Instituto Alberto Luiz Coimbra de Pós-Graduação e Pesquisa de Engenharia

COPPE UFRJ

A TWO-STAGE UNSUPERVISED GAN APPROACH FOR ANIME-TO-MANGA TRANSLATION

João Luiz Lagôas de Almeida Bertolino

Tese de Doutorado apresentada ao Programa de Pós-graduação em Engenharia de Sistemas e Computação, COPPE, da Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Doutor em Engenharia de Sistemas e Computação.

Orientadores: Carlos Eduardo Pedreira
Pedro Vieira Sander

Rio de Janeiro
Setembro de 2025

A TWO-STAGE UNSUPERVISED GAN APPROACH FOR ANIME-TO-MANGA TRANSLATION

João Luiz Lagôas de Almeida Bertolino

TESE SUBMETIDA AO CORPO DOCENTE DO INSTITUTO ALBERTO LUIZ COIMBRA DE PÓS-GRADUAÇÃO E PESQUISA DE ENGENHARIA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE DOUTOR EM CIÊNCIAS EM ENGENHARIA DE SISTEMAS E COMPUTAÇÃO.

Orientadores: Carlos Eduardo Pedreira
Pedro Vieira Sander

RIO DE JANEIRO, RJ – BRASIL
SETEMBRO DE 2025

Lagôas de Almeida Bertolino, João Luiz

A Two-Stage Unsupervised GAN Approach for Anime-to-Manga Translation/João Luiz Lagôas de Almeida Bertolino. – Rio de Janeiro: UFRJ/COPPE, 2025.

XV, 88 p.: il.; $29,7$cm.

Orientadores: Carlos Eduardo Pedreira
Pedro Vieira Sander

Tese (doutorado) – UFRJ/COPPE/Programa de Engenharia de Sistemas e Computação, 2025.

Referências Bibliográficas: p. 78 – 88.

1. image-to-image translation. 2. computer vision. 3. deep learning. I. Pedreira, Carlos Eduardo *et al.* II. Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia de Sistemas e Computação. III. Título.

À minha mãe, Cláudia, e à minha dinda, Lúcia, que foram as primeiras professoras da minha vida e me ensinaram tanto. Que eu possa seguir inspirando alunos e estudantes assim como elas me inspiraram. Essa conquista é nossa.

Agradecimentos

Trabalhar como Analista de Sistemas, atuar como Professor e ainda escrever uma tese tão técnica foi um grande desafio. Este trabalho é fruto de muita dedicação e paixão pela academia. Deixo aqui os meus agradecimentos a todos que, de alguma forma, estiveram envolvidos nesse processo e fizeram parte desta trajetória.

Agradeço primeiramente aos meus orientadores, Carlos Eduardo Pedreira e Pedro Sander, por todas as conversas, formais e informais, pelos direcionamentos ao longo dos anos, pela parceria e, especialmente, pela confiança e apoio incondicional nos momentos mais difíceis. Agradeço por terem topado misturar um pouquinho de suas áreas de atuação — Learning e Graphics, respectivamente —, de modo a viabilizar o desenvolvimento deste trabalho. Muito obrigado por tudo. Agradeço também à professora Jing Liao, por todas as dicas e feedbacks técnicos sobre o trabalho, sendo sempre objetiva, direta e contribuindo de forma significativa para seu desenvolvimento.

Gostaria de agradecer também aos amigos do Laboratório LC3, que sempre tornaram esse ambiente de troca acolhedor e leve, onde nunca faltou café, assim como os amigos do Laboratório LaSI.

Agradeço ao João Paixão (orientador de Mestrado) e à Luziane Ferreira (orientadora de Graduação), meus pais acadêmicos, que me inspiraram com entusiasmo e paixão pela profissão docente.

Aos professores e amigos do Colégio Pedro II — Flavio Costa, Cláudio Passos e Bianca Albuquerque — pelo apoio de sempre. Ao Pedro II, tudo!

À Natália Vidigal, minha grande companheira, agradeço pelo apoio constante, pela paciência e pelas inúmeras renúncias feitas ao longo deste percurso.

Agradeço à minha família: Alexandre, Lulu e André, que, mesmo nunca entendendo "o tanto que eu fazia na UFRJ", sempre me apoiaram.

Agradeço, por fim, por ter tido o prazer de ser formado pela Universidade Federal do Rio de Janeiro.

Resumo da Tese apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Doutor em Ciências (D.Sc.)

