to measure stellar mass since IR traces the emission of the majority of the stellar population. However, IR cannot trace the emissions of younger stars directly but may only do so indirectly by tracing dust emissions powered by star-forming activity. For this reason, IR-based SFR estimation leads to a significant underestimation of the true SFR in dust-deficient galaxies. Therefore, observing UV emissions as a direct indication of SFR is preferable to IR photometry. The problem with obtaining UV observations is a phenomenon referred to as astronomical extinction. For more details on stellar mass and SFR tracing as well as astronomical extinction, see [39].

In summary, astronomical extinction refers to the absorption and scattering of light by interstellar gas and dust grains, as it travels through space, causing a reduction in the observed intensity of light from celestial objects. Extinction poses a significant challenge to astronomers as it can obscure or distort the appearance of distant objects and affect measurements of their properties, including their brightness, colors, and spectral features. This phenomenon affects the wavelengths of light differently, with shorter wavelengths (such as UV and blue light) being more strongly absorbed than longer wavelengths (such as IR and red light). As a result, the observed UV flux may be significantly attenuated, compared to IR, leading to underestimation of the SFR. For this reason, using UV to trace SFR requires precise measurements of extinction. However, obtaining precise measurements for extinction requires accurate knowledge of the dust distribution, which can be challenging to determine, especially in complex or dusty environments.

It is for these reasons that multiple observations in different wavelengths are required for accurate estimates of stellar mass and SFR, again see [39]. Observations made in multiple ranges of wavelengths require fewer inferences and lead to stronger conclusions being drawn about the observed galaxies. Unfortunately, most surveys cover a limited range of wavelengths at sufficiently high quality for effective analysis, making feature extraction from particular bands impossible in certain regions of the sky.

Image-to-image models may be able to generate synthetic images of galaxies in a greater range of photometric bands than the input image(s). The idea here is

as follows: if we can learn the mapping between two photometric bands then this mapping may be used to augment existing observations by generating synthetic images in bands not previously observed. An example of such a transformation would be generating synthetic UV observations based on IR observations or vice versa.

A successful mapping can be used to augment survey datasets substantially and may provide benefit when studying the properties of galaxies in a specific region that has not yet been covered by high-quality surveys or that lacks sufficient band coverage. In addition to this, the trained model would have a very low computational cost for generating inferences; therefore, large batches of synthetic images may be generated with minimal computational expense.

Therefore, we seek a dataset that in addition to the criteria mentioned in chapter 2, provides us with images of galaxies taken in a sufficiently varied number of photometric bands. If the bands are not sufficiently distant from each other, then the difference observed between the corresponding images may not justify the trouble of learning the mapping between the bands; such a mapping may end up collapsing to identity anyway. For instance, translating between band G and band B would provide little to no added benefits in terms of understanding galaxy properties and morphologies.

It is for this reason that most sky surveys, our best resource for obtaining images of galaxies, will not suffice for our purposes. The most significant surveys include SDSS, the Panoramic Survey Telescope and Rapid Response System (Pan-STARRS), and the Dark Energy Survey (DES), [17], [20] and [18] respectively. None of which individually covers a wide enough range of wavelengths for our purposes. For this reason, getting a paired dataset of galaxy observations from one single source is out of the question.

Fortunately, image-to-image techniques exist for unpaired data but even getting our hands on an unpaired dataset requires coordinating and integrating data from various surveys which has proven to be quite complex and time-consuming. In addition to the challenge of obtaining such data, preprocessing data obtained from different surveys involves correcting for instrumental effects and accounting for varying background levels and noise characteristics. This requires a sophisticated image processing pipeline which requires a high level of domain knowledge of astronomical data processing.

These challenges made us question if obtaining observed data was necessary or if

we could seek other resources. Besides observations taken by sky surveys, simulations may provide us with training data that sufficiently encodes the cross-band mapping we seek to learn. If the cross-band mapping for simulated data matches the mapping for observed data, then we may leverage simulated data to train our model for the purpose of subsequently generating inferences based on observed data. In the next section, we will investigate one such simulation which we believe provides an adequate training set for our model.

## 3.5 Illustris Dataset

The Illustris simulations[1], [40], are a set of large-scale cosmological simulations along with galaxy formation simulations. For a complete accounting of the details in this section, please reference [41]. The goal of cosmological simulations is to test our current understanding of cosmology. Currently, the most favored model is the Lambda Cold Dark Matter, see [42], which contains three major components: ordinary matter (Bayrons), cold dark matter, and dark energy. It is the simplest model that provides a reasonably accurate accounting of the large-scale structure of the universe, observed in the distribution of galaxies, as well as the accelerating expansion of the universe. For this reason, it is referred to as the standard cosmological model. The model posits that the mass-energy density of the Universe is dominated by unknown forms of dark matter and dark energy.

Testing this model requires precise predictions on the formation of the Cosmic Web of sheets filaments and voids, inside which visible matter is embedded in the basic units of cosmic structure - galaxies. Cosmological simulations utilizing different models can be run to produce a simulated universe. These simulations are then compared to observed galaxy populations to test the underlying model; the utility of the underlying model can be measured by the degree of concordance between the simulation and real observations. To draw any conclusions from any such comparisons, we require simulated galaxies to be as detailed and realistic as possible.

Previous to Illustris, cosmological simulations had successfully recreated the large-scale cosmic web of galaxies seen in the universe but have unfortunately failed to recreate a mixed population of elliptical and spiral galaxies; this severely limited their

[1] https://www.illustris-project.org/about/

utility as a means to directly connect with observations. These previous cosmological simulations have been run to track the combined evolution of dark matter and dark energy and have only included the force of gravity. These dark matter-only (DM-only) simulations are limited by their inability to predict the distribution of galaxies made up of baryonic matter.

Illustris is set apart from previous attempts in its ability to directly account for baryonic matter by calculating hydrodynamics in addition to gravity. Since these simulations can follow the dynamics of both the dark matter and baryons, predictions can be made about the internal structure of galaxies. The Illustris simulations were run using the moving mesh code AREPO, [43], which, in addition to gravity and hydrodynamics includes comprehensive physics and feedback modules allowing for radiative gas cooling, heating, and ionization by a UV background, star formation with associated feedback and mass and metal return to the interstellar medium from aging stellar populations.

As concerns the image data generation, mock observations were created using stellar population synthesis models which assigned spectral energy distributions (SED) to each star particle in the galaxies. Stellar light is then spread using adaptive light spreading based on the local density of star particles. Images are created by projecting the stellar light using pinhole cameras located at various viewing angles. The spectra are then convolved with various broadband filters to create an equal number of broadband images for each galaxy. All details on the creation of the mock observation catalogue can be found in [44].

Before we proceed with these mock observations, we must ask how realistic the synthetic images are before utilizing data from this cosmological simulation. The generated images must match images captured in sky surveys. After all, we will train our model on simulated data for, eventually, running it on observed data. If our model is to perform as well on observed data as it does on simulated data, then the data distribution of the simulated galaxies has to match the distribution of observed galaxies. If the gap between both distributions is too large, then the model may well learn to navigate the distribution of simulated galaxies but will fail to generalize to observed data. In this case, then our model would be trained on unrealistic data and would be of no benefit as far as augmenting real observed data. Since there is no benefit to augmenting the Illustris data set, any successes we have with the mock

observations would be a theoretical proof of concept with little to no applicability. So, are the simulated galaxies realistic enough?

There are a few metrics that indicate high agreement between simulated data and observed data. Illustris successfully reproduces several observable properties as well as relationships between these properties. Firstly Illustris matches the present-day ratio of the quantity of stars to dark matter mass for galaxies of all masses. Additionally, Illustris matches the total amount of star formation in the universe as a function of time. These two properties are well constrained by observations and Illustris is constructed to agree with both properties. Furthermore, the simulation is observed to match how many galaxies of each mass and brightness exist at the present day as well as back in time and is also able to match the observed star formation rates. Illustris also precisely measures the gas composition of the universe as well as its location or in other words: the simulated galaxies' chemical composition is in agreement with observations. The simulation is also able to correctly predict that the majority of gas remains in intergalactic medium but that this gas contains only a minority of metals produced in the universe so far. Agreement across all of those metrics allows us to say with a fair degree of confidence that the Illustris-generated data distribution should be able to match a data distribution drawn from observed data.

### 3.5.1 Data processing

One advantage of dealing with simulated data such as Illustris, is that we have meta-data about each Galaxy. Metadata features include stellar mass, SFR, metallicity, and spin. Of those features, we have found the one feature that has the largest effect on band-to-band variation to be the star formation rate.

In Figure (3.6), we observe much higher variation across different bands with galaxies with higher SFRs compared to ones with lower SFRs. As we have mentioned previously, galaxies with a high SFR emit high levels of UV which is evident by the isolated clusters and spirals observed in the UV images of high star formation rate galaxies. These regions indicate areas of high stellar formation and are vitally important to account for when making inferences on the SFR of observed galaxies. Again, see [39] for further details.

Illustris has generated images for nearly 7000 galaxies, see [44], the IDs of these

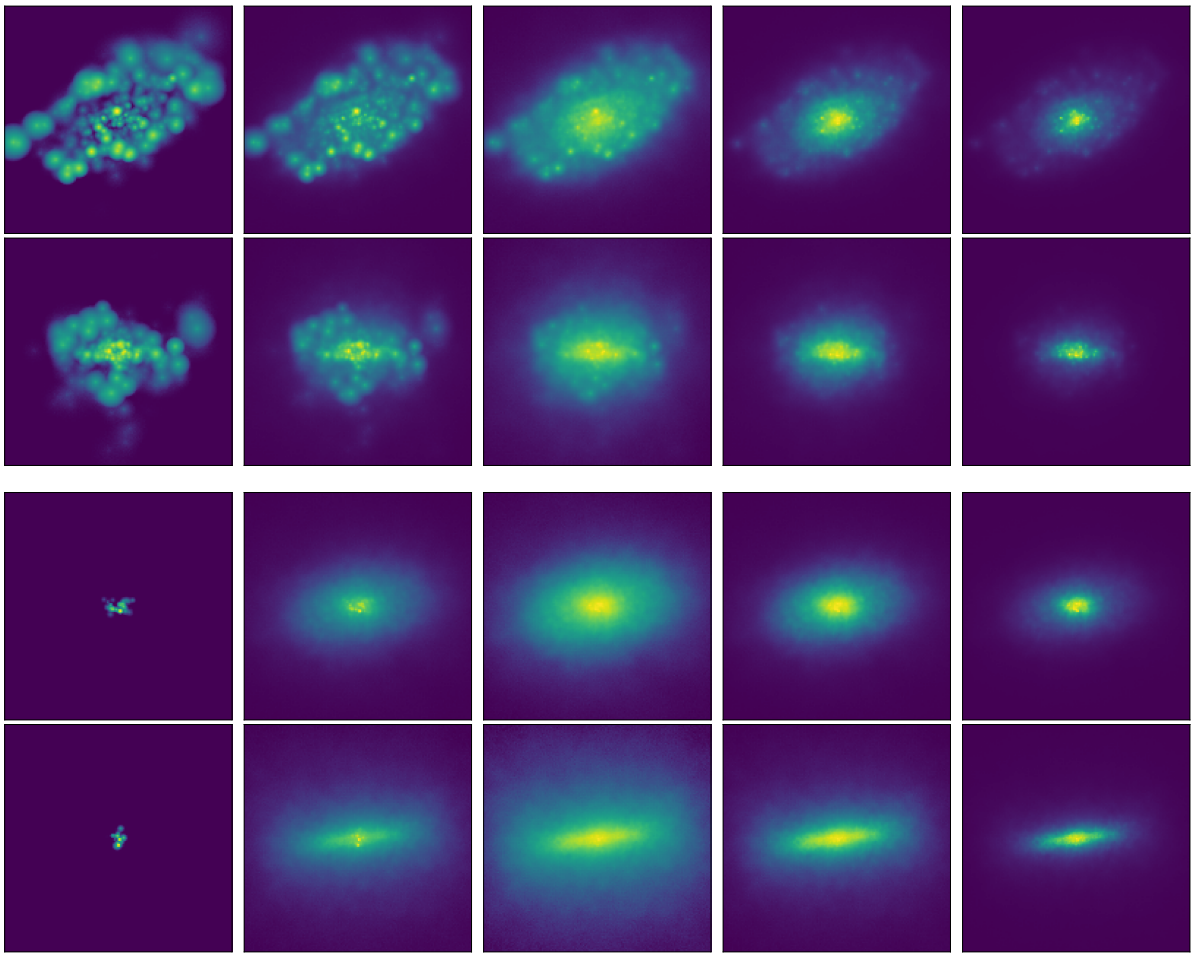


Figure 3.6: The above plot shows the progression of Galaxy observations from UV to IR. The top two rows show galaxies with high SFRs while the bottom two rows show galaxies with low SFRs. Note, the large variability between bands in galaxies with high SFRs compared to the mostly uniform observations of the low SFR galaxies.

galaxies were obtained using the web API found on the Illustris website. We sorted the galaxies based on total stellar mass and then selected the top 3000 galaxies in terms of stellar mass; the reason for this selection was that galaxies with higher stellar masses had a larger field of view and therefore had high-frequency features such as spirals resolved better compared to galaxies with lower stellar masses.

