Faculty of Fundamental Problems of Technology
Wrocław University of Science and Technology

Generative Adversarial Networks in comic art design

Szymon Kowalczyk
Student's ID number: 243079

Master's Thesis supervisor:
prof. dr. hab. inż. Halina Kwaśnicka

Wrocław 2023

Contents

1 Introduction

Cartoons, an influential art form, have left a lasting impact on popular culture throughout history. Their distinctive visual style, vibrant colors, and imaginative narratives have elevated cartoons from mere animated entertainment to an integral part of our everyday lives. They continue to captivate audiences with their ability to blend humor and social commentary, making them a beloved and enduring medium of entertainment and storytelling. Cartoons have become a cultural phenomenon, shaping our perceptions, sparking creativity, and providing a source of joy and inspiration [32]. From animated films and television shows to comic strips and digital media, cartoons have captured the hearts and minds of people across generations [28]. The history of cartoons can be traced back to the early 20th century when animated shorts and comic strips first emerged. Over time, cartoons have evolved and diversified, adopting various styles and techniques to cater to different artistic visions and social trends [48]. Cartoons have become an integral part of visual communication, captivating audiences of all ages and permeating countless aspects of our cultural and visual landscape. Animated creations have extended far beyond conventional mediums, finding their place in advertising, education, social media, and other digital applications, serving as an impactful tool in communication and narrative expression. The ability of cartoons to convey complex ideas through simplified visuals and humor makes them engaging for diverse audiences. However, the creation of cartoon pictures is a meticulous and time-consuming process that demands a high level of artistic skill and attention to detail [41]. From character design and expressive gestures to backgrounds and storytelling compositions, each element requires careful consideration to bring the cartoons to life [52]. Artists must possess a deep understanding of visual storytelling, anatomy, and composition to effectively convey the intended message and evoke emotions through their artwork. The time and effort required for manual creation present challenges in meeting the growing demand for fresh and visually captivating cartoon content [23].

The process of cartoonizing a photo has garnered significant interest in computer vision and image processing. Various methods have been explored to transform real-world photos into cartoon-style images, including the application of filters and style transfer techniques. Filters, such as edge detection and color quantization, have been widely employed to simplify image features and enhance cartoon-like characteristics [17]. However, these filter-based methods often result in oversimplified and visually inconsistent outputs, lacking the intricate details and artistic nuances associated with hand-drawn cartoons [31].

In recent years, style transfer methods have emerged as a promising approach to cartoonization. These methods aim to transfer the artistic style of a reference image onto a given photo. By leveraging deep learning and neural network architectures, style transfer algorithms attempt to blend the content of the photo with the desired style. Although significant success has been achieved with learning-based stylization, which can produce visually appealing results, state-of-the-art methods fail to produce cartoonized images with acceptable quality. They often face challenges in faithfully capturing the intricate stylistic details of cartoons. The inherent complexity and variation in cartoon styles make it difficult to achieve consistent and accurate style transfer, leading to artifacts and distortions in the final output. Furthermore, existing style transfer techniques heavily rely on paired datasets, where corresponding pairs of photos and stylized cartoons are required for training. This dependence on paired data limits the scalability and practicality of the methods, as acquiring large-scale paired datasets is laborious and time-consuming [17]. Moreover, style transfer methods struggle with preserving the semantic content of the original photo, often introducing unintended modifications or losing important visual details in the process.

Given the limitations and shortcomings of filter-based approaches and style transfer methods, there is a big need for more robust and effective techniques for cartoonizing photos. This thesis addresses these challenges by exploring the application of deep learning techniques, especially Generative Adversarial Networks (GANs), to automatically generate high-quality cartoon-style images from real photos. By training together two models,

the generator and the discriminator of the GAN, with a dataset of real photos and their corresponding cartoon-style images, the network learns to generate cartoons inspired by everyday scenes. This breakthrough not only accelerates the creation process for artists but also makes it easier for ordinary people to implement their own animation. Moreover, it opens up new avenues for exploring diverse visual styles and expanding the range of cartoon content available [51]. Various models of GANs for the automatic generation of cartoon-style images based on a given photo have been built, but there is still room for further improvements in this domain [8].
The goal of this thesis is to conduct a comprehensive evaluation and comparison of the results obtained from four state-of-the-art GAN architectures (CycleGAN, GANILLA, CartoonGAN, and AnimeGAN) [54, 20, 11, 9] on the same standardized dataset. The architectures discussed here have been specifically developed for the automated generation of stylized images, including the unique style of cartoons.
In addition to the models already discussed, the authors propose an alternative approach called LightAnimeGAN as an enhancement to the AnimeGAN network [9]. The LightAnimeGAN architecture aims to improve the generation of cartoon-style images by incorporating advancements and optimizations to the original AnimeGAN model. It leverages a lightweight generator model and uses layer normalization instead of instance normalization in the discriminator model. Our proposed architecture aims to reduce the number of model parameters, produce high-frequency artifacts, and enhance the visual aspects of the generated cartoon-style images. Lastly, we will calculate the Frechet Inception Distance metric and conduct a questionnaire-based comparison involving a diverse group of participants to evaluate the outputs generated by the proposed above GAN architectures. Through this analysis, we aim to gain valuable insights into the strengths, limitations, and overall performance of the models. By examining and analyzing the outputs, we can make informed assessments and draw conclusions about the effectiveness and suitability of the proposed GAN architectures for the task of generating cartoon or anime-style images.
The master's thesis is composed of seven chapters, each with its own distinct focus, forming a structured and logical flow of content. In the first chapter, the reader is acquainted with the topic, the core challenges it presents, and the overall structure of the thesis. Chapter 2 delves into a comprehensive background analysis. It aims to provide a deep understanding of the fundamental concepts and techniques of Generative Adversarial Networks, and discusses evaluation methods for GAN networks. Chapter 3 presents an overview of state of the art models from the literature, such as CycleGAN, CartoonGAN, GANILLA, and AnimeGAN. This chapter is devoted to outlining their architectural details, the loss functions they use, and summarizing the unique features each model offers. Chapter 4 introduces an alternative model, LightAnimeGAN, which builds on the structure of AnimeGAN but integrates key modifications designed to overcome its limitations and improve performance. Chapter 5 elaborates on the dataset used for training all the referenced models. It outlines the data preparation and preprocessing procedures applied to both real and style images. Chapter 6 outlines the conducted experiments, detailing the software and implementation process used, the training process, and providing an overall comparison between different models. This chapter presents the results from both Visual Inspection and Fréchet Inception Distance Analysis. It also includes a user study that evaluates the performance of the models by rating their cartoon outputs. Finally, Chapter 7 concludes the thesis, summarizing the key findings and outlining potential avenues for future research in this intriguing intersection of AI and art.

2 Background Analysis

The image-to-image (I2I) translation is a significant area of research in computer vision that focuses on transforming a source domain to a target domain, maintaining the underlying structure while modifying its appearance based on specific rules or styles. This technique has a wide range of applications, including style transfer, semantic segmentation, and image synthesis. At present, Neural Style Transfer and I2I Generative Adversarial Networks are two prominent methods that merge art and artificial intelligence [17, 25]. These techniques collectively present a novel approach to examining the potential of art and technology while constantly expanding the limits of AI-generated art.

Neural Style Transfer employs deep neural networks to apply the style of one image onto another while maintaining the content. It utilizes a pre-trained neural network to extract the style of a given image and apply it to a different image [17]. This method has been widely used to produce stylized images resembling well-known paintings and social media filters.

In contrast, I2I GANs focus on mapping images from a source domain to a target domain using generative adversarial networks. A notable example of I2I GANs is realistic human face synthesis [25]. By distinguishing between the style and structure of an image, I2I GANs offer enhanced control over the generated output, producing high-resolution images with intricate details. Additionally, these models can be customized using specific datasets to create artwork that emulates the appearance of the desired target domain.These techniques open up vast opportunities for delving into the convergence of art and technology. Neural Style Transfer allows artists to develop new artwork inspired by any painting, whereas I2I GAN offers a fresh means of producing original and realistic artwork.

In this chapter, the author presents and explains generative models, highlighting the differences between explicit and implicit generative models. Additionally, the fundamental concepts behind Style Transfer and Generative Adversarial Networks are elucidated. Lastly, the author explores the commonly used evaluation metrics that check the effectiveness of GANs.

2.1 Maximum Likelihood Estimation

Maximum Likelihood Estimation (MLE) is a statistical method defining a model that estimates probability distribution, parameterized by parameters θ based on observed data. The fundamental idea behind MLE is to find the values of the parameters that maximize the likelihood function $L(\theta|x^{(i)}) = \prod_{i=1}^{m} p_{model}$ where m is the dataset size for the training data $x^{(i)}$ with a given distribution. The likelihood function is defined as the probability of observing the given data, assumed distribution, and its parameters. It is calculated by multiplying the probabilities of individual data points together, assuming they are independent and identically distributed.

The goal of MLE is to find the parameter values that maximize the likelihood function. This is typically done by taking the derivative of the likelihood function with respect to the parameters and setting it to zero. However, instead of directly working with the likelihood function, it is often more convenient to work with the log-likelihood function (logarithm of the likelihood function) since it simplifies calculations and does not change the location of the maximum [18]. This leads to the concept of maximizing the log-likelihood function shown in Equation (2.1).

$$\arg\max_{\theta} \prod_{i=1}^{m} p_{\text{model}}(x^{(i)};\theta) = \arg\max_{\theta} \sum_{i=1}^{m} \log p_{\text{model}}(x^{(i)};\theta) \tag{2.1}$$

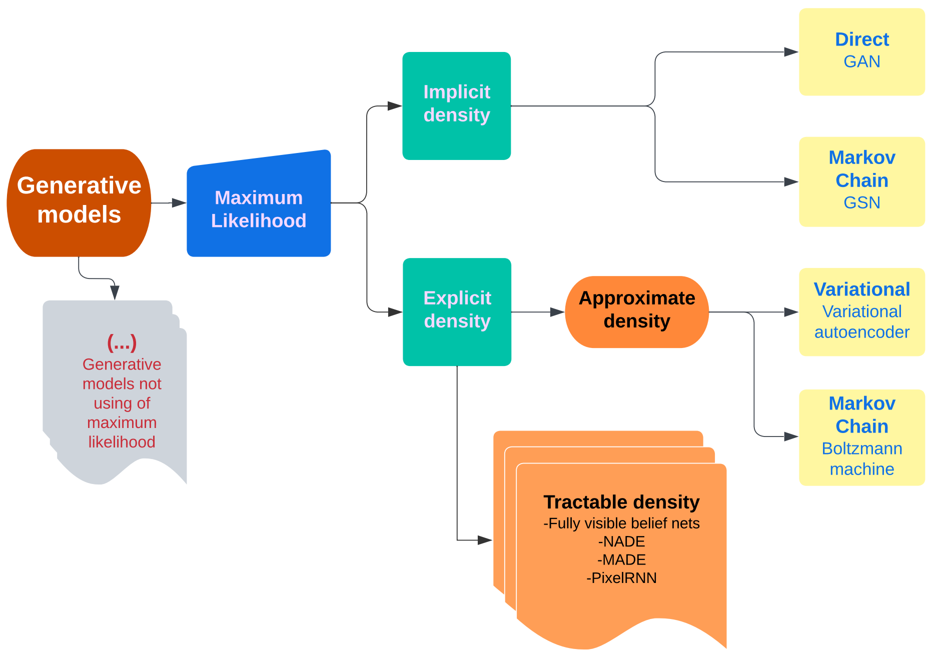


Figure 2.1: Taxonomy of generative models using the principle of maximum likelihood.

Maximizing the log-likelihood function is typically achieved using optimization algorithms, such as gradient descent or Newton's method. These methods iteratively update the parameter values in the direction of the steepest ascent of the log-likelihood function until convergence is reached.
We can also think of maximum likelihood estimation as minimizing the Kullback–Leibler divergence between the model and the data-generating distribution (see Equation (2.2)).

$$\theta^* = \arg\min_{\theta} D_{\mathrm{KL}}(p_{\mathrm{data}}(x) \| p_{\mathrm{model}}(x;\theta)) \tag{2.2}$$

In practical scenarios, we typically don't have a true distribution of p_{data}, but we estimate parameters that approximate probability in a model during the training process [18].

2.2 Generative Models

Generative models are a class of machine learning models that aim to learn and mimic the underlying training data distribution of a given dataset. These models are designed to generate new samples using learned probability distribution p_{model} that resemble the training data from a distribution p_{data}, enabling them to create new instances with similar characteristics and properties with some set variations [18]. But it is not always possible to learn the exact distribution of our training data either implicitly or explicitly and so we try to model a distribution that is as similar as possible to the true data distribution. For this, we can leverage the power of neural networks to learn a function that can approximate the model distribution to the true distribution. To simplify the discussion, we will focus only on generative models that work via the principle of maximum likelihood. By adopting this methodology, we can analyze and differentiate various models based on their approaches to calculating either the likelihood and its gradients or their approximations. Utilizing this method, we develop a taxonomy, as depicted in Figure 2.1, where each block possesses its own set of advantages and disadvantages.
Generative models can be broadly categorized into two main types: explicit and implicit generative models.

Explicit Generative Models

Explicit generative models, located in the bottom branch of the taxonomy illustrated in Figure 2.1, involve defining an explicit density function $p_{model}(x; \theta)$. In these models, maximizing the likelihood is straightforward, as we can directly substitute the model's density function into the likelihood function and follow the gradient uphill. However, a significant challenge in explicit density models lies in designing a model that can capture the complexity of the data while being computationally achievable [38]. To address this challenge, two strategies were designed. The first strategy called trackable density, involved the careful construction of models with guaranteed tractability. Tractable means that we can define a parametric function that is able to capture the distribution efficiently. These models are designed in a way that ensures their density function can be efficiently computed. Unfortunately, a lot of the distributions, for example, the distribution of images or speech waves, are complex and it is almost impossible to design a parametric function that explains them. In order to overcome the limitations of tractable density functions, an alternative approach was proposed. Approximate density maintains an explicit density function but uses variational methods or stochastic approximations. These models use approximations to maximize the likelihood. These approximations provide computationally feasible alternatives for modeling and training the explicit generative models. Gaussian Mixture Models (GMMs), Autoregressive Models (AMs), and Variational Autoencoders (VAEs) are among the commonly used explicit generative models [18]. GMMs capture the data distribution by combining multiple Gaussian components, where each component represents a distinct cluster or subpopulation. Autoregressive Models capture the dependencies between data points by modeling the conditional probability of each data point given the previous ones. VAEs leverage the power of both generative and inference models to learn a latent representation of the data, enabling the generation of new samples by sampling from the latent space.

Implicit Generative Models

Implicit generative models, located in the top branch of the taxonomy illustrated in Figure 2.1, are a class of generative models that do not explicitly model the probability distribution of the data. Instead, they offer a way to train the model sampling training data and interact indirectly with p_{model}. Instead of directly specifying the underlying density function as in explicit generative models, these models learn to capture the complex patterns and generate new samples through an optimization process [38]. These models have been successfully applied in various domains, including image synthesis, text generation, and audio generation. They can generate high-quality samples with rich details and capture the overall structure and diversity of the data distribution. One of the most well-known and widely used types of implicit generative models is the Generative Adversarial Network. GANs were specifically designed to address and overcome several disadvantages associated with other generative models. These disadvantages primarily revolved around issues related to modeling complex data distributions and generating high-quality samples. On the other hand, GANs have also presented a new drawback, as their training process requires finding the Nash equilibrium of a game, which is a really demanding optimization task [18].

2.3 Image-to-Image Translation

The image-to-image translation is the most popular application of GANs. It involves transforming images from one domain to another. Depending on the type of dataset used, the translation can be categorized into two types: paired image-to-image translation and unpaired image-to-image translation. An example depicting paired and unpaired image-to-image translation is shown in Figure 2.2.

Paired Image-to-Image Translation

Paired image-to-image translation characterize by a dataset containing pairs of corresponding images from two different domains. However, these pairs should represent the same or similar content but have different representations or styles. GANs that use paired datasets utilize supervised learning techniques to train the generator to transform images from the source domain to the target domain while maintaining content consistency [21]. Unfortunately creating a paired dataset can indeed be difficult and is only possible for specific applications. Some typical use cases for paired image-to-image translation include image colorization,

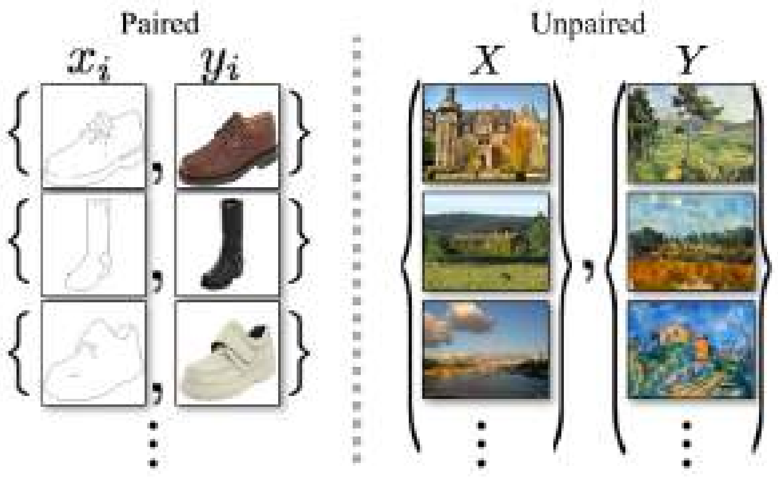


Figure 2.2: Example depicting the case where data in the two domains is paired (left) and data is unpaired (right) (source:[54]).

style transfer, and semantic segmentation, where corresponding images from different domains are available or can be generated with relative ease.

Unpaired Image-to-Image Translation

In unpaired image-to-image translation, a dataset containing images from two different domains without any explicit correspondence between them is used. This makes the task of learning a mapping between the two domains more challenging, as the generator must rely on the underlying structure and distribution of the data to identify and learn the appropriate transformation [54]. However such datasets are easier to acquire. The unpaired image-to-image translation is suitable for tasks where obtaining paired data is difficult or impossible, such as photo-realistic style transfer and domain adaptation.

Summary

The choice between paired and unpaired image-to-image translation mostly depends on the availability of corresponding images and the desired learning objective. The paired translation is suitable for GAN architectures that require direct supervision, whereas unpaired translation can be used with unsupervised GAN architectures that rely on cycle consistency or other unsupervised learning techniques to learn the mappings between the domains.

2.4 Neural Style Transfer

Since style is mostly affiliated with texture, image style transfer uses the texture to transfer the style onto the content image, generating a new output picture. The style image specifies the components an image should be made of, while the content controls the layout. To accomplish the desired output, Gatys et al. introduced a novel concept in their paper titled “A Neural Algorithm of Artistic Style” [17]. Their approach utilized Convolutional Neural Networks to create algorithms capable of transforming images based on artistic style. By extracting and manipulating the deep features learned by CNNs, they were able to generate images that combined the content of one image with the artistic style of another, resulting in visually appealing stylized outputs. To illustrate the capabilities of this method authors generated images that blend the content and style of well-known artworks taken from different periods of art depicted in Figure 2.3 [15].
This process is developed on the basis of VGG16 or VGG19 network to extract the high-level features from the style and the content from the photographed image. The “16” and “19” stand for the number of weighted convolutional layers in the model. The depicted network in Figure 2.4 has 16 deep layers and is trained on

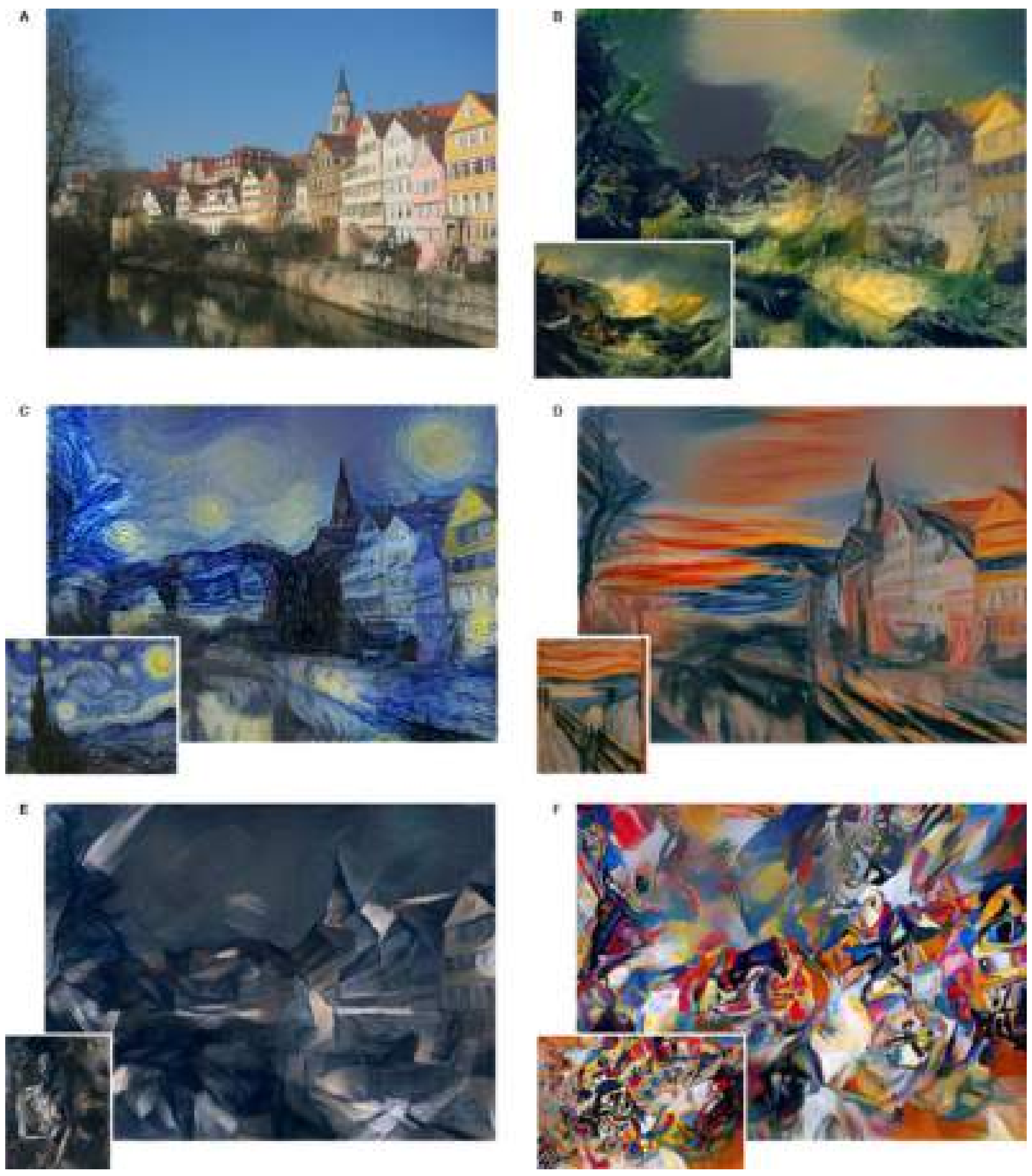


Figure 2.3: The displayed images combine the content of a photograph with the styles of popular paintings generated by Neural Style Transfer methods. The original photograph, which portrays the Neckarfront in Tübingen, Germany, is shown in panel A. The painting that provided the style for each respective generated image is displayed in the bottom left corner of each panel. Panel B features ”The Shipwreck of the Minotaur” by J.M.W. Turner, 1805. Panel C showcases ”The Starry Night” by Vincent van Gogh, 1889. Panel D presents ”The Scream” (Der Schrei) by Edvard Munch, 1893. Panel E displays ”Femme nue assise” by Pablo Picasso, 1910. Finally, panel F exhibits ”Composition VII” by Wassily Kandinsky, 1913 (source: [15]).

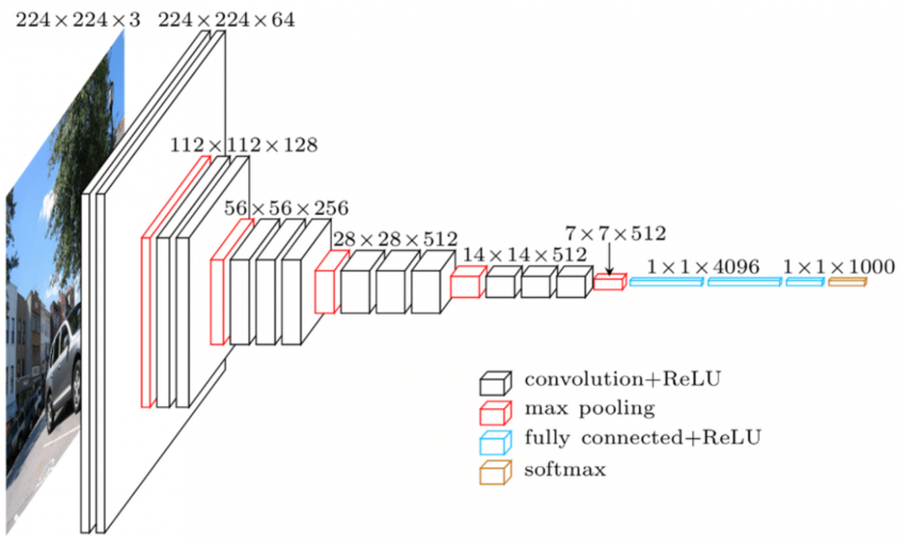


Figure 2.4: VGG16 network architecture. An input is a 224x224 RGB image that is processed through a sequence of convolutional layers, each equipped with 3x3 filters and followed by a rectified linear unit (ReLU) activation function. The network consists of 16 convolutional layers in total, with the first 13 layers containing 2 convolutional filters and the last 3 layers containing 4 convolutional filters. After every two convolutional layers, a max-pooling layer with a 2x2 filter and a stride of 2 is applied, which helps reduce the spatial size of the feature maps and extract the most important features. At the end of the network are fully connected layers, with the first two layers containing 4096 neurons each, and the final layer containing 1000 neurons, corresponding to the number of classes in the ImageNet dataset. The output of the last fully connected layer is passed through a softmax activation function to generate the predicted probabilities for each class (source [43]).

more than a million images from the ImageNet database [43, 37]. To extract the content features, CNN passes the input image $\vec{x}$ through the layers of the network, which is encoded. The content features are represented by the activations of the chosen layer. A layer with N_l different filters consists of N_l feature maps, each of size M_l, where M_l represents the height and width of a feature map.
By initializing a white noise image, which eventually becomes the output image, the network can obtain the equivalent of the feature responses of the original input photograph $\vec{p}$. Assuming that $\vec{x}$ is the generated image, F_l and P_l are the feature representations of the respective images $\vec{x}$ and $\vec{p}$ in layer l. The obtained feature maps are employed to reconstruct the content of the original image in the generated image. The squared-error loss between the two feature representations measures the difference between the activations of the chosen layer for the content image and the generated image can be defined like in Equation (2.3).

$$L_{content}(\vec{p}, \vec{x}, l) = \frac{1}{2} \sum_{i,j} (F_l^{ij} - P_l^{ij})^2 \tag{2.3}$$

To obtain a representation of the style, the correlations between the different filter responses of the neural network are calculated based on the Gram matrix $G_l \in \mathbb{R}^{N_l \times N_l}$ [16] . The element G_l^{ij} is the inner product between the vectors of feature maps i and j in the l-th layer:

$$G_l^{ij} = \sum_k F_l^{ik} F_l^{jk} \tag{2.4}$$

Assuming $\vec{a}$ and $\vec{x}$ are the style image painting and the generated image respectively, and A_l and G_l are their style representations in layer l. E_l is the contribution of layer l to the total loss. N_l represents the number of feature maps, M_l denotes the exact size of the flattened feature map in the l-th layer, and w_l is the weight given to the style loss of the l-th layer. The style loss equation can be expressed as:

$$L_{style}(\vec{a},\vec{x}) = \sum_{l=0}^{L} w_l E_l \text{ ,where } E_l = \frac{1}{4N_l^2 M_l^2} \sum_{i,j} (G_l^{ij} - A_l^{ij})^2 \tag{2.5}$$

Finally, the total loss can be calculated by adding up the style and content losses weighted with specified parameters α and β for content and style, respectively:

$$L_{total}(\vec{p},\vec{a},\vec{x}) = \alpha L_{content}(\vec{p},\vec{x}) + \beta L_{style}(\vec{a},\vec{x}) \tag{2.6}$$

By minimizing the distance of the feature representations of a white noise image from the content representation of $\vec{p}$ in one layer and the style representation $\vec{a}$ defined on a number of layers, one ensures that the resulting image captures both the content and style features of the input images. The gradient descent, Adam or L-BFGS optimizer can be used to calculate the minimum of the Equation (2.6).
Numerous Neural Style Transfer algorithms [43, 37, 10] have been devised to create unique images by blending the content of one image with the style of another (typically a painting). This technique is highly effective in transferring styles from intricate, textured, and color-rich paintings to photos, resulting in aesthetically pleasing and artistic outcomes.
Nevertheless, NST may not be the optimal approach for producing cartoons. One reason is that cartoons typically depend on a more straightforward, abstract visual representation, characterized by fewer details, exaggerated features, and simpler color schemes. NST concentrates on transferring style, texture, and color patterns from the style image but might not sufficiently capture the abstraction, simplification, and exaggeration inherent in cartoon styles. In short, existing neural style transfer methods are generally suitable only for specific style transfer tasks such as transferring styles from paintings. When applied to cartoon-style transfer, they tend to obtain unsatisfactory results.