A TWO-STAGE UNSUPERVISED GAN APPROACH FOR ANIME-TO-MANGA TRANSLATION

João Luiz Lagôas de Almeida Bertolino

Setembro/2025

Orientadores: Carlos Eduardo Pedreira
Pedro Vieira Sander

Programa: Engenharia de Sistemas e Computação

Essa tese tem como objetivo investigar como imagens de desenhos animados e quadrinhos podem ser melhor exploradas no contexto de conversão de seus estilos artísticos. Especificamente, buscamos investigar a tradução de ilustrações de anime para suas representações em mangá, tendo como referência um livro de mangá. Embora os modelos atuais de tradução de imagem para imagem do estado da arte sejam capazes de converter imagens entre diferentes domínios, os métodos existentes para traduzir ilustrações para este estilo ainda são escassos. Propomos explorar as características únicas das imagens de anime e mangá, permitindo uma saída preliminar que pode auxiliar no processo de tradução em duas etapas. Acreditamos que essa abordagem reduz a complexidade do modelo ao mesmo tempo em que gera saídas de alta qualidade. Além disso, buscamos impor restrições mínimas ao domínio de destino, tornando a tradução não supervisionada. Por fim, a saída do framework proposto pode ser utilizada para produzir conjuntos de dados ricos compostos por imagens coloridas e imagens sintéticas de mangá, o que apoiaria métodos de colorização que dependem de grandes quantidades de dados pareados para treinamento.

Abstract of Thesis presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Doctor of Science (D.Sc.)

A TWO-STAGE UNSUPERVISED GAN APPROACH FOR ANIME-TO-MANGA TRANSLATION

João Luiz Lagôas de Almeida Bertolino

September/2025

Advisors: Carlos Eduardo Pedreira
Pedro Vieira Sander

Department: Systems Engineering and Computer Science

This thesis aims to investigate how cartoon and comic images can be better explored in the context of converting their artistic styles. Specifically, we seek to study the translation of anime illustrations into their manga representations, given a manga book as a reference. Although state-of-the-art image-to-image translation models can convert images between different domains, methods for translating illustrations to this style are scarce. We propose to exploit the unique characteristics of anime and manga images, to produce a preliminary output, which supports a two-stage translation process. We believe this approach can reduce model complexity while generating high-fidelity outputs. Furthermore, we aim to impose minimal restrictions on the target domain, making the translation unsupervised. Finally, the proposed framework's output can be used to produce rich datasets composed of colored and synthetic manga images, which would support colorization methods that rely on large amounts of paired training data.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Manga is the main comic book style in Japan and has gained widespread popularity across the globe. Unlike Western comics, manga is typically published in black and white, with color reserved mostly for the cover. The production process is fast-paced, resulting in a continuous flow of content. Like most comics, a manga title is a set of pages composed of several panels separated by white lines. Those panels are filled with monochromatic illustrations and dialogue balloons that, as a whole, tell a specific story. Observe Figure 1.1 for an example showing the cover and selected pages from the Dragon Ball manga series [1].

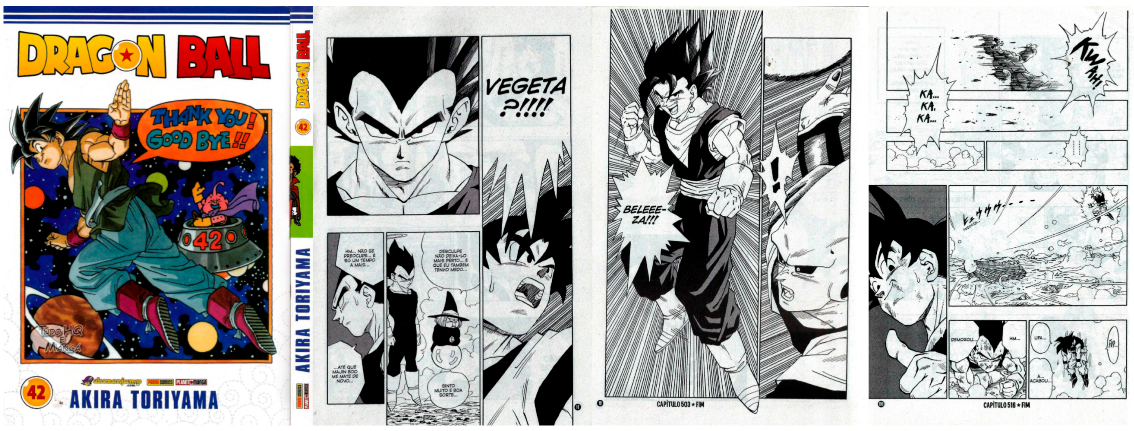


Figure 1.1: Cover and sample pages from the Dragon Ball manga series [1].