The data for each Galaxy was then downloaded for the specific bands that we wished to perform mappings to and from. We will discuss band selection in the next sections. The original images had resolutions of $256 \times 256$ however we have cropped

the images to $128 \times 128$ since we found the the majority of the galaxies fit within that central selection of $128 \times 128$; in other words, we discarded the background of the image.

We selected 1000 galaxies of the 3000 galaxies we had downloaded by stratified sampling. The stratification was done based on SFR; we felt it necessary to do so since the SFR had the largest impact on Galaxy morphology and we did not want our model to be biased in terms of it's performance with galaxies of different morphologies or structures. This stratified sampling was done once; all subsequent results in this paper, regardless of different band combinations, were obtained with models trained on the same stratified sample.

The data was processed in the following way: each image was loaded and then scaled using $asinh$ scaling, which is standard practice in astronomical image processing (see [45]), and normalized to a range of $[-1, 1]$. It is important to note that each Galaxy was captured from four different viewing angles which means that we have four sets of images for each Galaxy. The different cameras are notated by $Cam0$, $Cam1$, $Cam2$, and $Cam3$. To test our models' ability to generalize to data not included in the training set we will use $Cam0$ and $Cam2$ for training and reserve $Cam1$ and $Cam3$ for validation in all subsequent sections. This means if our training set is comprised of 1000 sampled galaxies, then it contains 2000 images.

The validation data set is comprised of 250 galaxies, which due to the two different viewing angles yields 500 images. The validation data set has also been stratified by SFR and is identical across all subsequent sections.

## 3.6 Model Selection

### 3.6.1 Performance Metrics

One of the main advantages of working with the Illustris dataset, besides ease of access, is the existence of ground truth labels. To fully take advantage of the ground truth labels, we must have a reliable metric that can provide an intelligent measure of distance between the generated images and the corresponding ground truth labels. Metrics such as the mean squared error or the mean absolute error while providing a general measure of distance are not particularly well suited to detecting subtle nuances

in how images may differ.

The structural similarity index or SSIM, see [46], is a metric that is much better suited for our purposes. It accepts as input two images and outputs a number in $[-1, 1]$ where 1 indicates perfect agreement, $-1$ indicates perfect disagreement and 0 indicates no relationship whatsoever. It is a much better predictor of distance than the other metrics like the mean squared error because it takes into account structural differences by measuring the variance and covariance between individual image patches; this is in contrast to metrics such as the mean squared error which only measure the difference in magnitude of individual pixels.

In the example visualized in Figure (3.7), both images have been constructed such that their mean squared error is equal, this is done by setting all pixels to have a value of 1 and changing an equal number of pixels to a value of 0.5. However, the way the altered pixels are distributed across each image differs. The SSIM is sensitive to such differences in structure while the mean squared error is only concerned with the mean pixel difference and has no regard for the structure or arrangement of the pixels. This renders the MSE not very helpful as far as assessing image data. The SSIM is a much smarter metric that will serve as a more accurate indication of model performance by being a better measure of agreement between the output image and the ground truth.

We will be measuring performance by looking at the a metric we will refer to as SSIM Gain which is given by

$$\text{SSIM}(\text{Output}, \text{Target}) - \text{SSIM}(\text{Input}, \text{Target}), \tag{3.5}$$

where the first term is the SSIM between the output image and the target image and the second term is the SSIM between the input image and the target image. The reasoning for this metric is as follows, the second term offers a baseline measure of distance between the input and target bands while the first term gives the distance between the output band and the target band (it is this difference that our model seeks to minimize). Looking exclusively at the SSIM taken between the output and target images as a measure of model performance would give a very crude measurement that lacks context. Considering how relative the model's performance is compared to the initial difference between input and target images provides a more comprehensive measure of success. We desire $\text{SSIM}(\text{Output}, \text{Target}) > \text{SSIM}(\text{Input}, \text{Target})$ since this

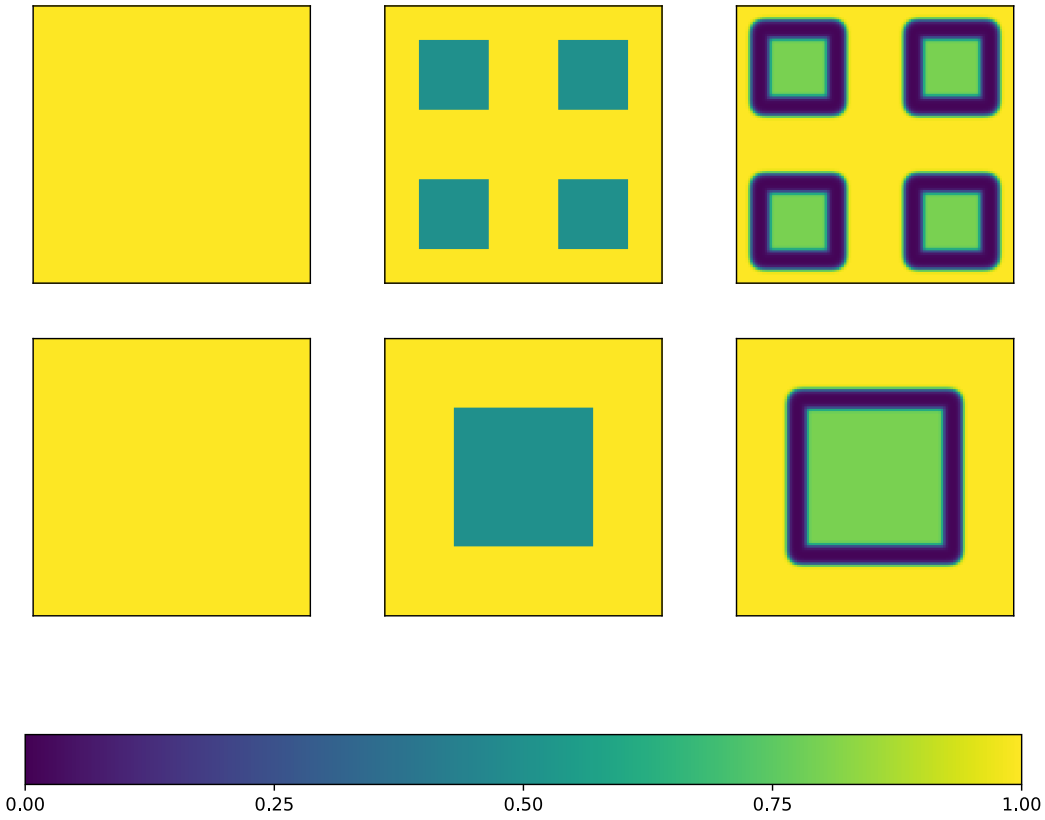



Figure 3.7: Each row pertains to a single example used to demonstrate the SSIM. The first column represents the first set input images; those are just identity. The second columns represent the second set of input images; to construct these images, we started with identity then modified an equal number of pixels in each image but with different arrangements. The final column shows the SSIM map, which is averaged for the final calculation, for each pair of images. Note the SSIM's ability to account for structure.

indicates that our model has successfully transformed the input image into an image that is smaller in distance to the target image compared to the initial distance of the input image. Thus, a positive SSIM Gain indicates satisfactory model performance, a negative SSIM Gain indicates undesirable performance while a score of 0 indicates that the model did not alter the image whatsoever.

### 3.6.2 Hyperparameter Tuning

Hyperparameters are parameters that control the configuration of the model and are not learned during training; hyperparameter tuning involves training different models with varying hyperparameter configurations to determine which configuration works best for the task at hand. For a detailed dive into hyperparameter tuning, see [47]. The authors of the original CycleGAN paper, [34], proposed that for images of size $128 \times 128$, that the generator architecture be comprised of 2 down-sampling blocks followed by a latent space containing 6 residual blocks and 2 upsampling blocks as in:

DB64-DB128-
6× RES256
UB128-UB64-
CT1

which, save for the output layer having only one channel instead of three, is identical to the model used in Section (3.3). To determine if this particular hyperparameter configuration was ideal for our specific purpose we ran nine models with various hyperparameter adjustments. The two dimensions across which we varied hyperparameters are as follows:

1. The number of downsampling and upsampling blocks; this number must be equal for both sets of blocks since we desire the output to have the same resolution as the input. We will refer to this as the latent scaling factor; each sampling block reduces the latent space resolution by a factor of 2. So, a latent scaling factor of 2, with a latent dimension of $64 \times 64$, is obtained with one downsampling block. A latent scaling factor of 4, with a latent dimension of $32 \times 32$, is obtained with two downsampling blocks and so on. The original paper proposed a latent scaling factor of 4 so we tried scaling factors of $\{2, 4, 8\}$.

2. The number of residual blocks. The original paper proposed 6 blocks, so we tried $\{3, 6, 9\}$ for each of the three different latent scaling factors giving us a total of 9 different hyperparameter configurations.

All other hyperparameters and aspects of training were kept equal across the 9 models as was the training dataset; the number of residual and sampling blocks was

changed but the contents of the blocks themselves were identical across all models. For this task of hyperparameter tuning, we have decided to train our models to map to and from bands U and K. This means that $G$ will be learning to map observations from band U to band K and $F$ will be doing the inverse mapping from K to U. We have selected these two bands as they are sufficiently distant from each other and are sufficiently varied to pose a decent challenge for our model. While such a mapping from UV (U) to IR (K) and back is somewhat extreme, for hyperparameter tuning it is better to have a challenging problem such that we allow sufficient room for the models' performance to improve with different hyperparameters. If the problem presented to the models is too trivial, then the model may well perform optimally with any hyperparameter configuration and we would learn very little as to the relationship between the hyperparameter space and model performance.

Each model was trained on the training set for 200 epochs with checkpoints saving each model's weights every 4 epochs. We logged the losses of each generator for all checkpoints and selected $G$ and $F$ that each had the lowest loss amongst all checkpoints. The selected models were then loaded, and the validation set was run on both $G$ and $F$ with the SSIM between output images and target images logged for each image in the validation set. This was repeated identically for all hyperparameter configurations. The mean SSIM Gain taken over the validation set for each hyperparameter configuration is shown in Tables (3.2) and (3.3) for $G$ and $F$ respectively.

| | **Residual Blocks** | | |
|---|---|---|---|
| **Latent Scaling Factor** | **3** | **6** | **9** |
| **2** | 0.026011 | 0.038999 | 0.039292 |
| **4** | 0.018069 | 0.015956 | 0.020942 |
| **8** | 0.010781 | 0.005545 | 0.007753 |

Table 3.2: The mean SSIM Gain returned by the various $G$ models on the validation set. We observe a trend of increased performance as the number of residual blocks is increased and the latent scaling factor is decreased.

We also plotted the distributions, see Figures (3.8) and (3.9), of the SSIM Gain returned over the validation set for the each hyperparameter configuration.

We observe a noticeable trend across the latent scaling factor and a slightly more subtle trend with the number of residual blocks. There is a significant decrease in performance as the latent scaling factor is increased with a latent scaling factor of two (only one down-block before the latent layers) performing best across all residual

| Latent Scaling Factor | Residual Blocks | | |
|---|---|---|---|
| | **3** | **6** | **9** |
| **2** | -0.01559 | -0.01460 | 0.00048 |
| **4** | -0.02392 | -0.02339 | -0.02583 |
| **8** | -0.03040 | -0.04354 | -0.0392 |

Table 3.3: The mean SSIM Gain returned by the various $F$ models on the validation set. We observe the same trend, as Table (3.2), of increased performance as the number of residual blocks is increased and the latent scaling factor is decreased. However, most of the are values in the table below zero indicating that the $F$ models' overall performance is lower than the corresponding $G$ models.

block configurations. Our reasoning for this is as such: the latent scaling factor determines the size of the bottleneck similar to how the latent dimension of a VAE determines its bottleneck. A smaller bottleneck does not transmit high-frequency features, such as spirals and bars, but only transmits low-frequency features such as overall shape and orientation. Recall that the VAE struggled with recreating high-frequency features due to this bottlenecking behavior. Given that the most significant differences between both bands are mainly high-frequency features, it makes sense that the more aggressive the bottleneck, the less adept our network is in resolving these high-frequency features.