2.5 Generative Adversarial Networks

Generative Adversarial Networks have been extensively used in various papers for addressing image-to-image (I2I) translation problems due to their ability to generate realistic and high-quality images [18]. In 2017 Isola et al. [21] , proposed a groundbreaking I2I translation method called Pix2Pix, which utilizes a Conditional-GAN (cGAN) and U-Net neural network to learn a mapping from an input image to an output image. The Pix2Pix model has been applied to tasks such as converting satellite images to maps, transforming sketches into photorealistic images, and synthesizing images from semantic label maps. Another notable work by Zhu et al. [54] introduces CycleGAN, an approach that enables I2I translation without requiring paired training examples. CycleGAN learns the mapping between two image domains using unpaired data by incorporating cycle consistency loss, which enforces the model to maintain the underlying structure of the input image. This method has been used for various applications, including style transfer, object transfiguration, and season transfer. In 2018 StarGAN model was proposed by Choi et al. [12]. Star GAN is a unified GAN framework that can perform multiple I2I translation tasks using a single model. It can handle multiple domains, thus reducing the need for training separate models for each domain pair. StarGAN has been successfully applied to tasks such as facial attribute transfer and facial expression synthesis (see Figure 2.5). To illustrate the capabilities of the StarGan authors generated examples of real images toward the target manifold [12].

2.5.1 Adversarial Nets

GANs are formally structured as probabilistic models containing latent variables $\mathbf{z}$ and observed variables $\mathbf{x}$. Both players in the game are represented by different functions with respect to their inputs and parameters. The discriminator, represented by the function $\mathbf{D}$ takes $\mathbf{x}$ as input and uses the parameters $\boldsymbol{\theta(D)}$ and the generator is defined by the function $\mathbf{G}$, takes $\mathbf{z}$ as input, with parameters $\boldsymbol{\theta(G)}$. Each player's cost functions depend on the other player's parameters, but neither player can control the other's parameters, making this a game rather than an optimization problem. The solution of this game is a Nash equilibrium. In this context, a local differential Nash equilibrium [36] which can be represented as a tuple ($\boldsymbol{\theta(D)}$, $\boldsymbol{\theta(G)}$) that express a local minimum of $\boldsymbol{J(D)}$ with respect to $\boldsymbol{\theta(D)}$ and a local minimum of $\boldsymbol{J(G)}$ with $\boldsymbol{\theta(G)}$ where $\boldsymbol{J}$ represent total cost function of model [18] .

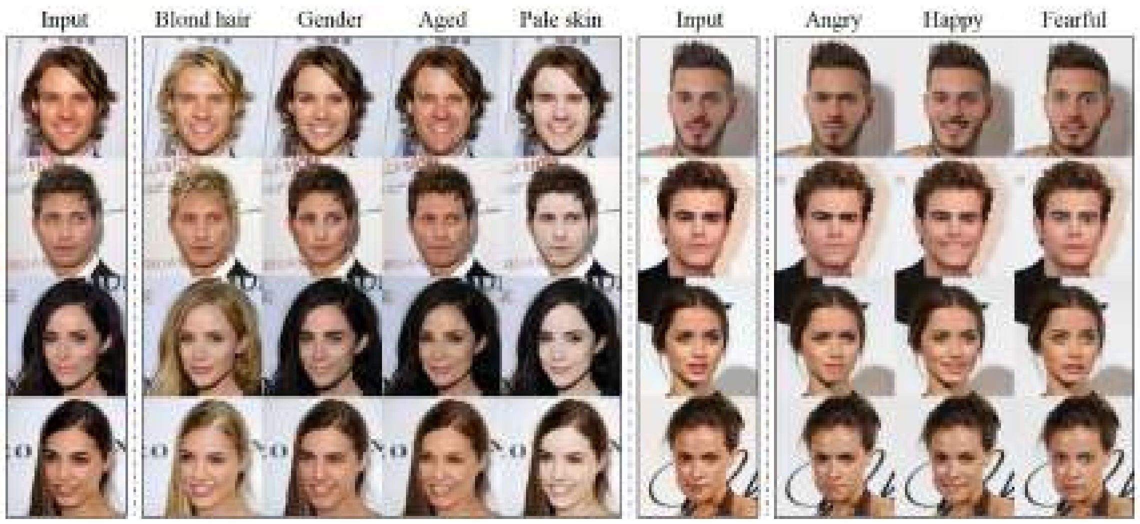


Figure 2.5: Multi-domain image-to-image translation results on the CelebA and RaFD dataset. The first and sixth columns show input images while the remaining columns are images generated by StarGAN toward the target manifold (source: [12]).

GANs working process is illustrated in Fgure 2.6. Discriminator's goal is to output the probability that its input is real rather than fake, assuming that half of the inputs it encounters are real and half are fake. In an ideal scenario, the discriminator $\boldsymbol{D(x)}$ should to be near 1 and $\boldsymbol{D(G(z))}$ near 0. On the other hand, it is desired to have a generator that creates images with a probability $\boldsymbol{D(x)}$ and $\boldsymbol{D(G(z))}$ near 1 which will state that the generated sample matches the real image. These two output probabilities are used to evaluate the performance of the models, so the generator is trained to maximize $\boldsymbol{D(G(z))}$ to be as close to 1 as possible but siumtanisly the discriminator is trained to minimize $\boldsymbol{D(G(z))}$ that's why GANs are often defined as a minimax game. If both models have sufficient capacity, the Nash equilibrium of this game corresponds to $\boldsymbol{G(z)}$ being drawn from the same distribution as the training data. The minimax game has a global optimum for $\boldsymbol{D(x)} = 0.5$ [36].

2.5.2 Training Process

The training process of proposed GANs involves simultaneously calculating Stochastic Gradient Descent (SGD) for the generator and discriminator. During each step, two mini-batches are sampled: a minibatch of $\boldsymbol{x}$ values from the dataset and a minibatch of $\boldsymbol{z}$ values drawn from the model's prior over latent variables. Then, two gradient steps are made simultaneously: one updating $\boldsymbol{\theta(D)}$ to reduce $\boldsymbol{J(D)}$ and the other updating $\boldsymbol{\theta(G)}$ to reduce $\boldsymbol{J(G)}$. In both cases, you can choose the gradient-based optimization algorithm that best suits your needs (most popular choice is Adam [27]).

Minimax Game

Training using more mathematical approach can be depicted as a two-player minimax game with value function $\boldsymbol{V(G,D)}$ (see Equation (2.7)). Because $\boldsymbol{J(G)}$ is tied directly to $\boldsymbol{J(D)}$, we can summarize the entire game with a value function $\boldsymbol{V(\theta(D),\theta(G) = -J_{(D)}(\theta(D),\theta(G))}$ specifying the discriminator's payoff. The discriminator $\boldsymbol{D}$ tries to maximize the probability of assigning the true label to samples from $\boldsymbol{G}$ and training samples. Simultaneously $\boldsymbol{G}$ is trained to minimize $\boldsymbol{\log(1 - D(G(z)))}$ [18].

$$\min_G \max_D V(D,G) = \min_G \max_D (\mathbb{E}_{x\sim pdata(x)}[\log D(x)] + \mathbb{E}_{z\sim p_z(z)}[\log(1 - D(G(z)))]). \tag{2.7}$$

In practice, above equation might not offer a sufficient gradient for the generator to learn effectively. In the early stages of learning, when the generator is still underperforming, the discriminator can continuously reject generated samples because they are noticeably different from the actual training data. To address this issue,

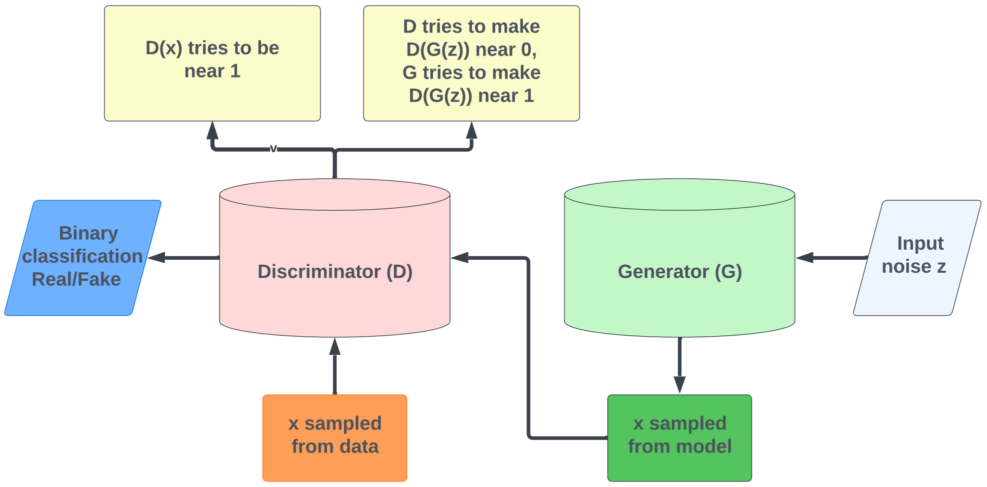


Figure 2.6: The generator generates images from random noise z, learning how to create realistic images over time. The random noise is fed into the generator, which produces an image. The generated images along with real images from the training set, are input to the discriminator. The output $D(x)$ represents the probability that an input image is real. The discriminator aims to make $D(G(z))$ near 0 (classifying generated images as fake), while the generator tries to make $D(G(z))$ near 1 (creating images that the discriminator classifies as real). This adversarial process helps the generator improve over training process.

instead of training the generator to minimize $\boldsymbol{\log(1 - D(G(z)))}$, we can train it to maximize $\boldsymbol{\log D(G(z))}$. Although this alternative objective function leads to the same fixed point in the dynamics of the generator and discriminator, it offers much stronger gradients during the early phases of learning, which helps the generator to learn more effectively.
Implementing the minimax game for GANs involves using an iterative, numerical approach. Directly optimizing the discriminator $\boldsymbol{D}$ to completion within the inner loop of training is computationally expensive and can lead to overfitting on finite datasets. Instead, the training process alternates between k steps of optimizing $\boldsymbol{D}$ and one step of optimizing the generator $\boldsymbol{G}$. This ensures that the discriminator stays near its optimal solution, provided that the generator changes slowly enough. The procedure can be illustrated using pseudocode as shown in Algorithm 1 [18].

Maximum Likelihood Game

It is possible to perform maximum likelihood learning with GANs. It requires minimizing the Kullback-Leibler (KL) divergence between the data and the model, as shown before in Equation (2.2). In 2014 Goodfellow et al.demonstrated that using Equation (2.8) is the optimal choice.

$$J(G) = -\frac{1}{2}\mathbb{E}_z \exp\left(\sigma^{-1}(D(G(z)))\right), \tag{2.8}$$

where σ represents the logistic sigmoid function. This equivalence is valid in expectation; however, in practice, stochastic gradient descent on the KL divergence and the GAN training process will show some variance around the true expected gradient due to the use of sampling (of x for maximum likelihood and z for GANs) to create the estimated gradient.

Loss Functions

In GANs, training loss function is a crucial component that determines the effectiveness and performance of the model. The training loss consists of two main parts: the generator loss and the discriminator loss. These losses work together during the learning process of both the generator and the discriminator networks.

The generator loss calculates how well the generator can create realistic images that can fool the discriminator. It encourages the generator to produce images that the discriminator classifies as real. The generator's goal is to minimize this loss, which makes the generated images as similar as possible to the target domain.
However, the discriminator loss measures the ability of the discriminator to differentiate between real and generated images. This loss consists of the real image loss (difference between the real images and the discriminator's predictions) and the fake image loss (difference between the generated images and the discriminator's predictions). The discriminator's goal is to minimize this loss, improving its ability to distinguish between real and generated images.
During the training process, the generator and discriminator losses are optimized simultaneously but in opposite directions (minmax game see: Section 2.5.2). This creates a dynamic equilibrium where the generator improves its image generation quality, and the discriminator improves its ability to gues between real and generated images [18]. This adversarial training process continues until a balance is reached, and the generator produces high-quality, realistic images.

Adversarial Loss

Adversarial loss is a crucial component in the training of GANs. It measures the discrepancy between the generated images and the target domain's distribution, guiding the generator to create more realistic images and the discriminator to distinguish between real and generated images. The generator aims to minimize the adversarial loss, while the discriminator tries to maximize it (minimize in the opposite direction) [6]. The adversarial loss can be represented as a minimax game with value function $V(G, D)$ (see: Equation 2.7). Where $V(G, D) = \mathcal{L}(G, D)$ is the loss function for the GAN defined as the sum of the losses for the generator and the discriminator.

$$\mathcal{L}(G, D) = \mathbb{E}x \sim p_{data}(x)[\log D(x)] + \mathbb{E}_{z\sim p_z(z)}[\log(1 - D(G(z)))] \tag{2.9}$$

By optimizing the adversarial loss, both the generator and discriminator improve their performance, leading to the generation of high-quality images that closely resemble the target domain.

Algorithm 1: Training of GANs using minibatch stochastic gradient descent with hyperparameter k (number of steps to apply to the network)[18].

```
for number of training iterations do
  • for k steps do
      – Sample minibatch of m noise samples $\{z^{(1)}, \dots, z^{(m)}\}$ from noise prior $p_g(z)$.
      – Sample minibatch of m examples $\{x^{(1)}, \dots, x^{(m)}\}$ from data generating distribution $p_{data}(x)$.
      – Update the discriminator by ascending its stochastic gradient:
            $\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log D\left(x^{(i)}\right) + \log\left(1 - D\left(G\left(z^{(i)}\right)\right)\right)\right].$
  • end for
  • Sample minibatch of m noise samples $\{z^{(1)}, \dots, z^{(m)}\}$ from noise prior $p_g(z)$.
  • Update the generator by descending its stochastic gradient:
            $\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log\left(1 - D\left(G\left(z^{(i)}\right)\right)\right).$
end for
```

2.5.3 Summary

GANs have been used in numerous applications, solving translation problems due to their ability to generate realistic images while preserving the underlying structure of the input. They have been used in a wide range of applications, including style transfer, semantic segmentation, and image synthesis.
In this section, we explored the fundamentals of Generative Adversarial Networks. GANs have been proven to be a powerful tool for generating content with diverse and novel results while preserving original image content by learning the underlying structure and patterns in the data. This is the reason why we, like many other well-known autors, decided to choose this approach for generating cartoons [54, 11, 9, 20].

2.6 Evaluation of Generative Adversarial Networks

Evaluating the performance of Generative Adversarial Networks can be challenging due to the absence of a definitive ground truth for generated samples. GAN evaluation typically relies on both qualitative and quantitative assessments to determine the quality and diversity of generated images. There are some quantitative metrics for image generation tasks, although their effectiveness has been a subject of ongoing debate. Various metrics like Inception Score and Frechet Inception Distance have been proposed, but they often don't perfectly align with human perceptual judgments of image quality. In this section, we discuss several standard methods for evaluating GANs, highlighting their limitations and difficulties specifically in the context of cartoon generation tasks.

Inception Score

The Inception Score (IS), proposed by Salimans et al. [39], is widely adopted as a score for evaluating Generative Adversarial Networks. It utilizes a pre-trained neural network, typically the Inception Net [46], which was trained on the ImageNet dataset [14]. The goal of the Inception Score is to capture desirable properties of generated samples, specifically their high classifiability and diversity with respect to class labels. The Inception Score mathematical representation is shown in Equation (2.10). It measures the average Kullback-Leibler (KL) divergence between the conditional label distribution, $p(y|x)$ of the generated samples, and the marginal distribution, $p(y)$, obtained from all the samples. It expects the conditional label distribution to have low entropy for easily classifiable samples, indicating better sample quality. On the other hand, it anticipates the marginal distribution to have high entropy if all classes are equally represented in the set of samples, indicating high diversity. The Inception Score favors low entropy in $p(y|x)$ but a large entropy in $p(y)$.

$$IS(G) = \exp\left(\mathbb{E}_x\left[\mathrm{KL}\left(p(y|x)||p(y)\right)\right]\right) = \exp\left(H(y) - \mathbb{E}_x\left[H(y|x)\right]\right) \tag{2.10}$$

where $p(y|x)$ represents the conditional label distribution for image x, estimated using a pre-trained Inception model [46], and $p(y)$ represents the marginal distribution. $H(x)$ denotes the entropy of the variable x.
IS measures both the diversity and quality of generated images. While it has been widely used, it assumes that the generated images have a similar data distribution to real-world images [39]. However, the Inception Score has several limitations. Firstly, it relies on a pre-trained model and assumes that the model captures the characteristics of the target domain, which may not be the case for datasets with unique distributions such as cartoon-like images. Secondly, IS emphasizes classifiability, which might not be the primary objective when evaluating cartoon generation, as cartoons often involve stylization. The Inception score will be low using classes outside of 1000 classes used to train the Inception Net dataset. Lastly, the Inception Score may be insensitive to specific cartoon characteristics and aesthetics as it is primarily designed for evaluating natural images.

Fréchet Inception Distance

Frechet Inception Distance (FID) introduced by Heusel et al. [19] is another widely used metric used to evaluate the performance of Generative Adversarial Networks. It calculates the distance between the distributions of real and generated images to extract features from both sets of images. Lower FID scores indicate that the generated images are more similar to the real images, and thus the GAN performs better.

FID embeds a set of generated samples into a feature space given by a specific layer of Inception-v3 Net [14] (also called GoogleNetv3 or any other CNN). The Inception-v3 model consists of 42 layers, including convolutions, average pooling, max pooling, concatenations, dropouts, and fully connected layers. Next, the output layer of the model is removed, and then, the output is taken as the activations from the last pooling layer to achieve a high-dimensional feature representation of the real and later, the generated images. The feature space is considered a continuous multivariate Gaussian. The mean and covariance are estimated for both the generated data and the real data distributions.
The FID between the real data distribution (μ_r, Σ_r) and the generated data distribution (μ_g, Σ_g) is calculated as follows:

$$FID(r,g) = \|\mu_r - \mu_g\|_2^2 + \mathrm{Tr}(\Sigma_r + \Sigma_g - 2(\Sigma_r\Sigma_g)^{1/2}) \tag{2.11}$$

Here, μ_r and μ_g represent the means of the real and generated data distributions, respectively. Σ_r and Σ_g are the covariance matrices of the real and generated data distributions, respectively. $\|\cdot\|_2^2$ denotes the squared Euclidean distance, and Tr represents the trace of a matrix. The result of (2.11) equation is a scalar value.
A lower FID value indicates smaller distances between the synthetic and real data distributions, indicating better quality of the generated samples. FID performs well in terms of discriminability, robustness, and computational efficiency. It has been shown to be a good measure, even though it only takes into consideration the first two moments of the distributions. However, FID assumes that the features are distributed according to a Gaussian distribution, which may not always be guaranteed.
FID has been found to be consistent with human judgments and is more robust to noise compared to other metrics such as Inception Score (IS). Unlike IS, FID is capable of detecting intra-class mode dropping, where a model generates only one image per class but still scores a high IS.
In conclusion, FID is a valuable metric for evaluating the quality and diversity of generated images in GANs. By comparing the feature statistics of the real and generated data distributions, FID provides quality insights into the performance of GAN models.

Visual Inspection

A straightforward way to evaluate GANs is to visually inspect the generated images. This can give you an idea of the quality of the images, their diversity, and whether the model has captured the essential characteristics of the target domain.
Visual Inspection metric, in the context of evaluating GANs for cartoon generation, involves conducting a questionnaire-based assessment where participants evaluate the generated cartoon images based on specific criteria such as preserving content, cartoonishness, and aesthetics. This rating scales metric captures the subjective aspects of cartoon quality and provides valuable insights into the perception and preferences of human evaluators. The questionnaire can be designed to include standard rating or Likert-scale questions [24] for participants to rate the generated cartoons on different criteria. For the cartoon domain participants can evaluate generated outputs on the criteria listed below:

- **Preserving Content** - participants rate which of the generated cartoons accurately preserves the content input image's recognizable features. This criterion ensures that the generated cartoons maintain the essence and identity of the original content.
- **Cartooniness** - participants evaluate the level of cartoon-like characteristics in the generated images. This criterion focuses on the stylization, exaggeration, simplification, and distinctive traits that define cartoons. It captures the degree to which the generated images resemble traditional or desired cartoon styles.
- **Aestheticness** - participants assess the overall aesthetic quality and visual appeal of the generated cartoons. This criterion considers the composition, color palette, line quality, balance, and other artistic elements that contribute to the perceived attractiveness and artistic merit of the images.

Participants can rank the generated cartoons on the above mentioned criteria using a numerical scale.
By aggregating the responses from multiple participants, statistical analysis can be conducted to derive insights regarding the performance of the cartoon generative GAN. This analysis may involve calculating mean scores, identifying trends, or identifying areas for improvement based on participant feedback.

The Visual Inspection metric through a questionnaire-based evaluation is valuable as it captures the human perception of generated image quality, which may not be fully quantifiable by mathematical metrics. It provides a comprehensive understanding of the strengths and weaknesses of the generated cartoons and guides the further refinement and development of the GANs models.

Summary

In conclusion, evaluating the performance of Generative Adversarial Networks, particularly for generating cartoons, presents unique challenges due to the subjective nature of cartoon art. Various metrics, such as Inception Score, Precision, Recall, F1 Score, Perceptual Path Length, and Frechet Inception Distance, have been used in the literature to evaluate GANs. However, not all of these metrics are suitable for evaluating GANs specifically designed for the automatic generation of cartoon-style images from real photos.
After conducting a literature analysis, we have identified the Visual Inspection and Fréchet Inception Distance as the most suitable metrics for evaluating GANs designed for cartoon generation. In upcoming experiments (see Chapter 6), we plan to utilize these metrics. The Visual Inspection metric captures subjective evaluations from human participants, while the FID metric provides a quantitative measure of similarity to real-world cartoon distributions. By employing both metrics, we aim to obtain a comprehensive evaluation of our cartoon generation model.

2.7 Summary

This chapter has reviewed the current knowledge concerning generative models, fundamental aspects of style transfer, as well as the theory behind GANs and their evaluation metrics.
In the upcoming chapters, we focus on understanding and applying state-of-the-art GAN models, specifically engineered for unpaired image-to-image translation tasks. These models, either purpose built or adaptable, serve to convert real life photographs into cartoons. We delve into the architecture and loss functions of models from the literature in Chapter 3 and propose our unique approach to enhancing the performance of the chosen AnimeGAN model in Chapter 4.
We further apply these models to a unique unpaired dataset that consists of both real and cartoon-style images, with detailed insights presented in Chapter 5. The process of implementing, training, and evaluating these selected models is thoroughly discussed in Chapter 6.
To wrap up this study, in Chapter 7 we perform a comprehensive evaluation and summary of the models and their outcomes, which provides conclusive insights and lays out potential directions for future research.

3 State of the Art Models

In recent years, Generative Adversarial Networks have become a highly popular and powerful tool for generating realistic and diverse results by learning the underlying structure and patterns in data. Various approaches have been proposed for using GANs. In Chapter 2, we introduced the fundamental concepts, advantages, and limitations of Style Transfer [17] and Generative Adversarial Networks [18]. Despite the impressive capabilities of Style Transfer and StyleGAN [25] in various domains, particularly in their ability to mimic the artistic style of one image onto the content of another, they seem to underperform in the field of cartoon generation. These methods tend to prioritize replicating detailed artistic textures and bright colors, which, is essential in copying art style, but don't work well for the cartoon generation. They often miss the important aspect of keeping the structure and content of the original image. As a result of these limitations, we decided that using Style Transfer and StyleGAN wouldn't be the optimal approach.

In this paper, we chose to examine four representative GAN-based models from the literature: CycleGAN, GANILLA, CartoonGAN, and AnimeGAN [54, 20, 11, 9]. In this section, we discuss the above-mentioned state-of-the-art GAN models in the field of cartoon generation, each characterized by unique network architectures and cost functions.

CartoonGAN and AnimeGAN are advanced architectures specifically designed for cartoon creation. On the contrary, CycleGAN and GANILLA are more general image-to-image translation models that are not explicitly tailored for cartoon generation. However, the GANILLA architecture has a specific objective: it's designed to transform images to resemble the style of children's book illustrations. This focused purpose differentiates it from more general models like CycleGAN, while still keeping it within the broader field of image-to-image translation. To assess their performance in generating cartoon images, it is crucial to compare these general models with those explicitly designed for this purpose, such as CartoonGAN and AnimeGAN. Through this comparison, we can identify the strengths and weaknesses of each approach, ultimately determining the most suitable architecture for generating high-quality cartoon images. This comparative analysis will provide invaluable insights for future research and development in the field of cartoon image synthesis.

3.1 CycleGAN

In this section, we delve into the details of the CycleGAN model, which was first introduced in the paper "Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks" by Zhu et al. in 2017. The model and paper have been continuously changed and updated till 2020 [54]. The model is designed for unpaired image-to-image translation tasks, which makes it suitable for generating cartoon images from real-world photos where paired and labeled datasets can't be created or it is a very difficult task.

The main task for CycleGAN is to learn the mapping functions between two domains X and Y, given training samples $\{x_i\}_{i=1}^N \in X$ and real samples $\{y_j\}_{j=1}^M \in Y$. CycleGAN uses two generators and two discriminators because it is designed to learn the mapping between two different, unpaired image domains, X and Y [54]. The simplified schema depicting discussed process is shown in Figure 3.1.

