**Internship Report**
M1 General Physics

# Inference of Weak Lensing Parameters from Blended Galaxies Using Generative Neural Networks

**By**

NGUYEN Thi Yen Binh

**Supervisors:**

Wassim KABALAN†
Cécile ROUCELLE†
Alexandre BOUCAUD†
Éric AUBOURG†

†*Laboratoire Astroparticule et Cosmology (APC), 75013 Paris, France*


**University Paris Saclay**
Gif-sur-Yvette, France
October 1, 2025

# Abstract

*Context.* The upcoming data from the Legacy Survey of Space and Time (LSST) at the Vera C. Rubin Observatory is expected to be affected by blending in about two-thirds of galaxies—where light from physically separate galaxies overlaps. This blending reduces the number of galaxies usable for weak lensing studies, making deblending a critical challenge.

*Aim.* Deblending algorithms aim to address this challenge using pixel-level, multi-band image data and to recover the weak lensing parameters.

*Method.* We use deep neural networks, including a generative model–Variational Autoencoder (VAE) and a regressor. The autoencoder reduces high-dimensional image data into a low-dimensional latent space and reconstructing the image from this space. While the regressor extracts weak lensing parameters from latent space.

*Result.* Using six-band simulated data for an LSST-like survey, we successfully developed a VAE to reconstruct isolated galaxies from blended scenes, along with a regressor that learns ellipticities from the latent space. The regressor outperforms the reconstruction by significantly reducing the bias toward round shapes that is present in the reconstructed images. We also applied the trained VAE and regressor to real DES data for evaluation.



**Keywords.** cosmology – deep learning – deblending – VAE – regressor


# Contents

# List of Figures

# List of Tables

# 1 Introduction

Gravitational lensing is the gravitational deflection of light from distant galaxies by intervening matter. This causes the galaxy images to appear distorted in shape and size. On average, the distortions of individual galaxies are very small, but when combined, a coherent stretching of the galaxies in a region in the same direction can be observed. Statistical measurements of shape distortion caused by weak lensing from a large-scale structure can be referred to as *cosmic shear*. It is directly linked to the observation of galaxy ellipticities[1] and can be used to statistically map the matter distribution in the universe.

Because the lensing signal accumulates along the line of sight, cosmic shear provides a powerful way to probe the growth of structure and the geometry of the universe. In particular, analyzing cosmic shear in multiple tomographic bins allows us to trace how the matter distribution evolves over time, making it a sensitive tool for constraining the nature and evolution of dark energy.

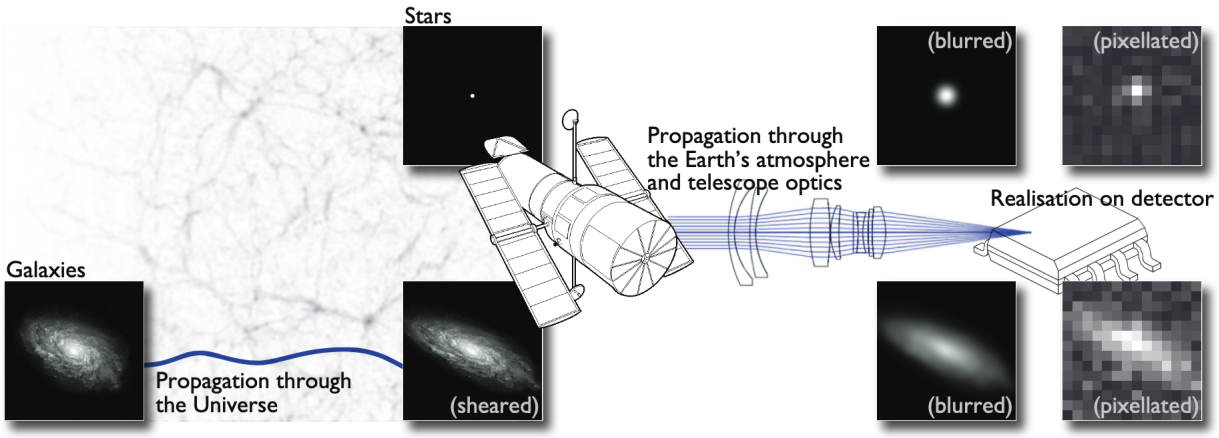



**Figure 1:** The observed galaxy image is affected by several processes (Mandelbaum 2018), including gravitational lensing and other sources of coherent shape distortion, such as convolution with the point spread function (PSF) caused by the atmosphere (for ground-based telescopes) and telescope optics.

The gravitational lensing studies are possible due to the advance in large-scale sky surveys. The next generation including Euclid (Refregier et al. 2010), Legacy Survey of Space and Time (LSST; Ivezić et al. 2019) and Nancy Grace Roman Space Telescope (Sanderson et al. 2024) will measure the dark energy equation of state with 1% precision when combined with data from the cosmic microwave background (CMB). Due to the large field of view, most of these surveys will not be able to employ adaptive optics that could minimize the atmospheric and instrumental effects on the point spread function (PSF). At the depth of the surveys, this would lead to blending, affecting over half of the objects. For example, Hyper Suprime-Cam (HSC) found that about 58% of detected objects are blended and LSST is expected to have 62% blended detections. Although it seems straightforward to discard blended objects, this would introduce systematic biases and reduce statistical power for cosmological analyses. As a result, effective galaxy deblending techniques are essential.

[1] Ellipticiy measures how much a galaxy shape deviates from a circle and in which direction it is stretched.

Classic deblenders such as `SExtractor` (Bertin & Arnouts 1996) and SDSS deblender (Lupton & Ivezić 2005) use threshold-based segmentation on each band and assign pixels exclusively to one object, but fail to redistribute flux between overlapping sources. This produces inaccurate morphologies and fluxes, particularly for complex blends and also limiting the only on single-band data.

Modern algorithms address these issues through (1) multi-band coherence and (2) physical modeling. For (1), tools like the Lambda Adaptive Multi-Band Deblending Algorithm in R (`LAMBDAR`; Wright 2016) use high-resolution priors to align photometry across bands, while `SCARLET` (Melchior et al. 2018) applies constrained matrix factorization to jointly model all bands, preserving spectral energy distributions. For (2), `The Tractor` (Lang et al. 2016) fits parametric galaxy models (e.g., Sérsic profiles) to optimize likelihoods, and `SCARLET` uses proximal gradient descent to enforce monotonicity and symmetry. However, these methods rely on predefined models, risking systematic errors when real galaxies deviate from these assumptions, especially for irregular shapes.

Machine learning, particularly neural networks, offers a paradigm shift for deblending challenges by learning deblending directly from data rather than relying on predefined models. This data-driven flexibility makes it easier to handle complex, multi-band images. Once trained, neural networks can also process data rapidly–a huge advantage given the massive volume of data expected from upcoming astronomical surveys. Among deep learning techniques, generative models have shown particular promise for galaxy deblending, such as Generative Adversarial Networks (GANs) by Reiman & Göhre (2019) and Variational Auto-Encoders (VAEs) with `Debvader` by Arcelin et al. (2020) and `MADNESS` by Biswas et al. (2024). Given that VAEs produce more consistent results than GANs (Ravanbakhsh et al. 2016), VAE-architecture of `Debvader` would be adopted in this project.

Similar to previous works including `Debvader` and `MADNESS`, our project is developed in preparation for the LSST Vera C. Rubin Observatory. These earlier methods used deep learning to (1) extract features from isolated galaxies and encode them into a latent space and (2) reconstruct galaxy images to measure lensing parameters such as ellipticities. `MADNESS` improves upon `DebVader` by incorporating a normalizing flow to better approximate the true posterior, reducing the bias toward round galaxy's shapes. Inspired by these studies, our project creates a method based on VAE that directly predicts lensing parameters, like ellipticities and redshift, from the latent space, skipping the reconstruction step to reduce biases. We also evaluated this approach on real DES data, whereas previous work only used LSST simulations.

The structure of this report is as follows: Section 2 introduces the main concepts of the Variational Autoencoder (VAE) framework. Section 3 provides an overview of the full pipeline, including dataset generation, the VAE-based deblender, the regressor architecture, and the training procedure. Section 4 presents the results and discussions. Finally, Section 5 concludes the report and outlines possible future directions.

This work is carried out within the framework of the *Astrodeep*[2] team, with the main goal of developing Bayesian deep learning methods for cosmology, at the Astroparticle and Cosmology Laboratory.

[2] https://astrodeep.net/

# 2 Variational Autoencoder

VAE, introduced by Kingma & Welling (2019), is a generative neural network composed of two main components: an encoder and a decoder. The encoder functions as a recognition model, transforming input data into a probability distribution in a latent space. This distribution is typically characterized by its mean and variance. The latent space is a compressed, lower-dimensional representation of the input, where each point corresponds to a possible configuration of the data. The decoder samples from the latent distribution and reconstructs the input data.

## 2.1 standard VAE

Mathematically, let $\mathbf{x}$ be the observed data and $\mathbf{z}$ the hidden latent variables. The goal is to find a function that maps from the latent space to the data, so that the model can generate $\mathbf{x}$ well. The decoder models this as $p_\theta(\mathbf{x}|\mathbf{z})$, where $\theta$ are the network parameters. We want to find $\theta$ that maximize the marginal likelihood of $\mathbf{x}$:

$$\hat{\theta} = \arg\max_\theta \int p_\theta(\mathbf{x}|\mathbf{z})p(\mathbf{z})d\mathbf{z} \tag{1}$$

This integral is hard to compute because it sums over many possible $\mathbf{z}$, especially when $\mathbf{z}$ is high dimensional. To solve this, we use a second function $q_\phi(\mathbf{z}|\mathbf{x})$ (the encoder), which approximates the true but unknown posterior $p_\theta(\mathbf{z}|\mathbf{x})$.

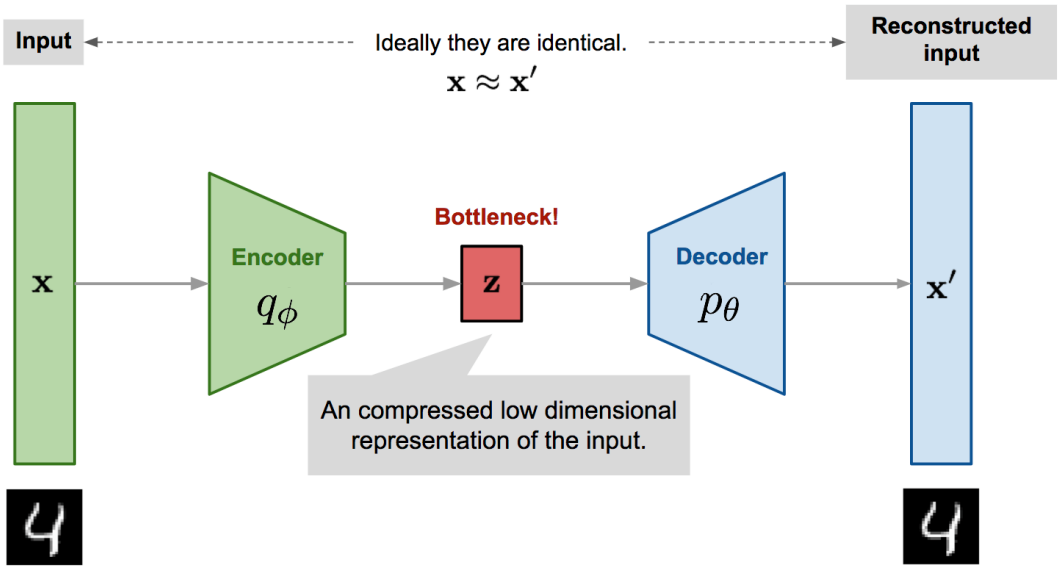



**Figure 2:** Variational Autoencoder Architecture (Source: Lilian Weng Blog)

We train the VAE by maximizing a lower bound on the log-likelihood of the data, called the Evidence Lower Bound (ELBO). Instead of maximizing ELBO, we minimize the negative ELBO as the loss:

$$\mathcal{L}_{\text{VAE}}(\mathbf{x}) = \mathcal{L}_{\text{KL}} + \mathcal{L}_{\text{recon}} = D_{\mathbf{KL}}\left(q_\phi(\mathbf{z} \mid \mathbf{x}) \| p(\mathbf{z})\right) - \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\log p_\theta(\mathbf{x} \mid \mathbf{z})\right] \tag{2}$$

The first term, the Kullback-Leibler (KL) divergence, is the expected information loss when we adopt an approximation, measuring how close the encoded distribution $q_\phi(\mathbf{z}|\mathbf{x})$ is to the prior $p(\mathbf{z})$. It helps to keep the latent space close to a known distribution and avoids overfitting.

The second term is the reconstruction term. It measures how well the decoder reconstructs the original input $\mathbf{x}$ from the sampled latent variable $\mathbf{z}$. Instead of the complex expected log-likelihood, we use a simple distance: the mean squared error (MSE) between $\mathbf{x}$ and its reconstruction $\hat{\mathbf{x}}$:

$$\text{MSE}(\mathbf{x}, \hat{\mathbf{x}}) = \|\mathbf{x} - \hat{\mathbf{x}}\|^2 \tag{3}$$

## 2.2 Reparameterization trick

The expectation term in the loss function involves generating samples from $\mathbf{z} \sim q_\phi(\mathbf{z} \mid \mathbf{x})$. Sampling is a stochastic operation, which makes it non-differentiable and prevents gradient backpropagation. To address this, the reparameterization trick is used. It allows us to express the random variable $\mathbf{z}$ as a deterministic function of $\mathbf{x}$ and an auxiliary noise variable $\epsilon$, such that $\mathbf{z} = \mathcal{T}_\phi(\mathbf{x}, \epsilon)$. This reformulation shifts the stochasticity into $\epsilon$, enabling gradients to flow through $\mathcal{T}_\phi$ during training.