Some manga books are adapted into Japanese animations, known as anime, often with the collaboration of the original author. In such cases, specific manga panels are modified to fit the animation format, treated as key frames, and then colorized to appeal to a wider market. As a full-length feature episode or film, popular anime franchises are composed of colored frames (illustrations) where characters and backgrounds from the adapted manga story are present. Figure 1.2 shows two manga panels from the Dragon Ball series [1] that have been adapted into corresponding key frames in the colored anime version Dragon Ball Z [2].

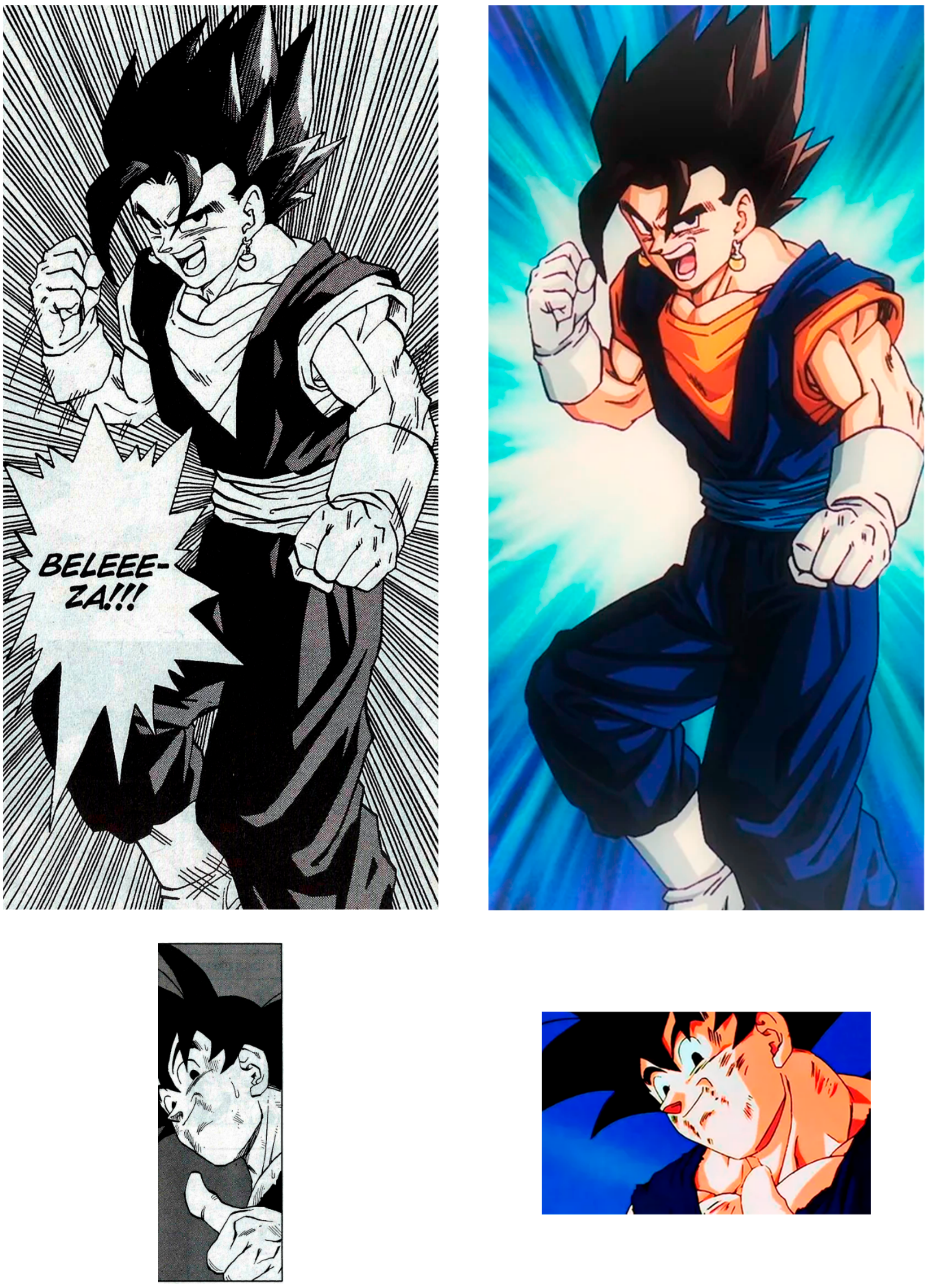


Figure 1.2: Two manga panels (left) and their corresponding key frames in the colored anime adaptation (right).