Generators: The generators are responsible for transforming images from one domain to another. There are two generators, G and F, because the model needs to learn mappings in both directions:

- Generator G learns the mapping from domain X to domain Y ($G : X \rightarrow Y$).
- Generator F learns the mapping from domain Y to domain X ($F : Y \rightarrow X$).

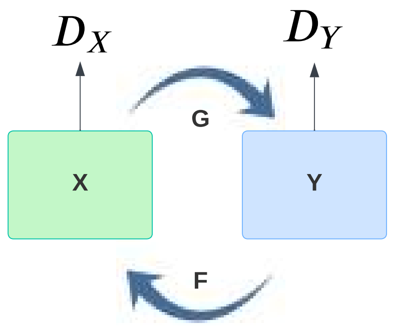


Figure 3.1: The model has two mapping functions, $G : X \to Y$ and $F : Y \to X$, with discriminators D_Y and D_X. D_Y helps G map X to Y, while D_X aids F in mapping Y to X.

Discriminators: The discriminators are responsible for distinguishing between real and generated (translated) images within their respective domains. There are two discriminators, D_X and D_Y, to evaluate the quality of generated images in both domains:

- Discriminator D_X aims to discriminate between real images in domain X and images generated by F ($F : Y \to X$).
- Discriminator D_Y aims to discriminate between real images in domain Y and images generated by G ($G : X \to Y$).

By having separate generators and discriminators for each domain, CycleGAN can learn the mappings between the two domains independently, while also enforcing cycle consistency constraints [53]. This ensures that if an image is translated from one domain to another and then back to the original domain, the reconstructed image closely resembles the initial input image, providing a more accurate and consistent translation.
An architecture generator G of CycleGAN consists of 3 main blocks. It starts with a downsampling convolution block followed by 9 residual blocks and ends with an upsampling convolution. However, the discriminator network is quite simple and consists of down-convolutions and normalization functions. The architectures for G, F, and D_X, D_Y are respectively identical. The detailed architecture can be found on the Github repository created by the authors (source [54]).

3.1.1 Cycle Consistency

In theory, adversarial training mappings G and F can generate images with distributions identical to target domains Y and X. However, in practice, only adversarial losses cannot guarantee that the learned mapping functions are adequate. In order to further constrain the possible mapping functions, the learned mappings should show both forward and backward cycle consistency (Figure 3.2). Forward cycle consistency means that for every image x from domain X, the translation cycle should return x (original or similar form). Similarly for backward cycle consistency: $y \to F(y) \to G(F(y)) \approx y$.
The cycle consistency loss is the average of the mean squared errors between the input images and their reconstructions obtained by passing through both generators in sequence. In other words, it calculates the difference between the original images and the images transformed from one domain to another and back to the initial domain. For the cycle $Y \to X \to Y$ cycle consistency loss can be written as:

$$\mathcal{L}_{cc}(G, D_Y) = \frac{1}{m}\sum_{i=1}^{m}(y^{(i)} - G_{X\to Y}(G_{Y\to X}(y^{(i)})))^2 \tag{3.1}$$

The loss for the $X \to Y \to X$ cycle is similar, with the roles of X and Y reversed. By minimizing this loss, the generators maintain content consistency of image [54].
To fully explain how cycle consistency works in practice, we illustrate it using an example of transforming from domain X (apples) to domain Y (oranges) and in the reverse direction, as shown in Figure 3.3.

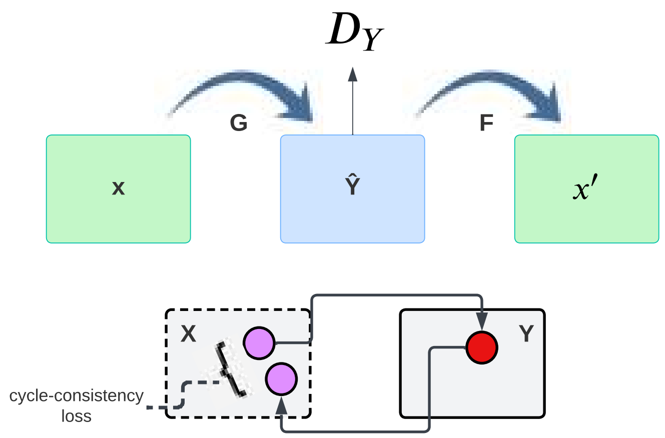


Figure 3.2: Cycle consistency losses in CycleGAN help maintain content preservation during image-to-image translation. Depicted forward cycle-consistency loss enforces that $x \to G(x) \to F(G(x)) \approx x'$. The backward cycle-consistency loss works the same but it enforces that $y \to F(y) \to G(F(y)) \approx y'$.

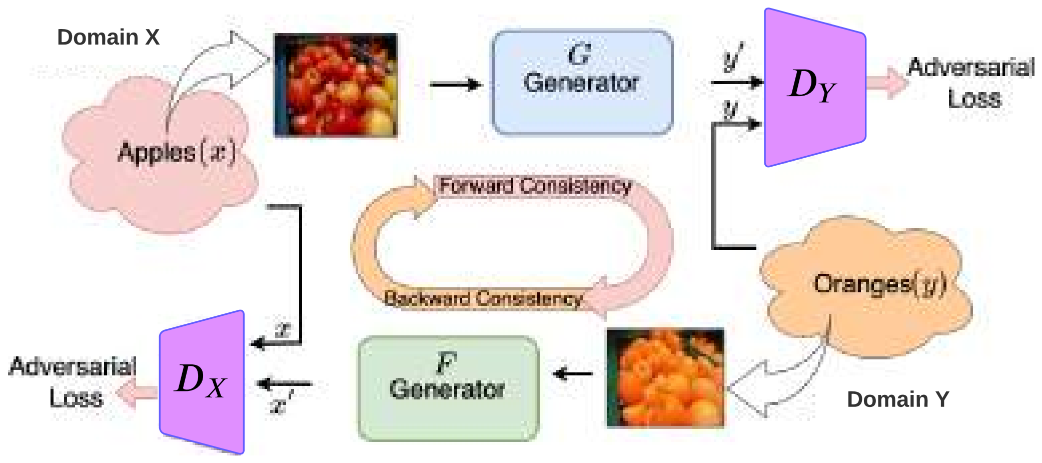


Figure 3.3: Image x from domain X (apples) is taken and passed through Generator G, which aims to generate an image belonging to domain Y distribution (oranges). Discriminator D_Y differentiates between generated samples y' and actual samples y. The generator is trained against this adversary using adversarial training, allowing it to generate images resembling domain Y distribution. Similarly, we take image y from domain Y (oranges) and pass it through Generator F, which aims to generate an image belonging to domain X distribution (apples). Discriminator D_X differentiates between generated samples x' and actual samples x.

3.1.2 Loss Functions

To gain a complete understanding of the CycleGAN training process, it is needed to introduce the full objective of the loss functions. CycleGAN network described by Jun-Yan Zhu consists of two loss functions: adversarial loss and cycle consistency loss.

Adversarial Loss

The adversarial loss, denoted as $\mathcal{L}_{GAN}(G, D)$, is designed to make the generated images indistinguishable from the target domain images. It can be defined for both generators G and F and their respective discriminators D_Y and D_X as follows:

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

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

The generator aims to minimize the adversarial loss, while the discriminator tries to maximize it. In other words, G,F, and D_Y, D_X respectively are adversaries, and they compete against each other during the training process.

$$\min_{G} \max_{D_Y} \mathcal{L}_{GAN}(G, D_Y, X, Y), \quad \min_{F} \max_{D_X} \mathcal{L}_{GAN}(F, D_X, Y, X) \tag{3.4}$$

Cycle Consistency Loss

The cycle consistency loss, denoted as $\mathcal{L}_{cyc}(G, F)$, ensures that the translations between domains are consistent, i.e., translating an image from one domain to the other and back should yield the original image (Section 3.1.1) [54].

$$\mathcal{L}_{cyc}(G, F) = \mathbb{E}_{x \sim p_{data}(x)}[||F(G(x)) - x||_1] + \mathbb{E}_{y \sim p_{data}(y)}[||G(F(y)) - y||_1] \tag{3.5}$$

Total Loss Function

The total loss function, $\mathcal{L}(G, F, D_X, D_Y)$ for CycleGAN training is the weighted sum of adversarial and cycle consistency losses and it is expressed in the equation below.

$$\mathcal{L}(G, F, D_X, D_Y) = \mathcal{L}_{GAN}(G, D_Y) + \mathcal{L}_{GAN}(F, D_X) + \lambda_{cyc}\mathcal{L}_{cyc}(G, F) \tag{3.6}$$

Here, λ_{cyc} is a hyperparameter that controls the relative importance of the cycle consistency loss. These losses regularize the learned mappings, ensuring that translations between domains and back again result in images close to the originals, which is especially important for unpaired datasets in applications like style transfer and domain adaptation [54].
Combining these terms, we can finally formulate an optimization problem that balances the adversarial losses with the cycle consistency losses to learn the desired mapping functions between the two domains.

$$G^*, F^* = \arg \min_{G,F} \max_{D_X, D_Y} \mathcal{L}(G, F, D_X, D_Y) \tag{3.7}$$

This enables the model to generate high-quality translations that exhibit both local and global consistency in the target domain.

3.2 CartoonGAN

In this section, we discuss the CartoonGAN model, which has been specifically designed for the task of photo cartoonization. In 2018 the network was introduced in the broadly cited paper “CartoonGAN: Generative Adversarial Networks for Photo Cartoonization” by Chen et al. [11]. This model has garnered significant attention due to its ability to generate high-quality cartoon images from real-world photographs.

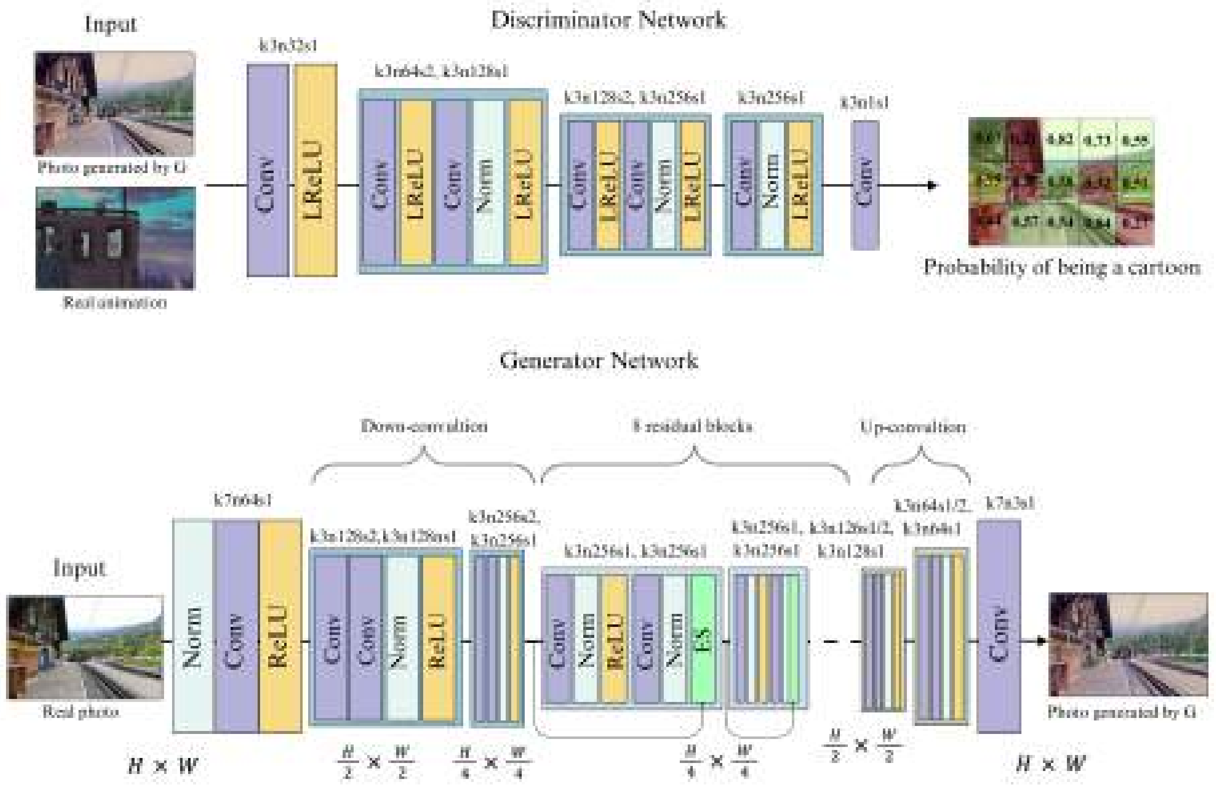


Figure 3.4: CartoonGAN Architecture. GAN framework consists of multiple convolutional layers with varying kernel sizes (k), number of feature maps (n), and strides (s). A normalization layer ($norm$) is added after each convolutional layer to stabilize and improve the learning process. Additionally, ES indicates elementwise sum, which is used for combining features from different layers.

3.2.1 Architecture

The architecture of CartoonGAN is based on the standard GAN framework, with a generator network G and a discriminator network D. During model training generator transforms the input real-world photos photo manifold $\mathcal{P}$ to the cartoon-like images manifold. The discriminator network is responsible for distinguishing between the generated by G images and real cartoon images and providing the adversarial loss for G. The network task is to solve minmax problem (Section 2.5.2). Let $\mathcal{L}$ be the loss function (Equation (3.9)), and the weights of the networks as G^* and D^*.

$$(G^*, D^*) = \arg \min_G \max_D \mathcal{L}(G,D) \tag{3.8}$$

Both networks are based on CNNs and they are composed of convolutional layers, residual blocks, and up/down convolutional layers [11].
In total, the generator network consists of 14 layers or blocks. The G architecture is depicted at the bottom of Figure 3.4. The network G begins with an initial flat convolution stage, then two down-convolution blocks with the task to encode and compress the content. This process extracts essential local signals and additional pieces of information. Next, G architecture consists of eight residual blocks with an undifferentiated layout. These blocks are used to construct the content and manifold features. As in the CartoonGan paper, we take the residual block layout proposed in [22]. At last, the cartoon-style output images are reconstructed through two up-convolution blocks and a final convolutional layer.
The discriminator network, responsible for classifying an image as either cartoon or real-life, consists only of five blocks. Given that determining whether an input is a real cartoon or not is a less demanding task, the authors propose a simple patch-level discriminator with a low number of parameters to minimize computational costs. The D architecture is depicted at the top of Figure 3.4. It starts with a flat convolution

layer. Followed by two down-convolutions with the task to reduce the resolution and encode important local features for binary classification. Finally, one feature construction block and one convolution layer classify whether the input is a real cartoon or generated one. The discriminator has attributes of patchGAN model [30] that's why instead of providing a single classification for an entire photo, it operates on smaller, cropped sections (patches) of each image. This results in a list of classification probabilities, with one corresponding to each patch from which the final binary classification is calculated.

3.2.2 Loss Functions

The total loss function $\mathcal{L}(G, D)$ used in Equation (3.8) comprises two parts. The adversarial loss $\mathcal{L}_{adv}(G, D)$ (Equation (3.12)), which guides the generator network to achieve the desired manifold transformation, and the content loss $\mathcal{L}_{con}(G, D)$ (Equation (3.11)), which ensures the preservation of image content during the cartoon stylization process. The loss function is formulated as a simple linear combination [11]:

$$\mathcal{L}(G, D) = \mathcal{L}_{adv}(G, D) + \omega\mathcal{L}_{con}(G, D) \tag{3.9}$$

where ω is the weight used to balance the contributions of the two losses.

Content Loss

In the cartoon generation field, besides the transformation between the manifolds, it is crucial to ensure that the resulting cartoon images retain the semantic content of the input photos. In order to accomplish this objective, the authors propose using high-level feature maps, which have been demonstrated to have strong object preservation abilities. These feature maps are derived from the VGG network [42], and pre-trained by [37]. The content loss based on the VGG network can be defined as:

$$\mathcal{L}_{con}(G, D) = \mathbb{E}_{p_i \sim S_{data}(p)}[||VGG_l(G(p_i)) - VGG_l(p_i)||_1] \tag{3.10}$$

where l indicates the feature mapping of a particular VGG layer.
Unlike other image generation methods [29, 17], the semantic content loss is defined using the ℓ_1 norm for sparse regularization of VGG feature maps between the generated cartoon image AND input photo. The ℓ_1 norm, also known as the L_1 norm or Manhattan norm, is a way of measuring the distance between two points in a vector space where x is a vector and xi are its components. Mathematical representation of ℓ_1 norm is shown below:

$$\ell_1(x) = \sum_{i=0}^{n} |x_i| \tag{3.11}$$

The ℓ_1 norm is often used in optimization problems, regularization, and sparse representations. This method is used because cartoon images possess distinct characteristics (e.g., clear edges and smooth shading) compared to photos. Even with a suitable VGG layer that aims to capture image content, the feature maps still might be influenced by the massive style difference. Such differences often concentrate on local regions where the representation and regional characteristics change dramatically. Norm ℓ_1 sparse regularization can handle these changes more effectively than the standard $\ell_2(x)$ norm.

Adversarial Loss

Usually, the adversarial loss in Generative Adversarial Networks serves as a measure of how well the generator's output mimics a cartoon image (Section 2.5.2). In previously mentioned GAN architectures such as CycleGAN and GANILLA [54, 20], this adversarial loss is standardly applied to both the generator G and discriminator D, influencing the transformation process within the min-max game between generator and discriminator in the task of creating cartoon images.
However, the authors observed that this standard approach to adversarial loss was inadequate for effectively translating photos into cartoon-like images. This inadequacy stems from the challenge of training the discriminator to distinguish between authentic and artificial images. One of the key features of cartoon images is the prominent edge definition, but such edges typically occupy a small portion of the entire image. Consequently,

an image output with accurate shading but poorly defined edges could easily mislead a discriminator trained with standard adversarial loss.
In response to this issue, a modified version of adversarial loss was proposed, which proved to be more effective for the specific task of cartoon generation [11, 9]. Authors claim that the solution is creating a new image set $S_{data}(e) = \{e_i | i = 1 \dots M\} \in \mathcal{E}$ which is automatically generated from the training cartoon images $S_{data}(c) \in \mathcal{C}$ to reduce clear edges and color shading (Section 5). For each image from the cartoon set $c_i \in S_{data}(c)$, three preprocessing steps are executed to yield an edge-smoothed version $e_i \in S_{data}(e)$. These steps are thoroughly described in Chapter 5. They briefly involve the detection of edge pixels using the Canny edge detector, the dilation of the detected edge regions, and the application of Gaussian smoothing within these dilated edge regions. Here, $\mathcal{C}$ symbolizes the cartoon manifold, and $\mathcal{E}$ the cartoon-like images without clear edges. A visual comparison between a cartoon image and its smoothed-edge counterpart can be found in Figure 5.4.
In the standard approach for every photo p_k within the photo manifold $\mathcal{P}$, the generator G generates cartoon-like image $G(p_k)$. However, in CartoonGAN, the aim of training the discriminator D is to maximize the sum of all of the probabilities of assigning the correct label to $G(p_k)$, the cartoon images without clear edges $e_j \in S_{\text{data}}(e)$, and the authentic cartoon images $c_i \in S_{\text{data}}(c)$. This ensures that generator G is correctly guided to transform the input towards the appropriate cartoon manifold. This modified (edge-promoting) adversarial loss is defined as follows:

$$L_{adv}(G, D) = \mathbb{E}_{c_i \sim S_{data}(c)}[\log D(c_i)] + \mathbb{E}_{e_j \sim S_{data}(e)}[\log(1 - D(e_j))] + \mathbb{E}_{p_k \sim S_{data}(p)}[\log(1 - D(G(p_k)))] \quad (3.12)$$

3.3 GANILLA

In the paper published in 2020 “GANILLA Generative Adversarial Networks for Image to Illustration Translation” by Hicsonmez et al. [20], children’s book illustrations are explored as a new target domain in unpaired image-to-image translation. These illustrations represent a unique challenge due to their qualitative differences from traditional art paintings and even cartoons. They exhibit a high level of abstraction, while also containing identifiable objects, which makes the balance between abstraction, style, and content more complex than in other domains. Existing models struggle to effectively handle this delicate equilibrium. The author claims that although current state-of-the-art image-to-image translation models successfully transfer either the style or the content, they fail to transfer both simultaneously. In contrast, the author’s proposed model demonstrates satisfactory performance in preserving content while transferring artistic style and illustration abstraction. Intriguingly, it has been shown that this architecture can be adapted for cartoon generation. By conducting a qualitative analysis and user study, the author has proven that this model surpasses CycleGAN [54] and CartoonGAN [11] in the realm of illustration generation [20].

3.3.1 Architecture

The GANILLA model architecture, like CycleGAN, incorporates two generators (G, F) and two discriminators (D_X, D_Y), all of which serve the function of converting images between domains [20]. This architecture aims to preserve the image content while transferring style and illustration abstraction by using low-level features. This dual-generator and dual-discriminator setup aids in creating a more reliable and robust image translation, resulting in better preservation of content and stylistic attributes.
The generator of the GANILLA network is depicted at the top of Figure 3.5. It includes two important stages downsampling and upsampling.
The GANILLA generator begins with a downsampling phase that basically is a ResNet-18 network with few modifications. The purpose of this phase is to leverage low-level features, which are integrated by concatenating features from previous layers at each step of downsampling. The process initiates with a convolutional layer that uses a 7x7 kernel, followed by instance normalization, a ReLU activation function, and max pooling layers. This phase continues with four more layers, each consisting of two residual blocks. Each residual block begins with a convolution layer, followed by instance normalization and ReLU layers. Another convolution layer and an instance normalization layer then follow. The output from these stages is then concatenated

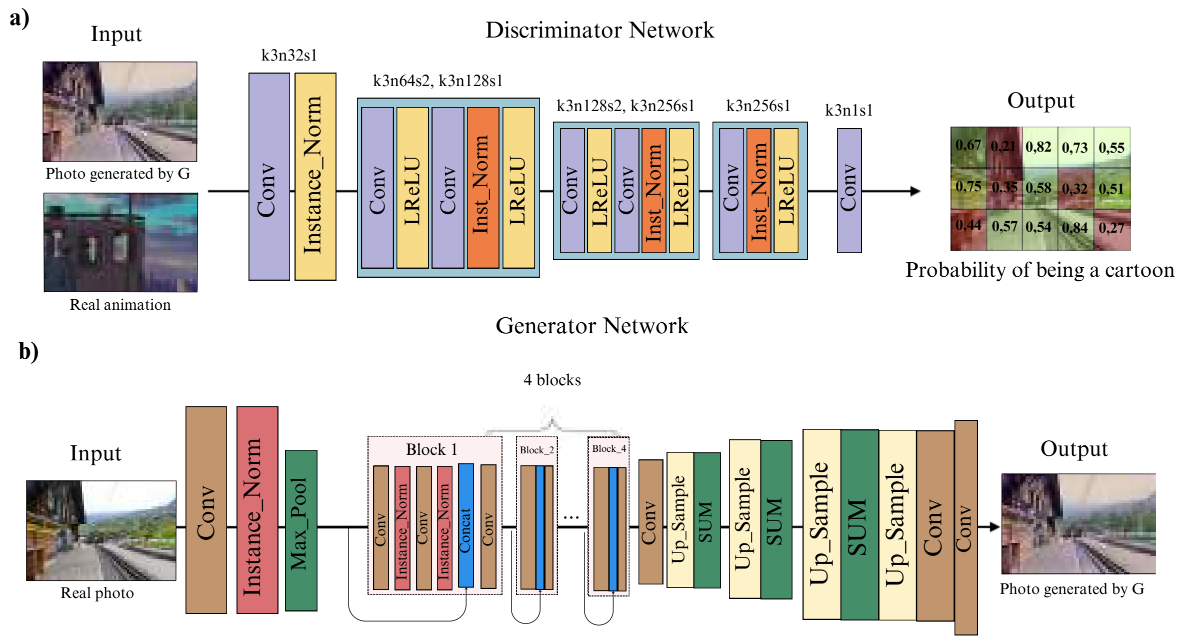


Figure 3.5: GANILLA Architecture. The generator network (b) is designed to include concatenative skip connections during the downsampling process (from the image up to Layer IV). Following this, the output from Layer IV is sequentially upsampled, with lower-level features from the downsampling stage being integrated via long skip connections (indicated by blue arrows in the original diagram). The final output produced by this process is a 3-channel stylized image, demonstrating the successful transfer of style from one domain to another while preserving the content. The discriminator network (a) is a 70x70 PatchGAN.

with the input of the residual block. This concatenated tensor is then passed through the final convolution and ReLU layers. The size of the feature map is reduced by half at each layer, except the first one, using convolutions with a stride of 2. All convolution operations within the residual layers use 3x3 kernels. This complex procedure ensures that the transferred image maintains the basic structure of the input content [20]. The subsequent upsampling stage in the uses the outputs of each layer from the downsampling phase, using these lower-level features to preserve the image content. These outputs are fed through long, skip connections to summation layers before the upsampling (Nearest Neighbor) operations commence. The upsampling stage comprises four consecutive layers, each containing convolution, upsample, and summation operations. The output of the fourth layer from the downsampling phase is first processed through convolution and upsample layers to increase the feature map size, aligning it with the feature size of the previous layer. This process of convolution and upsampling is repeated for each layer's outputs. All convolution filters in the upsampling stage utilize 1x1 kernels. Finally, a convolution layer with a 7x7 kernel is used to generate a 3-channel translated image [20].
The discriminator network implemented in the GANILLA architecture is a 70x70 PatchGAN, which has been successfully utilized in various image-to-image translation models. This network comprises three blocks of convolutions, each containing two convolution layers. The first block starts with a filter size of 64, and this filter size is doubled for each subsequent block. The discriminator of the GANILLA network is depicted at the bottom of Figure 3.5.

3.3.2 Loss Functions

Training the GANILLA model follows the concept of cycle consistency proposed in CycleGAN [54] (Section 3.1). Shortly, the first pair G, D_X aims to map source images to the target domain, while the second pair F, D_Y takes input from the target domain images and attempts to generate source images in a cyclic fashion. The total loss function consists of two adversarial losses (Section 3.1.2) for each generator and discriminator and one cycle consistency loss (Section 3.1.2) exactly like in CycleGAN (Section 3.1.2). The adversarial loss aims to optimize the generator and discriminator in a two-player minimax game, while the cycle consistency loss ensures that a generated sample can be mapped back to the source domain.

3.3.3 Summary

GANILLA is an innovative model purposed for unpaired image-to-image translation tasks, focusing on the transformation between two distinct image domains. GANILLA is heavily influenced by CycleGAN and shares the same loss functions but distinguishes itself through its unique generator architecture. This architecture comprises two stages: downsampling (which uses residual layers with concatenated convolutions) and upsampling. Authors a qualitative analysis and user study demonstrate that GANILLA successfully emulates children's book illustrations by leveraging low-level features to maintain content while transferring the style. Intriguingly, the study suggests that this architecture can be adapted for the generation of cartoons, highlighting its versatility and potential for diverse applications.

3.4 AnimeGAN

In this section, we discuss the AnimeGAN model proposed in 2020 by Chen et al. [9]. AnimeGAN is designed specifically for generating stylized anime images from real-world photographs using unpaired training data. The author suggests unique loss functions for generating high-quality anime images, including an adversarial loss to influence the animation process in the generator, content loss to retain the content from the input photo, grayscale style loss for anime-style textures and lines, and color reconstruction loss to maintain the original colors of the photos.