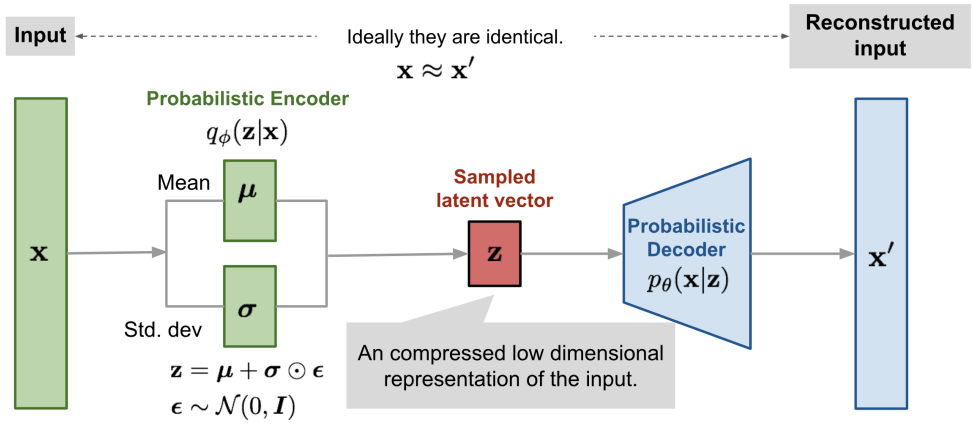



**Figure 3:** Reparameterization trick (Source: Lilian Weng Blog)

In this work, we adopt the common assumption that the approximate posterior $q_\phi(\mathbf{z} \mid \mathbf{x})$ is a multivariate Gaussian with a diagonal covariance matrix:

$$\begin{aligned} \mathbf{z} \sim q_\phi(\mathbf{z} \mid \mathbf{x}) &= \mathcal{N}\left(\mathbf{z}; \boldsymbol{\mu}, \boldsymbol{\sigma}^2 \mathbf{I}\right), \\ \mathbf{z} &= \boldsymbol{\mu} + \boldsymbol{\sigma} \odot \boldsymbol{\epsilon}, \quad \boldsymbol{\epsilon} \sim \mathcal{N}(0, \mathbf{I}), \end{aligned} \tag{4}$$

where $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$ are outputs of the encoder network and $\odot$ denotes element-wise multiplication.

With this choice, the KL divergence between the approximate posterior $q_\phi(\mathbf{z} \mid \mathbf{x})$ and the standard normal prior $p(\mathbf{z}) = \mathcal{N}(\mathbf{0}, \mathbf{I})$ can be computed analytically as:

$$D_{\mathbf{KL}}\left(q_\phi(\mathbf{z} \mid \mathbf{x}) \,\|\, p(\mathbf{z})\right) = \frac{1}{2} \sum_{i=1}^{d} \left(\mu_i^2 + \sigma_i^2 - \log \sigma_i^2 - 1\right), \tag{5}$$

where $d$ is the dimension of the latent space.

## 2.3 $\beta$-VAE

In a standard VAE, if the KL term can dominates the loss, it can degrade reconstruction quality. Conversely, if the reconstruction term dominates, it may cause overfitting and reduce generalization. $\beta$-VAE is a modification of the standard VAE that helps balance the two terms, encourages disentangled latent representations. In a disentangled representation, each latent variable $\mathbf{z}_i$ controls one specific generative factor of the data, and is mostly independent of other factors. This makes the model's latent space more interpretable and can improve generalization.

$\beta$-VAE introduces a hyperparameter $\beta > 0$ to the standard VAE loss, weighting the KL divergence term:

$$\mathcal{L}_{\beta\text{-VAE}}(\mathbf{x}) = \beta \mathcal{L}_{\text{KL}} + \mathcal{L}_{\text{recon}} \tag{6}$$

When $\beta = 1$, this is the standard VAE. When $\beta > 1$, the KL term is emphasized more. The model is pushed to compress the latent space and learn more disentangled representations, but possibly at the cost of reconstruction quality. When $\beta < 1$, the model focuses more on reconstruction. This can improve generation quality but often leads to less disentangled latent variables.

## 2.4 Normalizing Flow

Normalizing Flow is a method to build more flexible and expressive latent distributions in variational models. In a standard VAE, the posterior $q(\mathbf{z}|\mathbf{x})$ is usually assumed to be a simple Gaussian, which can limit the model's ability to capture complex data structures.

Normalizing Flow addresses this by applying a sequence of invertible and differentiable transformations $f_1, f_2, \ldots, f_K$ to a simple initial distribution $\mathbf{z} \sim q(\mathbf{z})$, transforming it in a more complex distribution:

$$\mathbf{z}_K = f_K \circ f_{K-1} \circ \cdots \circ f_1(\mathbf{z}) \tag{7}$$

Each transformation adjusts the shape of the distribution while allowing exact computation of the log-probability using the change of variables formula:

$$\log q_K(\mathbf{z}_K) = \log q(\mathbf{z}) - \sum_{k=1}^{K} \log \left| \det \frac{\partial f_k}{\partial \mathbf{z}_{k-1}} \right| \tag{8}$$

By learning these transformations jointly with the rest of the model, the VAE can better approximate the true posterior, leading to improved latent representations and more accurate generative models.

# 3 Pipeline Overview

The main goal of this project is to build a VAE network to denoise and deblend galaxy images, reconstruct the central galaxy, and train a regressor to directly extract ellipticities from the latent space. We will proceed in four steps: (1) generate a simulated multiband LSST galaxy dataset for training, (2) build a VAE-based deblender inspired by Arcelin et al. (2020), (3) develop a regressor to directly infer weak lensing parameters from blended galaxy images, bypassing explicit deblending, and (4) retrieve real DES data to validate the pipeline.

The code is implemented in `Python` using `Pytorch` (Paszke et al. 2019) and executed on the `Jean Zay` (IDRIS 2020) supercomputer due to the computational intensity of the training process.

## 3.1 Dataset Preparation

### 3.1.1 Data Normalization

The input for the VAE network should be normalized between -1 and 1 to ensure numerical stability during the training, we will adopt the nonlinear normalization from Arcelin et al. (2020) for each band $b$:

$$x_b = \tanh\left(\sinh^{-1}\left(\beta \frac{x_{\mathbf{raw},b}}{\langle \max\left(x_{\mathbf{raw},b}\right)\rangle_b}\right)\right) \tag{9}$$

where $\langle \max\left(x_{\mathbf{raw},b}\right)\rangle_b$ is the mean of the distribution of the maximum pixel values in the $b$ band of input images. $\beta$ is emperical constant set to 2.5 as we followed the original paper. This normalization makes the images easier for the network to learn. It increases the contrast between galaxies and their backgrounds, enlarges the visible galaxy structures, and makes edges clearer in normalized space. This helps VAE to better learn galaxy shapes and shear patterns. Since the transformation is invertible, we can always denormalize the images later using the inverse function.

### 3.1.2 Simulated LSST data

To train and evaluate our deblender, we use simulated LSST galaxy images generated with the `Blending ToolKit` (BTK; Mendoza et al. 2025), which is configured to reproduce the key observing characteristics of the Vera C. Rubin Observatory. The galaxies are sampled from the `CatSim` catalog (Connolly 2014), which contains bulge & disk models spanning redshifts from $z = 0$ to $z = 6$, with realistic distributions of sizes and ellipticities. All structural parameters are fixed across bands, except for the magnitudes. The distribution of galaxies after the simulation is shown in Figure 4, and it closely resembles real observational data.

The simulations follow the expected 10-year LSST survey strategy, using filter-specific properties (gain, zero-point, PSF, sky background, and number of visits) drawn from *SurveyCodex* and summarized in Table 1. The PSF in each band is modeled as wavelength-dependent but spatially constant, with full width at half maximum (FWHM) values based

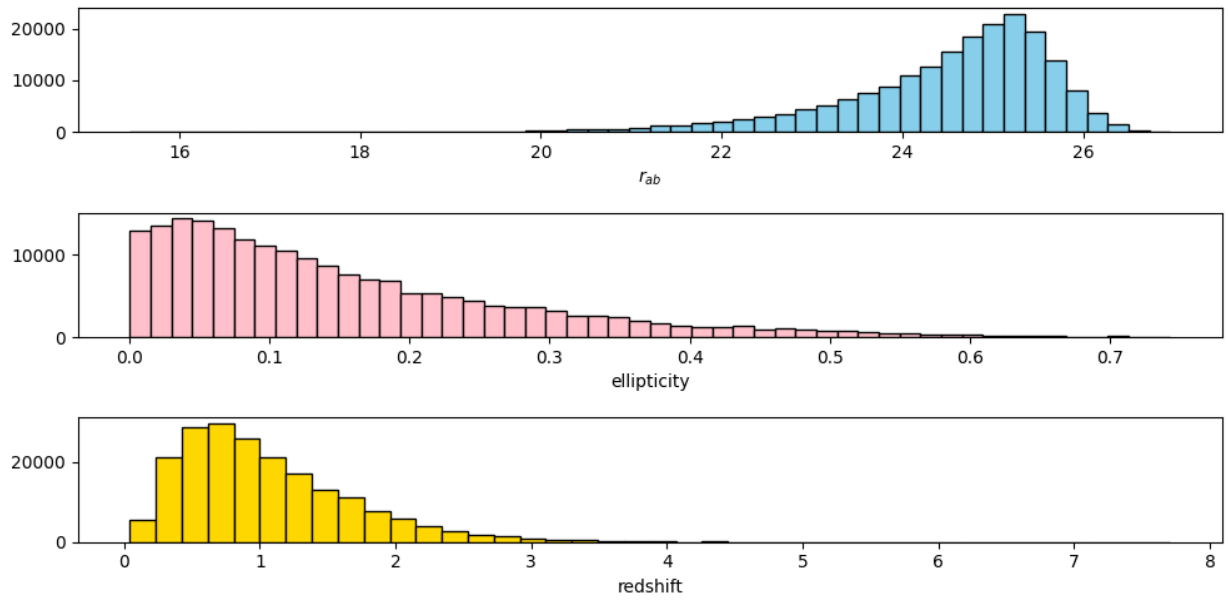



**Figure 4:** Distribution of Galaxies simulated from `Blending ToolKits` with configuration of LSST survey.

on airmass 1.2 seeing conditions from Ivezić et al. (2019). We adopt a pixel scale of 0.2″, and include both realistic atmospheric effects and telescope optics.

**Table 1:** LSST simulation specifications for each filter used in the galaxy catalog simulations.

| Filter | Zero Point | Sky Brightness | PSF FWHM (arcsec) | Number of Visits |
|---|---|---|---|---|
| $u$ | 26.40 | 22.99 | 0.90 | 56 |
| $g$ | 28.26 | 22.26 | 0.86 | 80 |
| $r$ | 28.10 | 21.20 | 0.81 | 184 |
| $i$ | 27.78 | 20.48 | 0.79 | 184 |
| $z$ | 26.56 | 18.61 | 0.76 | 160 |
| $Y$ | 27.39 | 19.60 | 0.77 | 160 |

Each galaxy is rendered using `GalSim` (Rowe et al. 2015), and image stamps of $45 \times 45$ pixels ($9'' \times 9''$ ) are created for all six LSST bands $u$, $g$, $r$, $i$, $z$, $Y$. The brightest objects are always centered and will act as targets for the deblender. Three datasets are generated:

1. Ground truth: noiseless isolated galaxy images centered in the frame.
2. Noisy isolated galaxy images.
3. Blended images containing up to 3 galaxies, with the target galaxy centered and the other galaxies shifted randomly.

Noise, including background noise and galaxy noise, is added to the second and third datasets. The noise characteristics follow those provided by the `Blending ToolKits`. Each

dataset contains 200,000 samples, divided into 90% for training and 10% for validation. We measure the ellipticities of the grouth truth images using the equations introduced in Section 4.3 to get the labels for *Training 3* (refer to Section 3.2).

