authentic (belonging to the problem domain) or counterfeit (generated by the generator model).

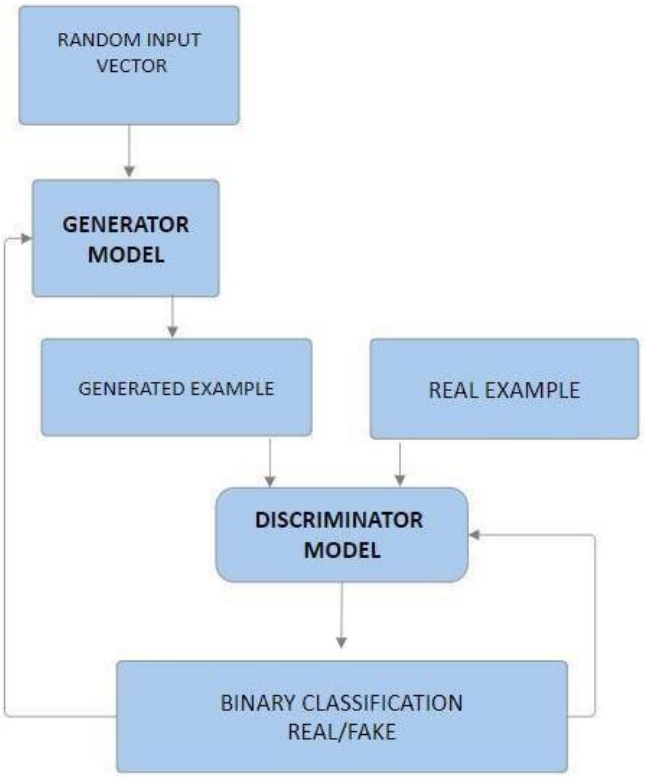

Figure 1.1 – Example of GAN model architecture

Generator component serves as a model that generates credible instances within the problem domain while the discriminator functions as a model for classifying instances as either authentic (representative of the domain) or synthetic (produced by the generator). Generative adversarial networks are grounded in a game-theoretic scenario wherein the generatornetwork engages in a competition with its adversary, the discriminator network. The generator network is tasked with directly creating samples, while the discriminator network strives to differentiate between samples drawn from the training data and those generated by the generator.

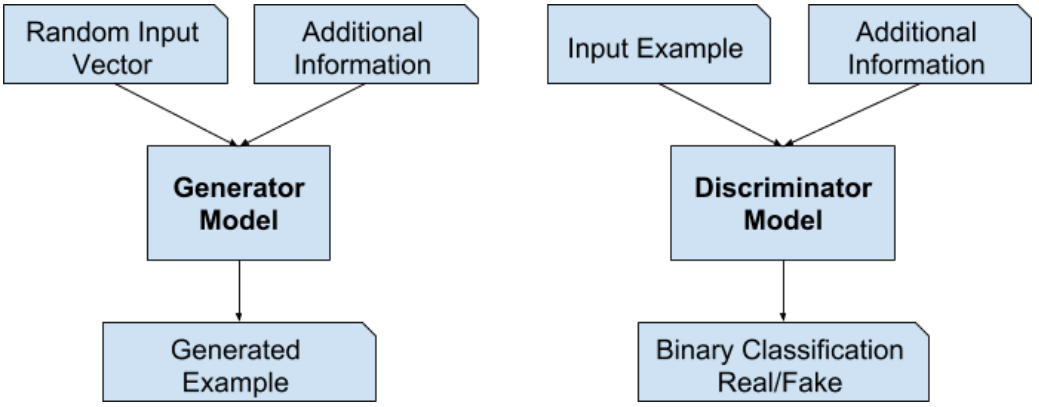

Figure 1.2 Example of Conditional GANs architecture

This supplementary input may consist of class labels, such as "male"or "female" for generating images of people, or numerical values, particularly in tasks like creating images of handwritten digits. The discriminator in conditional GANs is also conditioned, meaningit receives both an input image (real or fake) and the additional input information. For instance, in cases with classification labels as conditionalinput, the discriminator anticipates that the input belongs to a certain class. This dynamic encourages the generator to produce examples aligned withthe specified class, ultimately deceiving the discriminator.