3.4.1 Architecture

Chan et al., the authors of the previously mentioned CartoonGAN architecture (Section 3.2), introduced a novel model architecture known as AnimeGAN. This model builds upon the foundation laid by the CartoonGAN [11] model and the team's earlier work in the field of cartoon generation. Similar to CartoonGAN, AnimeGAN employs two convolutional neural networks. The generator G is responsible for converting real-world scene photos into anime-style images. On the other hand, the discriminator D judges whether the input images are from the actual target domain or were synthetically generated. Although the discriminator networks in both architectures bear a resemblance, key differences can be observed in the generator network and in the implementation of the supplementary loss functions in the AnimeGAN model.
As previously mentioned, the discriminator network in AnimeGAN is identical to that used in CartoonGAN. The architecture of this discriminator is depicted in Fig. 3.6 (a). Every convolutional layer in the discriminator consists of standard convolutions. Spectral normalization, as suggested by Miyato et al. [33], is applied to the weights of each convolutional layer to ensure the stability of network training.
The AnimeGAN generator can be characterized as a symmetric encoder-decoder network, predominantly composed of repetitive building blocks. The complete architecture of the AnimeGAN generator is depicted in Figure 3.6 (b). The key elements (blocks) include standard convolutions (Conv-Block), depthwise separable convolutions (DSConv) [13], inverted residual blocks [40], along with modules tailored for both upsampling (Up-Conv) and downsampling (Down-Conv). Figure 3.7 provides a detailed depiction of the structure of Conv-Block (a), DSConv (b), IRB (c), Up-Conv (d), and Down-Conv (e). The Conv-Block in AnimeGAN is structured with a standard convolution comprising 3x3 convolution kernels, followed by an instance normalization layer and an LRelu activation function. Similarly, the DSConv block consists of a depthwise separable convolution that uses 3x3 convolution kernels, which is then succeeded by an instance normalization layer and an LRelu activation function. The Inverted Residual Block (IRB), on the other hand, is more complex, containing a Conv-Block, a depthwise convolution, a pointwise convolution, and concluding with the instance normalization layer [49].
The downsampling method called Down-Conv is used to avoid the feature information loss that can occur due to max-pooling. The architecture of Down-Conv includes a DSConv module with a stride of 2 and another with a stride of 1. The Down-Conv reduces the resolution of the feature maps to half of the input feature

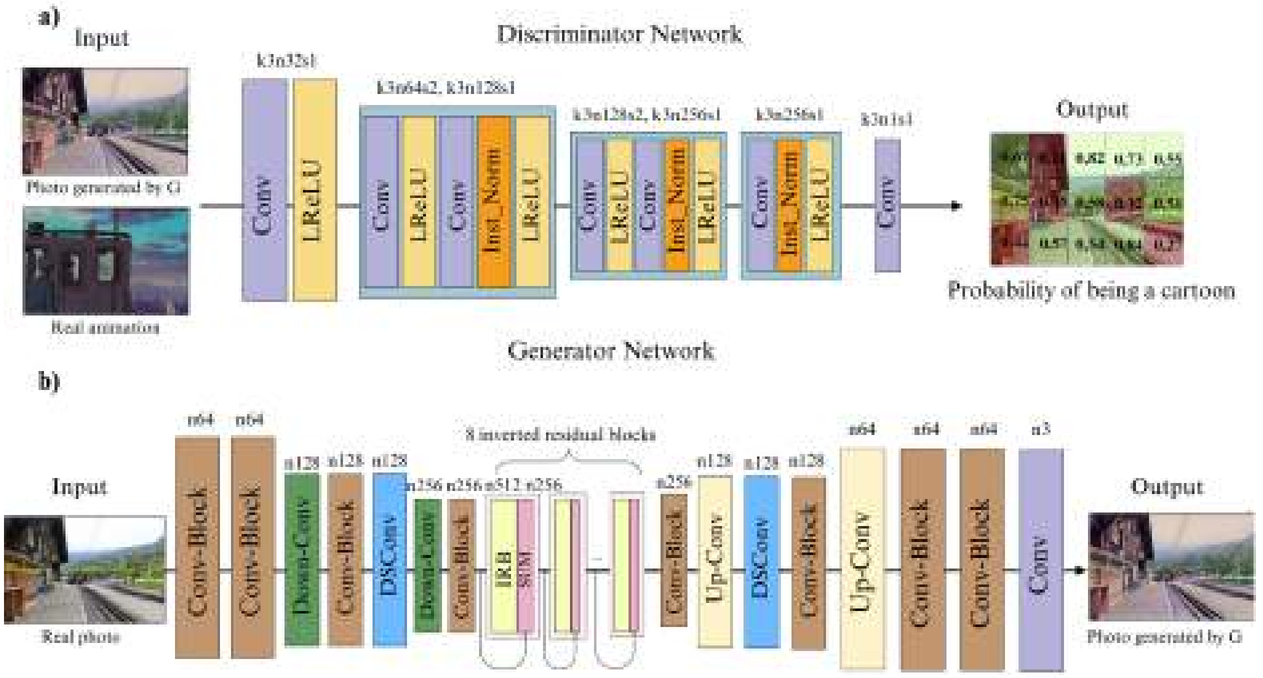


Figure 3.6: The architecture of the generator (b) and the discriminator (a) in the proposed AnimeGAN model. In the generator, on all boxes represent the number of channels and “SUM” means the element-wise sum. In the discriminator, each of convolutional layers vary in kernel sizes (k), number of feature maps (n), and strides (s). The “Inst Norm” indicates the instance normalization layer.

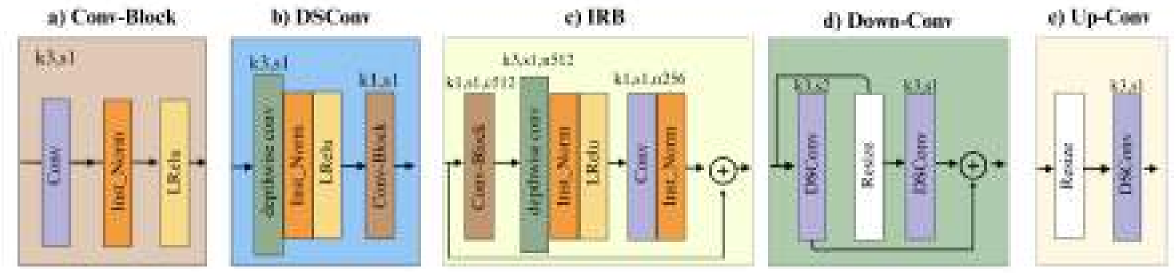


Figure 3.7: The AnimeGAN architecture includes various blocks of components: a) Conv-Block, b) DSConv, c) IRB, d) Down-Conv, and e)Up-Conv. Each of these components consists of convolutional layers that vary in kernel sizes (k), number of feature maps (n), and strides (s). The “Resize” block corresponds to the interpolation method used to set the size of the feature maps, and the symbol $\oplus$ represents the element-wise addition operation.

maps. The output from the Down-Conv module is the sum of the outputs from the DSConv modules with strides of 2 and 1.
Another critical part of the AnimeGAN generator network architecture is the use of inverted residual blocks to effectively reduce the number of parameters. Eight identical IRBs are used consecutively in the middle of the network. These IRBs are not only significantly less computationally intensive compared to standard residual blocks, but they also drastically reduce the parameter count of the network. The IRB used in the generator includes pointwise convolution with 512 kernels, depthwise convolution with 512 kernels, and pointwise convolution with 256 kernels. Interestingly, the final convolution layer does not utilize an activation function.
The upsampling technique referred to as Up-Conv is used to increase the resolution of the feature maps. The Up-Conv architecture effectively enlarges the size of the input feature maps by a factor of two. Instead of resorting to the commonly adopted method of a fractionally strided convolutional layer with a stride of 1/2 as in CartoonGAN. AnimeGAN employs the Up-Conv module. This decision was motivated by the observation that the traditional method tends to induce checkerboard artifacts in the synthesized images, thus compromising the quality of the final output [34].
In the generator, the last convolutional layer with 1 × 1 convolution kernels does not use the normalization layer and is followed by the tanh nonlinear activation function.

3.4.2 Loss Functions

As above mentioned models AnimeGAN is designed to transform real-world photographs into animation images by establishing an image-to-image mapping model, which bridges the domain of real-world photos $S_{data}(p) \in \mathcal{P}$ and the animation domain $S_{data}(a) \in \mathcal{A}$ using unpaired training data.
In the transformation process, authors focus more on transferring the texture rather than the color of the animated images to the photo images. To achieve this, the grayscale Gram matrix is proposed. The training animation set $S_{data}(a)$ is transformed into grayscale animation set $S_{data}(x) \in \mathcal{X}$ in order to eliminate color interference. Moreover, identically to CartoonGAN the dataset $S_{data}(e)$ is constructed by removing the edges of the animation images. After applying a smoothing transformation, the smooth-edge dataset is also converted into a grayscale set, denoted as $S_{\text{data}}(y) \in \mathcal{Y}$.
The total loss function in AnimeGAN, denoted as $L(G{,}D)$, is a weighted amalgamation of four individual loss functions, each having a distinct role:

$$L(G{,}D) = \omega_{\text{adv}} L_{\text{adv}}(G{,}D) + \omega_{\text{con}} L_{\text{con}}(G{,}D) + \omega_{\text{gra}} L_{\text{gra}}(G{,}D) + \omega_{\text{col}} L_{\text{col}}(G{,}D) \tag{3.13}$$

In Equation (3.13), $L_{adv}(G{,}D)$ represents the adversarial loss which influences the animation conversion process within the generator G, while $L_{con}(G{,}D)$ denotes the content loss that aids in retaining the content of the input photo in the output image. $L_{gra}(G{,}D)$ is a grayscale style loss that allows the generated images to possess the distinct anime style in textures and lines. However, the application of the grayscale style loss may result in the generated image being presented as a grayscale image. To solve this, $L_{col}(G{,}D)$ is proposed as the color reconstruction loss, ensuring that the generated images maintain the color of the original photos. The weights ω_{adv}, ω_{con}, ω_{gra}, and ω_{col} are used to establish a balance among these four loss functions.
AnimeGAN utilizes the least squares loss function from LSGANs to create high-quality images and stabilize the network's training process. Unlike conventional GANs which use the sigmoid cross-entropy loss function, LSGANs penalize samples distant from the decision boundary. This promotes the generator to produce samples closer to the real data manifold and helps mitigate the vanishing gradient problem, enhancing learning stability.
The content loss $L_{con}(G{,}D)$ and the grayscale style loss $L_{gra}(G{,}D)$ are computed using pre-trained VGG19 networks, which extract high-level semantic features from images. The loss formulations are as follows:

$$L_{con}(G{,}D) = E_{p_i \sim S_{data}(p)}[|VGG_l(p_i) - VGG_l(G(p_i))|_1] \tag{3.14}$$

$$L_{gra}(G{,}D) = E_{p_i \sim S_{data}(p)}, E_{x_i \sim S_{data}(x)}[|\text{Gram}(VGG_l(G(p_i))) - \text{Gram}(VGG_l(x_i))|_1] \tag{3.15}$$

These losses are calculated using ℓ_1 sparse regularization.

To improve the reconstruction of image color, the RGB format color image is transformed into the YUV format. For the loss $L_{col}(G,D)$, the ℓ_1 loss is used for the Y channel, and the Huber Loss is applied to the U and V channels. This can be formally defined as:

$$L_{col}(G,D) = E_{p_i \sim S_{data}(p)}\left[|Y(G(p_i)) - Y(p_i)|_1 + |U(G(p_i)) - U(p_i)|_H + |V(G(p_i)) - V(p_i)|_H\right] \tag{3.16}$$

In this equation, $Y(p_i)$, $U(p_i)$, and $V(p_i)$ symbolize the three channels of the image p_i in the YUV format, with H representing the Huber Loss.
The generator's loss function can be expressed as:

$$L(G) = \omega_{\text{adv}}\mathbb{E}_{p_i \sim S_{\text{data}}(p)}[(G(p_i) - 1)^2] + \omega_{\text{con}}L_{\text{con}}(G, D) + \omega_{\text{gra}}L_{\text{gra}}(G, D) + \omega_{\text{col}}L_{\text{col}}(G, D) \tag{3.17}$$

In addition to introducing an edge-promoting adversarial loss to the discriminator loss function, which ensures clear edges in AnimeGAN-generated images, a unique grayscale adversarial loss is incorporated to prevent the generated image from displaying as a grayscale image. The loss function of the discriminator can then be written as:

$$\begin{aligned} L(D) = &\omega_{\text{adv}}[\mathbb{E}_{a_i \sim S_{\text{data}}(a)}[(D(a_i) - 1)^2] + \mathbb{E}_{p_i \sim S_{\text{data}}(p)}[(D(G(p_i)))^2] \\ &+ \mathbb{E}_{x_i \sim S_{\text{data}}(x)}[(D(x_i))^2] + \omega_s\mathbb{E}_{y_i \sim S_{\text{data}}(y)}[(D(y_i))^2]] \end{aligned} \tag{3.18}$$

Here, $\mathbb{E}_{y_i \sim S_{\text{data}}(y)}[(D(y_i))^2]$ denotes the edge-promoting adversarial loss, and $\mathbb{E}_{x_i \sim S_{\text{data}}(x)}[(D(x_i))^2]$ signifies the grayscale adversarial loss.

3.5 Summary

This chapter presents an overview of literature state of the art models for image-to-image translation, particularly focusing on the transformation of real world images to cartoon style illustrations.
CycleGAN, a prominent GAN-based model in the field, allows for unpaired image translation between two different domains, using adversarial loss and cycle consistency loss. Although not originally intended for cartoonization, CycleGAN has demonstrated promising results in related fields. Its flexible nature and potential ability to create high quality cartoon images from real world photos make it a promising candidate for evaluation in this study.
Conversely, CartoonGAN is one of the most early models from literature purpose built for photo cartoonization, leveraging the robust capabilities of GANs to produce visually appealing and stylistically consistent cartoon images. It combines content loss and adversarial loss to maintain the structure of input photos while applying the desired cartoon style.
GANILLA, heavily influenced by CycleGAN, is another unique model used for unpaired image-to-image translation tasks. Despite sharing loss functions with CycleGAN, it differs in its generator architecture. This model demonstrates versatility and potential adaptability for diverse applications, including cartoon generation, by successfully emulating children's book illustrations.
Lastly, AnimeGAN is a lightweight GAN designed to transform real world images into anime style illustrations. It introduces a novel grayscale style loss and color reconstruction loss for better transformation and color conservation. To ensure rapid transformation, the generator design involves depthwise separable convolutions and inverted residual blocks. This architecture, being the most recent among the discussed models from the literature, shows promising potential and is anticipated to deliver superior performance. The authors test results affirm that AnimeGAN effectively transforms photographs of real world scenes into anime style images, surpassing the performance of other leading models [11, 54, 20].
In summary, each model has unique strengths and applications in the task of transforming real-world images into cartoon-style illustrations.

4 Author's Approach: LightAnimeGAN

The field of cartoon generation has seen a rapid evolution, enriched by an array of architectures, as referenced in Chapter 3. Among these are CycleGAN [54], GANILLA [20], CartoonGAN [11], and AnimeGAN [9], each distinguished by its unique advantages, disadvantages, and model characteristics. Despite these models demonstrating satisfactory results in cartoon generation tasks, there remains considerable scope for further improvements across all these architectures. Notably, the evaluation of GANs presents significant challenges, as outcomes can greatly vary based on the input image and the chosen cartoon style, as discussed in Section 2.6. Therefore, selecting the best model is not a straightforward decision.

However, in their pioneering experiments, Chen et al. have underscored the distinct effectiveness of AnimeGAN [9]. Their findings demonstrate that AnimeGAN has the capability to transform real-world photos into cartoons, exhibiting high-quality results. In conclusion of the paper, is written: *"The experiments show that AnimeGAN can transform photos of real-world scenes to anime style images with high-quality animated visual effects and significantly outperforms the state-of-the-art stylization method"*. [9] Such insights suggest that, relative to other models examined thus far, AnimeGAN potentially offers an optimized architecture for cartoon generation tasks that's why this model was chosen as a baseline for our approach.

Nevertheless, it's important to acknowledge that AnimeGAN, like any model, is not devoid of limitations. A significant drawback of AnimeGAN is its tendency to generate high-frequency artifacts within the images it creates. These artifacts, even though typically minor, can considerably impact the overall image quality, thereby marking a clear opportunity for enhancement. Furthermore, although AnimeGAN possesses fewer parameters than the CartoonGAN model (as demonstrated in Table 6.2), we think there is potential to reduce the number of parameters even further.

In light of these areas of potential improvement, we propose a modified version of AnimeGAN titled LightAnimeGAN. This refined model builds on the foundational structure of AnimeGAN but integrates key modifications designed to address its identified limitations. The primary aim of LightAnimeGAN is to boost the efficiency and quality of cartoon generation tasks.

Model details

Despite AnimeGAN's noteworthy capabilities in cartoon generation, it has its share of limitations. One of the most pronounced is the generation of high-frequency artifacts in the images created by the model. These artifacts can significantly detract from the overall quality and appeal of the generated cartoons. In this research to address these challenges, we introduce a modified version called LightAnimeGAN. This refined model prevents the generation of high-frequency artifacts by implementing a simple yet effective change: modifying the normalization of features in the network. Further, we have scaled down the size of the generator network by skipping not effective layers and reducing the number of feature maps, enhancing the efficiency of the animation style transfer process.

Traditionally, instance normalization has been considered a superior normalization method in style transfer tasks. This method enables different channels in the feature map to possess diverse feature characteristics, thereby promoting a rich variety of styles and high frequency in the model-generated images. However, for cartoon generation, instance normalization might not be the most effective approach. Cartoons often rely on a more abstract visual representation, marked by a reduction in details, exaggerated features, and simpler color schemes.

Our inspiration for this modification stems from the observation of StyleGAN models [25, 26]. Like AnimeGAN, StyleGAN applies instance normalization and tends to produce high-frequency images. Unfortunately, total variation loss cannot completely suppress the generation of high-frequency noise. Although this

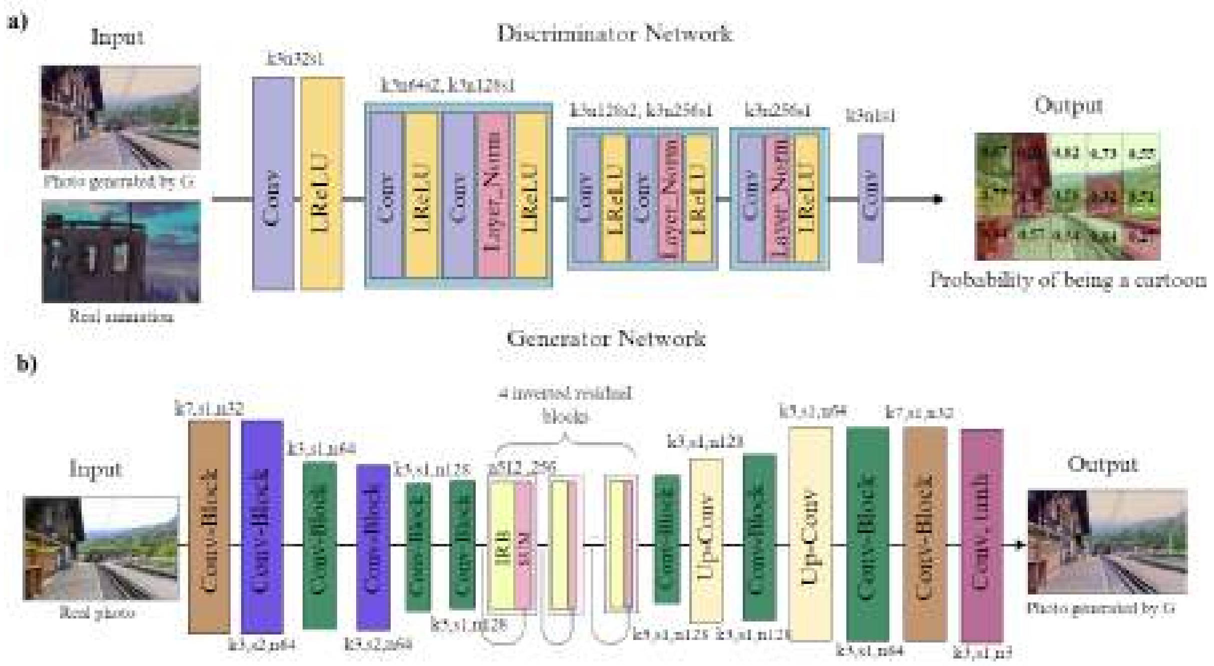


Figure 4.1: This figure depicts the architecture of the generator (b) and the discriminator (a) in the proposed LightAnimeGAN model. In the generator diagram, each box represents the number of channels. The term ”SUM” indicates an element-wise sum. Each convolutional layer has differing attributes such as kernel sizes (k), number of feature maps (n), and strides (s). Distinct color assignments in the ”Conv Block” represent different combinations of (k) and (s). Detailed specifics of each block can be found in Figure 4.2.

is required in StyleGAN models for effective style, texture, and color pattern transfer, it might not adequately capture the abstraction, simplification, and exaggeration inherent in cartoon styles.

By employing layer normalization instead, we allow different channels in the feature map to have the same distribution of feature properties. This effectively curbs the generation of local noise and high-frequency images.

In our pursuit of improved cartoon generation, we have made several modifications to the network of AnimeGAN. It is noteworthy to mention that we deemed the discriminator network in AnimeGAN to be optimal, therefore, the only alteration we introduced to this network was to switch from instance normalization to layer normalization. This change was primarily introduced to mitigate the high-frequency artifact problem associated with instance normalization. A graphical representation of the revised LightAnimeGAN discriminator architecture is shown in Figure 4.1 (a).

Subsequently, we focused on modifying the generator network of LightAnimeGAN. We aimed to resolve the high-frequency issue and to improve the efficiency of the animation style transfer process. This was achieved by reducing the size of the generator network, a process that involved skipping select layers and decreasing the number of feature maps.

As in the discriminator network, we replaced instance normalization with layer normalization in the generator network. Furthermore, we decided to remove the DSConv blocks and substituted the Down-Conv block with a standard convolution block. Each convolutional layer possesses unique attributes, the specifics of which are determined by the color designation of each block (Figure 4.2). We did this with the belief that, since we were reducing the number of feature maps at the input layer from 64 to 32, there was no need to reduce the resolution of the feature map.

Lastly, we streamlined the architecture further by decreasing the number of inverted residual blocks from 8 to 4. This not only simplified the model but also boosted the efficiency of the style transfer process. Through the modifications made to the generator network, we managed to achieve a substantial reduction in the model's parameter size and trainable parameters number. It decreased from 15.8MB (size), 39.6 mln (n_o parameters)

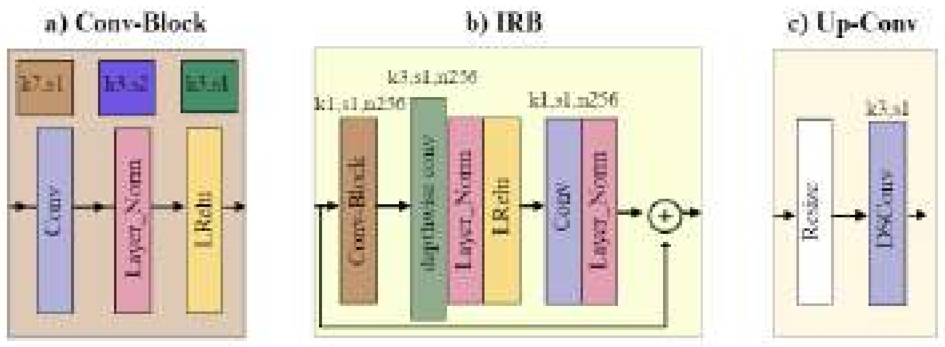


Figure 4.2: The LightAnimeGAN architecture includes various blocks of components: a) Conv-Block, b) IRB, c) Up-Conv. Each of these components consists of convolutional layers that vary in kernel sizes (k), number of feature maps (n), and strides (s). The "Resize" block corresponds to the interpolation method used to set the size of the feature maps, and the symbol $\bigoplus$ represents the element-wise addition operation.

in the original AnimeGAN to 8.6MB, 2.2 (mln) in our proposed LightAnimeGAN model.
The restructured architecture of the LightAnimeGAN generator is depicted in Figure 4.1 (b). The goal of these modifications is to address the known limitations of AnimeGAN and to optimize its performance, consequently improving the quality of cartoon generation.
The restructured architecture of the LightAnimeGAN generator is depicted in Figure 4.1 (b). The goal of these modifications is to address the known limitations of AnimeGAN and to optimize its performance, consequently improving the quality of cartoon generation.
In our exploration and modification of the AnimeGAN model, we found certain elements to be already optimized for our purposes. As such, we made the decision to keep these components unchanged in our proposed LightAnimeGAN model. That's why, loss functions used in the original AnimeGAN have been carried over to LightAnimeGAN without any alteration (Section 3.4.2). Experiments indicate that these loss functions, as implemented in the original AnimeGAN, already perform optimally for the tasks at hand and hence do not require further modifications.

Summary

This research aimed to improve the AnimeGAN model, which had a significant issue of producing high-frequency artifacts in generated images. A modified model, LightAnimeGAN, was introduced, which used layer normalization instead of instance normalization to address this problem. Key modifications also included downsizing the generator network and streamlining its architecture to improve efficiency. The discriminator network was largely left unchanged, except for the normalization switch. Certain elements, such as loss functions, remained the same due to their optimal performance in the original AnimeGAN model.
In Chapter 6, we conduct a performance assessment and evaluation of our proposed LightAnimeGAN model, comparing its effectiveness with the previously discussed state-of-the-art models.

5 Dataset

In this chapter, we discuss the dataset used for training and evaluating the GAN models for cartoon generation. The dataset consists of two distinct sets: real-world photos as content images and cartoons exactly anime images as style images. Both sets were used as the training data, while the test data includes only real-world photos. The resolution of all training images is set to 256 × 256. All of the mentioned architectures, including CycleGAN, AnimeGAN, CartoonGAN, and GANILLA [54, 11, 9, 20], were trained and evaluated on this dataset. This allowed for a fair comparison of their performance in generating cartoons in the specific styles of the artists. By using the same dataset for training and evaluation, we can better understand the strengths and weaknesses of each model in capturing the unique artistic direction of the chosen style images and transferring that style to the content images.

Content Images

In total 6,656 real-world photos are used for training as content images. The photos were crowdsourced from Flickr using the combination of tags ”landscape” and ”landscape photography”. Black and white photos were trimmed. The dataset consists of diverse images depicting various scenes, objects, and environments, providing a rich source of real-life content for the GAN models to learn from. Randomly chosen examples from the content dataset are depicted in Figure 5.1.