### 3.1.3 Real DES data

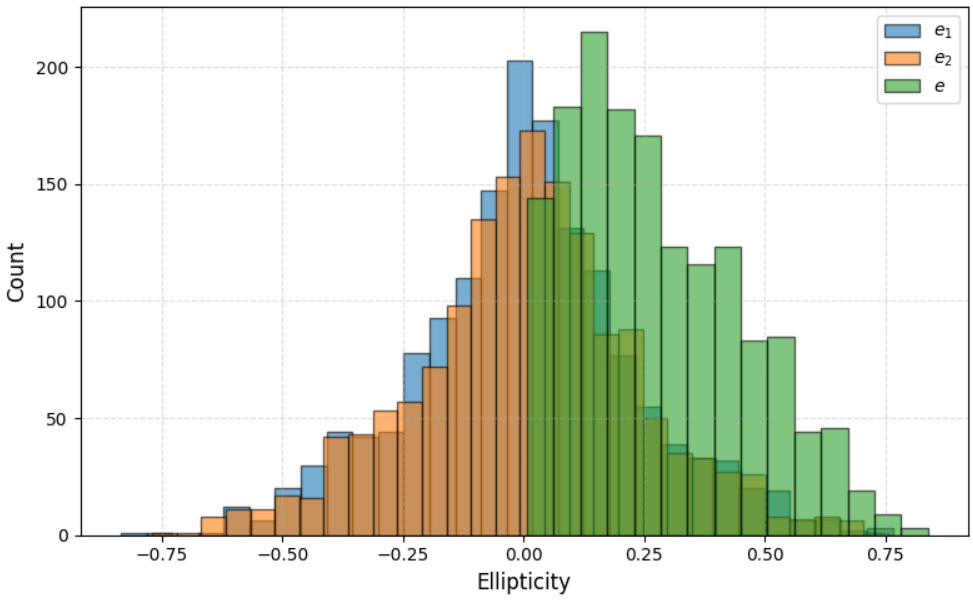



**Figure 5:** Distribution of ellipticities from `im3shape` catalog.

In this study, we selected galaxies from the DES[3] Science Verification (SV) `im3shape` catalog (Jarvis et al. 2016), which provides shape measurements optimized for weak gravitational lensing analyses. The `im3shape` algorithm is a maximum likelihood model-fitting method that fits galaxy shapes using a two-component bulge plus disk Sérsic model. It was applied to the DES SV data (referred to as SVA1), providing 5-band images in the *g, r, i, z, Y* filters.

To ensure a similar structure with the simulated LSST data that we generated, we applied the following quality cuts:

- Valid ellipticity components ($e_1$, $e_2$).
- A signal-to-noise ratio $\mathrm{snr_w} > 20$ to reduce noise bias.
- A match with the DES "gold" catalog to obtain accurate positions `(RA, Dec)`.

The distribution of ellipticities from the catalog after quality cuts is shown in Figure 5. These positions were used to retrieve image cutouts from the DES DR1 coadded tiles using the NOIRLab Astro Data Lab[4] portal. We used the `Image Cutout Tool`, which allows image

[3] https://www.darkenergysurvey.org/
[4] https://datalab.noirlab.edu/

stamps to be downloaded based on sky coordinates, filter, and cutout size. For each galaxy, we requested the cutout in each band centered on its position. The brightest galaxy was considered the central galaxy. We then applied a final cut to obtain $45 \times 45$ pixel input images. After all selections, we obtained 1546 images.

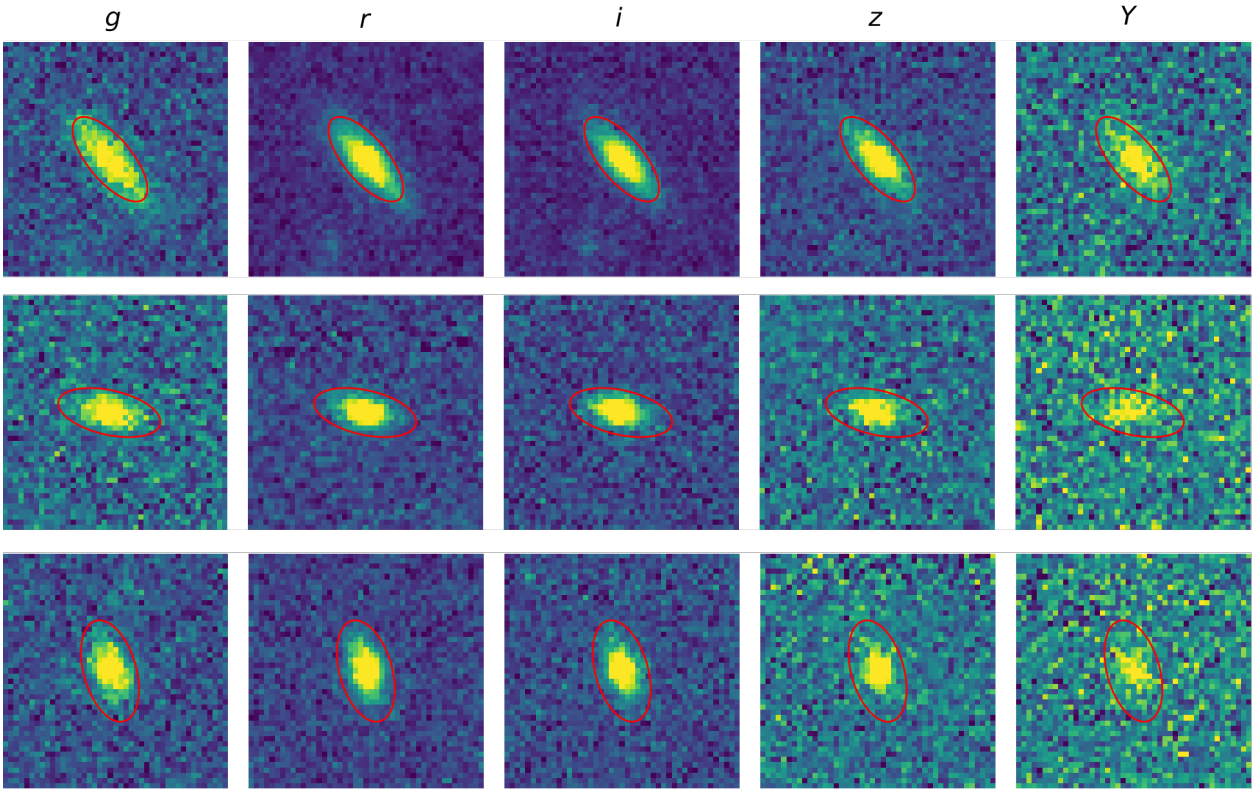



**Figure 6:** Example results from DES data queries with the ellipses overplot using $(e_1, e_2)$ from `im3shape` catalog.

To compare the shapes in the catalog with the observed galaxy images, we overplotted ellipses representing the `im3shape` ellipticity values. Each ellipse was drawn using the components $(e_1, e_2)$, where negating $e_2$ accounts for the difference in coordinate system between image pixel space and sky coordinates.

### 3.1.4 Dynamic Range

The dynamic range of an image refers to the range of pixel intensities, from the faintest background to the brightest sources. To help the model generalize from simulation to real data, the simulated LSST images and real DES images should have similar pixel value distributions after normalization. We apply the same non-linear normalization function from Arcelin et al. (2020) (see previous section) independently to each band of both datasets. This ensures that the values are rescaled in a consistent way.

After normalization, we compare the pixel value distributions band by band. As shown in Figure 7, the $g$, $r$, $i$, and $z$ bands show good agreement between LSST and DES. The $Y$

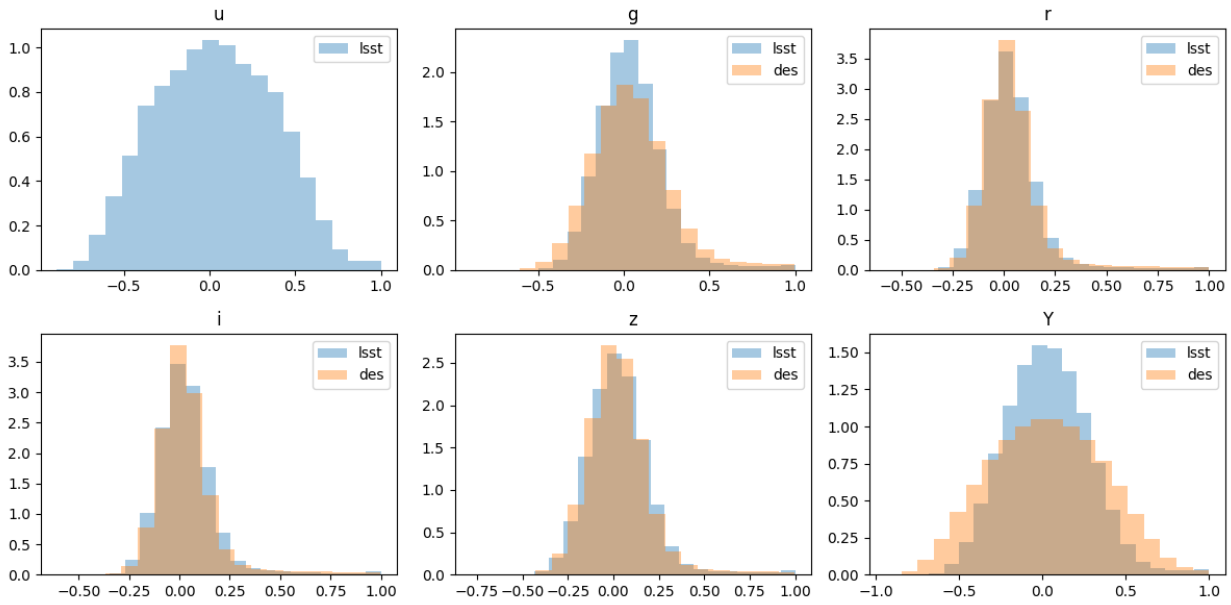



**Figure 7:** The comparison of the dynamical ranges between LSST and DES data

band shows a mismatch due to its low signal-to-noise ratio. The $u$-band is not present in DES, so it introduces missing information that we need to account for in later steps.

## 3.2 Deblender Architecture

For the implementation of the VAE, we use the `PyTorch` framework instead of `TensorFlow`, as in previous works, because `PyTorch` allows for more flexible construction of complex architectures and is more popular in deep learning research. Our training consists of three main steps to build the deblender:

1. *Training 1*: We train the VAE on noisy isolated galaxies as input. The goal is to learn features and representations of isolated galaxies in the latent space. For supervision, we use the clean isolated galaxies as the target in the loss function. After training, the network can denoise and reconstruct isolated galaxies from the latent space.

2. *Training 2*: We fix the decoder from Training 1 and train a new encoder on noisy blended galaxies as input. The decoder remains fixed to use the prior knowledge of isolated galaxies learned in *Training 1*. The supervision again uses the clean isolated galaxies as targets in the loss function, encouraging the network to deblend the noisy blended input into clean isolated images.

3. *Training 3*: We fix the encoder from *Training 2* and train a regressor that takes the latent vector from the encoder as input. The input remains noisy blended galaxies, but the labels are the ellipticities. The regressor learns to extract these physical parameters directly from the latent space using MSE loss.

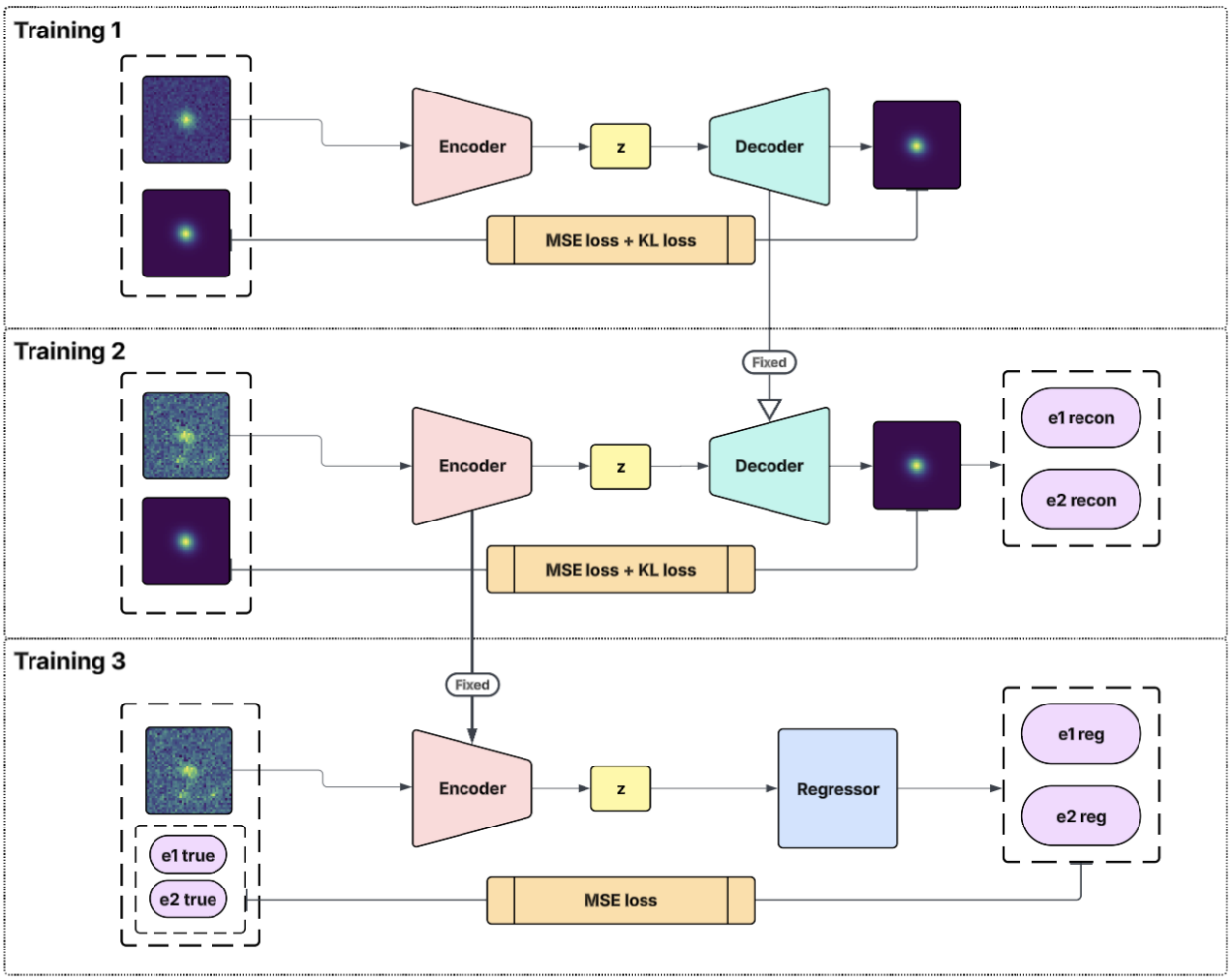



**Figure 8:** Diagram of the planned workflow for the neural network model.

**Encoder:** the encoder consists of four convolutional layers with 32, 64, 128, and 256 filters, each using a $3 \times 3$ convolution kernel with stride 2 to reduce the spatial size while capturing spatial features. These layers use the `PReLU` activation function for non-linearity. `PReLU` (Parametric Rectified Linear Unit) is a type of activation function that allows a small, learnable slope for negative input values, unlike the standard `ReLU` which outputs zero for all negative values. This helps the network avoid the "dying `ReLU`" problem, where neurons become inactive. Nonlinear activation functions like `PReLU` are essential in deep learning because they allow the network to learn complex, nonlinear patterns in the data. The encoder ends with two dense layers that produce the mean and variance of the latent space distribution. We applied the reparameteration trick here to allow backpropagation through the sampling process.