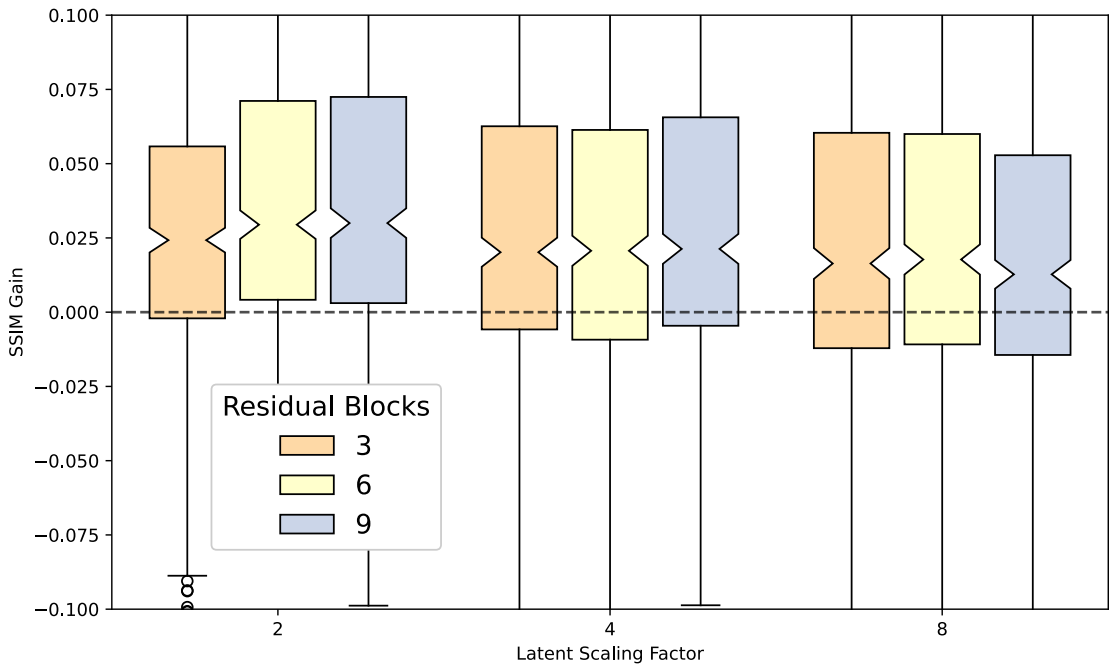



Figure 3.8: The distribution of SSIM gains for $G$ obtained from the validation set for different hyperparameter configurations. The horizontal line at 0 separates between desirable and undesirable performance in positive and negative values respectively. The top and bottom edges of the box show the 75th and 25th percentile respectively while the notch in the middle shows the 50th percentile. For example, if we find a notch at zero SSIM gain, this indicates that the given model performed satisfactorily with 50% of the validation set but unsatisfactorily for the other 50%of the validation set. The lines protruding from the boxes show the 0th and 100th percentiles while the points beyond represent outliers. Observe the noticeable trend of lower latent scaling factors having better performance as well, although less significant in effect, increased numbers of residual blocks also having better performance.

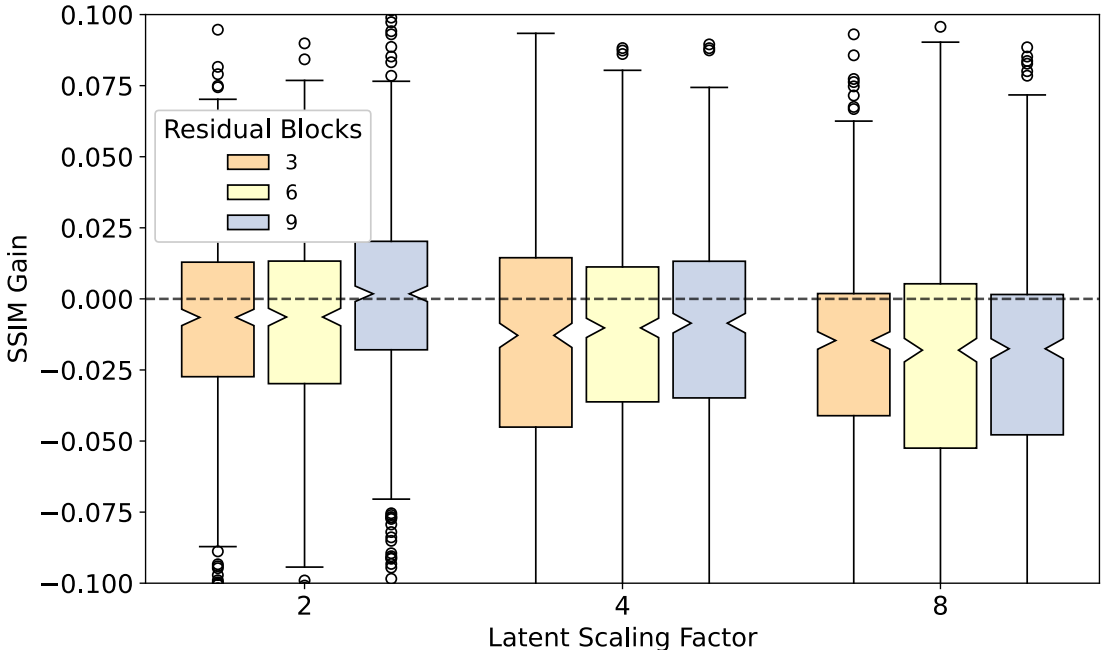



Figure 3.9: The distribution of SSIM gains for $F$ obtained from the validation set for different hyperparameter configurations. We observe the same noticeable trend, as in Figure 3.8, of models with lower latent scaling factors having better performance. The only $F$ model with that produced satisfactory performance with more than 50% of the validation set (indicated by the notch located above the 0 SSIM Gain line) is the model with a latent scaling factor of 2 and 9 residual blocks.

As for the number of residual blocks, we observe a slight increase in performance with more blocks being added. However, little to no increase in performance was observed in employing latent spaces with more than 9 residual blocks. We experimented briefly with 12 and 15 blocks and found no performance differences compared to models with 9 blocks. We also found that models with a latent scaling factor of 0 (no downsampling and upsamling blocks) performed no better than the model's with a latent scaling factor of 2 but required longer training times. Therefore, we can state that the hyperparameter configuration with a latent scaling factor of 2 along with 9 residual blocks, as in:

DB64-
9× RES128-
UB64-
CT1,

performs best. Therefore, we will use this configuration in all subsequent sections, and any results beyond this point were obtained with this hyperparameter configuration.

### SSIM Correlations

We also conducted a simple investigation into the relationship between the SSIM Gain and the variation exhibited between input and target band as measured by SSIM(Input,Target). The investigation was only conducted using the model with the best found hyperparameter configuration. The idea behind this analysis is that perhaps galaxies that had very little variation between input and target bands were more prone to producing a negative SSIM Gain since there was less room for improvement.

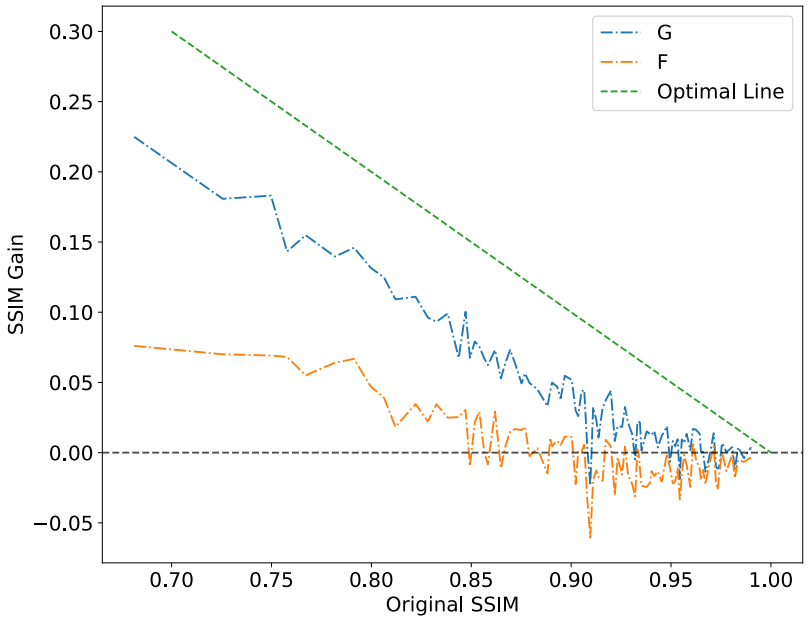



Figure 3.10: This plot shows the relationship between the models' performance as measured by the SSIM Gain and plotted on the $y-$axis, and the Original SSIM given by SSIM(Input,Target) and plotted on the $x-$axis. The green line shows the hypothetical optimal performance. While there is a slight offset for blue line, representing $G$, it's slope is nearly identical to the optimal line. This is a strong indication that model performs as intended but that it performs best with galaxies that have higher variability across different bands.The orange line, representing $F$, is farther away from the optimal line but still exhibits the same overall trend.

We found a significant negative correlation between the both measurements with $-0.86$ for G and $-0.61$ for $F$. These findings are visualized in Figure (3.10) and are a strong indication that our model performs optimally for galaxies with greater initial variation between input and target bands. This is in many ways good news since translating galaxies that have little to no variation across photometric bands is a less demanding task compared to galaxies with greater variation between photometric bands. It also gives a justification to results that may have a low or negative SSIM Gain. That justification being that there was little the model could do to change the input images to one closer to the target image as opposed to these failures being due to some inexplicable shortcoming of the model.

### 3.6.3 Supervised Training

Even though CycleGAN is designed to be trained with unpaired datasets, we can take advantage of our dataset being paired. This is one of the main advantages of working with Illustris data and provides us with a somewhat rare opportunity to turn this problem into a fully supervised problem. After all, ground truth labels are available in abundance. The following modifications were made to the original CycleGAN:

1. The discriminators were discarded since we are no longer conducting adversarial training.

2. Gone is the Cycle Consistency loss; this decouples both generators which now run independently of each other.

3. The loss function has been modified to be the mean squared error between the output image and the target image. In this way, the loss function is transformed from its original complex form, Equation(3.4), to the simplest form of supervised lossv seen in Equations (3.6) and (3.7) .

The new losses, which are minimized during training, for for $G$ and $F$ respectively are

$$\mathcal{L}_G = (\mathbf{y} - G(\mathbf{x}))^2, \tag{3.6}$$

and

$$\mathcal{L}_F = (\mathbf{x} - F(\mathbf{y}))^2, \tag{3.7}$$

where $\mathbf{x}$ and $\mathbf{y}$ are paired observations in different bands.

We trained the supervised model with the optimal hyperparameter configuration discussed previously and conducted analysis using the validation data set exactly as in the previous subsection. We compare this supervised model with the performance of the traditional CycleGAN with the optimal hyperparameter configuration and observe that the fully supervised model performs significantly better for both $G$ and $F$, see Figure (3.11). The cycle consistency loss could have been retained with the adversarial loss simply being replaced by the fully supervised objective. Perhaps this configuration would maintain the benefits of full supervision while observing the cycle consistency property.