Style Images

To obtain a series of anime images with the same style, we used key frames of animated films drawn and directed by the same artist as the style images. Given that the film frames are being utilized for academic purposes and no profit is being derived from their use, it is possible to apply fair use of a copyrighted work license principles section 106, allowing the use of the mentioned dataset without publishing it [50]. Different animation artists have their own unique animation creation styles, which are reflected in the chosen style images.
In our experiments, we used two sets of style images from two animated films:

- For the Miyazaki Hayao style model, we used 1,792 animation images from the movie ”The Wind Rises”

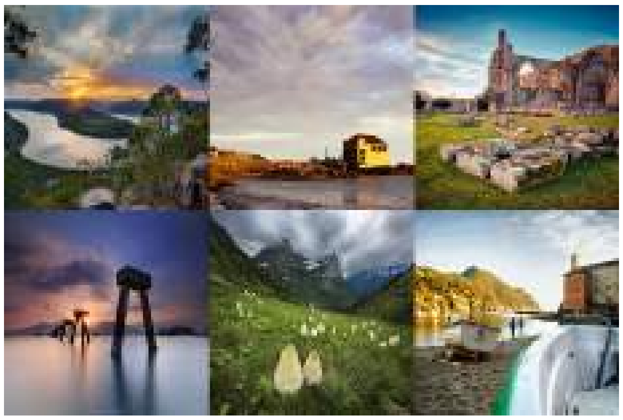


Figure 5.1: Randomly chosen examples from the content images set downloaded from Flickr.

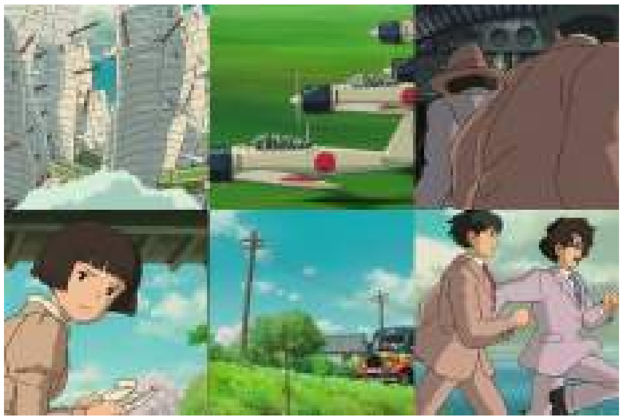


Figure 5.2: Randomly chosen examples from the Hayao style dataset [1].

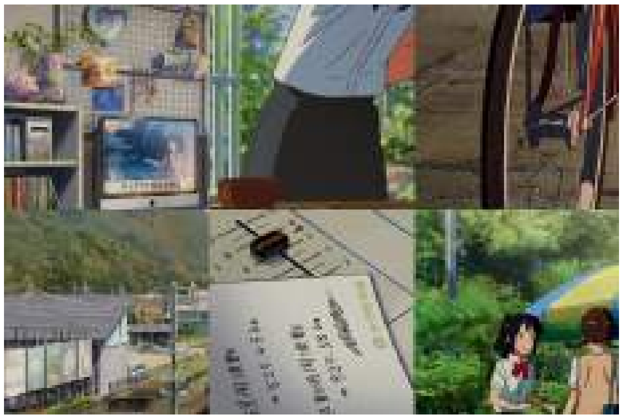


Figure 5.3: Randomly chosen examples from the Shinkai style datase [2].

directed by Miyazaki Hayao and produced by Studio Ghibli [1]. Randomly chosen examples from the Hayao style dataset are depicted in Figure 5.2.

- For the Makoto Shinkai style model, we used 1,650 animation images from the movie ”Your Name” directed by Makoto Shinkai and produced by CoMix Wave Films [2]. Randomly chosen examples from the Shinkai style dataset are depicted in Figure 5.3.

These style images provide a consistent artistic direction for the chosen GAN models to learn and generate cartoons in the specific styles of the artists.

Data Preparation and Preprocessing

To prepare the dataset, we used PySceneDetect (library which analyzes a video, looking for scene changes or cuts) to capture 3 images per movie scene (at the start, middle, and end of each scene) in a resolution of 256x256 pixels. Then using VisiPic (Windows application that will search for duplicate photos) we deleted images too similar to each other.
Deep neural networks can suffer from overconfidence, meaning they may rely on only a few features to classify an object. To address this issue, deep learning techniques such as regularization and dropout are used to prevent overconfidence [44]. Similarly, in GANs, if the discriminator relies on a limited set of features to identify real images, the generator may simply produce these features to deceive the discriminator. This could lead to overly greedy optimization, which does not provide long-term benefits [39]. In cartoon generation problems, the generator will unnaturally enhance clear edges and color shading. Therefore, inspired by other researchers (Section 3.2.2), we created a smoothed dataset from our style dataset (Figure 5.4). This preprocessing step

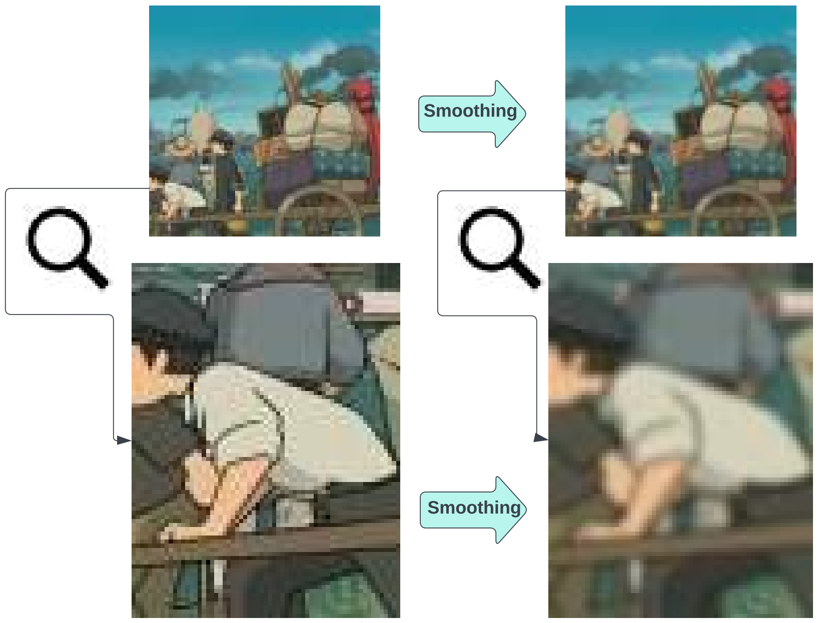


Figure 5.4: Visual example showing effects of the smoothing transformation on a random sample from the Hayao dataset (top) and enhanced fragment of an image (bottom). The left images show the original, unchanged samples, while the right images display the results after applying the smoothing transformation. By comparing the images, you can observe the impact of the smoothing process on the overall appearance and edge quality of the samples.

allows the (CartoonGAN [11], AnimeGAN [9]) generator to better learn and reproduce the desired style. This was done by first detecting edge pixels using a Canny filter with a threshold between 150 and 500 and thereafter applying a Gaussian blur filter on a 3x3 pixel dilation of the identified edges [7]. All image filter handling was done using OpenCV.

Summary

In this chapter, we presented the dataset used for training and evaluating the GAN models for cartoon generation. The dataset consists of real-world photos as content images and anime images from specific animated films as style images. The process of data preparation and preprocessing was also described, including the extraction of key frames, removal of similar images, and creation of a smoothed dataset. The dataset discussed in this study was only used for academic purposes and will not be published or distributed elsewhere. The style image set can be easily reproduced for any artist or style by assembling a customized dataset. By following the described data preparation techniques it is possible to create own style datasets. Described images provide a rich source of data for the GAN models to learn the underlying structure and patterns in the content and style domains, enabling them to generate diverse and visually appealing cartoon images.

6 Expirements

Each of the models detailed in Chapter 3, with the new approach LightAnimeGAN described in Chapter 4, were built and trained using PyTorch in the Python language. Regrettably, due to the high computational demand and time-consuming nature of training GANs, we were only able to train all of the discussed models on the Miyazaki Hayao dataset. However, as we anticipated that the AnimeGAN and LightAnimeGAN models would likely perform the best, these models were trained on both the Hayao and Shinakai datasets (Chapter 5) for a comparative evaluation of style transformation. During the implementation and training process, we followed the original authors' recommendations but also took measures to ensure a fair comparison among the different architectures by adopting as many uniform parameters and training techniques as possible.

In this section, we delve into the experiments, evaluation, and comparison of the models we described above. We explore each model individually, providing details on the implementation and training phases, as well as highlighting the unique parameters and characteristics inherent to each model. The software employed during the training phase is thoroughly discussed. We present the results achieved by each architecture, demonstrating their proficiency in converting real images into cartoons. The effectiveness of these conversions is visualized through combined plots and measured using Visual Inspection Analysis, User Study, and Fréchet Inception Distance metrics. Additionally, we conduct a comparison of the style translations between the selected architectures.

6.1 Software Used

For the implementation, we utilized the Python language in version 3.7.12 [5] and GCCcore in version 8.3.0 (Debian 8.3.0-6).

Libraries

The key library used for implementing all the models is **PyTorch**, version 1.9.1 [35], originally developed by Meta AI and later overseen by the PyTorch Foundation in 2022. [3]. The PyTorch library supports Tensor processing units (TPUs) computation and the construction of Deep Neural Networks (DNNs) built on auto differentiation in Python with strong graphics processing unit (GPU) acceleration.

In performing various tasks, we made use of specialized libraries. Among these, the **tqdm** library was used to show the progress of training, **Pillow** was instrumental for image loading and manipulation, **NumPy** was crucial for conducting numerical operations including reshaping image data arrays, and dealing with intricate matrix operations, while **Matplotlib** took on the task of visualizing images and graphing training metrics.

Google Cloud Platform

The model's training process was conducted thanks to the Google Cloud Platform and their 12-month free trial with a $300 credit to use with any Google Cloud services. Google Cloud Platform (GCP) is, along with Amazon Web Services and Microsoft Azure, among the three dominant players in the public cloud market. GCP is a platform consisting of cloud computing services used for developing, testing, and deploying solutions using Google infrastructure [4]. For the purpose of this research, a virtual machine equipped with one NVIDIA V100 GPU, 4 vCPU, and 15 GB RAM was chosen.

6.2 Implementation and Training

In the course of our experiments, we made a concerted effort to implement the parameters specified by the original authors. Our aim was to provide a fair comparison across the various architectures by employing as many consistent parameters and training techniques as possible. However, it's important to note that our primary objective was to attain optimal results in cartoon generation, rather than simply replicating the results documented by the authors.
While the authors' original parameters and training techniques served as our starting point, we found that in some instances, these did not yield the most desirable results for our specific dataset. In such cases, we took the liberty to fine-tune and adjust the parameters with the goal of achieving improved results.
Our approach was iterative and experimental. When the original parameters fell short in generating satisfactory outputs, we viewed this as an opportunity to explore other alternatives. This adaptive and flexible approach allowed us to extract the best performance from the given architecture for our specific dataset.
All models were trained and tested using the same dataset, as detailed in Chapter 5. The dataset was first prepared by organizing the data into the necessary folders and subfolders hierarchy as required by the model's data loaders. The Leaky ReLU (LReLU) was employed across all models, utilizing the parameters: $\alpha = 0.2$. To manage the extent of model training, we constrained the training phase to a maximum of 100 epochs. Given that the training of GAN model is a highly nonlinear optimization task and can easily become trapped in a sub-optimal local minimum with random initialization we address this issue, following the authors of the papers, we propose an initialization phase to enhance the model's convergence performance. It is essential to note that the generator network goal is to reconstruct the input photo in a cartoon style while maintaining the semantic content. The adversarial learning framework begins with a generator that reconstructs the content of input images. To this end, during the initialization phase, the generator network is pre-trained using only the semantic content loss $L_{con}(G,D)$. Consistent with the authors recommendations for the chosen models, an initialization phase comprising 10 initialization epochs was included in our study. During this phase, solely the content loss is utilized, which assists in stabilizing the training process.

CycleGAN

All of the experiments and training phases were conducted utilizing the repository shared by the CycleGAN authors. The PyTorch implementation of the repository can be accessed via this link: CycleGAN_Repo [54]. Researchers [20, 11] have found that the inclusion of an additional loss function that encourages the preservation of color composition between input and output is beneficial for generating painting like photos. This approach uses the technique introduced by Taigman et al. [47] called identity loss depicted in Equation (6.1). It aims to keep the generator close to an identity mapping when real samples from the target domain are used as input.

$$L_{ident}(G,F) = \mathbb{E}_{y\sim p_{\text{data}}(y)}[\|G(y) - y\|_1] + \mathbb{E}_{x\sim p_{data}(x)}[\|F(x) - x\|_1] \tag{6.1}$$

Without this identity loss L_{ident} with ℓ_1 distance function, the generators G and F could unnecessarily alter the color tone of input images. For instance, when translating between real photos and paintings, the generator could depict a daytime painting as a sunset photo, which is still valid under the adversarial and cycle consistency losses. This observation led us to believe that the identity loss would also be beneficial for generating cartoons, leading us to include it in the total loss function with the weight λ_{ident}. This modification yields the following total loss function:

$$\mathcal{L}(G,F,D_X,D_Y) = \mathcal{L}_{GAN}(G,D_Y) + \mathcal{L}_{GAN}(F,D_X) + \lambda_{cyc}\mathcal{L}_{cyc}(G,F) + \lambda_{ident}\mathcal{L}_{ident}(G,F) \tag{6.2}$$

In carrying out each experiment, we stayed faithful to the original author's guidelines. We adjusted the parameters λ_{cyc} and λ_{ident} to 10 and 0.5, respectively, in accordance with Equation (3.6). We didn't use any initialization epochs, instead, we trained all networks from scratch, with a learning rate that decreased over time. The initial weights were assigned based on a Gaussian distribution $N(0, 0.02)$. We made use of the Adam optimizer, with a batch size set to 1 because of cycle consistency, a learning rate at 0.0002, and $\beta_1 = 0.5$. According to the authors, this learning rate remained unchanged for the first 100 epochs and was then gradually brought down to zero over the next 100 epochs. However, since our training was limited to 100 epochs, we started applying linear decay after the 50th epoch.

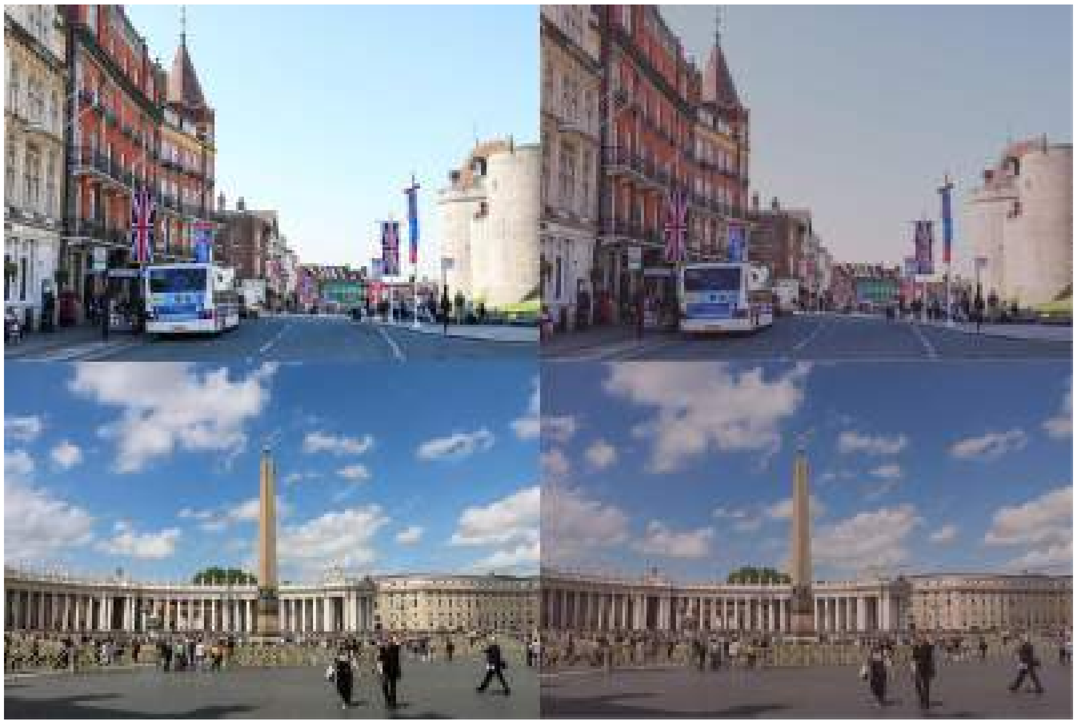


Figure 6.1: Comparison of original input images (left) and their corresponding cartoon like transformations generated by CartoonGAN model using the original paper's parameters (right).

GANILLA

All of the experiments and training phases were executed using the repository shared by the GANILLA authors. The PyTorch implementation of the repository can be accessed via this link: GANILLA_Repo [20]. Given that GANILLA is heavily inspired by CycleGAN model and shares the same loss functions, parameters, and learning techniques. The entire training and implementation process is the same as the one mentioned above for CycleGAN (Section 6.2).

CartoonGAN

All of the experiments and training phases were executed using the repository shared by the CartoonGAN authors. The PyTorch implementation of the repository can be accessed via this link: CartoonGAN_Repo [11].
In the CartoonGAN paper, Chen et al. proposed an initialization phase to stabilize the training process. This phase, conducted over ten epochs, utilized the Adam optimizer with a learning rate of 0.0001.
Our initial attempt at training the CartoonGAN model adhered to the hyperparameters suggested in the original paper. Parameter ω in Equation (3.9) was set to 10, which according to the CartoonGAN authors, achieves an optimal balance between style and content preservation. Each training iteration was conducted on a batch of 4 images, and the Adam optimizer was set to a constant learning rate of 0.0003 with a weight decay of 0.0001.
Despite strict adherence to these parameters, the results were unsatisfactory. The generated images were almost indistinguishable from the original images, contradicting our intent of creating a distinct style transformation. This issue is evident in Figure 6.1, where the left-hand images are the original input photos and the right-hand images are the cartoon renditions generated by our model using the original paper's parameters. The lack of cartoon like characteristics in the transformed images clearly indicates that our model did not perform as expected. This inconsistency may have arisen due to our use of a different dataset than the original authors. Our dataset contained four times fewer animation images compared to the original paper's dataset. Interestingly, we were not alone in encountering issues when attempting to replicate the original results using

the specified parameters [45]. Inspired by similar challenges faced by other researchers, we interpreted these issues as inherent characteristics of the network. Consequently, we strived to tune the parameters for optimal results.
This prompted us to experiment further and fine tune the hyperparameters to better suit our dataset. Subsequent changes resulted in significant improvements in the model's performance in cartoonization. However, these adjustments led to a decrease in the model's performance in content preservation. A key modification was replacing instance normalization with layer normalization in both the discriminator and generator. The decision was based on the significant performance difference and was validated by comparing the loss functions of the two normalization techniques.
Following a series of experiments, we refined the hyperparameters. Specifically, we adjusted the ω parameter to 1.5 and increased the learning rate of the Adam optimizer to 0.0002. These adjustments resulted in an enhanced style transfer, better emulating the cartoon aesthetics while still sufficiently preserving the content from the original images. However, it's important to note that the content often appears somewhat blurred and smudgy. Throughout the training process, we maintained a consistent learning rate of 0.0003 and a weight decay of 0.0001, executing each training iteration on a batch of 4 images.

AnimeGAN

All of the experiments and training phases were executed using the repository shared by the AnimeGAN authors. The PyTorch implementation of the repository can be accessed via this link: AnimeGAN_Repo [9].
The AnimeGAN model was implemented by adhering to specific procedural steps. Similar to the approach in CartoonGAN, an initialization process was conducted over ten epochs, with a learning rate set to 0.0001. Subsequent training of the AnimeGAN was characterized by distinct learning rates for the generator and the discriminator, set at 0.00008 and 0.00016 respectively. The training process spanned 100 epochs, utilizing a batch size of 4. To optimize the minimization of total loss, we employed the Adam optimizer.
In all experiments, the balance between style and content preservation was carefully regulated through the adjustment of the weights in Equation (3.13). These were used the same as in orginal AnimeGAN implementation set as $\omega_{\text{adv}} = 300$, $\omega_{\text{con}} = 1.5$, $\omega_{\text{gra}} = 3$, and $\omega_{\text{col}} = 10$.
Lastly, the scaling factor ω_s in Equation (3.18) was maintained at 0.1. This helped to prevent the edges of the generated image from appearing excessively sharp.

LightAnimeGAN

All of the experiments and training phases were executed using the repository shared by the LightAnimeGAN authors. The PyTorch implementation of the repository can be accessed via this link: LightAnimeGAN_Repo.
Given that LightAnimeGAN is heavily inspired by AnimeGAN model and shares the same loss functions, parameters, and learning techniques. The entire training and implementation process is the same as the one mentioned above for AnimeGAN (Section 6.2).

6.3 Overall Comparison

As shown in Table 6.2, there are differences between the considered models in terms of the number of parameters in the generator network and the training time required for one epoch on the described dataset.

	No of parameters G (10^6)	Generator size (MB)	Mean train time per epoch (sec)
CycleGAN	11.4	43.4	4408
GANILLA	7.2	27.5	2834
CartoonGAN	12.1	46.7	4498
AnimeGAN	39.6	15.8	3780
LightAnimeGAN	**2.2**	**8.6**	**2263**

Table 6.1: Comparison of models characteristics trained on the Hayao dataset.

From models taken from the literature, the GANILLA had the least number of parameters with only 7.2 million. This leads to a smaller size of the generator model at only 27.5MB, which can be advantageous in terms of the memory resources required to store and deploy the model. The training time for GANILLA was also relatively shorter than other models at 2834 seconds per epoch.
On the other hand, the AnimeGAN model had the highest number of parameters at 39.6 million, resulting in a larger model size of 15.8MB. However, despite the larger model size and the higher number of parameters, the generator size and the training time per epoch for the AnimeGAN were shorter than both CycleGAN and CartoonGAN.
CycleGAN and CartoonGAN had a moderate number of parameters and generator size, but they required the longest training time per epoch at 4408 seconds and 4498 seconds respectively. The long training time could potentially slow down the model development and tuning process.
The LightAnimeGAN model, as suggested by its name, features the smallest generator size of just 8.6MB, and had the shortest mean training time per epoch at 2263 seconds. However, it's noteworthy that the substantial difference in the number of parameters compared to other models does not correspondingly translate into large variations across other categories.
It should be noted that while the number of parameters, model size, and training time are important factors to consider when choosing a model, they do not dictate the performance of the model in generating cartoon-like images. Other factors such as the quality of the output images and the ability of the model to capture the unique characteristics of the target domain are the most crucial factor.

6.4 Results

After the implementation and training phase, we evaluated the proficiency of trained models in transforming real photographs into cartoon forms. This section contains a discussion and comparison of the results obtained from all models trained exclusively on the Hayao dataset. To extend the scope of our testing, we also trained the most promising architectures (AnimeGAN, LightAnimeGAN) on the unique styles of two artists (Hayao, Shinkai) and evaluated their performance in replicating the artist's styles.
We tested our models using a set of 21 high quality photographs depicting real world scenes. To diversify our dataset, we included 11 images of Japanese landmarks, in an effort to align the original photo domain as closely as possible with the animation domain. The remaining 10 images were selected from the author's private gallery and depict landmarks from various European countries. All of the high resolution cartoon images generated by the author are available online. To view and download these images, please follow this link.
Evaluating GANs introduces significant challenges as the outcomes can greatly fluctuate depending on the input image and chosen cartoon style. Moreover, the subjective nature of cartoon art adds an extra layer of complexity to this task, as detailed in Section 2.6. Following a thorough literature review, we selected Visual Inspection and Frechet Inception Distance as the most suitable metrics for evaluating GANs designed for cartoon generation. Moreover, to reduce the effect of subjectiveness in Visual Inspection, we conducted a user study to compare results.
We will apply these metrics to the mentioned above test set. Visual Inspection and User Study are human centric evaluation methods that offer insights into the aesthetic qualities and visual consistency of the generated images. In contrast, the FID provides a quantitative measure of similarity to a reference dataset distribution. The FID evaluates the model's ability to replicate the essential characteristics and diversity of the selected domain. Through the combined use of these metrics, we aim to achieve a comprehensive evaluation of our cartoon generation models, assessing both their visual attractiveness and their statistical alignment to dataset images.