**Decoder:** The decoder mirrors the encoder architecture in reverse, reconstructing images from the latent space, using the transposed convolutional layers followed by `PReLU`. The final layer uses a `Sigmoid` activation to ensure the output pixel values are between 0 and 1, improving previous work that used `ReLU` and produced outputs in $[0, \infty)$. To keep the output image size the same as the input, we applied bilinear interpolation, which resizes an

image by computing the output pixel value as a weighted average of the four nearest input pixels.

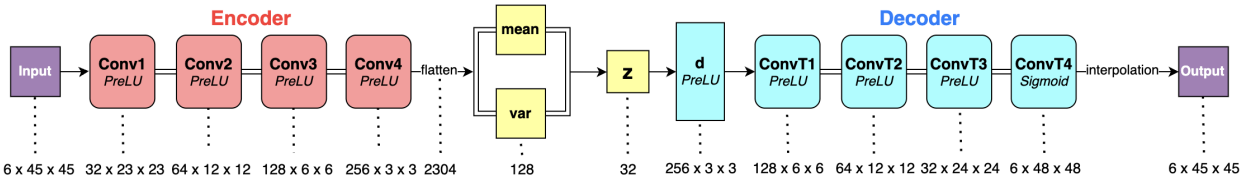



**Figure 9:** Diagram of the VAE architecture: Each block shows a layer name in bold, its activation function in italic, and the output tensor shape below. We use `Conv` for convolution layers, `ConvT` for transposed convolution layers, and `d` for dense (or fully-connected) layers.

**Regressor:** The regressor is a small neural network that predicts ellipticities from the latent space vector. It contains three fully connected layers with 128, 64 and 32 units, respectively, each followed by `PReLU` activations and dropout for regularization. Dropout is a regularization technique in which, during training, some neurons are randomly dropped out (i.e. set to zero) with a fixed probability. This prevents the network from becoming too dependent on any single neuron and helps reduce overfitting. The final layer outputs two values representing ellipticity components, passed through a `Tanh` activation to constrain the outputs to the range $[-1, 1]$. The input to the regressor is the mean of the latent distribution $q(\mathbf{z}|\mathbf{x})$ from the encoder.

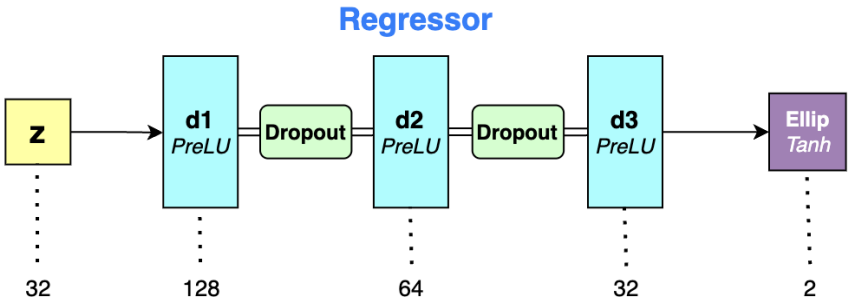



**Figure 10:** Diagram of the regressor. `d` are the dense layers, and `Dropout` layers in between for regularization. The input is the mean value $\mu$ from latent space and the output is the ellipticity $(e_1, e_2)$.

The overall architecture handles multiband astronomical data effectively because the convolutional layers learn cross-band features, and the latent bottleneck compresses spectral information efficiently. Optimization is performed by minimizing the loss of ELBO for the VAE and the loss of MSE for the regressor. We use *Adam* optimizer (Kingma & Ba 2017) with learning rate of $10^{-4}$, training for about 40 epochs until the loss functions reach the plateau.

After training, we want to assess the performance of our network on the lastest DES dataset of `im3shape` catalog as the true ellipticites. Unlike LSST, which has 6 bands, DES only has 5 bands and does not include the $u$-band. Since the ultraviolet $u$-band is not the most important for galaxy images, we replace it with an empty array and accept the loss of information for now.

# 4 Results

## 4.1 Deblenders training

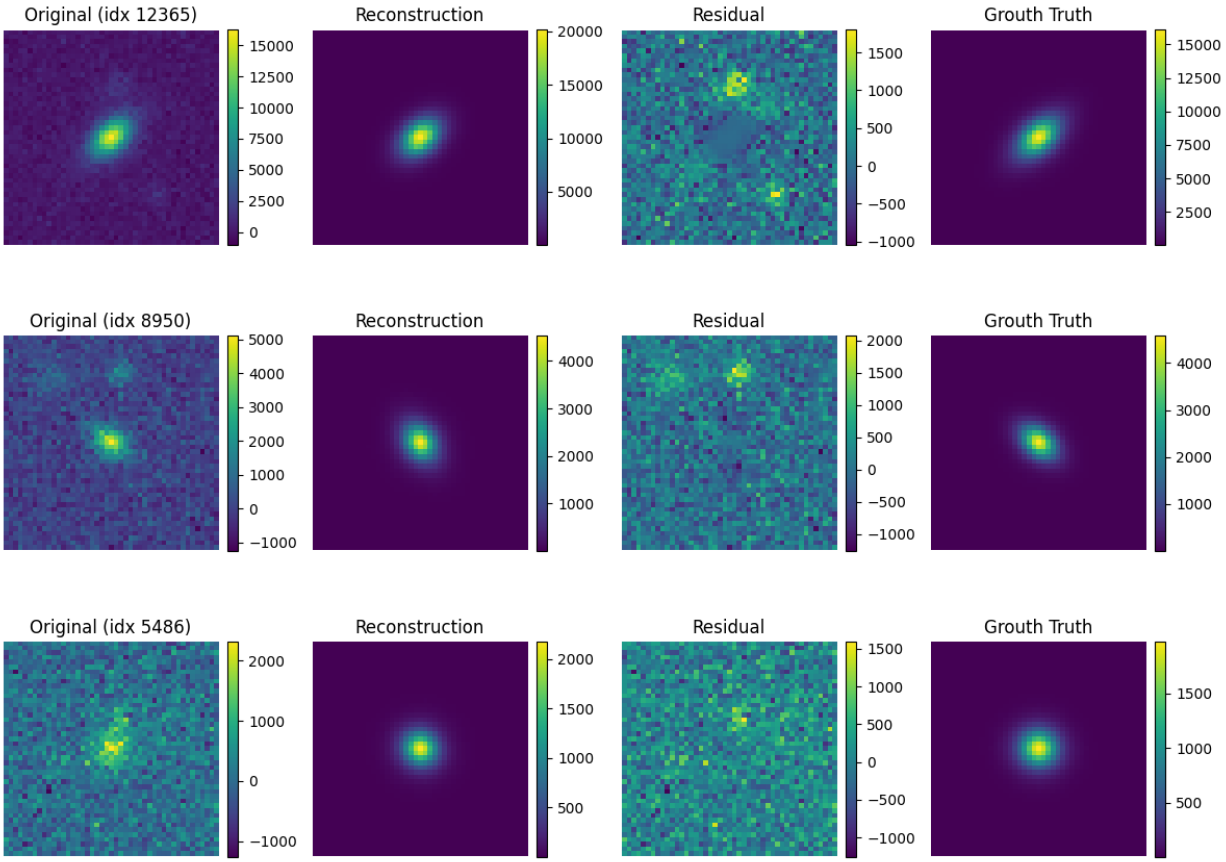



**Figure 11:** Demonstration for the reconstruction of deblender after *Training 2*.

Because the network tends to bias toward round shapes, due to the Gaussian prior in the latent space, we need to reduce $\beta$ (see the Section 2.3) so that the model focuses more on reconstruction and learns the ellipticities. After testing different values of $\beta$, we observed that the network starts learning meaningful reconstructions around $\beta \sim 10^{-3}$. This is similar to what Arcelin et al. (2020) found, where they used $\beta = 10^{-2}$. To further help the model, we constrain the training data to galaxies with ellipticity $e > 0.1$, focusing on galaxies with higher shear. After all of these steps, the network starts learning the shear structure of galaxies, as shown by the decreasing reconstruction loss in Figure 22. At the same time, the KL divergence increases because the encoder begins to shape the latent space to match the Gaussian prior.

Figure 11 shows examples of the deblending performance of the VAE-Deblender after *Training 2*. We see that the model successfully denoises the images and reconstructs the central galaxies with visible shear structures. However, the magnitude levels are not always

accurate. For example, in the first reconstructed image, the flux differs from the ground truth. This may be due to the nonlinear normalization, which emphasizes the edges more strongly. As a result, the central flux distribution may not be well reconstructed. However, since the main goal of this project is to estimate the shapes of galaxies, we accept this limitation for now.

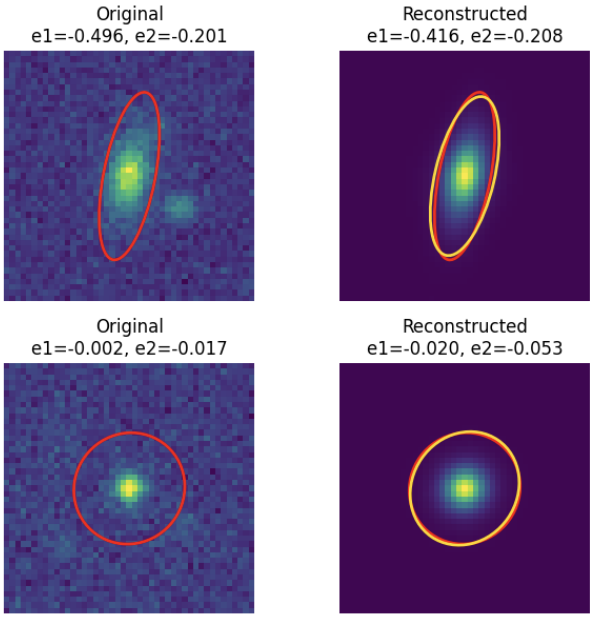



**Figure 12:** Demonstration of the regressor after *Training 3*, overplotted on the reconstructed images from the VAE. The red lines show the true ellipticities, while the yellow lines indicate the ellipticities predicted by the regressor.

Figure 12 shows the results of regressor from *Training 3*, we can see that the regressor has the ability to extract the information from the latent space.

## 4.2 Morphology and shapes of the deblended galaxies

To test the morphology of the reconstructions, we compute the Structural Similarity Index Measure (SSIM) and the cosine similarity. SSIM compares the 2D structure of the reconstructed and ground truth images, evaluating luminance, contrast, and pattern similarity.

$$\mathrm{SSIM}(x, y) = \frac{(2\mu_x\mu_y + c_1)\,(2\sigma_{xy} + c_2)}{\left(\mu_x^2 + \mu_y^2 + c_1\right)\left(\sigma_x^2 + \sigma_y^2 + c_2\right)} \tag{10}$$

where $\mu_x$ and $\mu_y$ are the means of $x$ and $y$, $\sigma_x^2$ and $\sigma_y^2$ are the variances, and $\sigma_{xy}$ is the covariance between $x$ and $y$. The constants $c_1$ and $c_2$ are used to stabilize the division when the denominator is small. The SSIM value ranges from $-1$ to 1, where 1 indicates perfect structural similarity. In practice, values are usually between 0 and 1 for image comparisons.

Cosine similarity compares the reconstructed and ground truth images flattened into 1D

vectors, measuring the angle between them in feature space.

$$S_C(A, B) := \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \cdot \sqrt{\sum_{i=1}^{n} B_i^2}}, \tag{11}$$

where $\mathbf{A}$ and $\mathbf{B}$ are the flattened versions of the reconstructed and ground truth images, respectively. Cosine similarity values range from $-1$ for opposite direction to 1 for the same direction, with 1 indicating perfect alignment in feature space.