Conditional GANs can be effectively employed to generate instances belonging to a specific category within a domain. Moreover, GAN models can take conditioning a step further by using an existing example from the domain, such as an image, as a reference. This feature enables various applications, including text-to-image and image-to-image translations. This functionality facilitates impressive applications of GANs, like style transfer, photo colorization, or the transformation of images from one season or time of day to another. In scenarios involving conditional GANs for image-to-image translation, such as converting day images to night, the discriminator is provided with both real and generated nighttime images and, conditioned on, real daytime photos as inputs. Meanwhile, the generator utilizes a random vector from the

inspecting the individual generator units, exploratory endeavors extend to the latent space. This is where we select the latent vector for input into the generator, unraveling the latent space's concealed,interpretable dimensions. In this context, the pre-trained generator is denoted and a random vector sampled from the latent space culminate in the output image. As distinct vectors are utilized, the produced images assume distinct forms. Thus, the latent space becomes a repository for diverse image attributes. If we can pinpoint latent subspaces aligned with human-comprehensible concepts, it becomes feasible to steer the image creation process by adjusting the latent code along these pertinent subspaces.

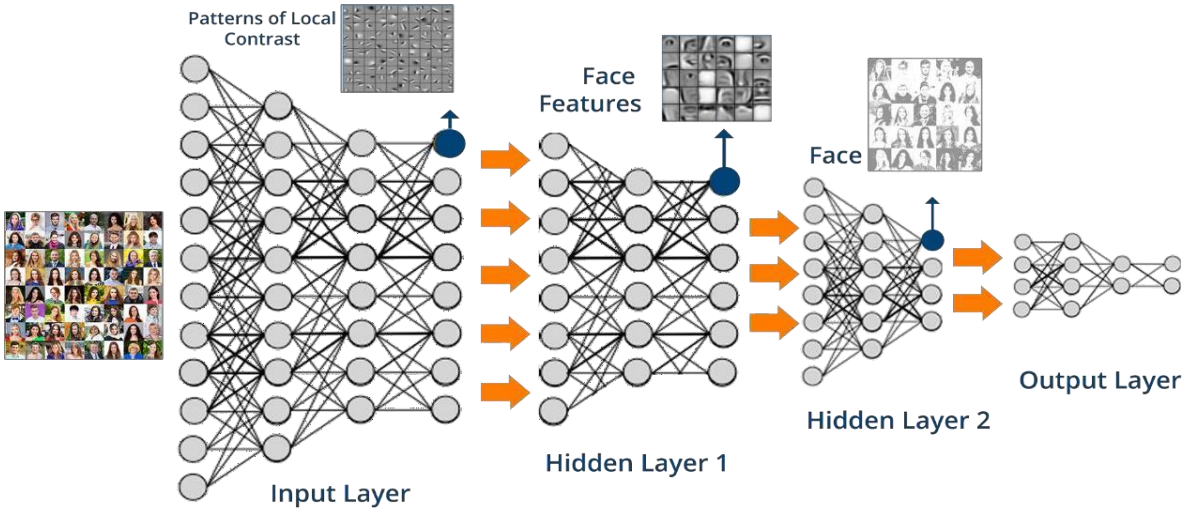

Figure 1.3 Using Contrast to identify Colors

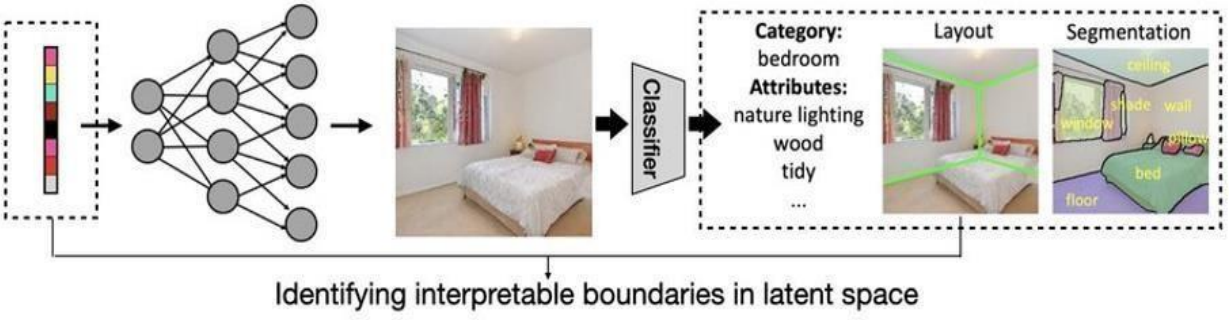

Figure 1.4 Exploring attribute boundaries

To establish a connection between the latent and semantic spaces, we can

CHAPTER 4

SYSTEM ARCHITECTURE

4.1 ARCHITECURE

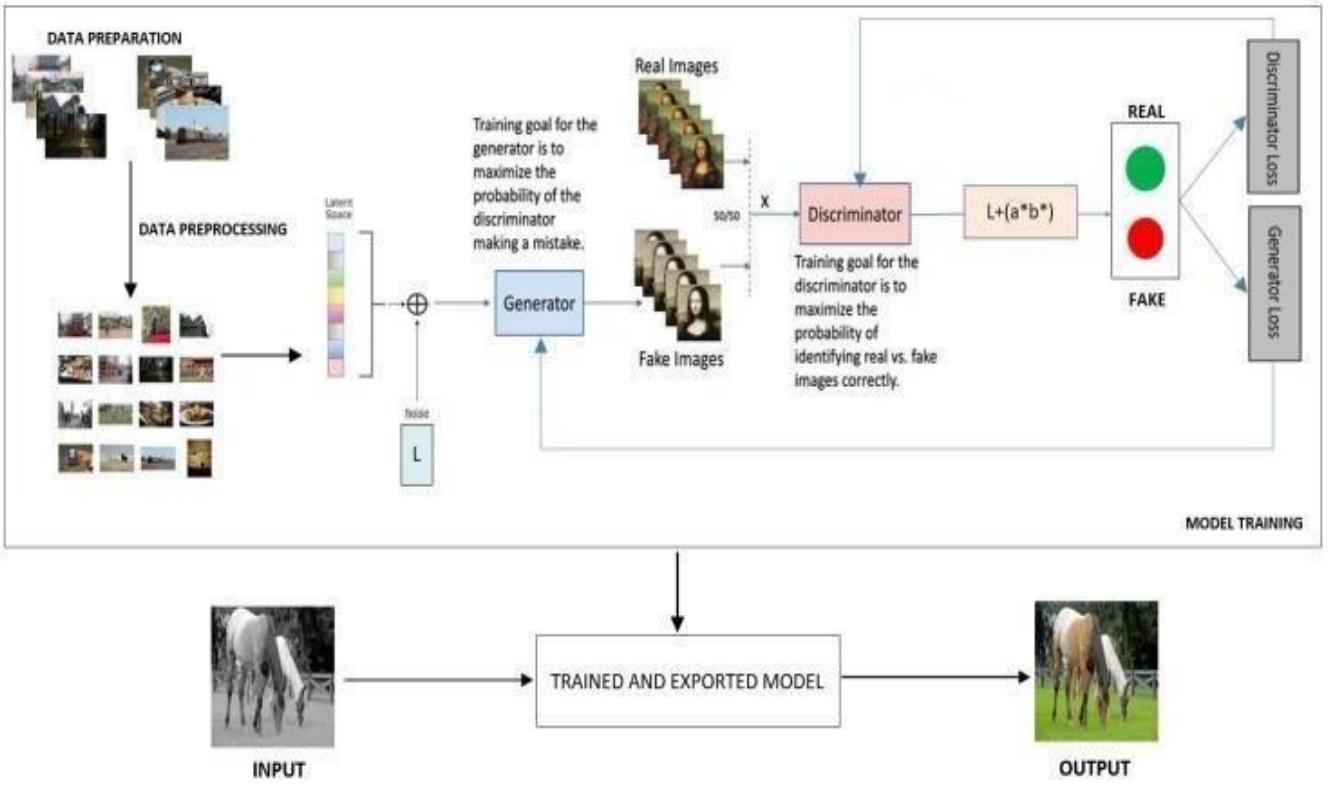

Figure 4.1 Architecture Diagram for GANs

4.2 COMPONENTS AND ITS WORKING

Colorization is a remarkable application of Generative Adversarial Networks (GANs), where the Generator, Discriminator, latent space, noise, loss functions, and fine-tune training all work in tandem to transform grayscale images into realistic, vibrant color images. Let's explore how these elements collaborate in the colorization process:

levels of image detail. As the network progresses through the UNetBlocks, it can capture both fine and coarse features, enabling it to produce high-quality colorizations. Furthermore, the UNet architecture incorporates dropout layers to prevent overfitting during training. These dropout layers introduce regularization, improving the network's generalization capabilities.