6.4.1 Visual Inspection Analysis

In the following section, we present a selection of generated images to underscore the distinctive characteristics and tendencies exhibited by the studied models. This selection is designed to provide a richer understanding of each model's unique behaviors, as identified by the authors. The focus is on a visual assessment of the

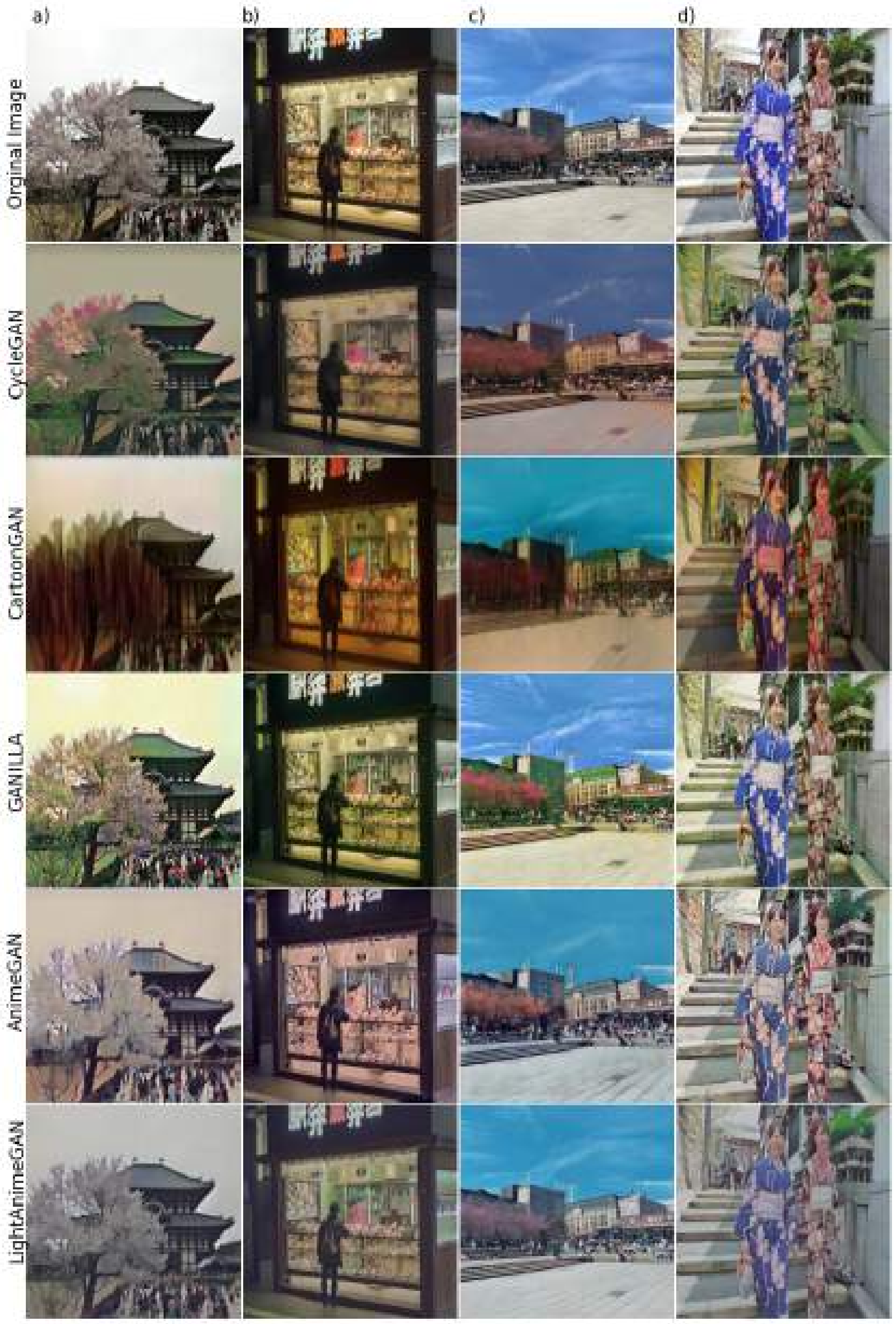
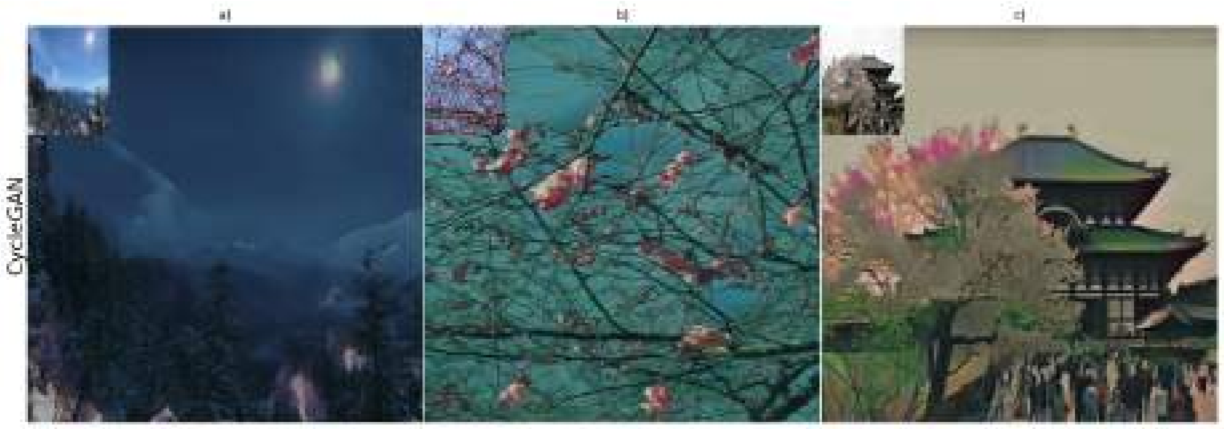
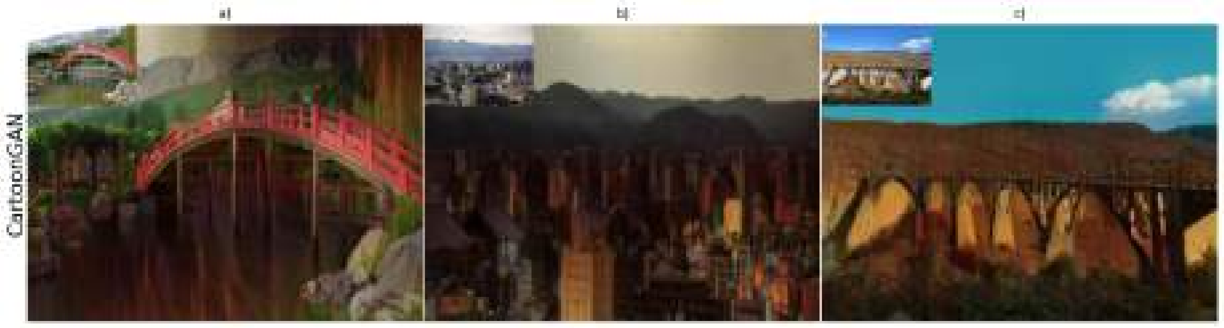


Figure 6.2: A selection of generated images for the test set, produced by the discussed architectures using original images as input. Each row represents the outputs from a different model.



(i) Images generated by CycleGAN architectire



(ii) Images generated by CartoonGAN architectire

Figure 6.3: Part1: Chosen test set images showcasing unique attributes of selected models, with the original input picture displayed in the top left corner of each image.

generated images, and on discussing the individual tendencies of each model. Furthermore, we perform a comparative analysis of style transformations using the two most promising architectures.

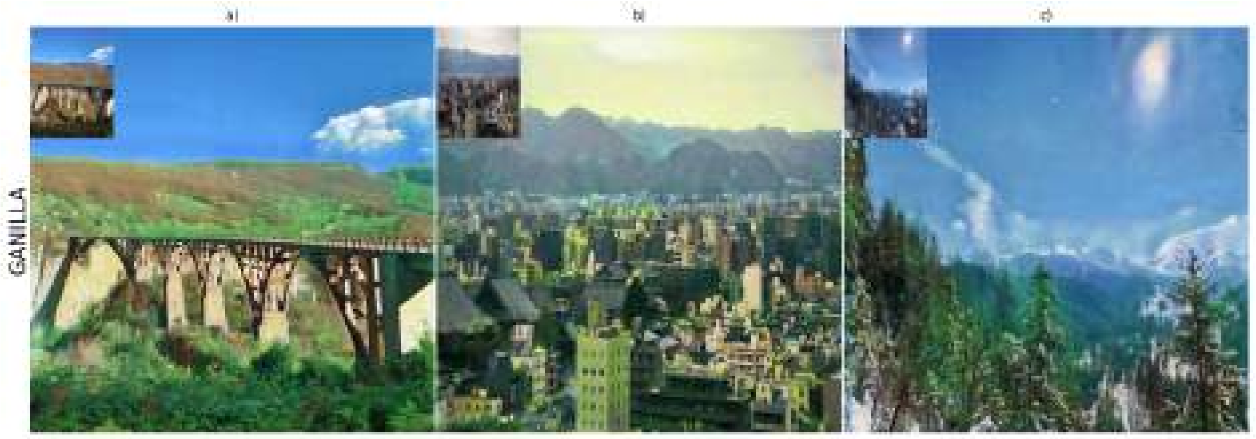
By meticulously examining the samples generated by each architecture in our test set, including the subset illustrated in Figures 6.2, 6.3, 6.5, several insights were observed.

From the initial visual assessment, it is evident that all examined models transformed input images and successfully embody the cartoon characteristics. While the degree of effectiveness varies from model to model, they all meet the criteria for satisfactory cartoon results. We depict and discuss these differences considering their performance and their approach to replicating the Hayao style.

First and foremost, observation is that despite each model being trained on the exact same dataset, they each learned somewhat distinct color distributions while attempting to mimic the Hayao style. The cartoonish color palette's importance is a key aspect of this study. The goal is to subtly modify the original image colors to create a more vibrant, surreal, and cartoonish output, but without drastically altering the color spectrum inherent to the image. Striking a balance between preserving the original colors and infusing the specific vibrant characteristics of Hayao's animation is essential for a more convincing cartoon style transformation. Achieving a balance between the preservation of original colors and the addition of the Hayao animation's specific color vibrancy is crucial for producing a more authentic cartoon transformation.

Secondly, the preservation of content, including shapes, details, and textures, is another critical factor. The transition to a cartoon style aims to simplify textures and eliminate unnecessary details. However, it's crucial that the image's core content remains recognizable. In this respect, smooth transitions play an integral role in maintaining a balance between enhancing stylistic elements and retaining the real world image's original composition and content. We notice varying degrees of simplification in the images generated by our models.

The third point of interest is the inclination to highlight the sharp edges of shapes, a feature common in



(iii) Images generated by GANILLA architectire

Figure 6.3: Part 2: Chosen test set images showcasing unique attributes of selected models, with the original input picture displayed in the top left corner of each image.

comic illustrations, such as pronounced black frames. The aim here is to strike a balance between visibly distinctive outlines without losing the content and oversimplifying the shapes.

Each model demonstrates unique characteristics that include preferences for certain color palettes, attention to detail, propensity for smoothing effects, and the tendency to accentuate sharp edges.

Our analysis reveals that the CycleGAN architecture strikes a noteworthy balance between preserving the original content of the image and incorporating cartoon like features. This is demonstrated by the smooth transitions and simplified textures it produces. Intriguingly, despite being trained on vibrant style images, CycleGAN often transforms the original image's color palette into darker and more muted tones, while still maintaining a reasonable color distribution (Figure 6.3i a,b). However, it occasionally generates artificial elements into empty areas that were not present in the original image, such as adding a green blur to leafless trees (Figure 6.3i b,c).

Additionally, as a way of visually demonstrating the concept of cycle consistency and testing the performance of the CycleGAN model, we performed transformations between the real and cartoon domains in both directions. This process involved taking an image from the real domain, transforming it to the cartoon domain using the forward cycle consistency, and then reversing the process using backward cycle consistency. These results are depicted in Figure 6.4.

To achieve the best results, we specifically selected real and cartoon images that depict similar content. The process was performed on both a real photo and a cartoon image. As can be seen from the top panel of Figure 6.4, our model can effectively switch manifolds between real photos and cartoon images. However, when it comes to generating realistic images from cartoon ones, as depicted in the bottom panel of Figure 6.4, the results are less satisfactory. The output image appears artificial and lacks the realistic details found in actual photographs. This observation suggests that while our model can reasonably capture and replicate cartoon styles, it struggles to accurately recreate realistic styles from cartoon images. This deficiency may be attributed to our training focus, which predominantly emphasized transformations from real photographs to cartoon images. Consequently, the model's ability to convert images in the reverse direction could have been inadvertently compromised.

In comparison, the primary limitation of the GANILLA architecture is its inclination to dramatically alter the original color palette, frequently changing original colors to bright and unrealistic shades of green or pink (Figure 6.3iii b, c). This characteristic is indeed comparable to what we observed in the CycleGAN architecture. However, in the GANILLA model, this tendency to drastically alter the original color palette is significantly more pronounced. While this occasionally results in visually captivating outputs for natural landscapes with green or pink elements, such as grass and flowers (Figure 6.3iii a), it typically fails to preserve the original color and content of the images, leading to unrealistic outcomes like green buildings (Figure 6.3iii b). In comparison to other architectures, GANILLA seems to display less pronounced cartoon

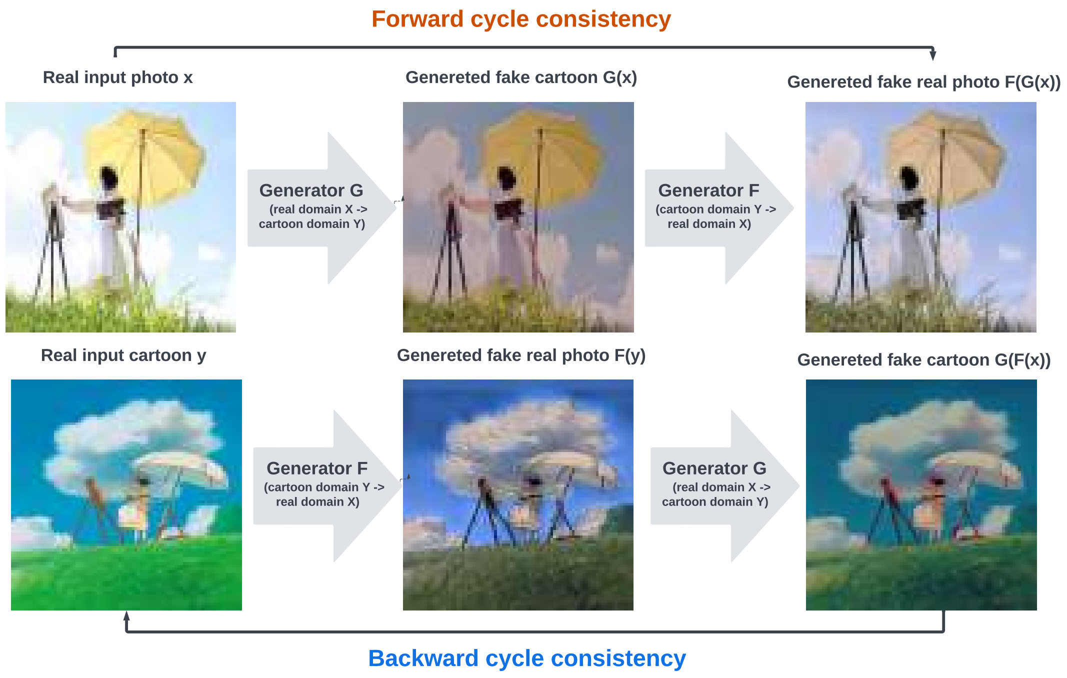

Forward cycle consistency
Real input photo x
Genereted fake cartoon G(x)
Genereted fake real photo F(G(x))
Generator G
(real domain X -> cartoon domain Y)
Generator F
(cartoon domain Y -> real domain X)
Real input cartoon y
Genereted fake real photo F(y)
Genereted fake cartoon G(F(x))
Generator F
(cartoon domain Y -> real domain X)
Generator G
(real domain X -> cartoon domain Y)
Backward cycle consistency

Figure 6.4: Illustration of cycle consistency. The figure depicts the transformation from a real image to a cartoon domain using a forward cycle, and the reverse transformation using a backward cycle consistency.

like characteristics, exhibiting less smooth transitions and more preserved textures, suggesting a stronger emphasis on content preservation over cartoonization. However, its ability to accentuate sharp edges with cartoon frames is commendable. Despite producing some visually attractive images, the performance of the GANILLA model is compromised by the overall distortion of the color palette, the absence of smooth transitions, and an overemphasis on the preservation of original content.

The CartoonGAN model, on the other hand, shows a tendency towards dark and brown colors, dramatically failing to retain the original colors. Additionally, the images it generates often appear smeared, causing a significant loss of original content. This imparts a dream like or even depressive quality to the output images, but this comes at the expense of realistic cartoon style replication. The presence of random straight lines in unexpected places further obscures the images and undermines the preservation of original content (Figure 6.3ii a, b, c). Among all the architectures we evaluated, CartoonGAN appears to face the greatest challenges with content preservation. It's important to note that these results deviate from the findings of the original CartoonGAN authors, which is likely due to the adjustments we made during training, specifically reducing the original ω parameter from 10 to 1.5. As discussed in Section 6.2, this modification allowed us to achieve the most desirable balance.

When comparing the results across all models (Figure 6.2, 6.5), as expected, AnimeGAN and LightAnimeGAN appear to provide the best outcomes, outperforming the other models. Both of these architectures successfully replicate the original image color palette without introducing unnecessary noise or smudges, likely due to a specialized loss function focusing on color replication. They also effectively preserve the content of the original image and apply well cartoon characteristics.

AnimeGAN, however, has a tendency to generate an excessive number of individual shapes separated by wavy lines, which ultimately disrupts the structure of the image and leads to a blurry yet overly detailed output. This pattern of curvy lines detracts from the aesthetic appeal of the images. Furthermore, we observe an unrealistic glow appearing around bright objects in images generated by AnimeGAN.

LightAnimeGAN, on the other hand, addresses these issues through architectural modifications. It solves the problem of curvy lines by producing cartoonish simple lines and shapes. The issue with brightness is also rectified, reducing the blur and emphasizing color transitions with simple frames splitting simple shapes. We

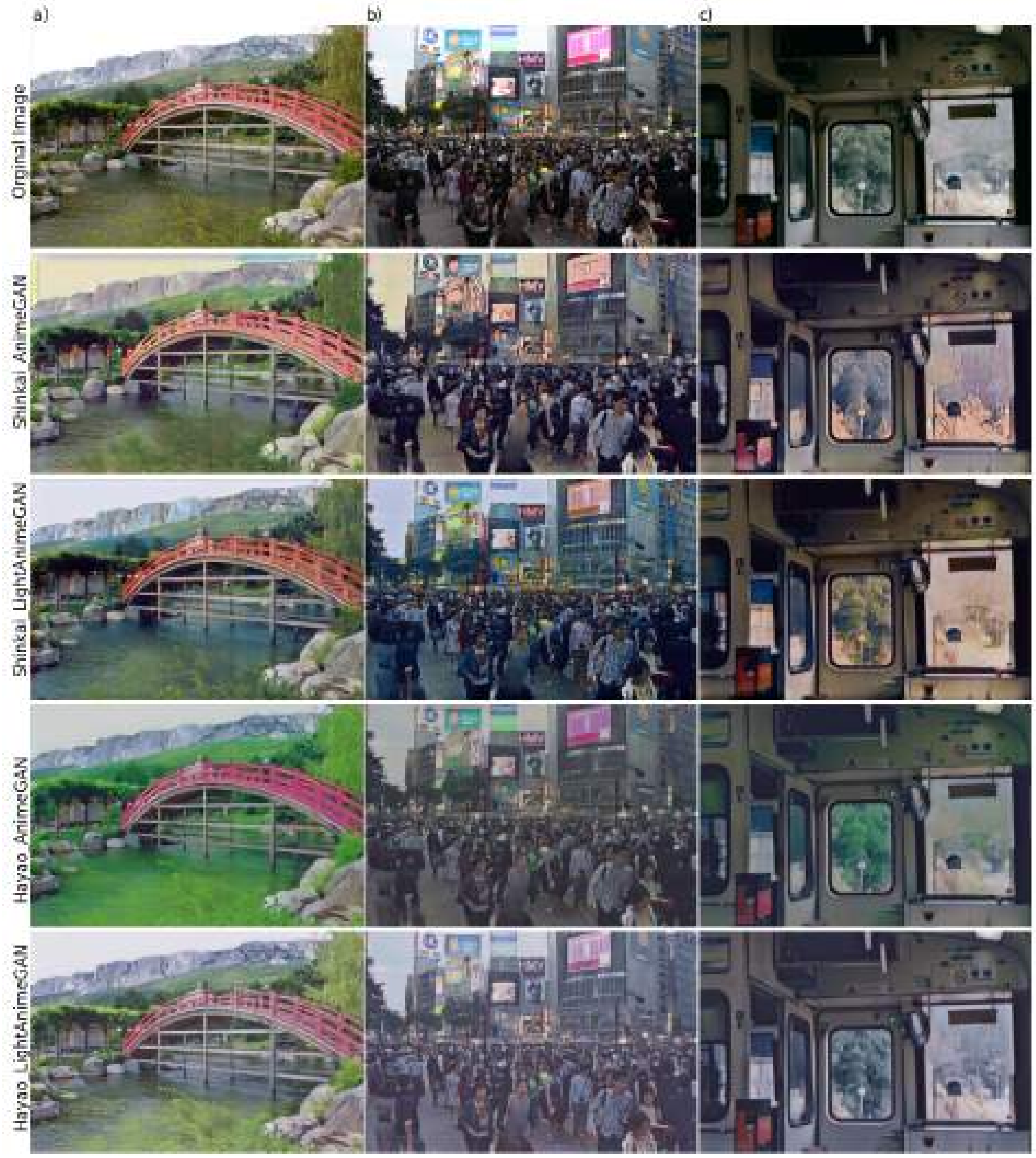


Figure 6.5: Selected images from the test set that exhibit the distinct traits and effectiveness of the AnimeGAN and LightAnimeGAN models, trained on specific style datasets, in accurately capturing the artistic styles of Hayao and Shinkai.

Table 6.2: Performance comparison of all models trained on the Hayao dataset using the FID metric.

Dataset \ Model	CycleGAN	GANILLA	CartoonGAN	AnimeGAN	LightAnimeGAN
Hayao Style	332.96	340.01	348.95	331.03	324.91
Input Images	160.32	109.46	165.34	129.97	84.39
Real Photo	299.31	290.22	312.12	302.34	299.84

observe that LightAnimeGAN tends to generate higher resolution images with superior visual appeal.
In conclusion, when comparing the two, LightAnimeGAN performs better overall, producing the most aesthetically pleasing cartoon like images while preserving original content.
To broaden our evaluation, we trained and tested the two most promising architectures, AnimeGAN and LightAnimeGAN, to emulate the distinctive styles of two artists, Hayao and Shinkai. Due to the significant computational demand of GAN training, we confined our comparison to these two models only. The selected samples, which are illustrated in Figure 6.5, highlight the unique attributes and effectiveness of the AnimeGAN and LightAnimeGAN models.
Both architectures displayed proficiency in style transformation in our test set. In the Hayao style, the models generated images with vibrant and colorful palettes, while in the Shinkai style, the output favored softer and subdued colors, aptly capturing the signature styles of the respective artists. This indicates the remarkable versatility of both the AnimeGAN and LightAnimeGAN models. However, based on our private judgment, LightAnimeGAN demonstrated superior performance, generating images of higher cartoon like quality.

6.4.2 Fréchet Inception Distance Analysis

To quantitatively evaluate the effectiveness of different GAN models in image generation, we utilize the Fréchet Inception Distance metric. As previously mentioned (Section 2.6), a lower FID value signifies superior performance in image generation, as it represents a smaller distribution distance between the generated images and the ground truth images from the analyzed set.
Evaluation of GANs is a complex task, and interpretation of the results from Table 6.2 necessitates a comprehensive understanding of what each column signifies, and the implications of calculating FID on different datasets within our image cartoonization task.
When considering the **Hayao Style** row, the FID calculation measures the distance between our generated set of test examples and the Hayao dataset. Similarly, in the **Input Images** row FID metrics mean the overall distance between the set of original input images and the images generated by each model. In these instances, a lower FID is preferable, as it implies that the distribution of our generated images closely mirrors the distributions of the cartoon (Hayao) and input images. This is indicative of successful content and style preservation, which is a fundamental objective of our task.
However, when considering the **Real Photo** row, we must interpret the FID results more discerningly. In this context, a lower FID indicates a close distribution to the images from **Real Photo** set, usually implying that the style and content of the real photos have been preserved too much. This may lead that the generated images retaining too many real life characteristics, resulting in a less satisfactory cartoonization of the images. Achieving the right balance between content preservation and cartoonization is the key challenge here, and hence, the FID metric for the **Real Photo** set cannot be interpreted in the same way as the other sets. A low FID for this set indicates that our generated images resemble real life photos that are distinct from the input images, which is not our desired outcome. Intriguingly, in some instances, a higher Fréchet Inception Distance might imply more effective cartoonization. This is due to the nature of the task. A greater deviation from the real photo set, indicated by a higher FID, might suggest that the model has successfully applied a more dramatic transformation to the images, effectively cartoonizing them. However, this is not always the case, as a high FID can also result from corrupted images that might contain random noise or other undesirable characteristics. Thus, we cannot analyze the FID for the **Real Photo** set in the same way as we do for others. For an optimal analysis, we should aim to identify the architecture that achieves the lowest FID in both **Hayao Style** and **Input Image** categories. However, to achieve a balanced result, we should also desire a high FID for the **Real Photo** category, signaling a successful divergence from the real life image features, and thereby effective cartoonization.

Table 6.3: Comparison of performance using FID metric of the AnimeGAN and LightAnimeGAN architectures trained on two different styles, Hayao and Shinkai artist datasets.

Style	Hayao Style		Shinkai Style	
Model / Dataset	AnimeGAN	LightAnimeGAN	AnimeGAN	LightAnimeGAN
Style	331.03	324.91	328.89	326.54
Input Images	129.97	84.39	86.48	126.83
Real Photo	302.34	299.84	280.39	296.43

This highlights the complexity of interpreting the FID metric, especially when applied to tasks like image cartoonization where a direct correlation between FID and image quality may not hold true. The FID metric can provide insights into the performance of different GAN models, but it may not comprehensively capture all aspects of image quality, such as style accuracy, content preservation, and the avoidance of artifacts or noise. Therefore, while we do take into account the FID results in our comparative analysis, we treat them more as suggestive indicators rather than absolute ground truth.

Given that we are comparing different datasets, it's important to remember that we can only compare the distance in distribution between them for each model. Essentially, the results obtained by different models can only be compared within the same dataset.

Analyzing FID for **Hayao Style** category from Table 6.2, which measures the model competence in replicating the unique Hayao cartoon style, we find that LightAnimeGAN outperforms the others. It achieves the lowest FID when tested on images in the Hayao style, implying its superior capability in mirroring the Hayao animation style. Not far behind in this category are AnimeGAN and CycleGAN, both showing impressive performances in creating images with distinct cartoon like features. Contrastingly, CartoonGAN and GANILLA exhibit less impressive results in the **Hayao Style** category, indicating a weaker ability to accurately capture the intended cartoon style. This conclusion aligns with our visual analysis and can likely be attributed to distinct issues within each model.

CartoonGAN, which is primarily designed to enhance cartoon like features through the parameter ω (as detailed in Section 6.2), faces difficulties in the remaining color palette replacing original colors with dark and brown ones. It also struggles with issues of random noise and unexpected straight lines, significantly deviating the output image from the input. This noise generation issue is also evident in the **Real Photo** and **Input Images** categories, where CartoonGAN records the highest FID, indicative of its failure to accurately preserve the content of the original image. Interestingly, while a high deviation from real life pictures is often a desirable trait in our context, this does not seem to be the case with CartoonGAN. The significant divergence here appears to be driven not by effective cartoonization but rather by the presence of random noise, as corroborated by its performance in the **Hayao Style** category.

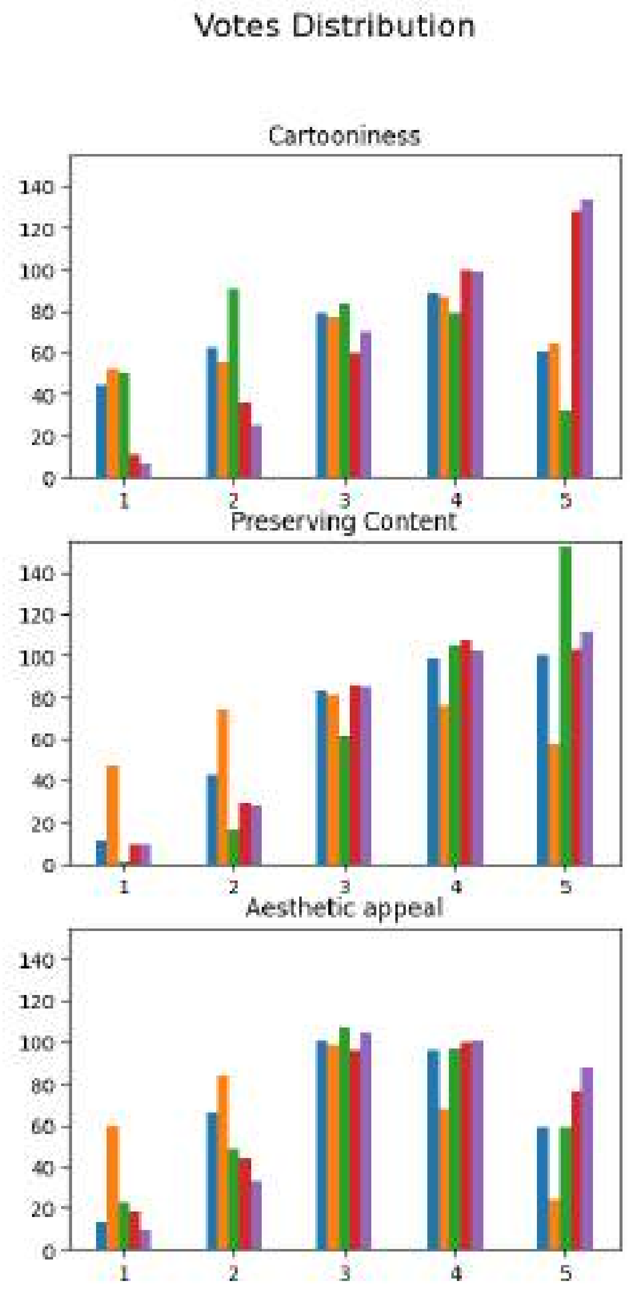
Contrastingly, GANILLA tends to excessively retain the properties of the original photo. This over preservation is especially apparent when examining the results with the **Input Images** and **Real Photo** set, where GANILLA achieves the lowest FID across all architectures. This suggests that the images produced by GANILLA are exceedingly similar to the input and real life photos, implying an inadequate degree of cartoonization. It points to a potential imbalance in GANILLA's transformation approach, seemingly prioritizing the preservation of real life image details over the creation of a cartoon like pictures.