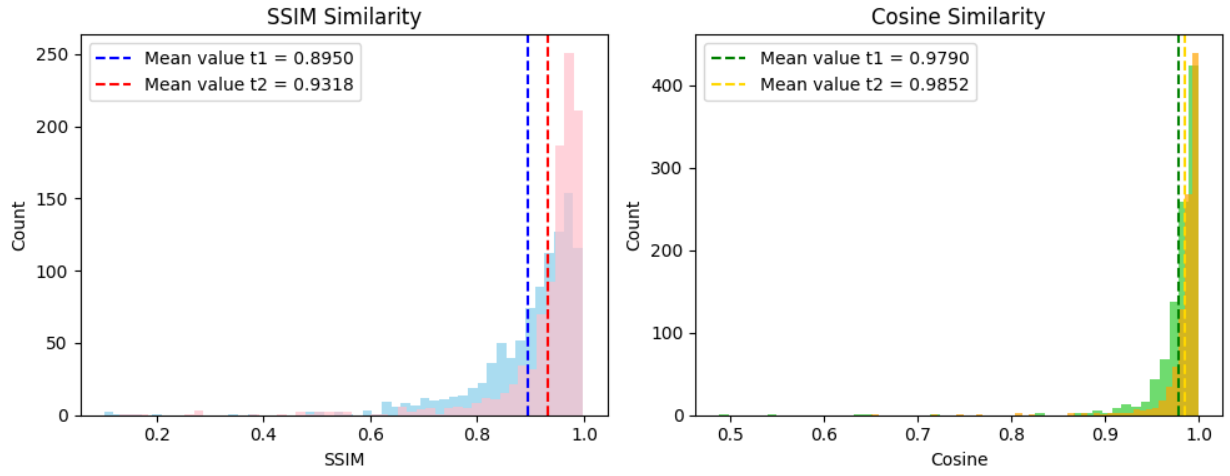



**Figure 13:** Structural Similarity and Cosine Similarity evaluated on the noisy deblended validation dataset after *Training 1* (t1) and *Training 2* (t2).

Figure 13 shows the evaluation of the deblender's performance after *Training 1* and *Training 2*. Even without access to blended inputs during *Training 1*, the network learns to reconstruct the central isolated galaxy with about 90% average structural similarity and 98% average cosine similarity. After *Training 2*, where the encoder is trained to extract the central galaxy from blended scenes, these scores improve to 93% and 98.5%, respectively. This suggests that the reconstructed galaxies have similar structure and intensity patterns to the true isolated galaxies. However, since the galaxies in the dataset tend to have round shapes, high SSIM and cosine similarity scores may not fully reflect errors in shape, ellipticity, or orientation.

## 4.3 Ellipticity estimation

Having the noiseless isolated galaxy image, we measure the ellipticity components $e_1$ ("+" mode, elongation along x and y axes) and $e_2$ ("×" mode, elongation along 45° diagonals). These components describe the shape and orientation of the galaxy and are computed from the second moments of the light distribution:

$$e_1 = \frac{Q_{xx} - Q_{yy}}{Q_{xx} + Q_{yy}}, \quad e_2 = \frac{2Q_{xy}}{Q_{xx} + Q_{yy}} \tag{12}$$

The second moments $Q_{ij}$ of the galaxy brightness profile $I(\mathbf{x})$, weighted by a function $W(\mathbf{x})$, are defined as:

$$Q_{ij} = \frac{\int d^2\mathbf{x}\, W(\mathbf{x})\, I(\mathbf{x})\, x_i x_j}{\int d^2\mathbf{x}\, W(\mathbf{x})\, I(\mathbf{x})} \tag{13}$$

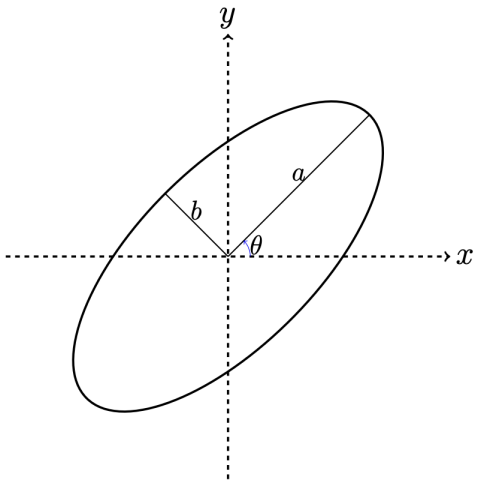
where $\mathbf{x} = (x, y)$ is the position relative to the galaxy centroid, and $i, j \in \{x, y\}$. As we use a uniform weight, we could choose $W(\mathbf{x}) = 1$. The total ellipticity can be estimated as:

$$e = \sqrt{e_1^2 + e_2^2} \tag{14}$$

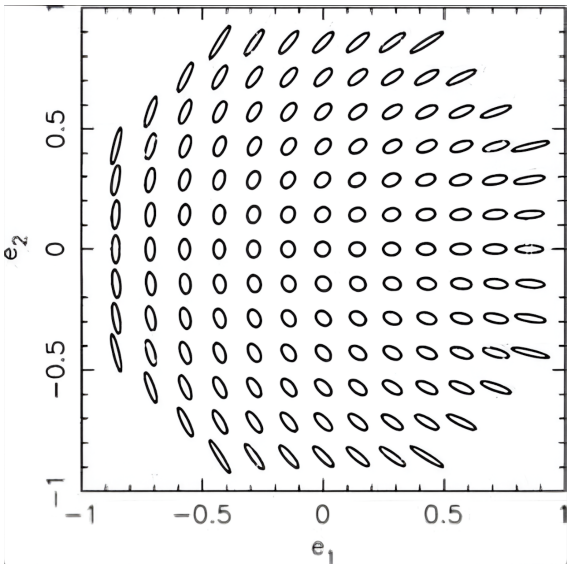
The ellipticities $(e_1, e_2)$ can be transformed into the semi-major axis $a$, semi-minor axis $b$, and the orientation angle $\theta$ (with respect to the x-axis) via a bijective mapping:

$$\theta = \frac{1}{2}\arctan\left(\frac{e_2}{e_1}\right), \quad \frac{b}{a} = \frac{1-e}{1+e} \tag{15}$$

$b/a$ quantifies the ellipticity of a galaxy: values near 1 mean round, values near 0 mean very flat.




**(a)** Ellipse with semi-major axis $a$, semi-minor axis $b$, and the orientation angle $\theta$.




**(b)** Ellipticities represents by $(e_1, e_2)$

)

**Figure 14:** two representations of ellipticity parameters (Source: Bridle et al. 2008

### 4.3.1 simulated LSST data

Figure 15 shows the distribution of predicted ellipticity parameters $(e_1, e_2)$ from both the reconstructed images and the regressor, as a function of the ground truth values in the first three scatter plots. Points closer to the diagonal indicate more accurate predictions. We observe a degeneracy in the reconstructed images: some points form a horizontal line, showing that the model sometimes reconstructs nearly round galaxies regardless of the input. In contrast, the regressor shows less bias and no such degeneracy, likely because it has

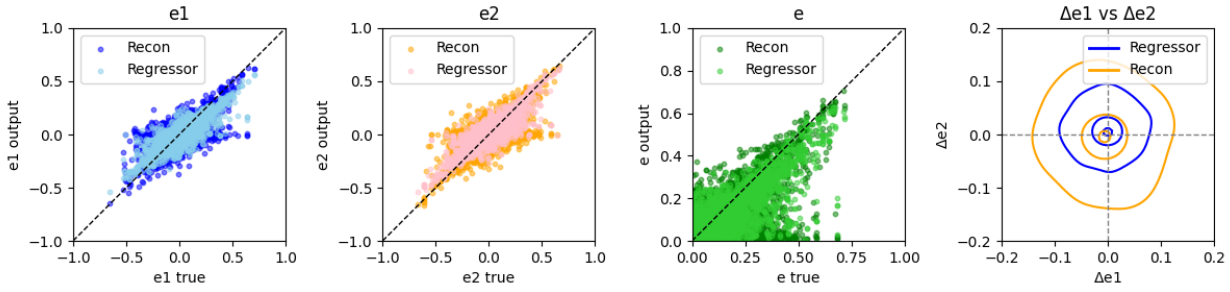



**Figure 15:** Distributions of the output shape parameters as functions of the true parameters measured on simulated LSST data. Scatter plots show $e_1$, $e_2$, and ellipticity magnitude $e$, along with the error contours on $e_1$ and $e_2$ within $3\sigma$.

fewer parameters and a simpler structure. The final plot shows the errors on $e_1$ and $e_2$, computed as $\Delta e = e_{\text{output}} - e_{\text{input}}$. The regressor has smaller error contours, confirming better performance. Figure 16 shows smoothed distributions of $e_1$ and $e_2$, which further confirms that the reconstruction is more biased toward round shapes compared to the regressor.

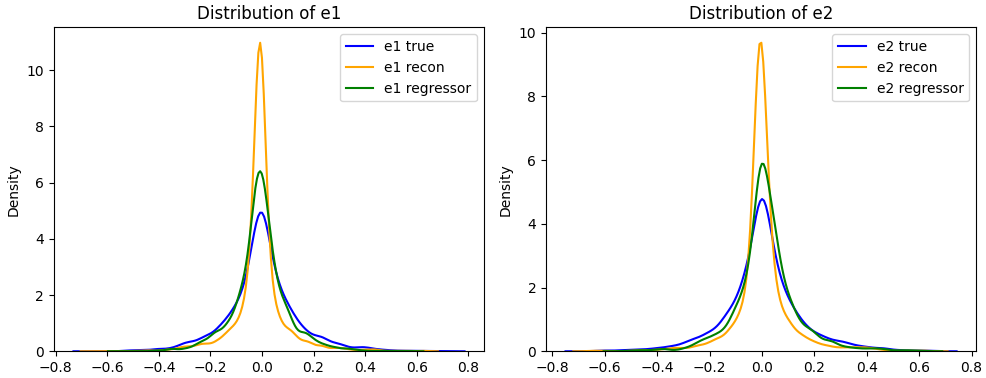



**Figure 16:** Distribution of ellipticities parameters validated on simulated LSST data, showing the level of bias towards the round shape between VAE-reconstruction and regressor.

As explained in Section 2, the KL divergence term in the VAE encourages the latent space to follow a Gaussian prior, which limits how much information can be stored and leads to a bias toward round shapes. Also, if the decoder cannot fully recover all the shape variations, it may produce a generic average galaxy, which is usually round. By using a regressor with fewer weights, we avoid some of the biases introduced during the reconstruction process. However, there are still residual biases due to the Gaussian prior in the latent space. This problem could be reduced by using a normalizing flow, as introduced in Section 2.4. Because normalizing flows can apply an invertible transformation to the Gaussian prior, they allow the latent space to follow a more complex distribution that better matches the true data distribution.

### 4.3.2 DES data

DES is a highly successful wide-field multi-band imaging survey and a natural precursor to LSST. It offers similar multi-band coverage but with shallower depth and no $u$-band, making it a valuable test case for methods we aim to apply to LSST data in the future. No VAE-based deblender has yet been tested on real survey data, so this is an important step toward applying such models in practice.

We applied the trained pipeline to the DES dataset. The input images are 45×45 pixels, with the brightest galaxy at the center, and the ellipticities from `im3shape` are used as ground truth. The results are shown in Figure 17 and Figure 18.

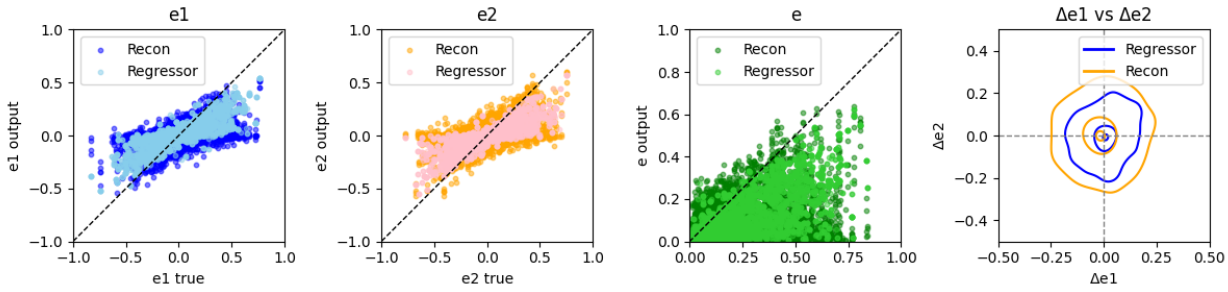



**Figure 17:** Distributions of the output shape parameters as functions of the true parameters measured on simulated DES data with empty array for $u$-band. Scatter plots show $e_1$, $e_2$, and ellipticity magnitude $e$, along with the error contours on $e_1$ and $e_2$ within $3\sigma$.

We observe that both the VAE and the Regressor show strong biases toward rounder shapes on the real DES data, and the errors are also large. This suggests that the encoder cannot capture the correct information in the latent space. As a result, both the regression and the reconstruction perform worse than in the simulated LSST data case. However, the Regressor still performs slightly better than the VAE reconstruction.

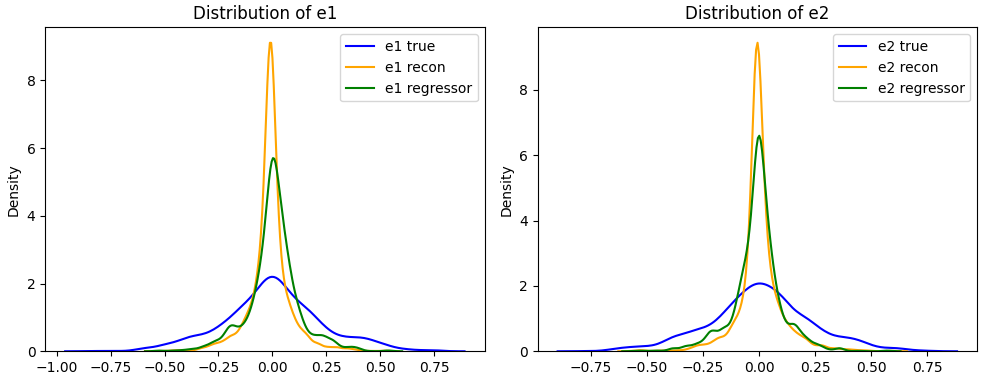



**Figure 18:** Distribution of ellipticities parameters validated on DES data with empty array for $u$-band, showing the level of bias towards the round shape between VAE-reconstruction and regressor.