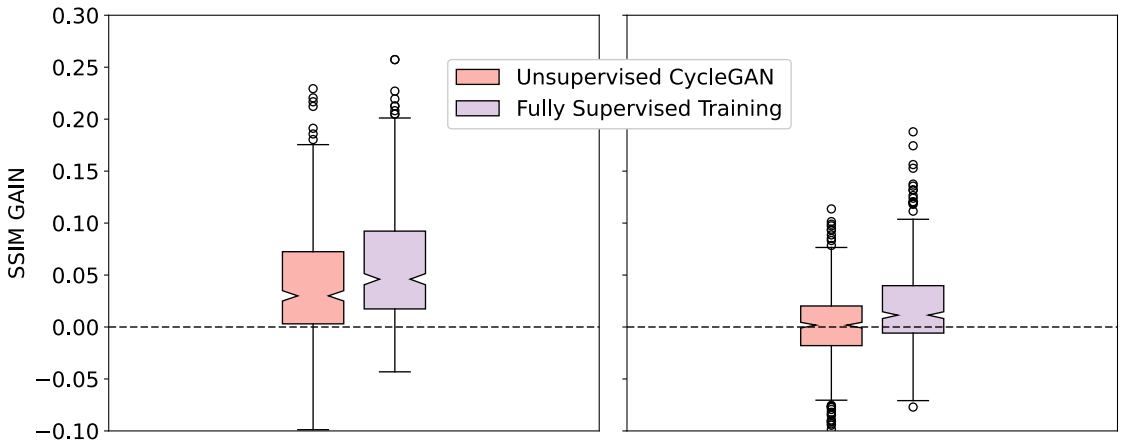



Figure 3.11: This pair of box plots, $G$ on the left and $F$ on the right, compares the performance of fully supervised training to the unsupervised training routine of the original CycleGAN model. We observe a noticeable increase in performance using full supervision to train the models. All models used the optimal hyperparameter configuration discussed previously.

### 3.6.4 Ultraviolet to Infrared Results

What we expect in the $G : U \to K$ transformation is the following:

1. Attenuation of the spiral arms since they emit mostly in UV.

2. Amplification of the central bulge that emits mostly in IR.

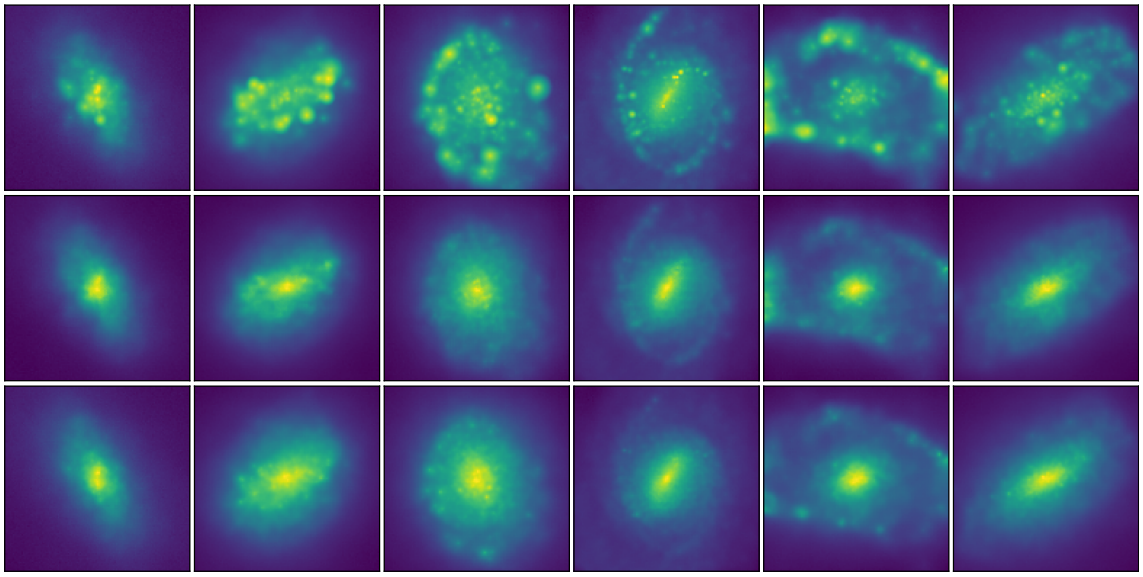


Figure 3.12: Outputs for $G$ for the fully supervised model. The top row contains the inputs (U-band), the middle row contains the mapped outputs (K-band) and the bottom row contains the ground truth labels (K-band).

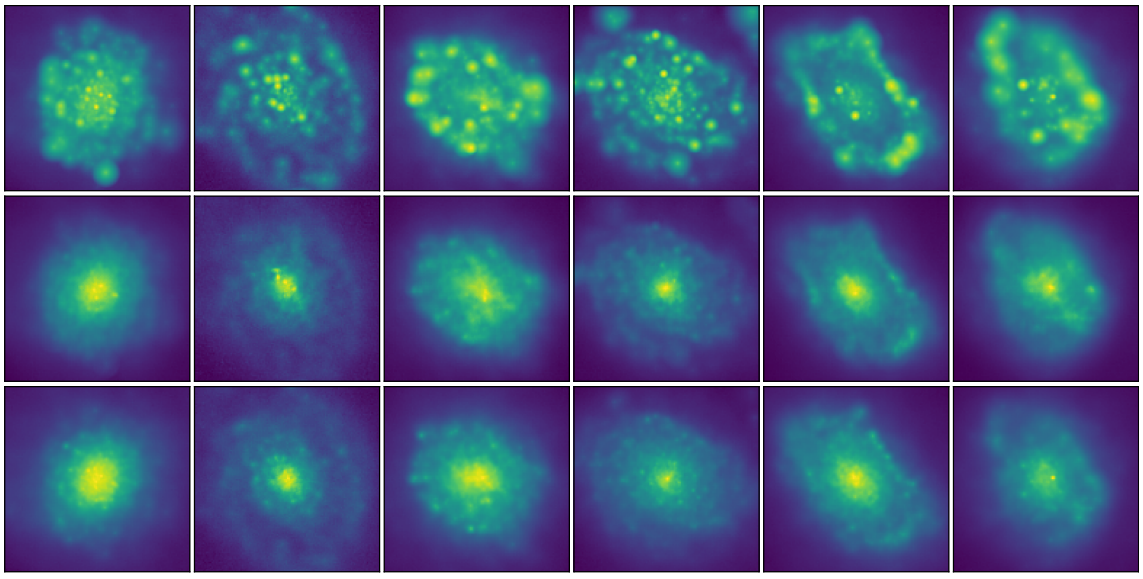


Figure 3.13: More output examples for $G$, the same layout as Figure (3.12) is used.

We observe desirable qualities in outputs of $G$. We can see in Figures (3.12) and (3.13) that the model is able to learn the correct transformations and apply them to

the input image while preserving structure. There is a fair bit of agreement between the outputs and the ground truth labels in the remaining structure of the spiral arms. There are examples of the model removing too much structure such as in the second column of Figure (3.12) but overall, the model's transformations are aligned with the physically motivated differences we expect to observe in the $U \to K$ transformation.

### 3.6.5 Infrared to Ultraviolet Results

What we expect in the $F : K \to U$ transformation is the following:

1. Attenuation of the central bulge since it emits mostly in IR.
2. Amplification of the spiral arms that emit mostly in UV.

This $K \to U$ transformation is significantly more challenging than the previous $U \to K$ transformation since the model has to infer a lot more structure in the spiral arms. This is the reason for the lower performance of $F$ compared to $G$ observed in Sections (3.6.2) and (3.6.3). The results can be seen in Figures (3.14) and (3.15).

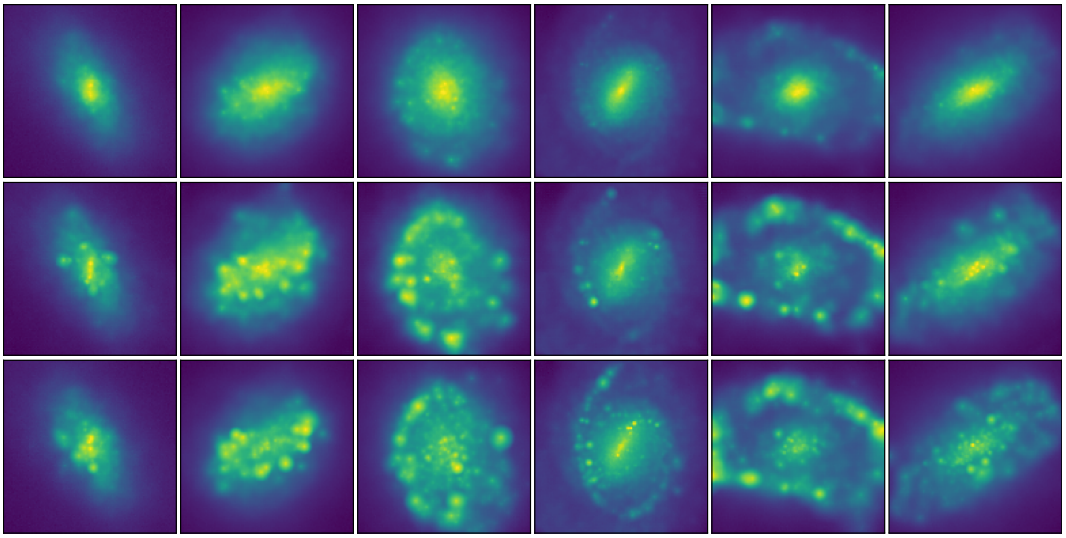


Figure 3.14: Outputs for $F$ for the fully supervised model. The top row contains the inputs (K-band), the middle row contains the mapped outputs (U-band) and the bottom row contains the ground truth labels (U-band).

While the overall style of the output images is in line with expectations, the model occasionally struggles with inferring the correct structure of the spiral arms. This is

especially true for input images that lack any traces of spiral arms' structure such as columns one in Figure (3.14) and columns one and five in Figure (3.15). Without any traces of structure, the model resorts to assigning areas of star formation somewhat haphazardly. This results in the lower qualitative accuracy observed in subsections 3.6.2 and 3.6.3. There are however encouraging results with input images where traces of the UV emitting structure are visible such as in columns three, four and five of Figure (3.14).

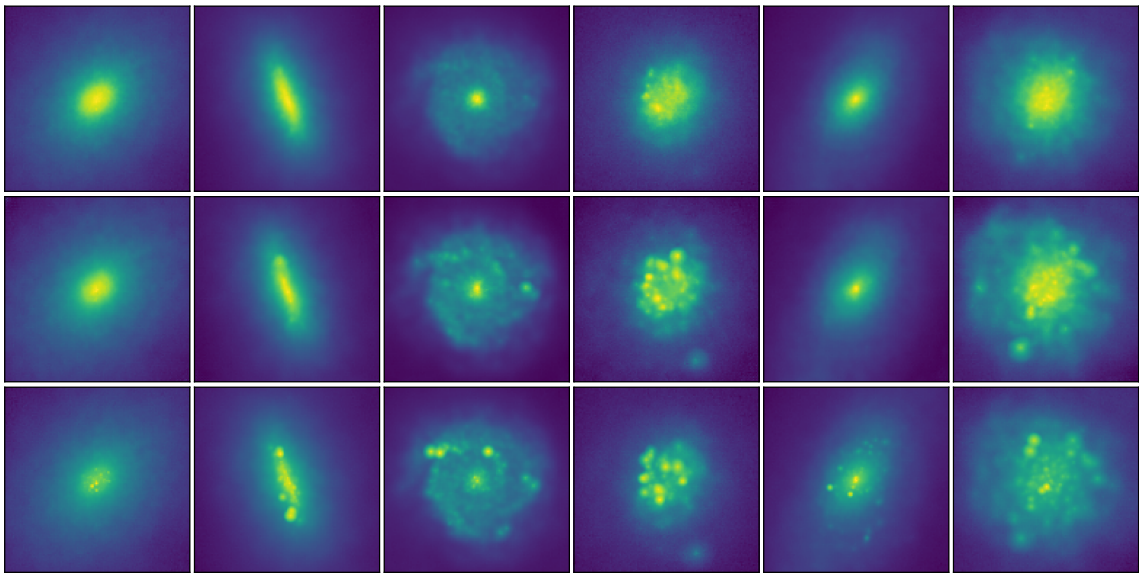


Figure 3.15: More examples for $F$, the same layout as Figure (3.14) is used.

## 3.7 Band Interpolation

In Section (3.6), we selected two bands, U and K, with a large difference in wavelength to perform hyperparameter tuning and test different training paradigms. In this section, we will investigate interpolating between two distant bands. The idea here is to generate a sequence of observations at intermediate bands. For instance, we could select bands U and K as input and train a model to map to a specific band delineated by both input bands such as bands B, G, R or Z. To clarify, we have now multiple models that each accept the same two inputs but are trained to map to a specific intermediate band. A unique model needs to be trained for each output band; we have not tried developing a single model that can output multiple bands but given the relatively low cost of making inferences with multiple models, we do not believe having one model that outputs all bands would be of any significant advantage.