In the **Original Images** category, LightAnimeGAN notably surpasses other models, achieving an impressively low FID score of 84.39, followed by AnimeGAN with a score of 129.97. This suggests a high degree of similarity between the images generated by these models and the original input images preserving input content. However, CycleGAN falls short in the **Input Images** category, registering a high FID score. This outcome seems to conflict with our earlier Visual Analysis, where we recognized CycleGAN's commendable ability to balance the preservation of original content with the introduction of cartoon like attributes. The discrepancy in CycleGAN's performance in the **Input Image** category could be attributed to its proclivity for injecting non existenting elements into the generated images, which might affect the consistency between the input and the output, thus leading to a less than optimal FID score. It's important to note that the observations from our Visual Analysis do align with the performance of CycleGAN in other evaluation categories.

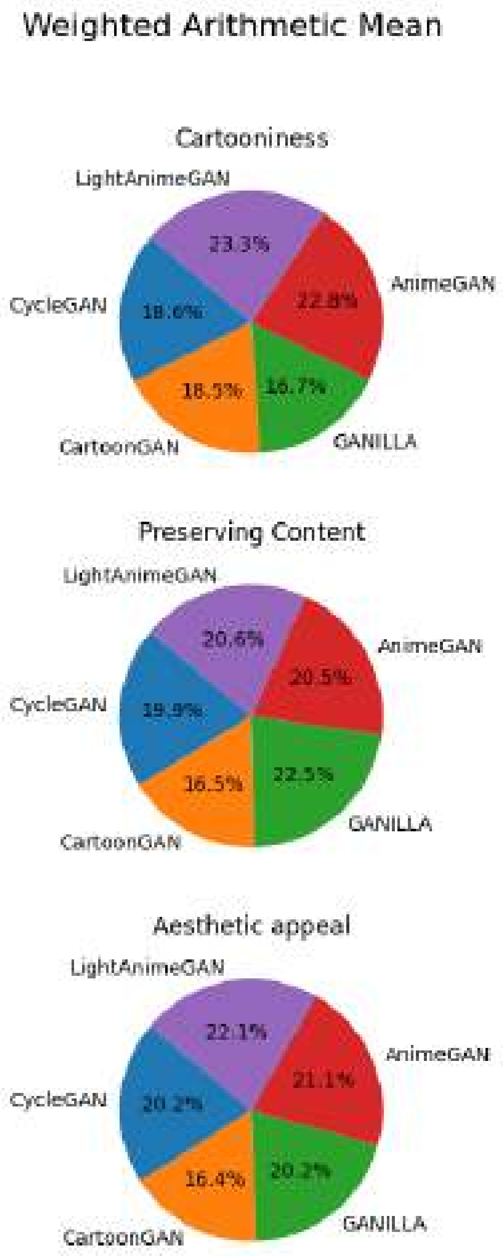
From this analysis, we can conclude that AnimeGAN and LightAnimeGAN are the two architectures achieving the best FID results in the analyzed categories. It is also worth highlighting that when considering the cumulative FID metrics, LightAnimeGAN outperforms AnimeGAN and all other models under consideration.
In Table 6.3, the performance comparison of AnimeGAN and LightAnimeGAN when trained on both Hayao and Shinkai artist styles reveals similar trends. LightAnimeGAN outperforms AnimeGAN across all categories and styles, suggesting its consistent and superior performance.
In conclusion, LightAnimeGAN performs most proficiently across all evaluated categories and styles, evidenced by the best FID scores. The consistent performance of LightAnimeGAN also suggests its effectiveness for cartoon style transfer tasks.

6.4.3 User Study

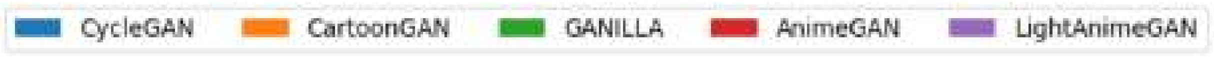
Relying solely on qualitative analysis for visual inspection can lead to subjective results. To bring more objectivity to our evaluation, we carried out a user study.
In the study, users were presented with five images from the test set, alongside their corresponding transformed cartoon versions generated by different model architectures discussed in this thesis. Utilizing Microsoft Forms, these image sets and associated questions were consolidated into a structured form, facilitating direct comparison, and enabling a straightforward evaluation by the users.
A total of 67 survey responses were collected from diverse participants to gain insights into the various model performances from a users perspective. It is worth mentioning, that the respondents of our questionnaire were not experts in the field of cartoons or generative AI models. Their choices were purely subjective, based on personal preference and perception. Thus, their responses provide interesting insights to supplement our model analysis and reduce the subjectivity of the Visual Analysis through a multitude of opinions. For a more comprehensive analysis, a stripped down version of the user study can be found in Appendix **??** of this thesis, and a full version with detailed responses can be accessed through this link.
The user study was conducted using a set of five images, with two tasks assigned for each image. In the first task, users were requested to rank the images according to their visual appeal, cartoonishness, and their capacity for content preservation. This was done using a scale of 1 to 5, where 5 indicated the most appealing image. The distribution of votes for each model is summarized in Figure 6.6i. For a more general understanding of the model's performance in each category, we calculated a weighted arithmetic mean, using the vote index as the weight. These results can be found in Figure 6.6ii. The weighted average provides a clearer picture of the obtained results, taking into account both the frequency and the value of the votes.
These results allow us to observe several insights. In terms of cartoonization, users identified GANILLA as the least effective model, consistent with other metrics demonstrating a failure to effectively transform the image to a cartoon style. CycleGAN and CartoonGAN were perceived to achieve comparable results, with LightAnimeGAN and AnimeGAN substantially outperforming them. Notably, LightAnimeGAN was rated slightly better than AnimeGAN in this category.
Unexpectedly, GANILLA emerged as the most successful model at preserving content, this observation aligns with the previously mentioned FID metric analysis and our visual analysis, where we noted that GANILLA tends to preserve content well but struggles to replicate the cartoon style. An examination of the detailed vote distribution (Figure 6.6i) reveals that many users gave GANILLA the highest score for content preservation. It's plausible that the respondents in this category concentrated on the preservation of details and textures, as opposed to the retention of the general essence of the image as anticipated by the authors. Following GANILLA in the content preservation category, we have LightAnimeGAN, AnimeGAN, and CycleGAN, all achieving fairly similar scores. As anticipated, CartoonGAN scored poorly in content preservation, presumably due to issues with random noise.
When considering an artistic appeal, all models, except CartoonGAN, received similar ratings, around 20-22%. LightAnimeGAN emerged as the most preferred, closely followed by AnimeGAN with a 1% difference (Figure 6.6ii). CycleGAN and GANILLA tied in the next position. Once again, CartoonGAN achieved the worst result securing only 16.4% of user votes.
Across all evaluated categories, LightAnimeGAN consistently emerges as the top performer based on a detailed examination of model outputs in the first part of discussed User Study.
The second task presented users with a random assortment of five images selected from the test set, asking them to identify the best overall cartoon image produced by the models across all categories. Since the



(i) User Study Votes Distribution

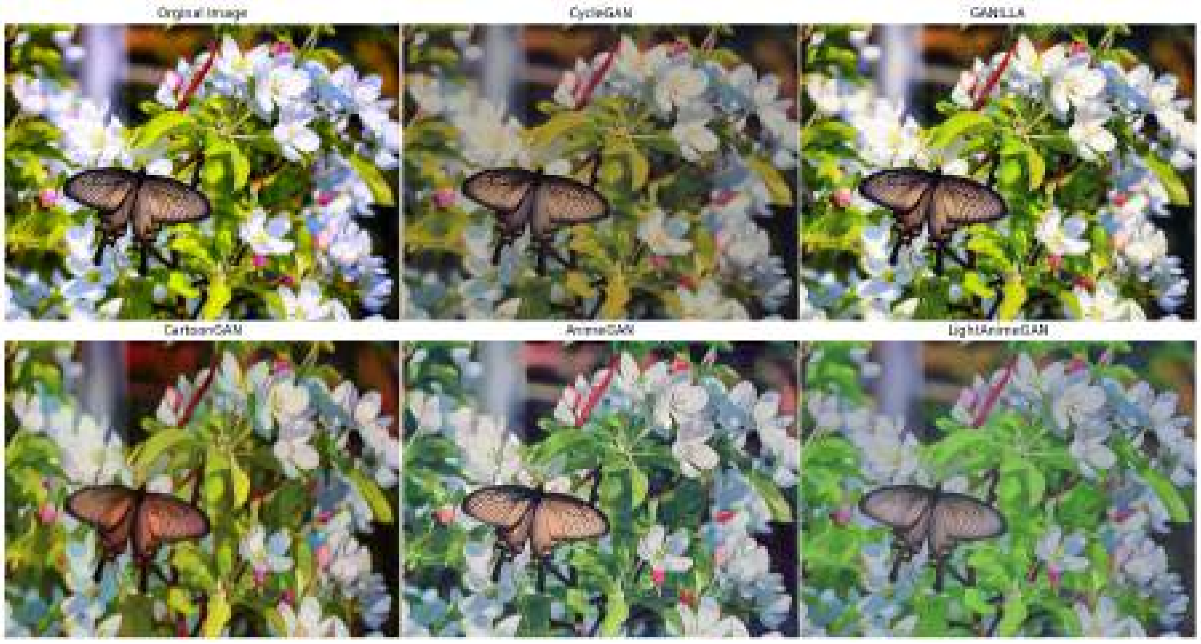


(ii) User Study Weighted Arithmetic Mean

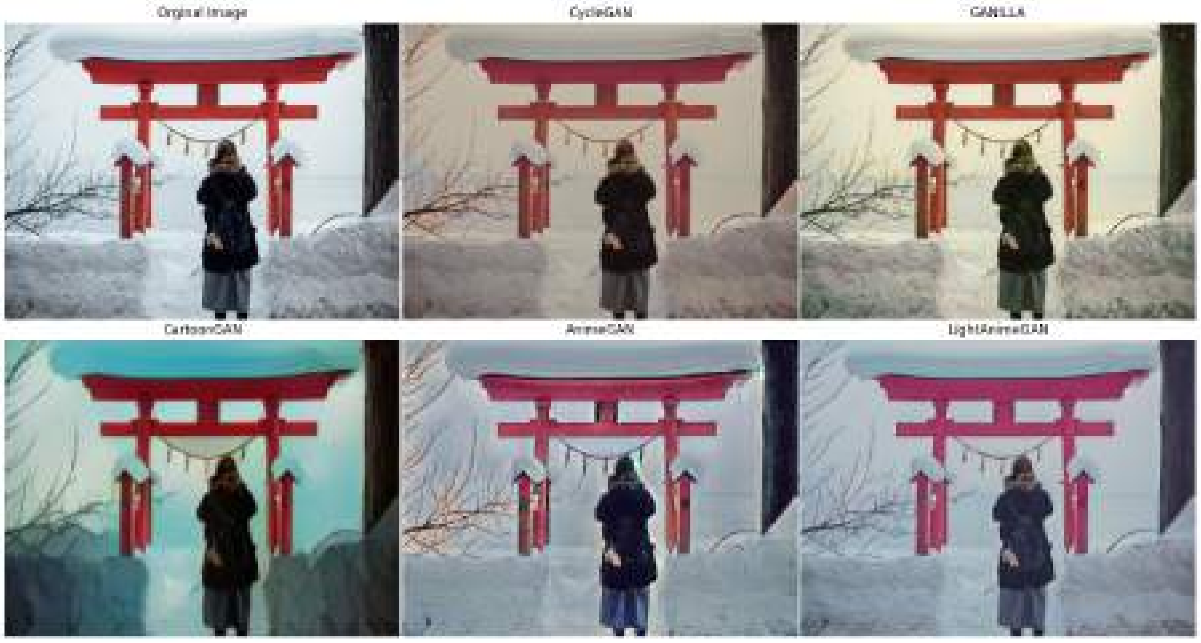


(iii) Legend

Figure 6.6: The plots demonstrate the distribution of votes and the calculated weighted arithmetic mean, where the weight is the vote index, reflecting the performance of the discussed models in categories such as visual appeal, degree of cartoonization, and preservation of content.

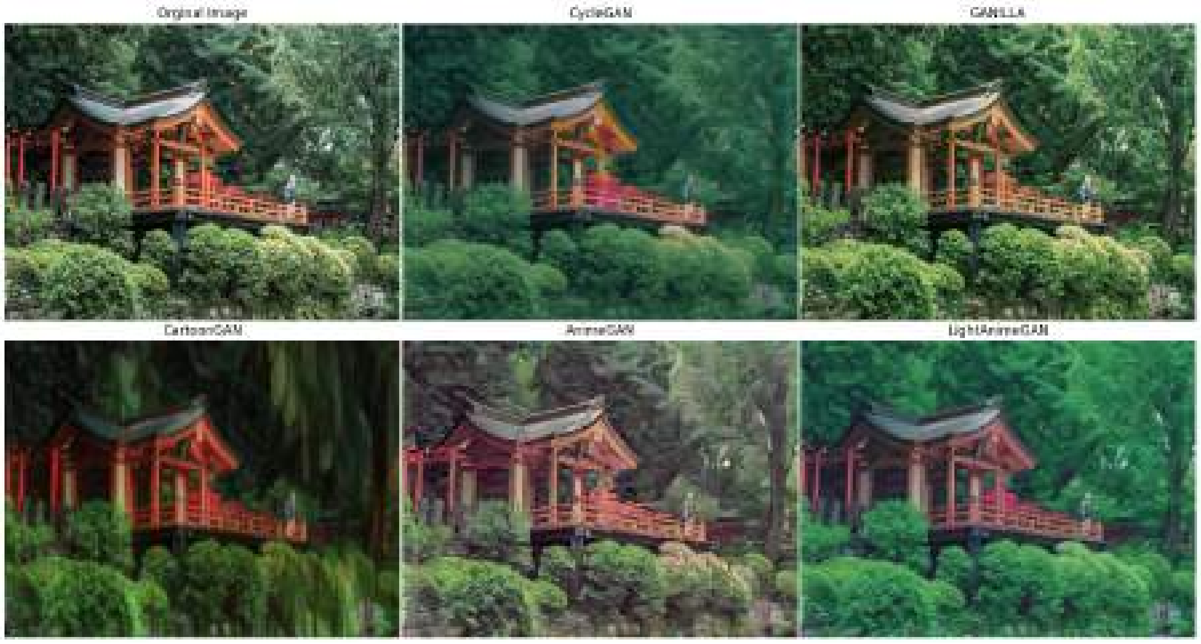


(i) For Figure 1, AnimeGAN's image received the most votes 33, followed by GANILLA with 14 votes. LightAnimeGAN got 12 votes, while CartoonGAN received 5 and CycleGAN received the least votes, 3. This suggests that, for the image in Figure 1, AnimeGAN's model was perceived to have the best translation of the original figure.

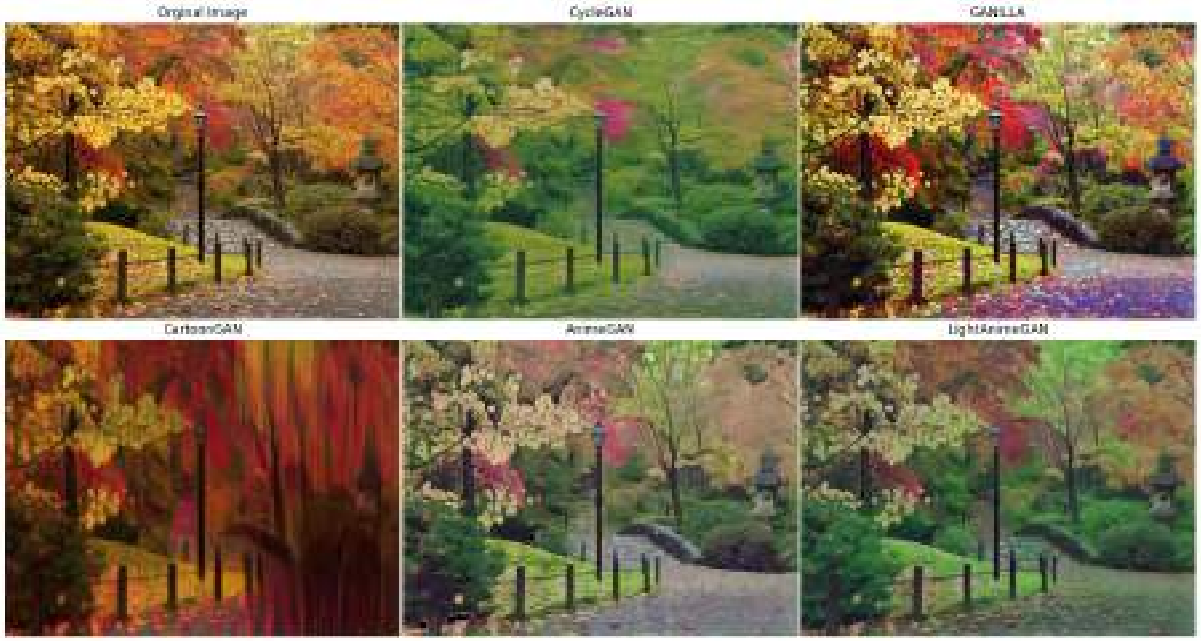


(ii) Moving to Figure 2, the distribution of votes significantly shifted. CartoonGAN emerged as the favored model, securing 22 votes, whereas LightAnimeGAN received 18 votes and GANILLA received 11 votes. The performance of AnimeGAN was not as dominating as before, only managing to get 9 votes. The CycleGAN model received the least votes for this figure, with only 7 participant preferring its output.

Figure 6.7: Part 1: Continiud figure depicting user study the visual representation of cartoon outputs generated by each model, with detailed results from the questionnaire corresponding to each input image.

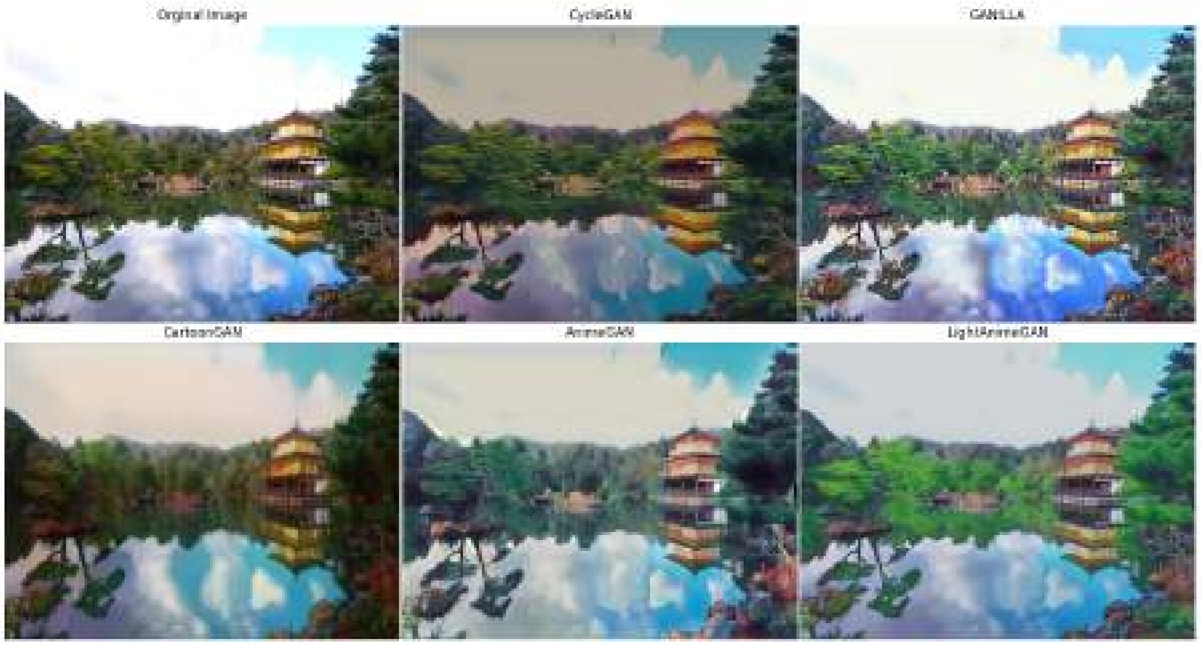


(iii) For Figure 3, LightAnimeGAN's image received the most votes 25, followed by AnimeGAN with 16 votes, and CycleGAN with 14 votes. GANILLA's image received 8 votes, and CartoonGAN received the least number of votes, 4. The results indicate that for the image in Figure 3, the majority of participants preferred the image generated by LightAnimeGAN



(iv) For Figure 4, AnimeGAN's image received the most votes 22, closely followed by LightAnimeGAN with 19 votes, and GANILLA with 16 votes. CycleGAN's image received 9 votes, while CartoonGAN received the least number of votes, 1. This suggests that for Figure 4, participants preferred the image generated by AnimeGAN, but the image generated by GANILLA was also quite popular.

Figure 6.7: Part 2: User study the visual representation of cartoon outputs generated by each model, with detailed results from the questionnaire corresponding to each input image.



(v) Finally, for Figure 5, LightAnimeGAN's image received the most votes 25, followed by AnimeGAN with 19 votes, and GANILLA with 15 votes. CycleGAN's image received 8 votes, and CartoonGAN received no votes. This result suggests that for Figure 5, participants found the image generated by LightAnimeGAN to be the best translation of the original image.

Figure 6.7: Part 3: User study the visual representation of cartoon outputs generated by each model, with detailed results from the questionnaire corresponding to each input image.

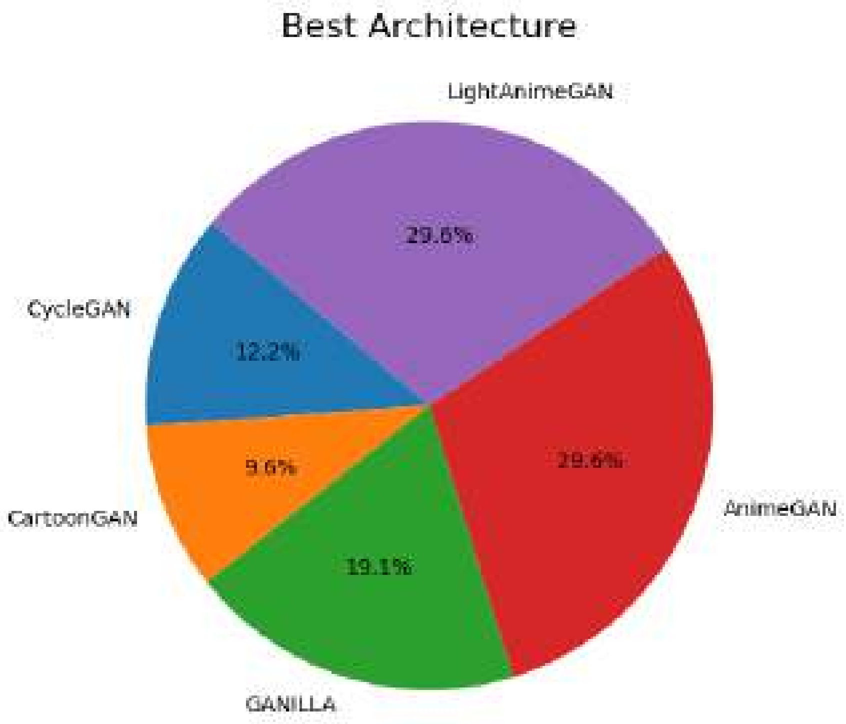


Figure 6.8: Overall distribution of user votes for the best performing model.

results can be significantly influenced by the chosen input image, Figure 6.7 offers a visual representation of the cartoon outputs generated by each model, along with detailed results from the questionnaire for each image. This broad comparison provides a more balanced evaluation, beyond our subjective analysis, highlighting nuances in results and the challenges users encountered.
In an effort to consolidate the findings obtained from this analysis and reduce the impact of the input picture on the findings, we calculated the total sum of votes received for each model. A visual representation of these summarized results is exhibited in Figure 6.8.
From these results, we can deduce that both LightAnimeGAN and AnimeGAN perform comparably well, each receiving approximately 30% of the total votes. Interestingly, GANILLA, despite its subpar performance in the cartoonization aspect during the first part of the study (Figure 6.6), was chosen as the best performing architecture by 19.1% of the users. CycleGAN managed to garner only 12.2% of the total votes, ranking it towards the lower end of preference among users. CartoonGAN had the least appeal, securing merely 9.6% of the votes, marking it as the least favored among the considered architectures.
In conclusion, based on a large sample of participants in our user study, LightAnimeGAN and AnimeGAN were the top performers, securing the same number of votes in the second task. However, LightAnimeGAN slightly outperformed AnimeGAN in the first task across all categories. This suggests that LightAnimeGAN might be the superior architecture for the cartoon style transfer task.

6.5 Summary

In this chapter, we've delved into the experimental segment of our thesis, shedding light on our training methodology, the software we employed, and the steps taken for implementing and training the discussed models. A thorough comparison of model parameters highlighted LightAnimeGAN as the standout, exhibiting the smallest generator size and the shortest mean training time per epoch. Subsequently, we conducted a comparative analysis of the results generated by these models, utilizing a mix of Visual Analysis, the FID metric, and a User Study.
According to these analysis methods, CartoonGAN consistently underperformed, receiving overall the fewest votes in the User Study. Generated images by this model were often characterized by dark colors, significant deviations from the original image, and apparent random noise, failing to convincingly retain the content and distinctive cartoonish style of the images. However, it's important to highlight that in our implementation, we used a modified version with parameter $\omega = 1.5$ due to the poor results with the original parameters from the paper. This adjustment undoubtedly influenced the results.
CycleGAN, while performing reasonably well in the author's Visual Analysis and FID metrics, only ranked fourth in the user study. This model effectively cartoonized images and preserved content, demonstrating smooth transitions and simplified texture as a result of used special identity loss. However, it sometimes had problems with empty areas and color palette, potentially leading to its lower User Study score.
Surprisingly, GANILLA secured the third position in the User Study, despite the Visual Analysis and FID metrics suggesting its main strength is in content preservation rather than in transforming real images into cartoons. The Visual Analysis revealed that the generated images were almost identical to the original images, contradicting our intent of creating a distinct cartoon style transformation. The unexpectedly high results for GANILLA in the User Study could likely be influenced by the randomly selected pictures for the questionnaire. In these pictures, the main drawbacks of the architecture, which were observed in the User Analysis (such as altering the original color palette and generating unrealistic outcomes like green buildings depicted in Figure 6.3iii), were not as visible.
Lastly, AnimeGAN and LightAnimeGAN secured the second and first places respectively in our evaluation. The slight advantage of LightAnimeGAN over AnimeGAN is due to its superior ability to transform input images into high quality cartoons while still preserving the original content and color palette. Additionally, AnimeGAN, has a tendency to generate an excessive number of individual shapes separated by wavy lines, which ultimately disrupts the structure of the image and leads to a blurry yet overly detailed output. This pattern of curvy lines detracts from the aesthetic appeal of the images. Moreover, we observed an unrealistic glow around bright objects in images generated by AnimeGAN. These problems were not present in the output from LightAnimeGAN. Based on these indicators, we consider LightAnimeGAN to be superior to AnimeGAN.