One reason for the stronger bias is that the data are more spread out and the regressor tends to be biased toward the average representation of the round shape. This bias does not show any new systematic failure beyond what was mentioned before. This issue highlights the need for a better latent-space distribution. Another possible reason is the missing $u$ band in DES, which causes loss of information. To improve performance, we could either handle the missing band better or train the model using DES simulation.

# 5 Conclusion

In this work, we developed a VAE-based Deblender that can reconstruct isolated galaxies from blended images and extract ellipticities from the latent space using a regressor. Overall, the regressor performs better than the decoder reconstruction, although some bias toward round shapes remains. This bias could be reduced in future work by using normalizing flows to improve the latent space representation. With normalizing flows, we could also relax the non-linear normalization function, which may help improve the reconstruction quality, especially for the galaxy's central brightness.

For the test on DES data using the `im3shape` catalog, the results are as good as those of the simulated data. This is understandable because, in simulations, we fully control the input conditions and have access to the true values. In contrast, for DES data, we assume that `im3shape` gives the true ellipticities. The performance drop could be due to the differences between LSST and DES, especially the missing $u$-band. It could also be caused by how we apply quality cuts, which could create a different data distribution from our simulations. To improve this, we could try training on simulated DES-like data instead of LSST-like data or using some domain adaptation methods to align simulated and real data distributions.

We initially tried to train a regressor to predict both redshift and ellipticities from the latent space, but it did not work well because of their different physical meanings and scales. As a result, we focus only on ellipticities in this work. In the future, we would like to explore the extraction of redshift with another regressor.

# References


Arcelin, B., Doux, C., Aubourg, E., & Roucelle, C. 2020, Monthly Notices of the Royal Astronomical Society, 500, 531–547

Bertin, E. & Arnouts, S. 1996, , 117, 393

Biswas, B., Aubourg, E., Boucaud, A., et al. 2024, MADNESS Deblender: Maximum A posteriori with Deep NEural networks for Source Separation

Bridle, S., Shawe-Taylor, J., Amara, A., et al. 2008, The Annals of Applied Statistics, 3

IDRIS. 2020, Jean Zay: Supercalculateur convergé pour le calcul intensif et l'intelligence artificielle, accessed: 2025-03-27

Ivezić, , Kahn, S. M., Tyson, J. A., et al. 2019, The Astrophysical Journal, 873, 111

Jarvis, M., Sheldon, E., Zuntz, J., et al. 2016, Monthly Notices of the Royal Astronomical Society, 460, 2245–2281

Kingma, D. P. & Ba, J. 2017, Adam: A Method for Stochastic Optimization

Kingma, D. P. & Welling, M. 2019, Foundations and Trends® in Machine Learning, 12, 307–392

Lang, D., Hogg, D. W., & Mykytyn, D. 2016, The Tractor: Probabilistic astronomical source detection and measurement, Astrophysics Source Code Library, record ascl:1604.008

Lupton, R. & Ivezić, Ž. 2005, in Astronomical Society of the Pacific Conference Series, Vol. 338, Astrometry in the Age of the Next Generation of Large Telescopes, ed. P. K. Seidelmann & A. K. B. Monet, 151

Mandelbaum, R. 2018, Annual Review of Astronomy and Astrophysics, 56, 393–433

Melchior, P., Moolekamp, F., Jerdee, M., et al. 2018, Astronomy and Computing, 24, 129–142

Mendoza, I., Torchylo, A., Sainrat, T., et al. 2025, The Open Journal of Astrophysics, 8

Paszke, A., Gross, S., Massa, F., et al. 2019, PyTorch: An Imperative Style, High-Performance Deep Learning Library

Ravanbakhsh, S., Lanusse, F., Mandelbaum, R., Schneider, J., & Poczos, B. 2016, Enabling Dark Energy Science with Deep Generative Models of Galaxy Images

Refregier, A., Amara, A., Kitching, T. D., et al. 2010, Euclid Imaging Consortium Science Book

Reiman, D. M. & Göhre, B. E. 2019, Monthly Notices of the Royal Astronomical Society, 485, 2617–2627

Rowe, B., Jarvis, M., Mandelbaum, R., et al. 2015, GalSim: The modular galaxy image simulation toolkit

Sanderson, R. E., Hickox, R., Hirata, C. M., et al. 2024, Recommendations for Early Definition Science with the Nancy Grace Roman Space Telescope

Wright, A. H. 2016, LAMBDAR: Lambda Adaptive Multi-Band Deblending Algorithm in R, Astrophysics Source Code Library, record ascl:1604.003

# 6 Appendix

## 6.1 VAE formalism

The generative model VAE is a member of the machine learning class known as unsupervised learning. Learning probabilistic models of different natural and artificial phenomena from data is typically of interest to machine learning researchers. Such phenomena are described mathematically by probabilistic models. They are helpful for comprehending such events, forecasting future unknowns, and for a variety of automated or assisted decision-making processes.

Probabilistic models always contain unknowns and data rarely represent the whole picture, so a level of uncertainty is always inherent in the form of probability distribution. The most complete form of probabilistic model is the joint distribution over all variables.

Let's $\mathbf{x}$ be the vector representing the set of all observed variables, whose joint distribution we would like to model.

**Assumption:** observed variable $\mathbf{x}$ is a random sample from an *unknown underlying process*, whose true distribution $p^*(\mathbf{x})$ is unknown.

We want to approximate this process with a chosen distribution $p_\theta(\mathbf{x})$ parameterized by $\theta$:

$$\mathbf{x} \sim p_\theta(\mathbf{x}) \tag{16}$$

**Learning:** the process of finding $\theta$ so that $p_\theta(\mathbf{x}) \approx p^*(\mathbf{x})$ for any observed $\mathbf{x}$.

For practical purposes like classification or progressor tasks, we are interested in learning a conditional model $p_\theta(\mathbf{y}|\mathbf{x})$ that approximates the underlying conditional distribution $p^*(\mathbf{y}|\mathbf{x})$

$$p_\theta(\mathbf{y}|\mathbf{x}) \approx p^*(\mathbf{y}|\mathbf{x}) \tag{17}$$

where $\mathbf{x}$ is the *input* and $\mathbf{y}$ is the vector of variables we want to know. For example, for image classification, $\mathbf{x}$ is an image and $\mathbf{y}$ is its class.

### 6.1.1 Dataset

We collect a dataset containing $N \geq 1$ datapoints:

$$\mathcal{D} = \{\mathbf{x}^{(i)}\}_{i=1}^{N} \tag{18}$$

The data points are assumed to be sampled from an underlying distribution. Then, the log-probability assigned to the data by the model:

$$\log p_\theta(\mathcal{D}) = \sum_{\mathbf{x}\in\mathcal{D}} \log p_\theta(\mathbf{x}) \tag{19}$$

### 6.1.2 Marginal Likelihood

We could present a joint distribution $p_\theta(\mathbf{x}, \mathbf{z})$ and marginalized over latent variables $\mathbf{z}$:

$$p_\theta(\mathbf{x}) = \int p_\theta(\mathbf{x}, \mathbf{z})\, d\mathbf{z} \tag{20}$$

This is the *marginal likelihood* or the *model evidence*. We can also factorize it in this form:

$$p_\theta(\mathbf{x}, \mathbf{z}) = p_\theta(\mathbf{z}) p_\theta(\mathbf{x} \mid \mathbf{z}) \tag{21}$$

where the distribution of $p(\mathbf{z})$ is referred to as the *prior* distribution over $\mathbf{z}$.

Latent variables $\mathbf{z}$ are variables that are part of the model, but we cannot observe them and they are not part of the dataset. The latent distribution $p(\mathbf{z})$ is the most effective way to represent our data distribution $p(\mathbf{x})$. To go from one representation to the other, we need mappings. The first mapping from $\mathbf{x}$ to $\mathbf{z}$ is the posterior probability $p(\mathbf{z} \mid \mathbf{x})$ or the probability that we can infer latent vectors $\mathbf{z}$ giving particular images $\mathbf{x}$. The second mapping from $\mathbf{z}$ to $\mathbf{x}$ is the likelihood probability $p(\mathbf{x} \mid \mathbf{z})$ or the probability that we can generate the data $\mathbf{x}$ from the latent vector $\mathbf{z}$.

To optimize $p_\theta(\mathbf{x})$, we need to compute the integral:

$$p_\theta(\mathbf{x}) = \int p_\theta(\mathbf{z}) p_\theta(\mathbf{x} \mid \mathbf{z}) \, d\mathbf{z} \tag{22}$$

However, this integral is intractable, as it lacks an analytical solution and cannot be efficiently computed numerically. The VAE tackles this issue by maximizing $p_\theta(\mathbf{x})$ as much as possible given a data set, within the Bayesian inference framework.

### 6.1.3 Bayes rules

Bayes rule is a fundamental result in probability theory. It allows us to update our beliefs about unknown variables after observing the data.

$$p_\theta(\mathbf{x} \mid \mathbf{z}) = \frac{p_\theta(\mathbf{z}) \cdot p_\theta(\mathbf{z} \mid \mathbf{x})}{p_\theta(\mathbf{x})} = \frac{p_\theta(\mathbf{x}, \mathbf{z})}{p_\theta(\mathbf{x})} \tag{23}$$

where

- *Posterior* $p_\theta(\mathbf{z} \mid \mathbf{x})$: distribution of $\mathbf{z}$ given the observed data $\mathbf{x}$.
- *Likelihood* $p_\theta(\mathbf{x} \mid \mathbf{z})$: probability of observing $\mathbf{x}$ for a given $\mathbf{z}$.
- *Prior* $p_\theta(\mathbf{z})$: model for $\mathbf{z}$ before observing $\mathbf{x}$.
- *Evidence* $p_\theta(\mathbf{x})$: marginal probability of the data.

### 6.1.4 Variational inference

Because the marginal likelihood $p_\theta(\mathbf{x})$ is intractable, the posterior $p_\theta(\mathbf{z} \mid \mathbf{x})$ is also intractable. Therefore, we use variational inference to approximate the posterior using a simpler distribution $q_\phi(\mathbf{z} \mid \mathbf{x})$, parameterized by $\phi$:

$$q_\phi(\mathbf{z} \mid \mathbf{x}) \approx p_\theta(\mathbf{z} \mid \mathbf{x}) \tag{24}$$

The distribution $q_\phi(\mathbf{z} \mid \mathbf{x})$ should be flexible enough so that, by optimizing the parameters $\phi$, it can be made close to the true posterior $p_\theta(\mathbf{z} \mid \mathbf{x})$.

### 6.1.5 Kullback-Leibler divergence (KL divergence)

The Kullback-Leibler divergence measures the expected loss of information when using $q$ to approximate the true distribution $p$. In other words, it quantifies how much the distribution $q$ diverges from $p$:

$$
\begin{aligned}
D_{KL}(q_\phi \,\|\, p_\theta) &= \int_{\mathbf{z}} q_\phi \cdot \log \frac{q_\phi}{p_\theta} \, d\mathbf{z} \geq 0 && (25)\\
&= \mathbb{E}_{q_\phi}\left[\log \frac{q_\phi}{p_\theta}\right] && (26)\\
&= \mathbb{E}_{q_\phi}\left[\log q_\phi\right] - \mathbb{E}_{q_\phi}\left[\log p_\theta\right] && (27)\\
&= \mathbb{E}_{q_\phi}\left[\log q_\phi\right] - \mathbb{E}_{q_\phi}\left[\log \left(\frac{p_\theta(\mathbf{z},\mathbf{x})}{p_\theta(\mathbf{x})}\right)\right] && (28)\\
&= \mathbb{E}_{q_\phi}\left[\log q_\phi\right] - \mathbb{E}_{q_\phi}\left[\log p_\theta(\mathbf{z},\mathbf{x})\right] + \mathbb{E}_{q_\phi}\left[\log p_\theta(\mathbf{x})\right] && (29)\\
&= \log p_\theta(\mathbf{x}) - \mathbb{E}_{q_\phi}\left[\log p_\theta(\mathbf{z},\mathbf{x}) - \log q_\phi\right] && (30)
\end{aligned}
$$

where $p_\theta = p_\theta(\mathbf{z} \mid \mathbf{x})$ and $q_\phi = q_\phi(\mathbf{z} \mid \mathbf{x})$. The KL divergence helps us to rewrite the log-likelihood $\log p_\theta(\mathbf{x})$ as the sum of the KL divergence plus a term called the Evidence Lower Bound (ELBO). Because the KL divergence is always non-negative, the ELBO is a lower bound on the log evidence:

$$
\begin{aligned}
\log p_\theta(\mathbf{x}) &= D_{KL}\left(q_\phi \| p_\theta\right) + \mathbb{E}_{q_\phi}\left[\log p_\theta(\mathbf{z},\mathbf{x}) - \log q_\phi\right] && (31)\\
\log p_\theta(\mathbf{x}) &\geq \mathbb{E}_{q_\phi}\left[\log p_\theta(\mathbf{z},\mathbf{x}) - \log q_\phi\right] && (32)
\end{aligned}
$$

We can interpret the KL divergence as the gap between the log likelihood $\log p_\theta(\mathbf{x})$ and the ELBO. The smaller the KL divergence, the tighter the bound. Therefore, maximizing the ELBO both maximizes the log likelihood and minimizes the KL divergence. In this way, the generative model parameters $\theta$ and the inference model parameters $\phi$ are optimized simultaneously by maximizing the ELBO.