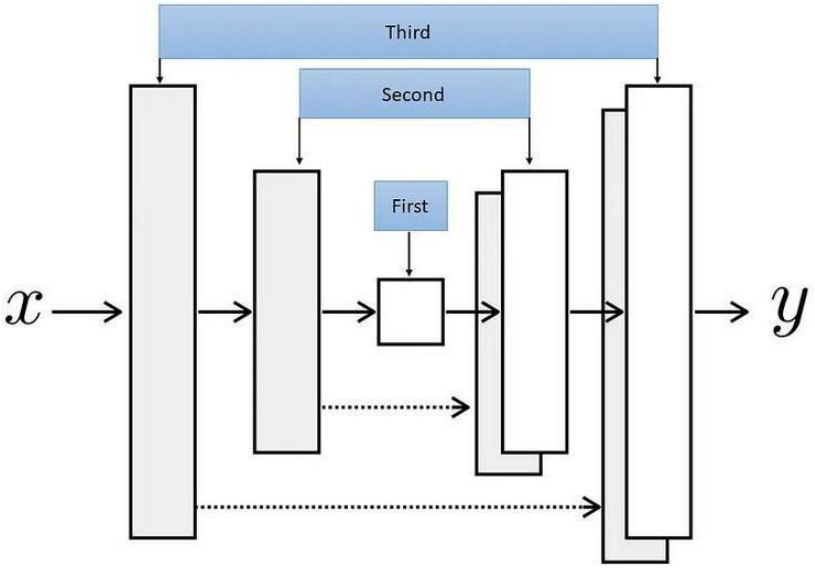

Figure 6.1 UNet from the paper Image-to-Image translation with Conditional Adversarial Networks

6.4 DISCRIMINATOR USING PatchGAN

The discriminator, known as a PatchGAN, plays a crucial role in the adversarial learning process of the colorization model. This component is responsible for distinguishing between real and generated colorized images. The discriminator is designed as a neural network that assesses localized patches within an image to provide feedback to the generator.

The PatchDiscriminator is defined as a neural network module. It acceptsan input image or feature map and processes it to make binary judgments aboutthe realism of the image patches. To achieve this, the discriminator is equippedwith

Log Results

Logging is a critical aspect of research, and the `log_results` function fulfills this role by printing the average values of loss meters tothe console. This real-time feedback aids in monitoring training and evaluating the model's convergence. These utility functions are the backbone of the research framework, offering efficiency and transparency in managing the training process, monitoring model performance, and ensuring the robustness of the colorization model.

6.8 RESULTS

TRAINING PROCESS AND LOSS METRICS

Iteration 200/500:

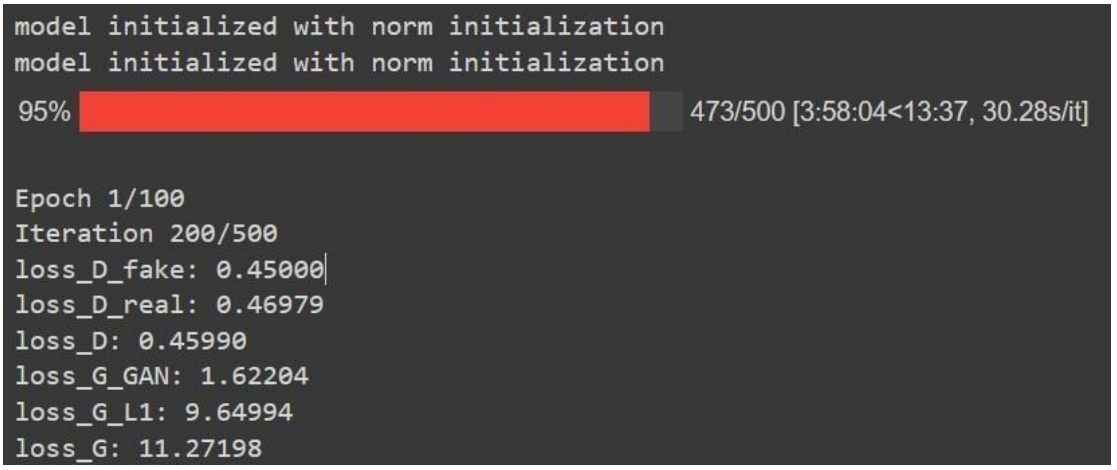

Figure 6.2 Metrics Visualization

In the provided output, we have two main sections:

Model Initialization:

"model initialized with norm initialization" is a message indicatingthat the model has been initialized with normalized weights. This typically means that the model's weights have been set using a normalization technique to ensure stable and efficient training. Proper weight initialization is crucial for the convergence and

performance of neural networks.

Training Progress:

- "95%" suggests the completion progress of a training process,specifically, 95% of the process is complete.
- "473/500" indicates that the training process has iterated throught
- 473 out of 500 total iterations.
- "[3:58:04<13:37, 30.28s/it]" shows the training time progress.

Following this, the output provides details about the training process for a specific epoch (Epoch 1/100) and iteration (Iteration 200/500):

"loss_D_fake," "loss_D_real," "loss_D," "loss_G_GAN," "loss_G_L1," and "loss_G" are various loss metrics associated with training a GAN (Generative Adversarial Network). The numbers associated with these loss metrics indicate the valueof each loss at the given training stage. For example, "loss_D_fake: 0.45000" suggests the loss value for the discriminator when evaluating fake data.

OUTPUT

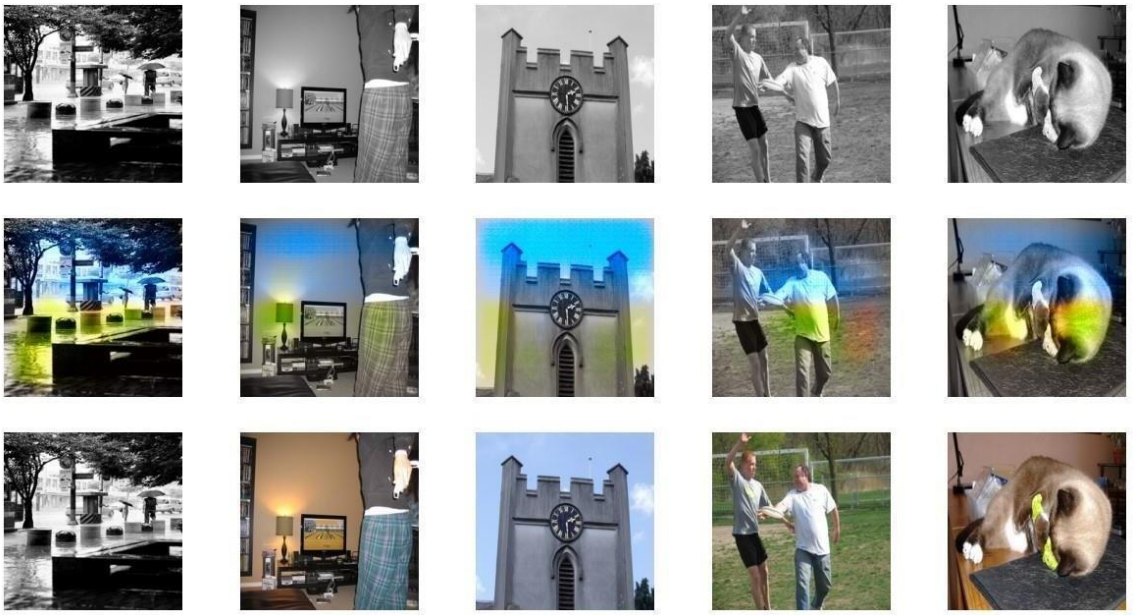

Figure 6.3 Output after 200 iterations

Iteration 400/500:

```
Epoch 1/100
Iteration 400/500
loss_D_fake: 0.47620
loss_D_real: 0.49065
loss_D: 0.48342
loss_G_GAN: 1.55394
loss_G_L1: 9.96936
loss_G: 11.52330
```