In conclusion, while all of the examined models generate satisfactory results, each applying its unique style of transformation, LightAnimeGAN consistently surpasses other models across all evaluated categories and styles, backed by the best FID scores in each category. The consistent high performance of LightAnimeGAN emphasizes its effectiveness in cartoon style transfer tasks while preserving original content. Therefore, it emerges as the recommended model for these types of applications.

7 Conclusion and Future Work

The objective of this master's thesis was to delve into the captivating realm of machine learning, particularly focusing on its unique application in transforming real-world photographs into cartoon-style images while preserving the original content. We aimed to comprehend and demonstrate how machine learning methodologies, specifically Generative Adversarial Networks, can be employed in the sphere of art and design for the automatic creation of cartoons from real images.

We began with a thorough introduction to the fundamentals required in order to understand the complex nature of GANs. This included delving into training and evaluation procedures and providing an explanation of the key principles underlying generative networks.

After a thorough literature review, we identified four state-of-the-art models exhibiting promising results in the field of image-to-image translation, namely CycleGAN, GANILLA, CartoonGAN, and AnimeGAN [54, 20, 11, 9]. Taking our study a step further, we introduced LightAnimeGAN a novel modification to the AnimeGAN model. This involved architectural changes including a reduction in the number of parameters and switching the normalization method from instance normalization to batch normalization. The goal was to investigate how well our models performed and see whether changes made to AnimeGAN may result in better performance.

For this task, we developed a specialized dataset for training our models. This dataset was an amalgamation of real-life images and corresponding cartoon versions, primarily in the Hayao and Shinkai styles. To ensure a fair and accurate comparison of the models performance, we followed the training methodologies stated in the original papers and employed our constructed dataset consistently across all models.

Our evaluation strategy consisted of a threefold approach. We evaluated the models using Visual Inspection, Fréchet Inception Distance, and User Study. Each method of analysis was meticulously executed, allowing us to dissect the results and gain a thorough understanding of the unique features and behaviors exhibited by each model.

Through our meticulous analysis, we found that AnimeGAN and proposed modified architecture, LightAnimeGAN, demonstrated superior performance compared to the other discussed models across all evaluated categories. Our observations revealed that despite their potential to generate visually captivating results and satisfactorily preserve the content, the models CartoonGAN, CycleGAN, and GANILLA were unable to reach superior performance. Each of these models faced unique challenges which affected their overall performance. These challenges included difficulties in adequately transforming the stylistic elements of the input images into the cartoon style, maintaining color consistency, and effectively handling noise in the output images. This led to a less than ideal performance in terms of both qualitative and quantitative evaluation metrics.

Our analysis aligns with the literature review, which denoted AnimeGAN as the top performing architecture in this task among currently available models. However, our proposed modification, LightAnimeGAN, presented similar or even better results across all evaluated categories in analysis. When considering the overall model characteristics, LightAnimeGAN not only demonstrated efficiency and superior results in transforming real life images into cartoon style while preserving original content but also proved to be a simpler and faster model to train. This finding is substantial, highlighting the efficiency of LightAnimeGAN in carrying out cartoon style transfer tasks while making the training process less computationally demanding. This presents a promising direction for future research in this field, encouraging further exploration into the optimization and performance of GAN architectures.

As we look to the future, several promising directions for further exploration present themselves. Our models can be tested on a variety of datasets beyond the current scope. Datasets featuring various cartoon styles, more abstract, imaginative, or even surreal artistic depictions could be utilized to evaluate the adaptability

and versatility of the models. A recommended future step would be to expand the training sample size, as a larger volume of data could further refine the training process and potentially enhance performance.
Another potentially exciting direction for future work could be the application of these models to a wider range of photographs beyond just landscapes. Portraits, detailed close-up shots, intricate scenery, and a variety of other real-life images including historical images, fantasy-themed photographs, images depicting different seasons, or photos centered around specific events or celebrations could be cartoonized using the models. The possibilities are vast, and such an approach would help in testing the models' capacity to adapt to various styles and themes, thereby making the applications of this research even more wide-ranging.
The application of these models to a broader range of pictures beyond only landscapes might be another intriguing possible path for future development. The models might be used to cartoonize close-up portraits, incredibly detailed close-up shots, and a wide range of other real life images, such as historical or seasonal photographs. The opportunities are endless, and such a method would aid in assessing the models ability to adapt to varied styles and topics, broadening the range of applications for this study.
Finally, a fascinating potential application of our models lies in the field of filmmaking. Although the task of converting a sequence of film frames into a cartoon style might seem straightforward at first glance, producing high-quality cartoon films would undoubtedly necessitate the incorporation of sophisticated film analysis techniques with our models. This intriguing convergence of machine learning, film analysis, and artistic creativity could lead to groundbreaking advancements in the realm of animation. Such developments could potentially revolutionize traditional methods of animation, enabling the rapid creation of animated sequences, and thus expanding the possibilities for storytelling in animated formats.
In conclusion, this master's thesis has contributed valuable insights into the field of automatic cartoonization using GANs, and the findings provide a strong base for further research. The work done provides a novel perspective on the intriguing intersection of machine learning and art introducung modified architecture best performing LightanimeGAN, pushing the boundaries of what can be achieved when these two fields collide.

Bibliography

[1] The wind rises, 2013. Tokyo, Japan: Studio Ghibli.

[2] Your name, 2016. Tokyo, Japan: CoMix Wave Films.

[3] Announcing the pytorch foundation: A new era for the cutting-edge ai framework. https://ai.facebook.com/blog/pytorch-foundation/, 2023. Accessed: May 8, 2023.

[4] Google cloud: Cloud computing services. https://cloud.google.com/, 2023. Accessed: May 14, 2023.

[5] Python. https://www.python.org/, 2023. Accessed: May 7, 2023.

[6] Arjovsky, M., Chintala, S., and Bottou, L. Wasserstein gan, 2017.

[7] Canny, J. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence PAMI-8*, 6 (1986), 679–698.

[8] Chen, W., Song, Z., Zhao, R., Yang, H., Wang, Q., and Zhang, C. Generative adversarial networks for cartoon face generation. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision* (2020), pp. 1704–1713.

[9] Chen, X., Zhou, Y., Wu, Y., and Wu, Q. Animegan: A novel lightweight gan for photo animation, 2020.

[10] Chen, Y., Lai, Y.-K., and Liu, Y.-J. Transforming photos to comics using convolutional neural networks. In *International Conference on Image Processing* (2017).

[11] Chen, Y., Yuan, J., Yu, J., Wang, C., and Woodford, O. Cartoongan: Generative adversarial networks for photo cartoonization. In *Proceedings of the IEEE International Conference on Computer Vision* (2018), pp. 9465–9474.

[12] Choi, Y., Choi, M., Kim, M., Ha, J.-W., Kim, S., and Choo, J. Stargan: Unified generative adversarial networks for multi-domain image-to-image translation. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (2018), pp. 8789–8797.

[13] Chollet, F. Xception: Deep learning with depthwise separable convolutions. In *Proceedings of the 30th IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2017), IEEE, pp. 1800–1807.

[14] Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. Imagenet: A large-scale hierarchical image database. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2009).

[15] Gatys, L. A., Ecker, A. S., and Bethge, M. A neural algorithm of artistic style. *arXiv preprint arXiv:1508.06576* (2015).

[16] Gatys, L. A., Ecker, A. S., and Bethge, M. Texture synthesis and the controlled generation of natural stimuli using convolutional neural networks. *arXiv preprint arXiv:1505.07376* (2015).

[17] Gatys, L. A., Ecker, A. S., and Bethge, M. Image style transfer using convolutional neural networks. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2016), 2414–2423.

[18] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. Generative adversarial nets. *Advances in neural information processing systems* (2014), 2672–2680.

[19] Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., and Hochreiter, S. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *Advances in Neural Information Processing Systems* (2017).

[20] Hicsonmez, S., Samet, N., Akbas, E., and Duygulu, P. Ganilla: Generative adversarial networks for image to illustration translation. *arXiv preprint arXiv:1902.03410* (2019).

[21] Isola, P., Zhu, J.-Y., Zhou, T., and Efros, A. A. Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (2017), pp. 1125–1134.

[22] Johnson, J., Alahi, A., and Fei-Fei, L. Perceptual losses for real-time style transfer and super-resolution. In *European Conference on Computer Vision* (2016), pp. 694–711.

[23] Jones, E., and Smith, D. The art of cartoon animation. *Journal of Animation Studies 17*, 2 (2019), 123–138.

[24] Joshi, A., Kale, S., Chandel, S., and Pal, D. Likert scale: Explored and explained. *British Journal of Applied Science & Technology 7* (01 2015), 396–403.

[25] Karras, T., Laine, S., and Aila, T. A style-based generator architecture for generative adversarial networks. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2019), 4401–4410.

[26] Karras, T., Laine, S., Aittala, M., Hellsten, J., Lehtinen, J., and Aila, T. Analyzing and improving the image quality of stylegan. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2020), 8110–8119.

[27] Kingma, D. P., and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[28] Larson, S., and Smith, R. The impact of cartoons on popular culture. *Journal of Popular Culture 45*, 3 (2020), 567–582.

[29] Ledig, C., Theis, L., Husz'ar, F., Caballero, J., Cunningham, A., Acosta, A., Aitken, A., Tejani, A., Totz, J., Wang, Z., et al. Photo-realistic single image super-resolution using a generative adversarial network. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2017).

[30] Li, C., and Wand, M. Precomputed real-time texture synthesis with markovian generative adversarial networks. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* (2016), vol. 9907 LNCS.

[31] Liu, X., Li, Z., Luo, P., Loy, C. C., and Tang, X. Learning to cartoonize using white-box cartoon representations. *IEEE Transactions on Multimedia 20*, 11 (2018), 3059–3071.

[32] McDonald, P. *The Art of Cartooning.* North Light Books, 2018.

[33] Miyato, T., Kataoka, T., Koyama, M., and Yoshida, Y. Spectral normalization for generative adversarial networks. *arXiv preprint arXiv:1802.05957* (2018).

[34] Odena, A., Dumoulin, V., and Olah, C. Deconvolution and checkerboard artifacts. *Distill* (2016).

[35] Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32* (2019), Curran Associates, Inc., p. 8024–8035.

[36] Ratliff, N., Bagnell, J. A., and Zinkevich, M. (online) subgradient methods for structured prediction. In *Artificial Intelligence and Statistics* (2013).

[37] Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision 115*, 3 (2015), 211–252.

[38] Ruthotto, L., and Haber, E. An introduction to deep generative modeling. Tech. rep., Department of Mathematics, Emory University; Department of Earth and Ocean Sciences, University of British Columbia, Atlanta, GA, USA; Vancouver, BC, Canada, April 2021.

[39] Salimans, T., Goodfellow, I., Zaremba, W., and Cheung, V. Improved techniques for training gans. In *Advances in Neural Information Processing Systems* (2016), pp. 2234–2242.

[40] Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., and Chen, L. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the 31st IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2018), IEEE, pp. 4510–4520.

[41] Schwarz, M. The art and craft of cartooning. *Visual Arts Review 12*, 4 (2020), 87–102.

[42] Simonyan, K., and Zisserman, A. Very deep convolutional networks for large-scale image recognition. *CoRR abs/1409.1556* (2014).

[43] Simonyan, K., and Zisserman, A. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556* (2015).

[44] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research 15*, 1 (2014), 1929–1958.

[45] Sunderdiek, T. Cartoongan, 2023. [Online; accessed 25-May-2023].

[46] Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., and Wojna, Z. Rethinking the inception architecture for computer vision. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2016).

[47] Taigman, Y., Polyak, A., and Wolf, L. Unsupervised cross-domain image generation. In *International Conference on Learning Representations (ICLR)* (2017).

[48] Thompson, J. The evolution of cartoons: From silent shorts to digital animation. *Animation Journal 29*, 3 (2021), 234–249.

[49] Ulyanov, D., Vedaldi, A., and Lempitsky, V. Instance normalization: The missing ingredient for fast stylization. *arXiv preprint arXiv:1607.08022* (2016).

[50] U.S. Intellectual Property Enforcement Coordinator. 2013 joint strategic plan on intellectual property enforcement. Tech. rep., Office of the U.S. Intellectual Property Enforcement Coordinator, June 2013.

[51] Wang, X., Yu, Y., Wu, X., Gu, Y., Liu, Y., Dong, C., and Loy, C. C. Towards automated cartooning via generative adversarial networks. In *Proceedings of the European Conference on Computer Vision* (2018), pp. 836–851.

[52] Williams, R. *The Animator's Survival Kit.* Faber & Faber, 2008.

[53] Yi, Z., Zhang, H., Tan, P., and Gong, M. Dualgan: Unsupervised dual learning for image-to-image translation. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)* (2017).

[54] Zhu, J.-Y., Park, T., Isola, P., and Efros, A. A. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE international conference on computer vision* (2017), pp. 2223–2232.

Appendix A Questionnaire for evaluation of generated by AI Cartoon Images

Cartoon Image Evaluation Questionnaire generated by AI

Introduction:
In this questionnaire, you will examine an original input image alongside cartoon-like images generated from this input by five different GAN-based models. Your responses will assist the author in comparing the models performance.
Instructions:
Please review the set of generated cartoon images from the original input image based on the given criteria below. Each image should be evaluated separately, on a scale of 1-5, where 1 is "very poor" and 5 is "excellent".
For each image, consider:

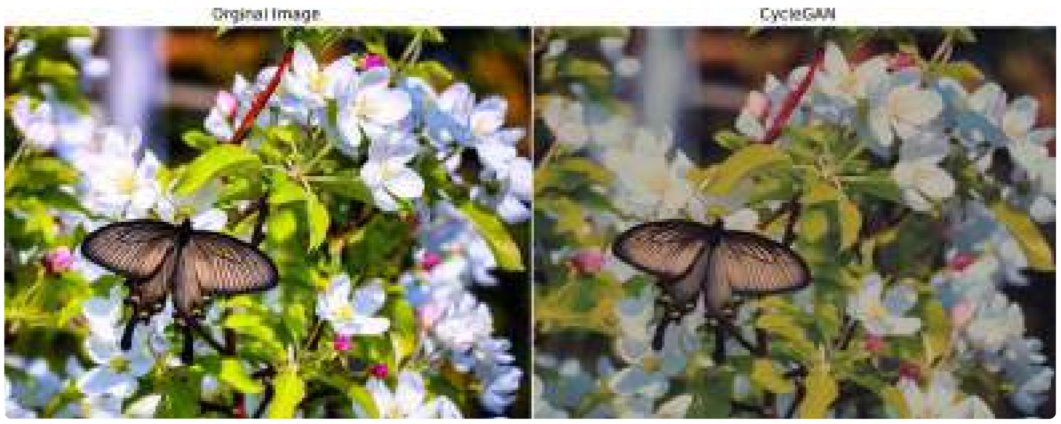
1. **Preserving Content**: Does the cartoon image accurately retain the recognizable features of the input image? Does it maintain the essence and identity of the original content?
2. **Cartooniness**: Does the image have cartoon-like characteristics such as stylization, exaggeration, simplification, and distinctive traits that define cartoons? How well does it resemble a traditional or desired cartoon style?
3. **Aesthetic appeal (Aestheticness)**: How do you rate the overall aesthetic quality and visual appeal of the cartoon image? Consider factors such as composition, color palette, line quality, balance, and other artistic elements that contribute to its attractiveness and artistic merit.

Thank you for your participation in this assessment!

Image 1

1

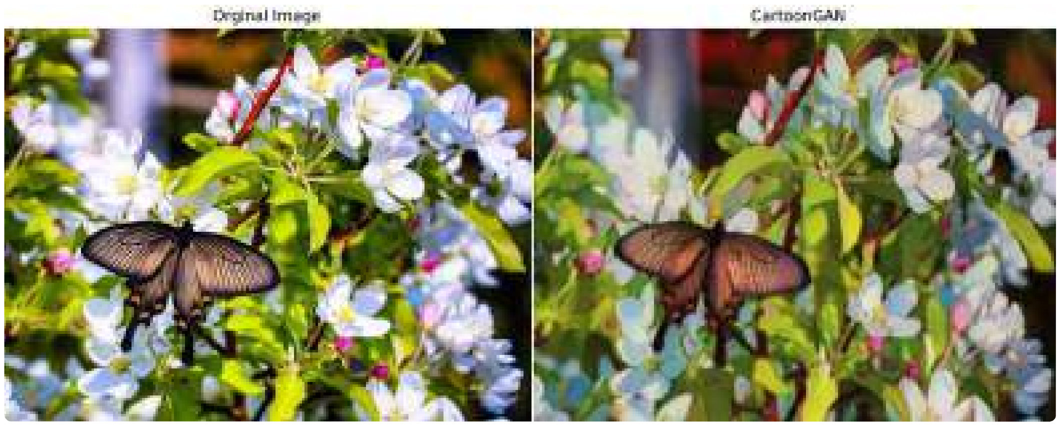
Please rate generated image by CycleGAN model on a scale of 1-5 based on the criteria below *



	1	2	3	4	5
Cartooniness	○	○	○	○	○
Preserving Content	○	○	○	○	○
Aestheticness	○	○	○	○	○

2

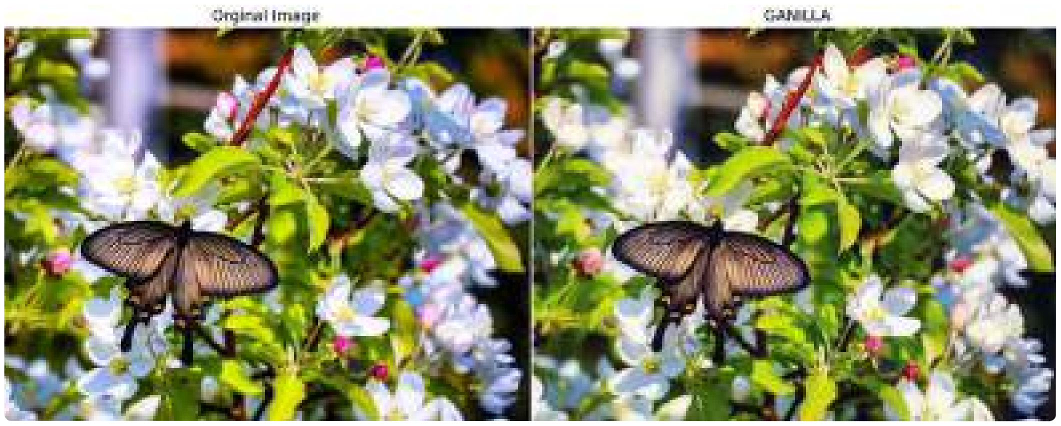
Please rate generated image by CartoonGAN model on a scale of 1-5 based on the criteria below *



	1	2	3	4	5
Cartooniness	○	○	○	○	○
Preserving Content	○	○	○	○	○
Aestheticness	○	○	○	○	○

3

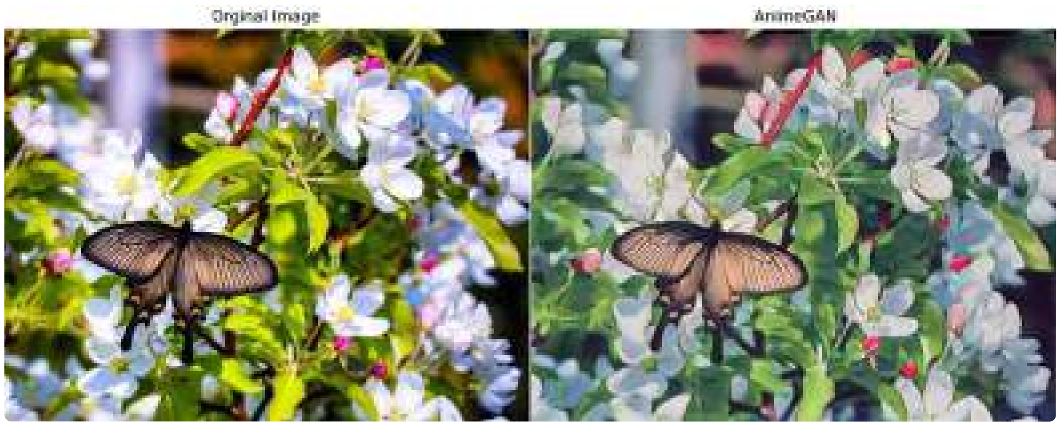
Please rate generated image by GANILLA model on a scale of 1-5 based on the criteria below *



	1	2	3	4	5
Cartooniness	○	○	○	○	○
Preserving Content	○	○	○	○	○
Aestheticness	○	○	○	○	○

4

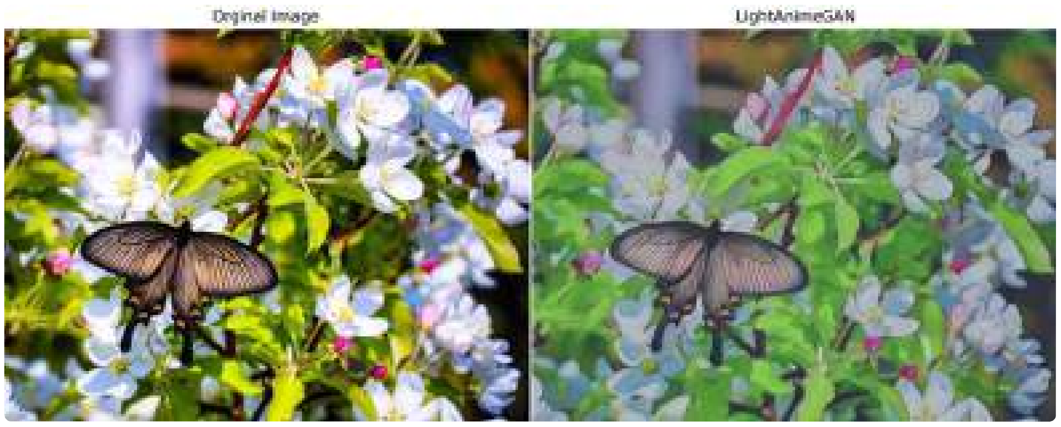
Please rate generated image by AnimeGAN model on a scale of 1-5 based on the criteria below *



	1	2	3	4	5
Cartooniness	○	○	○	○	○
Preserving Content	○	○	○	○	○
Aestheticness	○	○	○	○	○

5

Please rate generated image by LightAnimeGAN model on a scale of 1-5 based on the criteria below *

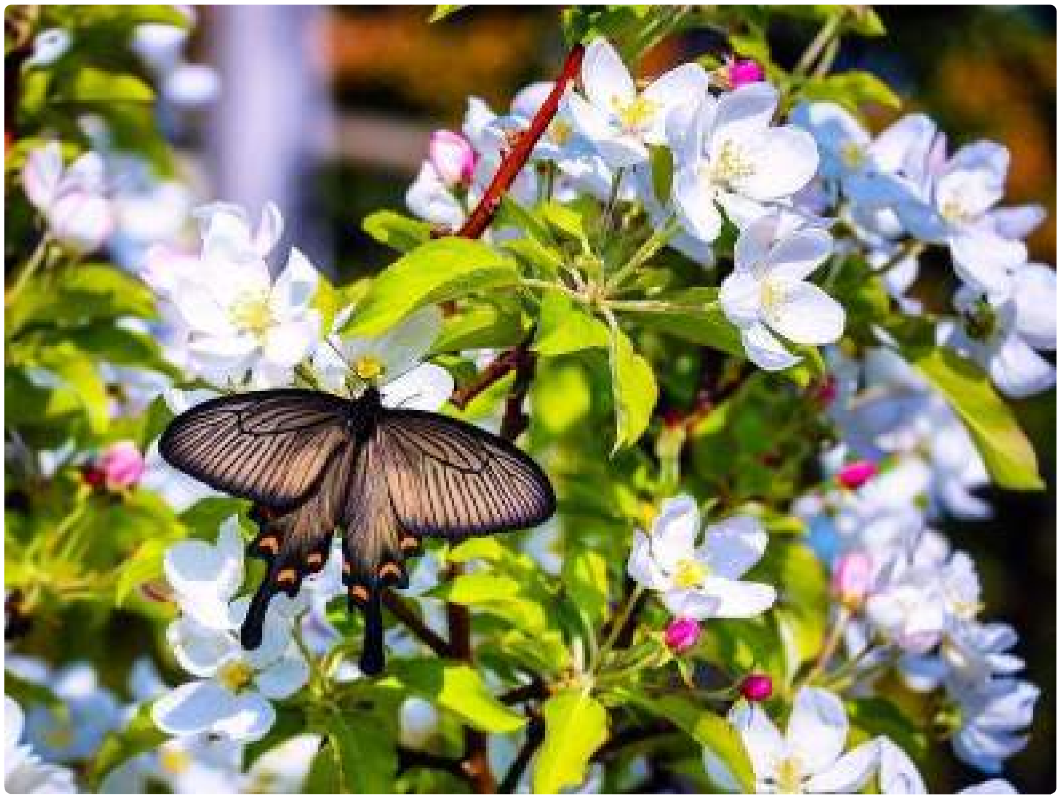


	1	2	3	4	5
Cartooniness	○	○	○	○	○
Preserving Content	○	○	○	○	○
Aestheticness	○	○	○	○	○

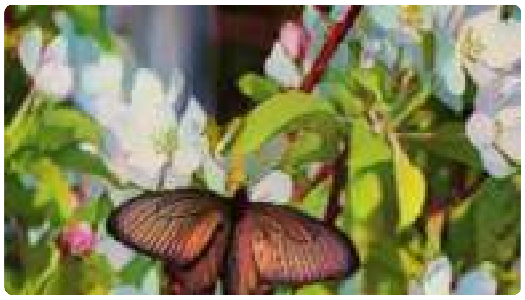
6

Please carefully examine each image and choose the one that you believe performs best in terms of successfully transforming a real image into a cartoon. *

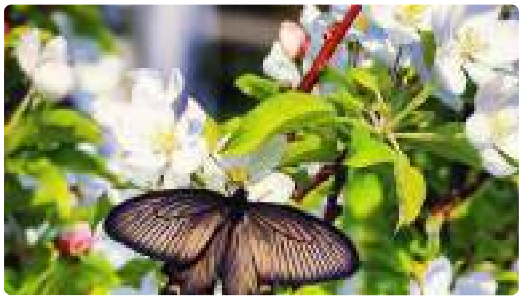
Please mark the checkbox below.



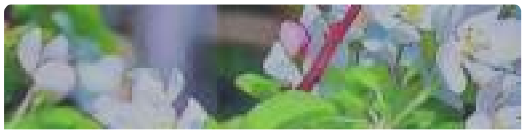
Tekst alternatywny: Orginal Image

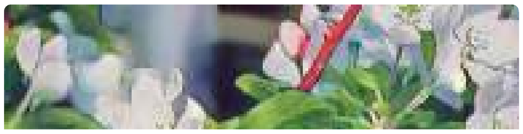


◯ CartoonGAN



◯ GANILLA





○ LightAnimeGAN

○ AnimeGAN

○ CycleGAN