### 6.1.6 Stochastic Gradient Descent

To maximize the ELBO, given a dataset $\mathcal{D}$ with samples $\mathbf{x} \sim \mathcal{D}$, we define:

$$
\mathcal{L}(\mathbf{x}) = -\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\log \frac{p_\theta(\mathbf{z},\mathbf{x})}{q_\phi(\mathbf{z} \mid \mathbf{x})}\right] \tag{33}
$$

We maximize $\mathcal{L}(\mathbf{x})$ by stochastic gradient ascent over mini-batches. The gradient with respect to parameters $\theta$ and $\phi$ is:

$$
\nabla_{\theta,\phi}\mathcal{L}(\mathbf{x}) = -\nabla_{\theta,\phi}\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\log \frac{p_\theta(\mathbf{z},\mathbf{x})}{q_\phi(\mathbf{z} \mid \mathbf{x})}\right] \tag{34}
$$

Differentiating the ELBO with respect to $\theta$ is straightforward because $q_\phi(\mathbf{z} \mid \mathbf{x})$ does not depend on $\theta$. Thus, we can move the gradient inside the expectation and approximate it by

sampling:

$$\nabla_\theta \left( \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} \left[ \log p_\theta(\mathbf{z}, \mathbf{x}) - \log q_\phi(\mathbf{z} \mid \mathbf{x}) \right] \right) \tag{35}$$

$$= \nabla_\theta \left( \int_{\mathbf{z}} q_\phi(\mathbf{z} \mid \mathbf{x}) \left[ \log p_\theta(\mathbf{z}, \mathbf{x}) - \log q_\phi(\mathbf{z} \mid \mathbf{x}) \right] d\mathbf{z} \right) \tag{36}$$

$$= \int_{\mathbf{z}} q_\phi(\mathbf{z} \mid \mathbf{x}) \nabla_\theta \left[ \log p_\theta(\mathbf{z}, \mathbf{x}) - \log q_\phi(\mathbf{z} \mid \mathbf{x}) \right] d\mathbf{z} \tag{37}$$

$$= \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} \nabla_\theta \left[ \log p_\theta(\mathbf{z}, \mathbf{x}) - \log q_\phi(\mathbf{z} \mid \mathbf{x}) \right] \tag{38}$$

$$\approx \frac{1}{L} \sum_{i=1}^{L} \nabla_\theta \left[ \log p_\theta(\mathbf{z}, \mathbf{x}) \right] \tag{39}$$

However, differentiating with respect to $\phi$ is more difficult because $q_\phi(\mathbf{z}|\mathbf{x})$ itself depends on $\phi$. This creates a problem: the gradient involves terms that are not simple expectations and cannot be easily estimated by sampling.

$$\nabla_\phi \left( \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} \left[ \log p_\theta(\mathbf{z}, \mathbf{x}) - \log q_\phi(\mathbf{z} \mid \mathbf{x}) \right] \right) \tag{40}$$

$$= \nabla_\phi \left( \int_{\mathbf{z}} q_\phi(\mathbf{z} \mid \mathbf{x}) \left[ \log p_\theta(\mathbf{z}, \mathbf{x}) - \log q_\phi(\mathbf{z} \mid \mathbf{x}) \right] d\mathbf{z} \right) \tag{41}$$

$$= \int_{\mathbf{z}} \nabla_\phi \left[ q_\phi(\mathbf{z} \mid \mathbf{x}) \cdot \mathrm{ELBO} \right] d\mathbf{z} \tag{42}$$

$$= \int_{\mathbf{z}} q_\phi(\mathbf{z} \mid \mathbf{x}) \cdot \nabla_\phi \mathrm{ELBO} \; d\mathbf{z} + \int_{\mathbf{z}} \mathrm{ELBO} \cdot \nabla_\phi q_\phi(\mathbf{z} \mid \mathbf{x}) \; d\mathbf{z} \tag{43}$$

$$= \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} \left[ \nabla_\phi \mathrm{ELBO} \right] + \int_{\mathbf{z}} \mathrm{ELBO} \cdot \nabla_\phi q_\phi(\mathbf{z} \mid \mathbf{x}) \; d\mathbf{z} \tag{44}$$

The second term cannot be easily estimated because it is not in the form of the expectation, and there is no simple analytical solution available for it.

### 6.1.7 Reparameterization trick

To solve this, we use the reparameterization trick. Instead of sampling $\mathbf{z}$ directly from $q_\phi(\mathbf{z} \mid \mathbf{x})$, we express $\mathbf{z}$ as a deterministic function $\mathcal{T}(\phi, \mathbf{x}, \epsilon)$ of $\phi$, $\mathbf{x}$, and a noise variable $\epsilon$ drawn from a fixed and simple distribution $p(\epsilon)$. In this way, the randomness is isolated in $\epsilon$, and the gradient with respect to $\phi$ can flow through $\mathcal{T}$.

$$\mathcal{L}(\mathbf{x}) = -\mathbb{E}_{p(\epsilon)} \left[ \log p_\theta(\mathbf{z}, \mathbf{x}) - \log q_\phi(\mathbf{z} \mid \mathbf{x}) \right] \tag{45}$$

$$\nabla_{\theta,\phi} \mathcal{L}(\mathbf{x}) \approx -\frac{1}{L} \sum_{i=1}^{L} \nabla_{\theta,\phi} \left[ \log p_\theta(\mathbf{z}, \mathbf{x}) - \log q_\phi(\mathbf{z} \mid \mathbf{x}) \right] \tag{46}$$

### 6.1.8 ELBO

We can rewrite the ELBO as follow:

$$\begin{aligned}
\text{ELBO} &= \mathbb{E}_{p(\epsilon)}\left[\log p_\theta(\mathbf{z},\mathbf{x}) - \log q_\phi(\mathbf{z}\mid\mathbf{x})\right] && (47)\\
&= \mathbb{E}_{p(\epsilon)}\left[\log\left(p_\theta(\mathbf{x}\mid\mathbf{z})\cdot p_\theta(\mathbf{z})\right) - \log q_\phi(\mathbf{z}\mid\mathbf{x})\right] && (48)\\
&= \mathbb{E}_{p(\epsilon)}\left[\log p_\theta(\mathbf{x}\mid\mathbf{z}) + \log p_\theta(\mathbf{z}) - \log q_\phi(\mathbf{z}\mid\mathbf{x})\right] && (49)\\
&= \mathbb{E}_{p(\epsilon)}\left[\log p_\theta(\mathbf{x}\mid\mathbf{z})\right] - \mathbb{E}_{p(\epsilon)}\left[\log\frac{q_\phi(\mathbf{z}\mid\mathbf{x})}{p_\theta(\mathbf{z})}\right] && (50)\\
&= \mathbb{E}_{p(\epsilon)}\left[\log p_\theta(\mathbf{x}\mid\mathbf{z})\right] - D_{KL}\left(q_\phi(\mathbf{z}\mid\mathbf{x})\|p_\theta(\mathbf{z})\right) && (51)
\end{aligned}$$

To make sampling differentiable with respect to $\phi$, we use the reparameterization trick. We model $q_\phi(\mathbf{z}\mid\mathbf{x})$ as a Gaussian distribution with mean $\boldsymbol{\mu}$ and standard deviation $\boldsymbol{\sigma}$, both predicted by a neural network $f(\mathbf{x})$. Instead of sampling $\mathbf{z}$ directly, we sample noise $\boldsymbol{\epsilon}$ from a simple distribution $p(\boldsymbol{\epsilon})$ (usually standard normal), and compute $\mathbf{z} = \mathcal{T}(\phi,\mathbf{x},\boldsymbol{\epsilon}) = \boldsymbol{\mu} + \boldsymbol{\sigma}\odot\boldsymbol{\epsilon}$. The prior $p_\theta(\mathbf{z})$ is typically chosen as a standard normal distribution $\mathcal{N}(\mathbf{0},\mathbf{I})$.

For Gaussian latent variables, both the approximate posterior $q_\phi(\mathbf{z}\mid\mathbf{x})$ and the prior $p_\theta(\mathbf{z})$ are normal distributions:

$$\mathcal{N}(\mathbf{z};\boldsymbol{\mu},\boldsymbol{\sigma}) = \frac{1}{\boldsymbol{\sigma}\sqrt{2\pi}}\exp\left(-\frac{1}{2}\left(\frac{\mathbf{z}-\boldsymbol{\mu}}{\boldsymbol{\sigma}}\right)^2\right) \tag{52}$$

This allows us to write an analytical form for the KL divergence. Starting from the definition of KL divergence between two Gaussians, we substitute the Gaussian PDF and simplify the expression step by step. Using known properties of expectations under a Gaussian distribution, the integrals can be solved, leading to a closed-form formula for the KL term:

$$\begin{aligned}
D_{KL}\left(q_\phi(\mathbf{z}\mid\mathbf{x})\|p_\theta(\mathbf{z})\right) &= \int_{\mathbf{z}} q_\phi(\mathbf{z}\mid\mathbf{x})\cdot\log\left(\frac{q_\phi(\mathbf{z}\mid\mathbf{x})}{p_\theta(\mathbf{z})}\right)d\mathbf{z} && (53)\\
&= \int_{\mathbf{z}} q_\phi(\mathbf{z}\mid\mathbf{x})\cdot\log\left(\frac{1}{\boldsymbol{\sigma}}e^{-\frac{1}{2}\left[\left(\frac{\mathbf{z}-\boldsymbol{\mu}}{\boldsymbol{\sigma}}\right)^2-\mathbf{z}^2\right]}\right)d\mathbf{z} && (54)\\
&= -\frac{1}{2}\int_{\mathbf{z}} q_\phi(\mathbf{z}\mid\mathbf{x})\left[\log\boldsymbol{\sigma}^2 - \mathbf{z}^2 + \frac{1}{\boldsymbol{\sigma}^2}(\mathbf{z}-\boldsymbol{\mu})^2\right]d\mathbf{z} && (55)\\
&= -\frac{1}{2}\left[\log\boldsymbol{\sigma}^2\int_{\mathbf{z}} q_\phi(\mathbf{z}\mid\mathbf{x})d\mathbf{z} - \int_{\mathbf{z}}\mathbf{z}^2 q_\phi(\mathbf{z}\mid\mathbf{x})d\mathbf{z} + \frac{1}{\boldsymbol{\sigma}^2}\int_{\mathbf{z}}(\mathbf{z}-\boldsymbol{\mu})^2 q_\phi(\mathbf{z}\mid\mathbf{x})d\mathbf{z}\right] && (56)
\end{aligned}$$

We use the following known moment identities of the Gaussian distribution:

$$\int_{\mathbf{z}} q_\phi(\mathbf{z}\mid\mathbf{x})\,d\mathbf{z} = 1 \quad \text{(Normalization)} \tag{57}$$

$$\int_{\mathbf{z}}\mathbf{z}^2\,\mathcal{N}(\mathbf{z};\boldsymbol{\mu},\boldsymbol{\sigma})\,d\mathbf{z} = \boldsymbol{\mu}^2 + \boldsymbol{\sigma}^2 \quad \text{(Second moment)} \tag{58}$$

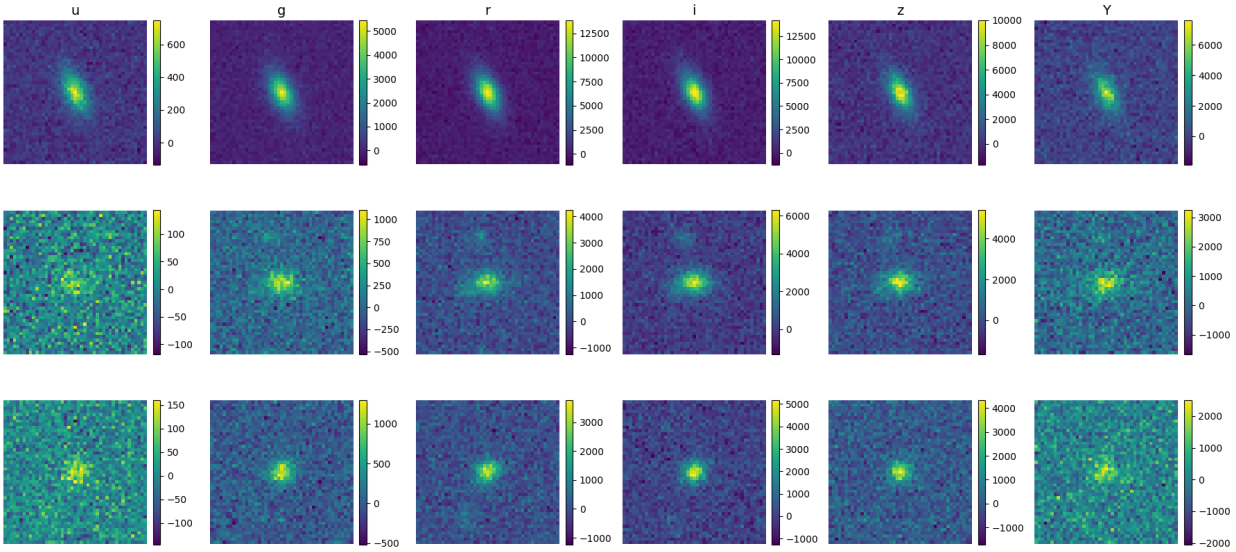
$$\int_{\mathbf{z}}(\mathbf{z}-\boldsymbol{\mu})^2\,\mathcal{N}(\mathbf{z};\boldsymbol{\mu},\boldsymbol{\sigma})\,d\mathbf{z} = \boldsymbol{\sigma}^2 \quad \text{(Variance)} \tag{59}$$