Figure 6.4 Metrics of a later stage

Figure 6.5 Output after 400 iterations

VISUALIZATION IN EARLIER STAGES FOR ANALYSIS

```
_, axes = plt.subplots(4, 4, figsize=(10, 10))
for ax, img_path in zip(axes.flatten(), train_paths):
    ax.imshow(Image.open(img_path))
    ax.axis("off")
```

Figure 6.6 Creates a 4x4 grid for visualization

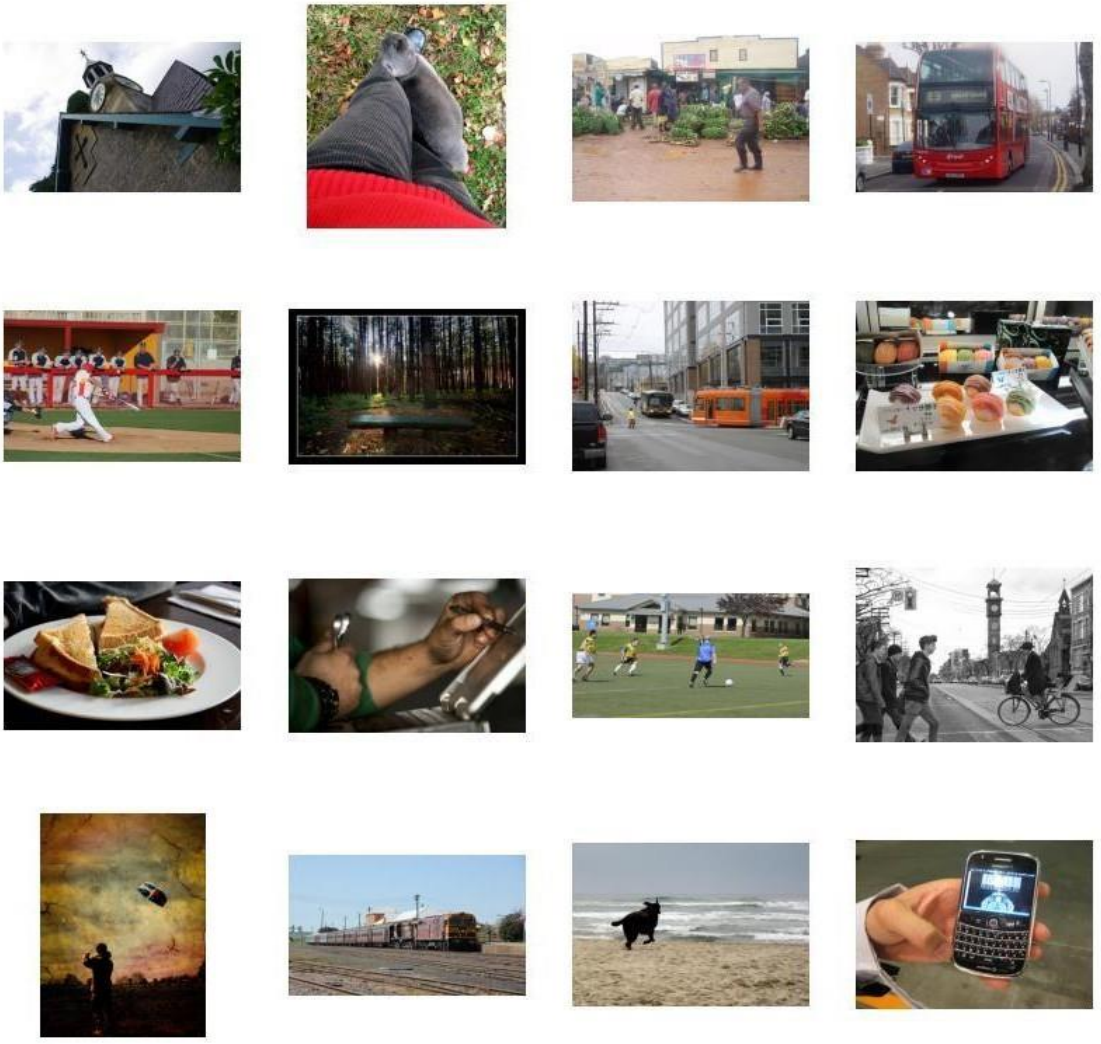

Figure 6.7 4x4 Grid