We desired to experiment with bands corresponding to observations taken with GALAX and 2MASS telescopes (see [19] and [16] respectively), so we have selected the FUV as well as K bands as input, see Figure (3.16). We observe significant differences between both input bands since we have selected the bands to be even farther apart than the bands used in Section (3.6).

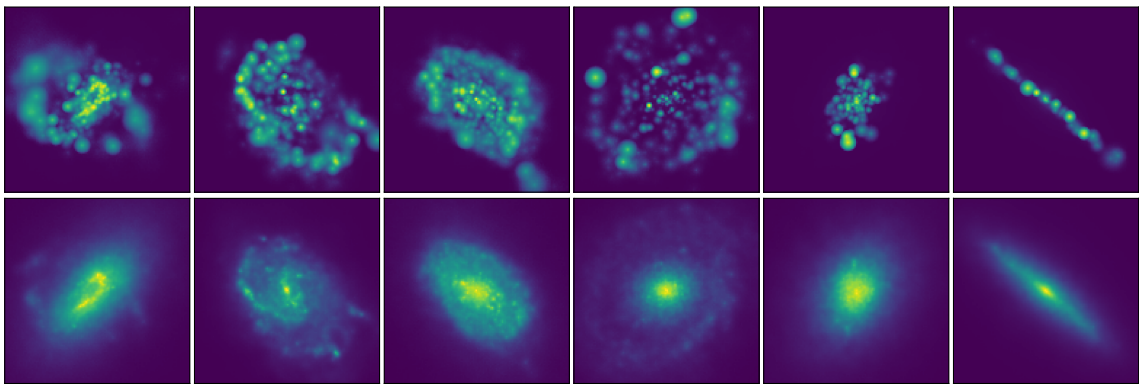


Figure 3.16: Each column shows an individual Galaxy observed in both FUV (top row) and K (bottom row) bands which represents a single input accepted by the model. Note that there is a substantial difference between both inputs since each traces particular sections of the stellar population; the FUV band observations trace the younger star population in the spiral arms, while the K band observations trace the majority of the stellar population of older, cooler stars located mostly in the central galactic bulge.

We trained three models to map to the G,R and Z bands with results for each model shown in Figures (3.17), (3.18), and (3.19) respectively.

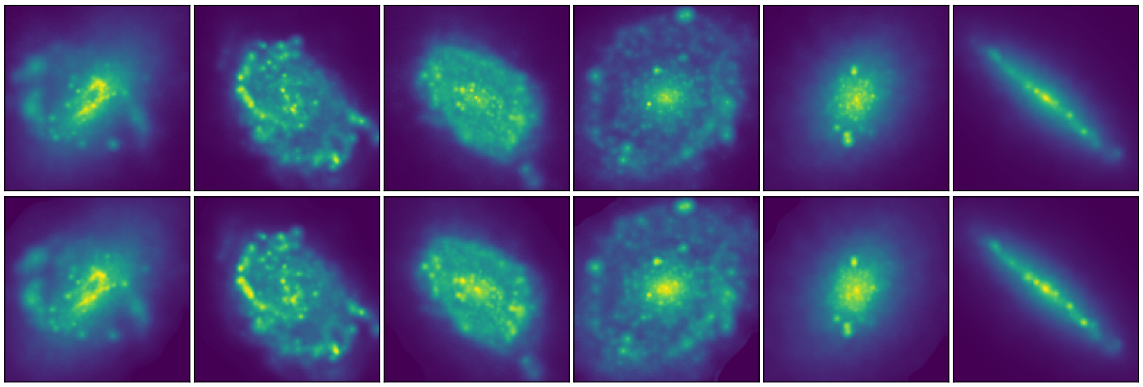


Figure 3.17: Here we show outputs of the model trained to infer G band observations from pairs of FUV and K band inputs. The top row shows the ground truth labels and the bottom row shows corresponding inferences from the model. The images are nearly identical with almost no visible differences beyond slight differences in the luminosity of some sections. The structure recreated in the inferred images is identical to the ground truth images which is very promising indeed.

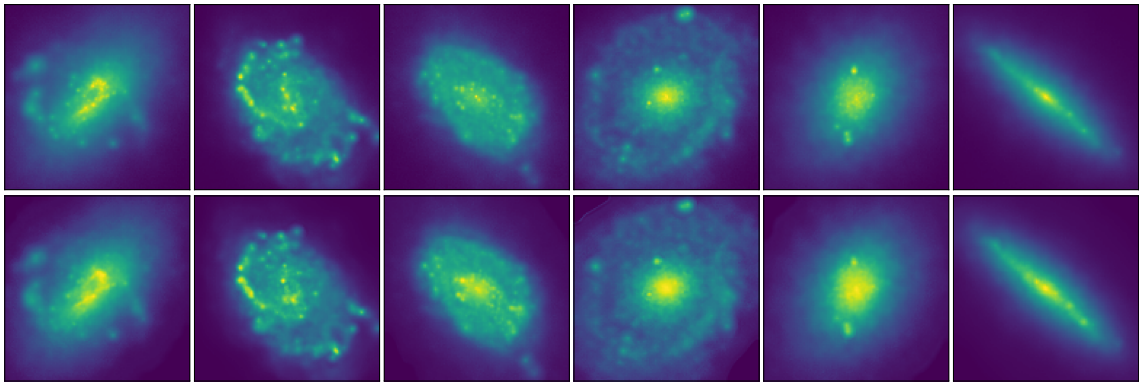


Figure 3.18: Outputs of the model trained to map to the R band. The top row is the ground truth labels and the bottom row are the model outputs. The structure is less visible than in Figure (3.17) which is to be expected with this higher wavelength band.

This sort of band interpolation yields the most promising results; the outputs are nearly identical to the ground truth labels. This is because the model has a significant advantage in being able to detect areas of high star formation from the FUV band while also having access to the distribution of the majority of the stellar matter through the K band. Leveraging information from both bands allows it to make

much more precise predictions than was previously possible with only one band's input.

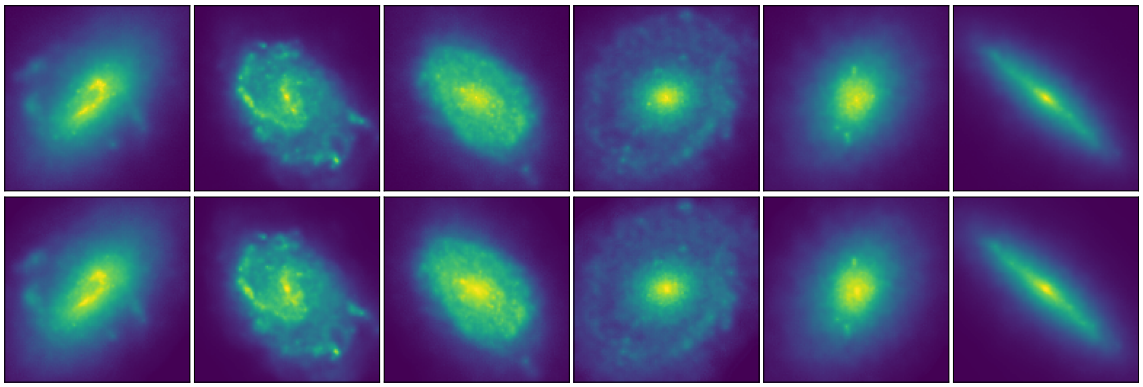


Figure 3.19: Outputs of the model trained to map to the Z band. Again, the top row is the ground truth labels and the bottom row are the model outputs. There is significantly less structure than in Figures (3.17) and (3.18) which is in line with expectations of IR observations.

Since we now require a paired dataset from potentially different instruments, the application of this type of band interpolation faces significant hurdles as regards to running inferences based on real observations. However, the performance is very promising and shows how powerful these models can be.

We selected bands G, R, and Z to map to, since SDSS, [17], collects observations in these bands. We also chose bands FUV and K as inputs since GALAX and 2MASS collect observations in those respective bands. The motivation was that a paired set of observations taken from both GALAX and 2MASS could be used to significantly augment the SDSS database. The task of obtaining and preprocessing such a paired data set is not one we have investigated but is theoretically possible and would potentially produce a high volume of high-quality images to augment SDSS observations with minimal inference cost.

Finally, we have plotted all results side-by-side to show the progression of learned bands in Figure (3.20).

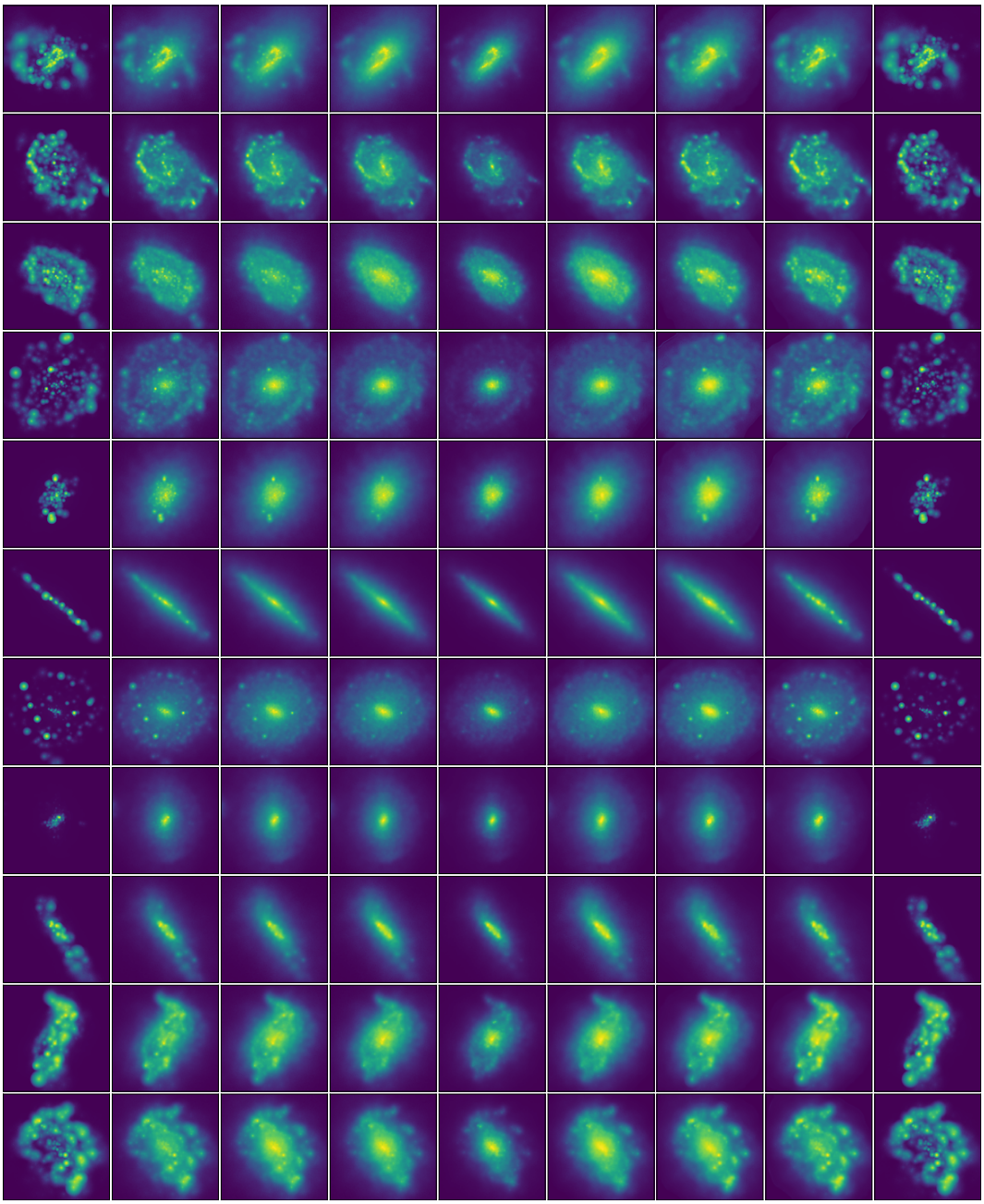


Figure 3.20: Each row represents a single galaxy observation. The first and last columns are observations of the FUV band, the 5th (middle), column is the K band observation, and the intermediate bands are plotted as and arranged sequence with Z, R and G plotted outwards from K respectively ( Z in columns 4 and 6, R in 3 and 7 and G in 2 and 8). One set of sequences (Z, R and G columns) shows ground truth labels while another set shows inferences produced by the model. The sequence on the right was generated by our models while the one on the left is of the ground truth labels.