Substituting into the KL divergence expression, we get the final closed form:

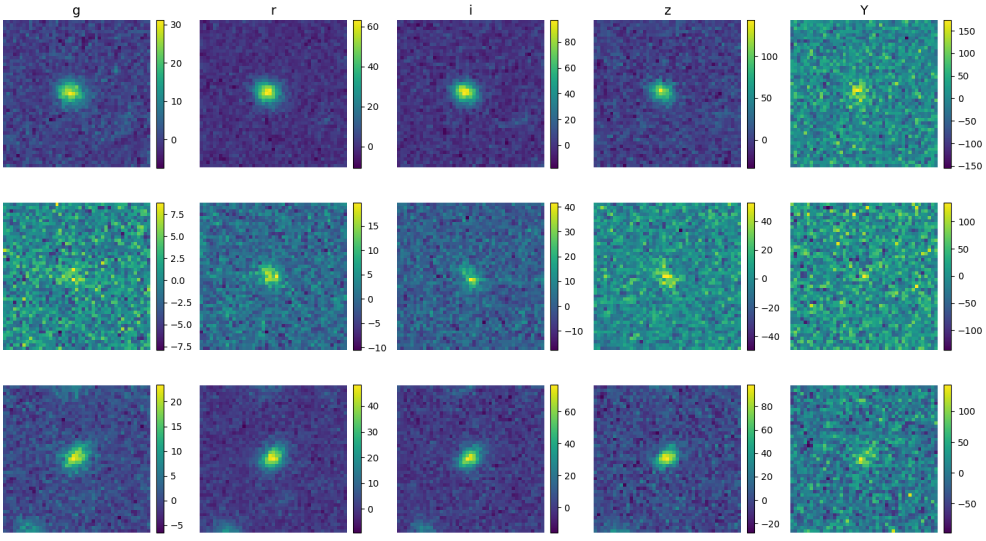
$$D_{KL}\left(q_\phi(\mathbf{z}\mid\mathbf{x})\|p_\theta(\mathbf{z})\right) = -\frac{1}{2}\left[\log\boldsymbol{\sigma}^2 - \boldsymbol{\mu}^2 - \boldsymbol{\sigma}^2 + 1\right] \tag{60}$$

## 6.2 Galaxy images in different bands

Both the Dark Energy Survey (DES) and the Vera C. Rubin Observatory's Legacy Survey of Space and Time (LSST) observe galaxies in multiple photometric bands, each capturing light at different wavelengths.




**(a)** sim LSST 6 bands




**(b)** DES 5 bands

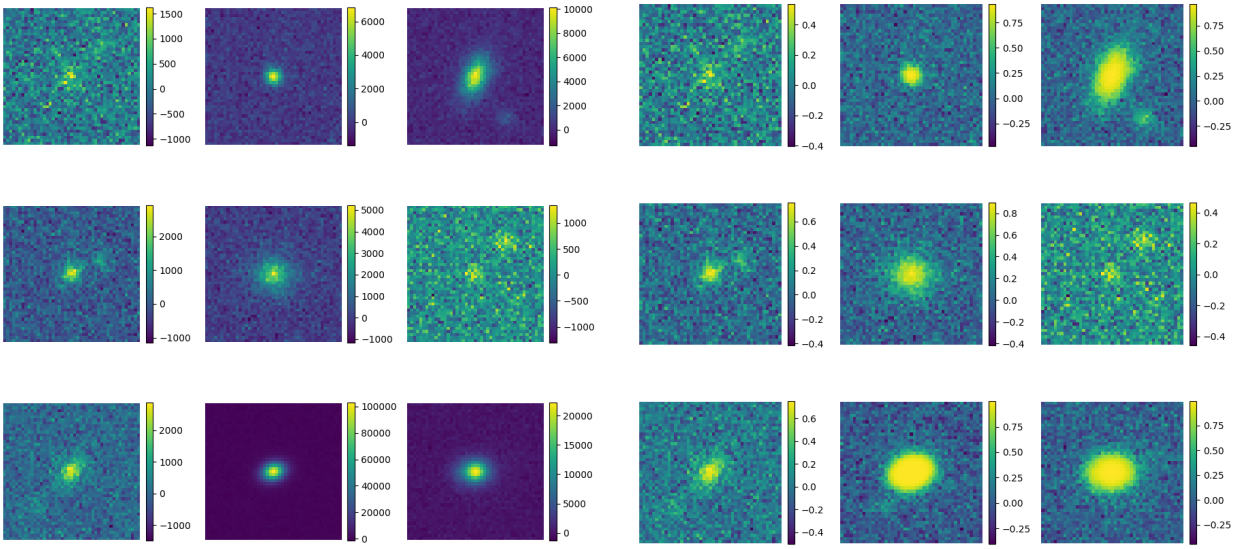
**Figure 19:** sim LSST 6 bands and DES 5 bands

DES uses five bands: $g$, $r$, $i$, $z$, and $Y$. These bands cover light from the blue-green part of the spectrum (g-band) through the red and near-infrared regions ($r$, $i$, $z$, and $Y$-bands).

The LSST observes in six bands: $u$, $g$, $r$, $i$, $z$, and $Y$. Compared to DES, LSST adds an extra band in the near ultraviolet ($u$-band). These multiband observations help capture more information about galaxy colors and redshifts, which is important for weak lensing studies and galaxy characterization.

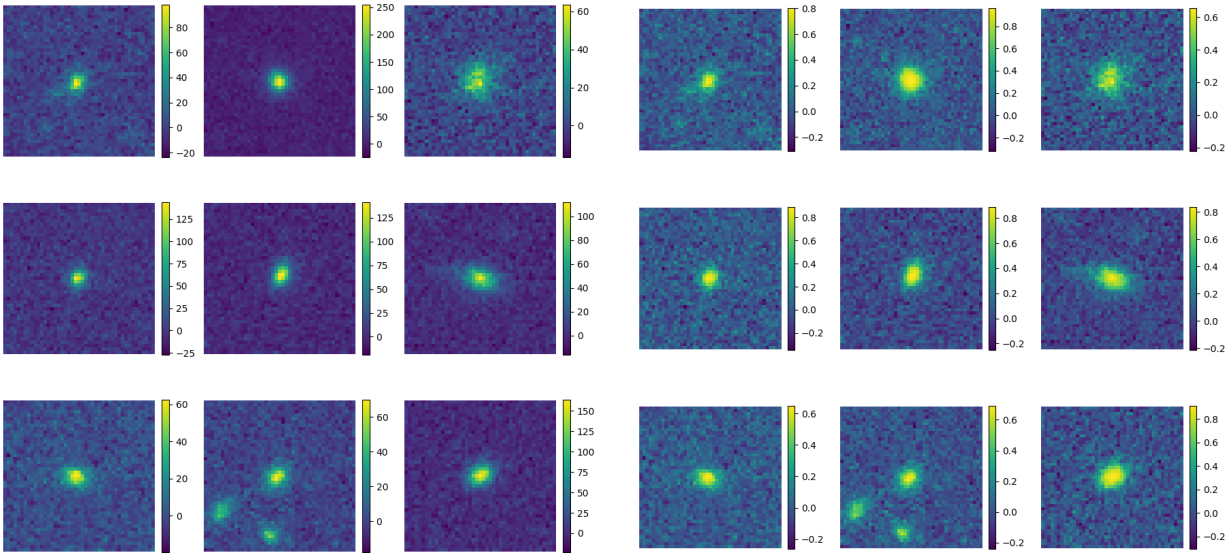
## 6.3 The effects of normalization function

Figures 20 and Figure 21 present examples of simulated galaxy images from LSST and real images from DES in the $r$-band. For both datasets, the left panels show the raw images with the original pixel values. The right panels display the normalized images, where the pixel values are rescaled to a standard range.



**(a)** Raw simulated LSST images ($r$-band) **(b)** Normalized simulated LSST images ($r$-band)

**Figure 20:** Comparison between raw and normalized simulated LSST images in the $r$-band.

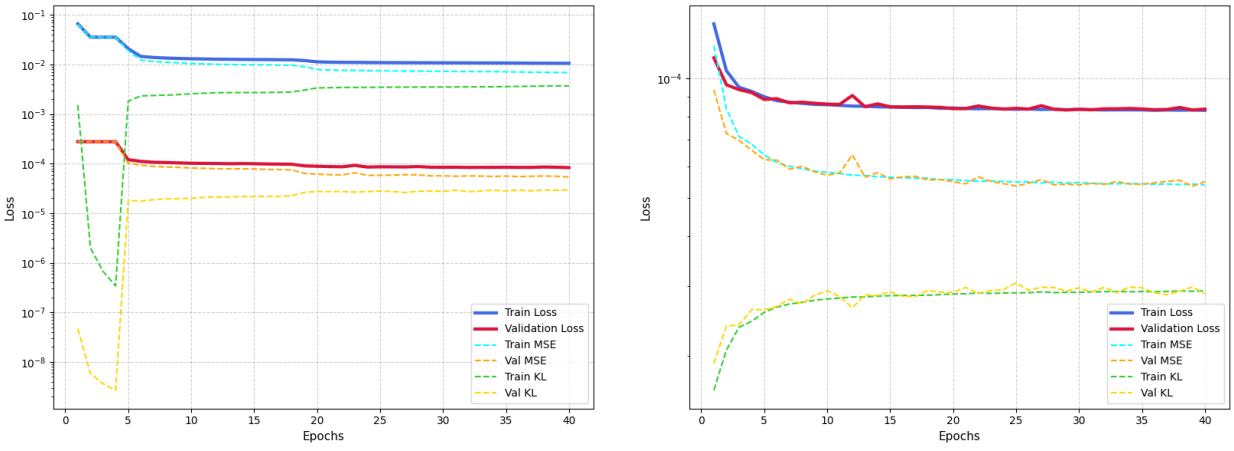


**(a)** Raw DES images ($r$-band) **(b)** Normalized DES images ($r$-band)

**Figure 21:** Comparison between raw and normalized DES images in the $r$-band.

## 6.4 Loss Functions

Figure 22 shows the evolution of the loss function during VAE training. Panel (a) presents the loss after *Training 1*, which focuses on learning the general features of isolated galaxies. Panel (b) shows the loss after *Training 2*, where the VAE is fine-tuned on blended galaxy images. In both cases, the total loss decreases over epochs, indicating that the model is learning to reconstruct the input data while keeping the latent space distribution close to the prior target.




**(a)** Loss function of VAE after *Training 1* **(b)** Loss function of VAE after *Training 2*

**Figure 22:** Loss function of VAE

Figure 23 shows the evolution of the loss function during the training of the regressor. The loss decreases smoothly over epochs, showing that the regressor learns to map the latent variables to the correct galaxy parameters. This steady reduction in loss indicates good convergence of the training process.

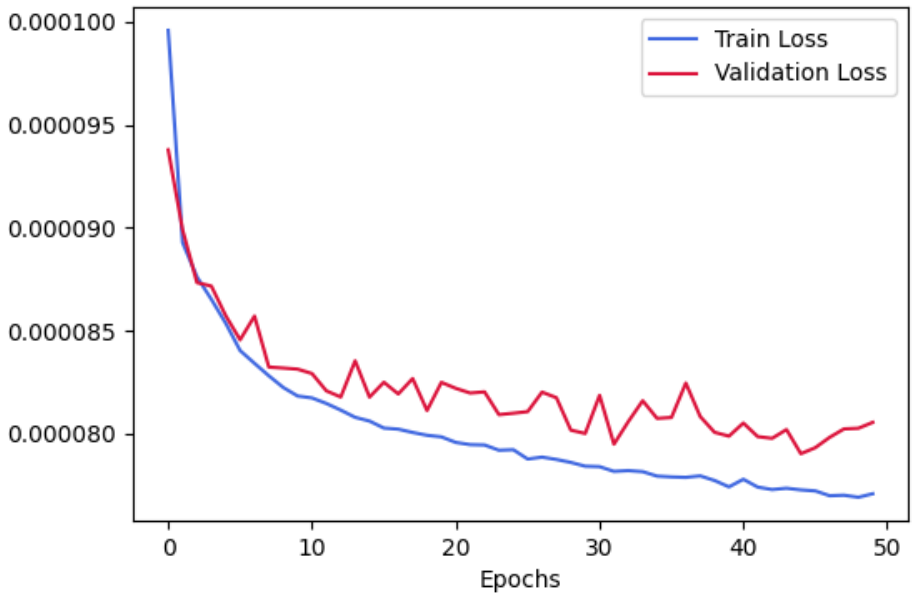



**Figure 23:** Loss function of Regressor