## 3.8 Band Extrapolation

The focus of Section (3.7) was on band interpolation where an intermediate band was inferred from two surrounding bands. This section will investigate band extrapolation where a sequence of bands is extended to infer bands. Specifically, we will look extrapolating low wavelength bands since this is an inherently challenging task. The challenge presented here is twofold: firstly, the model is at disadvantage compared to the setup in Section (3.7) where it could use information from both high and low wavelength bands; extrapolation is inherently more challenging that interpolation. Secondly, the nature of low wavelength inference presents more challenges since models must infer structures such as spiral arms, bars and star forming areas.

The input bands we have selected are bands B and R; the reason for this selection is that most sky surveys cover these two bands in the visible light segment. Furthermore, there is sufficient variation between bands to encode enough information about the pattern of progression between bands to the model. Using pairs of B and R as input data, we trained two models to perform extrapolation to NUV and FUV bands. Note that there is a significant difference between the input bands and the extrapolated output bands; this makes the task further challenging for the model.

Despite the challenging nature of the inference, we note significant agreement between the model outputs and corresponding ground truth labels for both NUV inference, Figure (3.21), and FUV inference, Figure (3.22). The halos observable in the input bands are reduced to match the signatures in the ground truth labels. Additionally, the star forming regions are also correctly inferred making this a successful translation at least on qualitative terms.

As noted in Section (3.7), the challenges of interpolation include having to source the input pairs from different surveys. This challenge is not present in extrapolation. Since the two input bands are relatively close enough to each other and are in the same band of electromagnetic radiation (visible light in our case), we can source both inputs from the same survey. This makes extrapolation a much more appealing application as far as practicality is concerned.

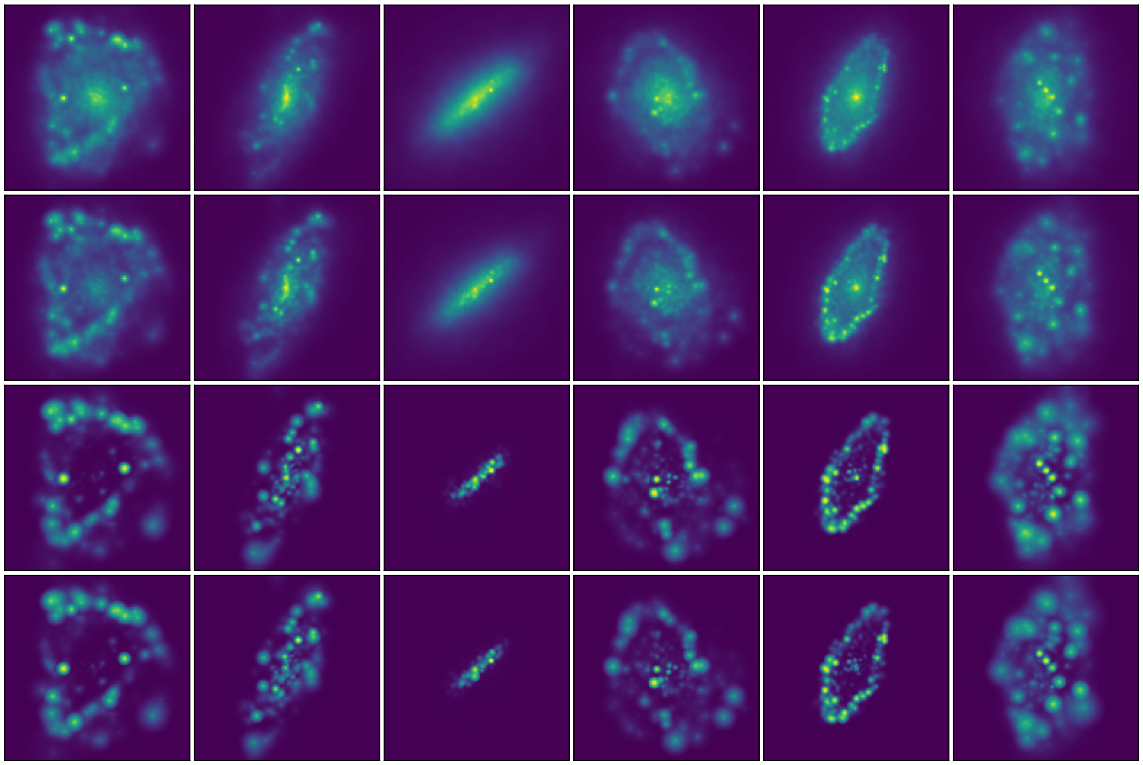


Figure 3.21: The top two rows show the R and B band inputs respectively. The third and fourth rows show the NUV inferences and the ground truth NUV labels respetively. Note the agreement between the inferences and labels despite the complexity of the structure.

We could for instance use bands inputs from G and R ( bands B and G are so close that they are practically interchangeable) from the SDSS to make inferences to FUV and NUV bands from the GALAX survey. Further work could be done to expand this approach to include the Z band as an additional input to benefit from further observations that the SDSS offers.

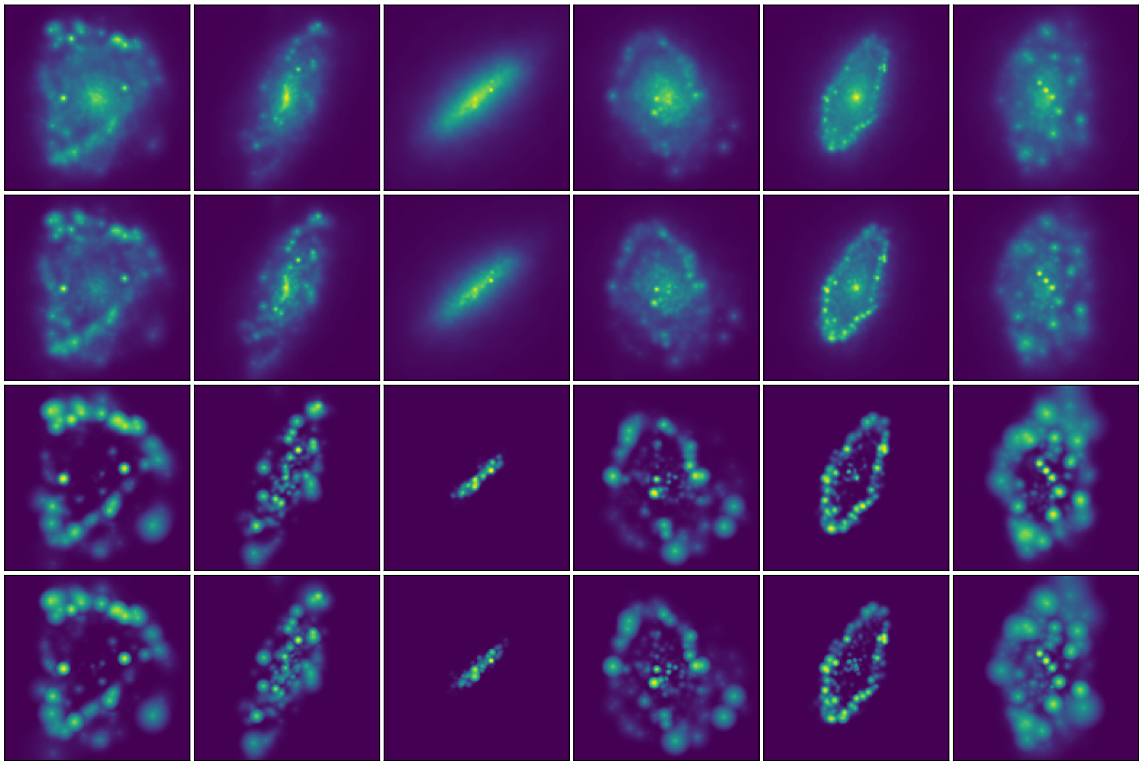


Figure 3.22: The top two rows show the R and B band inputs respectively. The third and fourth rows show the FUV inferences and the ground truth FUV labels respetively. Note the agreement between the inferences and labels despite the complexity of the structure.

### 3.8.1 Astronomical Validation

The bulk of the background information in this section references [48] which we consider to be a comprehensive reference on non-parametric galaxy morphology classification.

So far, we have investigated the outputs of our models qualitatively, by visual inspection and comparison, and quantitatively by means of the SSIM metric. As mentioned in Section (3.6.1), the SSIM metric accounts for local structural differences between images and is a much smarter metric compared to pointwise metrics such as the MSE.

However, the SSIM is a general metric that is not specialized for analyzing galaxy images and while it is a decent indicator of overall image agreement, it does not encode

any semantically meaningful information about the galaxies in question. Therefore, using only the SSIM does not provide any assurances that the generated images are free of spurious effects and reliably resemble natural images.

Astronomers utilize specialized metrics that encode certain features about galaxies; the utility of these metrics lies in their ability to transmit complex features compactly which facilitates the large-scale analysis of vast numbers of galaxies. These metrics are adapted specifically for galaxy images and therefore encode semantically meaningful information about the given galaxy.

Astronomers often employ non-parametric methods to quantify galaxy morphology, focusing on specific structural features. Three widely used non-parametric measures are concentration, asymmetry, and clumpiness (often abbreviated as the CAS parameters), see [49]. Concentration quantifies how centrally concentrated the light from a galaxy is, indicating whether the galaxy has a dense core or a more diffuse distribution. Asymmetry measures the degree to which a galaxy's shape deviates from perfect symmetry, providing insights into recent interactions, mergers, or other dynamic processes. Clumpiness assesses the distribution of light within the galaxy, highlighting the presence of star-forming regions or other irregularities. These parameters collectively offer a compact, yet detailed, description of a galaxy's morphology, capturing essential features that are critical for understanding the galaxy's evolutionary history.

In addition to the CAS parameters, astronomers also utilize Gini and M20 measurements to further characterize the morphological features of galaxies, see [48]. The Gini coefficient is a statistical measure originally used to quantify income inequality, but in galaxy morphology, it quantifies the distribution of light among a galaxy's pixels. A high Gini coefficient indicates that most of the galaxy's light is concentrated in a few pixels, often corresponding to a bright, compact core, whereas a low Gini coefficient suggests a more evenly distributed light profile. This metric is particularly useful for distinguishing between galaxies with different central light concentrations, such as elliptical galaxies and spirals, and for identifying galaxies that may have undergone recent mergers.

M20, on the other hand, is a measure of the relative contribution of the brightest 20 percent of a galaxy's light to its overall light distribution. Specifically, it quantifies the spatial distribution of the brightest regions within a galaxy, helping to identify

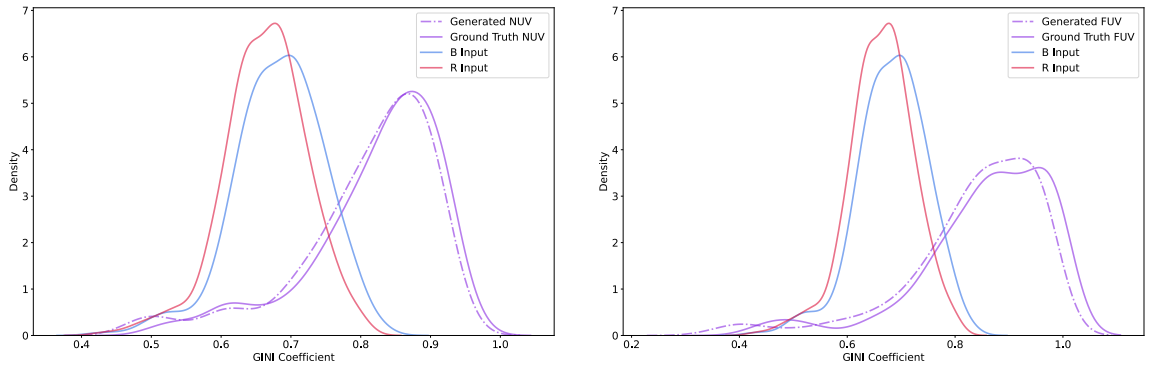
whether the galaxy has a single dominant core or multiple bright regions, which could indicate complex internal structures or recent mergers. When used together, Gini and M20 provide a powerful tool for differentiating between various morphological types, especially in galaxies with irregular or disturbed structures. These measurements are particularly valuable in large-scale surveys where they enable astronomers to classify and analyze thousands of galaxies efficiently, offering insights into the processes driving galaxy formation and evolution. For more details on the calculation of the Gini and M20 measurements, see Section 2 in [48].

The Gini and M20 measurements offer several advantages over the CAS parameters when it comes to analyzing certain types of galaxies, particularly those with more complex or irregular structures. One key advantage is that while the CAS parameters—concentration, asymmetry, and clumpiness—are useful for analyzing relatively well-defined, symmetric galaxies, they may struggle to capture the nuances of galaxies that have undergone mergers or other disturbances. In contrast, Gini and M20 excel at quantifying the light distribution in galaxies that are not well-described by traditional, symmetric models. The Gini coefficient is adept at revealing variations in light concentration without relying on assumptions about the galaxy's overall shape, making it more flexible for analyzing galaxies with irregular or clumpy structures.

Additionally, the M20 measure offers more specific information about the brightest regions of a galaxy, which can be crucial for detecting signs of recent mergers or interactions that cause multiple bright cores or irregular features. Unlike the concentration parameter, which only measures how light is concentrated overall, M20 directly assesses how this concentration is distributed spatially within the brightest 20 percent of the galaxy's light. This gives astronomers a more detailed understanding of internal structures, making Gini and M20 particularly useful for distinguishing between galaxies that might look similar in terms of CAS parameters but differ significantly in their internal dynamics or recent interaction history. These advantages make Gini and M20 ideal for analyzing more diverse galaxy populations in large-scale surveys. It is for this reason that we will use the Gini coefficient and M20 to confirm that the outputs are free of spurious effects and reliably resemble natural images.

We conducted the following analysis. First, we took our trained extrapolation models, one for NUV and one for FUV bands. We then selected a validation set of 500 images using stratified sampling by SFR as discussed in Section (3.5.1). These images

are completely independent from the training dataset. We then passed the validation set through each of the models and recorded the outputs. We calculated the M20 and Gini measurements for both the model outputs and the ground truth labels. A kernel density plot was obtained comparing the Gini and M20 measurement distributions for both the generated images and the ground truth labels. Large agreement between both distributions indicates that the model produces realistic images and is able to preserve the distributions observed for ground truth labels.

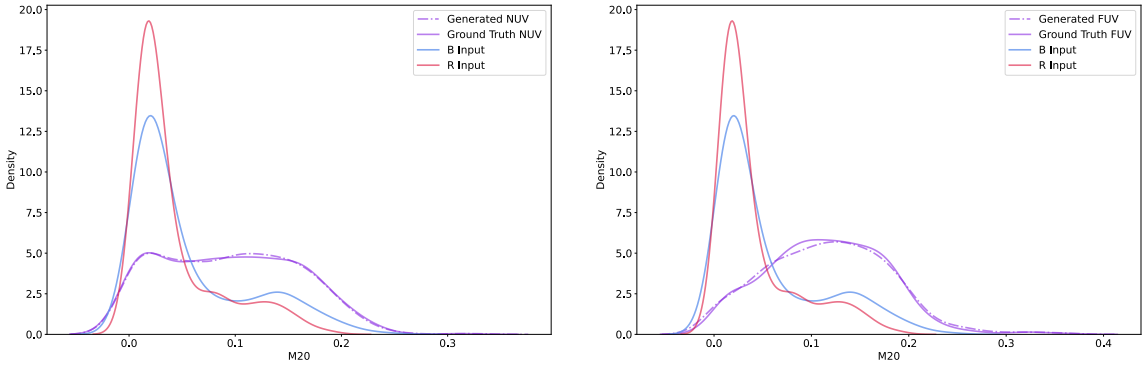



(a) Gini coefficient distributions for NUV labels and outputs

(b) Gini coefficient distributions for FUV labels and outputs

Figure 3.23: Gini coefficient distributions for NUV (right) and FUV (left) labels and outputs. The input bands are plotted as well to visualize the extent to which the networks shift the input distributions. We note the models ability to produce outputs that match the ground truth label GINI distributions.

As seen in Figures (3.23) and (3.24) for the GINI coefficient and M20 distributions respectively, we observe many desirable properties. Firstly, there is a significant shift between the input distributions and the ground truth distributions. Despite this, our networks are capable of producing images that follow the ground truth label distribution for both measurements. There is a slight discrepancy between the generated images and the ground truths labels but given how significant the shift is compared to the inputs, this discrepancy is not a significant cause for concern. Note in Figure (3.24a) how the generated distribution follows along the ground truth distribution despite the ground truth distributions being bimodal (the second mode, centered on M20 = 0.15, of the ground truth distribution lines up with the second mode of the B input). This is an indication that the model is sensitive to subtle differences in the input images.




(a) M20 distributions for NUV labels and outputs

(b) M20 distributions for FUV labels and outputs

Figure 3.24: M20 distributions for NUV (right) and FUV (left) labels and outputs. The input bands are plotted as well to visualize the extent to which the networks shift the input distributions. We note the models ability to produce outputs that match the ground truth label M20 distributions.

## 3.9 SDSS Validation

While the results of Sections (3.6) and (3.7) show promise, they were all obtained with a validation set comprised of Illustris data. This leaves open the following question: to what extent are the models we have developed and trained capable of generalizing the learned mappings to real observations? As mentioned previously, the mock observations were obtained from a cosmological simulation that matches many observed properties both in large-scale structure and in galaxy morphology and composition. In theory, our model should generalize to observed image data with little to no issues however, until this is done in practice, our speculations will remain as such.

In this section, we show promising results in using a model, trained on mock observations from Illustris, to make inferences using real observations from SDSS. To begin, we train a fully supervised model, with the optimal hyperparameter configuration discussed in Section (3.6), to map from band R to band U. This model was then used to make inferences using data gathered from SDSS as in Section (2.2). We selected the R band from the SDSS images by extracting the red channel of the original 3-channel images obtained from the image cut-out tool. These SDSS R band images were fed into the trained model to make inferences. We were unable to obtain U band images

for these galaxies. Therefore, we have no ground truth and will assess the inferred outputs on purely qualitative grounds.

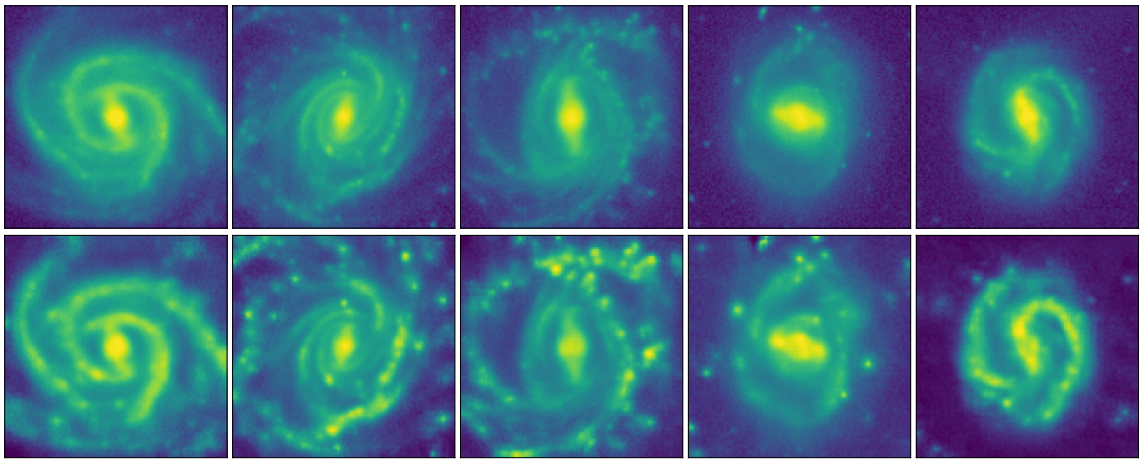


Figure 3.25: The top row shows SDSS inputs while the bottom row show the mapping produced by the Illustris trained model. Note that the model can detect structures and apply the appropriate transformations to these structures as seen with transformations applied to the Illustris data as input.

We observe in the mapped images, Figure (3.25), the patterns that were observed when translating Illustris inputs to U bands (see Figures (3.14) and (3.15)). Traces of spiral arms, which usually indicate areas of high star formation, are amplified and are used by the model to infer surrounding structures which are much more distinctly outlined and highlighted. Also, note that some galaxies have central regions (galactic bulges) attenuated significantly by the mapping, see Figure (3.26). Again, this is in line with expectations for such a translation.

However, there are instances of the model performing little to no transformation on the input image, see Figure (3.27 ). This usually happens when there is little to no underlying structure that the model can use to make inferences. These restrictions are discussed further in the next chapter.

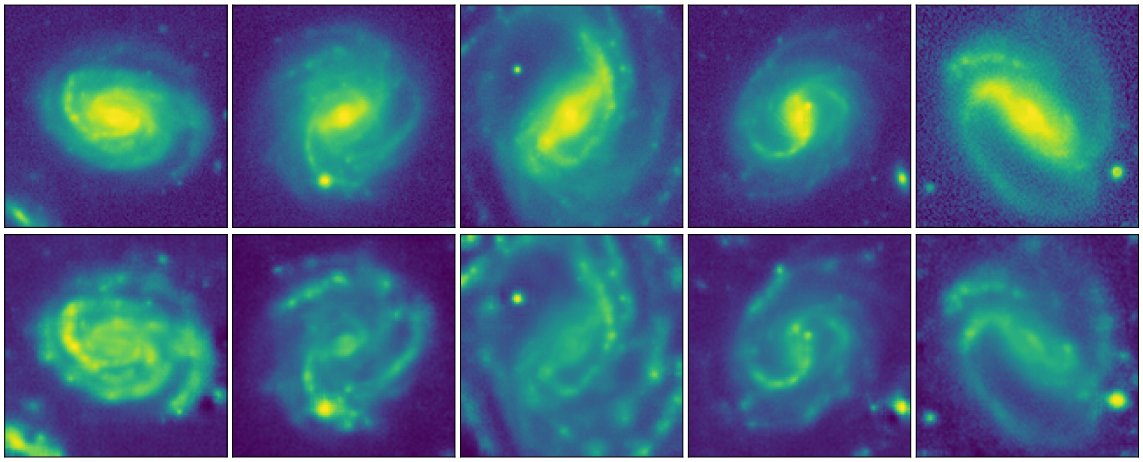


Figure 3.26: The top row shows input galaxies while the bottom row shows the transformed output. Note the attenuation of the central galactic bulge which has little star-forming activity and therefore would contribute little to UV observations.

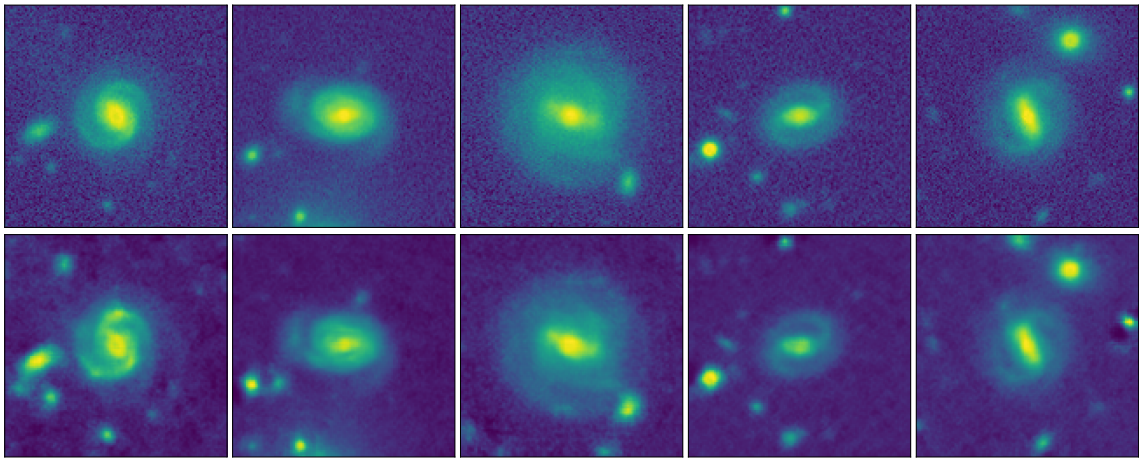


Figure 3.27: Again,the top row is SDSS inputs and the bottom row are the model outputs. We observe little to no transformation applied to these galaxies due to a lack of structure which the model uses to make inferences.

# Chapter 4

# Discussion

We have shown that the image-to-image models can learn semantically meaningful, bijective mappings that are in line with the expected transformations based on our current understanding of galaxies' stellar population distributions. We have also shown that band interpolation yields very promising results and that such models could be used to significantly augment existing observations.

We have done all of our training using mock observations from the Illustris simulations which is, to date, a novel use of this dataset. Training with Illustris data allowed us to take advantage of the existence of ground truth labels and turn the unsupervised CycleGAN objective into a fully supervised objective which yielded better results than its unsupervised counterpart.

There are however restrictions to the models and to the training routine we use. Firstly, training in a fully supervised manner requires a paired data set with ground truth labels. This places a significant restriction on our training data set; specifically, getting a paired data set consisting of real astronomical observations can prove to be quite challenging for various reasons such as scarcity of data and the need for complex pre-processing pipelines. This restriction is particularly cumbersome for the band interpolation models which cannot be run using the original unsupervised CycleGAN training routine.

We also observe a restriction as to the type of galaxies on which our models perform best. We note that our model performs best with galaxies that have a high variation across different band observations. These galaxies tend to be ones with higher SFRs

which tend to be spiral galaxies. Also, for transformations from high wavelength bands, such as IR bands, to low wavelength bands, such as UV bands, our models require traces of the star-forming regions to be visible in the high wavelength input band. Without these traces, our models struggle to recreate the correct structure observed in the low wavelengths band. This was particularly noticeable in Section (3.9), where the model performed little to no transformation on the SDSS inputs due to the lack of structure in the inputs.

As for future work, it is not uncommon for research to beg more questions than it answers and this one is no exception. As it currently stands, our work presents many different avenues of exploration. First and most importantly, we would like to train our models using real observations, whether paired or unpaired. We would then quantitatively compare, all else being equal, the performance between an Illustris-trained model and one that is trained using real observations. This would yield a great deal of insight into any biases the Illustris data contains over real observations. Furthermore, we would like to perform quantitative analysis on an Illustris-trained model using a validation set comprised of real observations; we have only performed a qualitative analysis in Section (3.9) due to a lack of ground truth labels.

As regards band interpolation, we would like to experiment with different input band combinations such as using infrared and visible observations to predict ultraviolet observations. An example would be using Z and R bands to infer the U band. Such inferences are more challenging than the ones presented in Section (3.7) but are less challenging than the single input transformation we presented in Section (3.6).

Finally, our hyperparameter training indicates that an increase in the number of residual blocks that make up the latent space could produce models with better performance. We also would like to experiment with the objective function of the supervised models; perhaps retaining the cycle consistency loss in the supervised objective could yield better results.

Overall, our results are very promising, and we believe that further research and development of image-to-image models for the purpose of photometric band translation would be very beneficial to astronomers.

# Bibliography


[1] Linda S Sparke and John S Gallagher III. *Galaxies in the universe: an introduction*. Cambridge University Press, 2007.

[2] Françoise Combes, Patrick Boissé, Alain Mazure, and Alain Blanchard. *Galaxies and cosmology*. Springer Science & Business Media, 2004.

[3] Yanxia Zhang and Yongheng Zhao. Astronomy in the big data era. *Data Science Journal*, 14:11–11, 2015.

[4] Kyle W Willett, Chris J Lintott, Steven P Bamford, Karen L Masters, Brooke D Simmons, Kevin RV Casteels, Edward M Edmondson, Lucy F Fortson, Sugata Kaviraj, William C Keel, et al. Galaxy zoo 2: detailed morphological classifications for 304 122 galaxies from the sloan digital sky survey. *Monthly Notices of the Royal Astronomical Society*, 435(4):2835–2860, 2013.

[5] Jorge De La Calleja and Olac Fuentes. Machine learning and image analysis for morphological galaxy classification. *Monthly Notices of the Royal Astronomical Society*, 349(1):87–93, 2004.

[6] Ajay Shrestha and Ausif Mahmood. Review of deep learning algorithms and architectures. *IEEE access*, 7:53040–53065, 2019.

[7] Brandon Buncher, Awshesh Nath Sharma, and Matias Carrasco Kind. Survey2survey: a deep learning generative model approach for cross-survey image mapping. *Monthly Notices of the Royal Astronomical Society*, 503(1):777–796, 2021.

[8] Ashley Spindler, James E Geach, and Michael J Smith. Astrovader: astronomical variational deep embedder for unsupervised morphological classification of galaxies and synthetic image generation. *Monthly Notices of the Royal Astronomical Society*, 502(1):985–1007, 2021.

[9] Michael J Smith, James E Geach, Ryan A Jackson, Nikhil Arora, Connor Stone, and Stéphane Courteau. Realistic galaxy image simulation via score-based generative models. *Monthly Notices of the Royal Astronomical Society*, 511(2):1808–1818, 2022.

[10] S Santhosh Baboo and I Kadar Shereef. An efficient weather forecasting system using artificial neural network. *International journal of environmental science and development*, 1(4):321, 2010.

[11] Mingxing Tan and Quoc V. Le. Efficientnet: Rethinking model scaling for convolutional neural networks, 2020.

[12] David Foster. *Generative Deep Learning*. O'Reilly Media, 1005 Gravenstein Highway North, Sebastopol, CA 95472, 2023.

[13] Trevor Hastie, Robert Tibshirani, Jerome Friedman, Trevor Hastie, Robert Tibshirani, and Jerome Friedman. Unsupervised learning. *The elements of statistical learning: Data mining, inference, and prediction*, pages 485–585, 2009.

[14] Douglas M Hawkins. The problem of overfitting. *Journal of chemical information and computer sciences*, 44(1):1–12, 2004.

[15] Sidney Van den Bergh. *Galaxy morphology and classification*. Cambridge University Press, 1998.

[16] MF Skrutskie, SE Schneider, R Stiening, SE Strom, MD Weinberg, C Beichman, T Chester, R Cutri, C Lonsdale, J Elias, et al. The two micron all sky survey (2mass): overview and status. In *The Impact of Large Scale Near-IR Sky Surveys: Proceedings of a Workshop held at Puerto de la Cruz, Tenerife (Spain), 22–26 April 1996*, pages 25–32. Springer, 1997.

[17] Donald G York, J Adelman, John E Anderson Jr, Scott F Anderson, James Annis, Neta A Bahcall, JA Bakken, Robert Barkhouser, Steven Bastian, Eileen Berman, et al. The sloan digital sky survey: Technical summary. *The Astronomical Journal*, 120(3):1579, 2000.

[18] Dark Energy Survey Collaboration:, T Abbott, FB Abdalla, J Aleksić, S Allam, A Amara, D Bacon, E Balbinot, M Banerji, K Bechtol, et al. The dark energy survey: more than dark energy–an overview. *Monthly Notices of the Royal Astronomical Society*, 460(2):1270–1299, 2016.

[19] D Christopher Martin, James Fanson, David Schiminovich, Patrick Morrissey, Peter G Friedman, Tom A Barlow, Tim Conrow, Robert Grange, Patrick N Jelinsky, Bruno Milliard, et al. The galaxy evolution explorer: a space ultraviolet survey mission. *The Astrophysical Journal*, 619(1):L1, 2005.

[20] Nicholas Kaiser, Herve Aussel, Barry E Burke, Hans Boesgaard, Ken Chambers, Mark Richard Chun, James N Heasley, Klaus-Werner Hodapp, Bobby Hunt, Robert Jedicke, et al. Pan-starrs: a large synoptic survey telescope array. In *Survey and Other Telescope Technologies and Discoveries*, volume 4836, pages 154–164. SPIE, 2002.

[21] Junhai Zhai, Sufang Zhang, Junfen Chen, and Qiang He. Autoencoder and its various variants. In *2018 IEEE international conference on systems, man, and cybernetics (SMC)*, pages 415–419. IEEE, 2018.

[22] Lev Yasenko, Yaroslav Klyatchenko, and Oksana Tarasenko-Klyatchenko. Image noise reduction by denoising autoencoder. In *2020 IEEE 11th International Conference on Dependable Systems, Services and Technologies (DESSERT)*, pages 351–355. IEEE, 2020.

[23] Zhaomin Chen, Chai Kiat Yeo, Bu Sung Lee, and Chiew Tong Lau. Autoencoder-based network anomaly detection. In *2018 Wireless telecommunications symposium (WTS)*, pages 1–5. IEEE, 2018.

[24] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[25] Joseph Rocca. Understanding variational autoencoders (vaes) https://towardsdatascience.com/understanding-variational-autoencoders-vaes-f70510919f73, 2019.

[26] Calvin Luo. Understanding diffusion models: A unified perspective. *arXiv preprint arXiv:2208.11970*, 2022.

[27] Getao Du, Xu Cao, Jimin Liang, Xueli Chen, and Yonghua Zhan. Medical image segmentation based on u-net: A review. *Journal of Imaging Science & Technology*, 64(2), 2020.

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

[29] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[30] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models, 2020.

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

[32] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. *Advances in neural information processing systems*, 29, 2016.

[33] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1125–1134, 2017.

[34] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, pages 2223–2232, 2017.

[35] Kamyar Nazeri, Eric Ng, and Mehran Ebrahimi. Image colorization using generative adversarial networks. In *Articulated Motion and Deformable Objects: 10th International Conference, AMDO 2018, Palma de Mallorca, Spain, July 12-13, 2018, Proceedings 10*, pages 85–94. Springer, 2018.

[36] Samarth Shukla, Luc Van Gool, and Radu Timofte. Extremely weak supervised image-to-image translation for semantic segmentation. In *2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW)*, pages 3368–3377. IEEE, 2019.

[37] Yongcheng Jing, Yezhou Yang, Zunlei Feng, Jingwen Ye, Yizhou Yu, and Mingli Song. Neural style transfer: A review. *IEEE transactions on visualization and computer graphics*, 26(11):3365–3385, 2019.

[38] Michael S Bessell. Standard photometric systems. *Annu. Rev. Astron. Astrophys.*, 43:293–336, 2005.

[39] Konstantinos Kouroumpatzakis, Andreas Zezas, Elias Kyritsis, Samir Salim, and Jiri Svoboda. Star formation rate and stellar mass calibrations based on infrared photometry and their dependence on stellar population age and extinction. *Astronomy & Astrophysics*, 673:A16, 2023.

[40] Mark Vogelsberger, Shy Genel, Volker Springel, Paul Torrey, Debora Sijacki, Dandan Xu, G Snyder, Simeon Bird, Dylan Nelson, and Lars Hernquist. Properties of galaxies reproduced by a hydrodynamic simulation. *Nature*, 509(7499):177–182, 2014.

[41] Illustris. illustris-project.org/about, 2018.

[42] James D Bjorken. Cosmology and the standard model. *Physical Review D*, 67(4):043508, 2003.

[43] Rainer Weinberger, Volker Springel, and Rüdiger Pakmor. The arepo public code release. *The Astrophysical Journal Supplement Series*, 248(2):32, 2020.

[44] Paul Torrey, Gregory F Snyder, Mark Vogelsberger, Christopher C Hayward, Shy Genel, Debora Sijacki, Volker Springel, Lars Hernquist, Dylan Nelson, Mariska Kriek, et al. Synthetic galaxy images and spectra from the illustris simulation. *Monthly Notices of the Royal Astronomical Society*, 447(3):2753–2771, 2015.

[45] Robert H Lupton, James E Gunn, and Alexander S Szalay. A modified magnitude system that produces well-behaved magnitudes, colors, and errors even for low signal-to-noise ratio measurements. *The Astronomical Journal*, 118(3):1406, 1999.

[46] Dominique Brunet, Edward R Vrscay, and Zhou Wang. On the mathematical properties of the structural similarity index. *IEEE Transactions on Image Processing*, 21(4):1488–1499, 2011.

[47] Li Yang and Abdallah Shami. On hyperparameter optimization of machine learning algorithms: Theory and practice. *Neurocomputing*, 415:295–316, 2020.

[48] Jennifer M. Lotz, Joel Primack, and Piero Madau. A new nonparametric approach to galaxy morphological classification. *The Astronomical Journal*, 128(1):163–182, July 2004.

[49] Christopher J. Conselice. The relationship between stellar light distributions of galaxies and their formation histories. *The Astrophysical Journal Supplement Series*, 147(1):1–28, July 2003.