Despite some similarities, it is important to note that manga and anime are not identical kinds of artwork. A manga image is not simply a monochromatic illustration; it is a black-and-white image conforming to a specific art style developed by its original artist. In this context, screen patterns — bitonal textures applied to the drawings — are an important artistic choice that contributes to the identity and recognizable style of manga titles. In Figure 1.3, one can distinguish between an anime illustration, its monochromatic format, and its corresponding manga representation. Observe that a screen pattern was applied in the background by the manga artist.

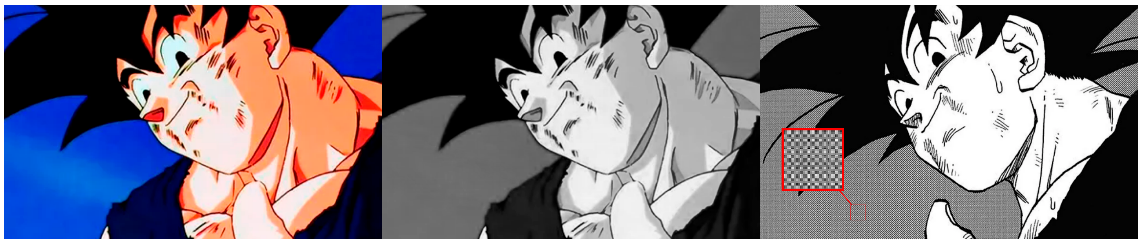


Figure 1.3: An example of an anime illustration (left), its monochromatic format (middle), and its corresponding manga representation (right). The screen pattern applied by the manga artist is highlighted with a red square. Zoom in to check the screen pattern.

Knowing that there is a strong demand for colorization techniques for manga books, it is worth considering the use of their adapted animations to create image pairs consisting of colored frames and their corresponding manga versions. These pairs can support semi-automatic colorization methods that rely on paired data [24–28].

Additionally, considering that the process of creating manga is labor-intensive and time-consuming, an automated and effective method for generating manga images from anime-style illustrations could offer practical benefits for the manga industry, since many publishers have started adopting tools that support the conversion of generic digital illustrations into manga-style outputs [29, 30].

1.1 Objective

The main goal of this thesis is to develop a method capable of converting anime images into high-quality manga illustrations, taking into account the diverse screen patterns used in the target manga style. By achieving this, we believe that it can serve two purposes: it enables the synthetic generation of paired datasets to support supervised colorization techniques, and it contributes to the automation of manga production workflows by transforming digital illustrations into stylized manga outputs.

Given an anime image or a digital illustration, our method is capable of producing a manga-style version. Figure 1.4 presents an example based on a frame extracted from the anime Dragon Ball Z series [2], while Figure 1.5 showcases additional input-output pairs generated from digital illustrations available in the Danbooru dataset [3] — a large-scale, publicly available collection of anime images widely used in the research community as a benchmark for data-driven approaches. We invite the reader to explore additional results presented in Chapter 6, particularly in Section 6.3.

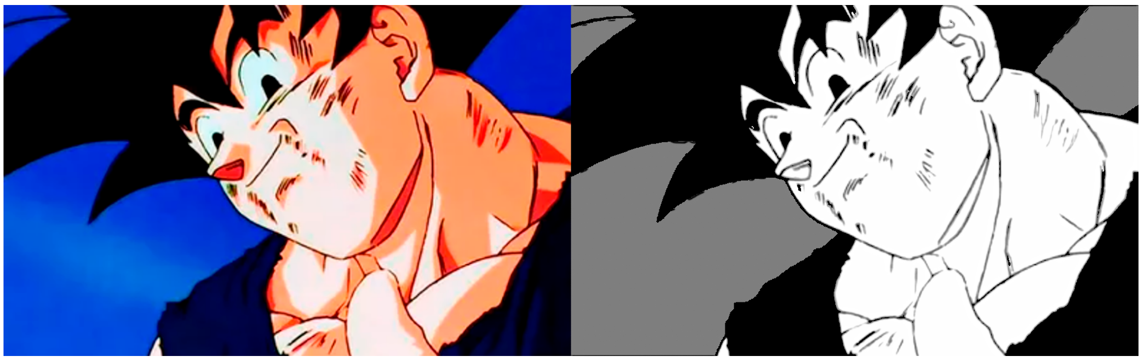


Figure 1.4: An anime frame extracted from the Dragon Ball Z series [2] and its corresponding manga-style version generated by our method. Zoom in to see the applied screen patterns.

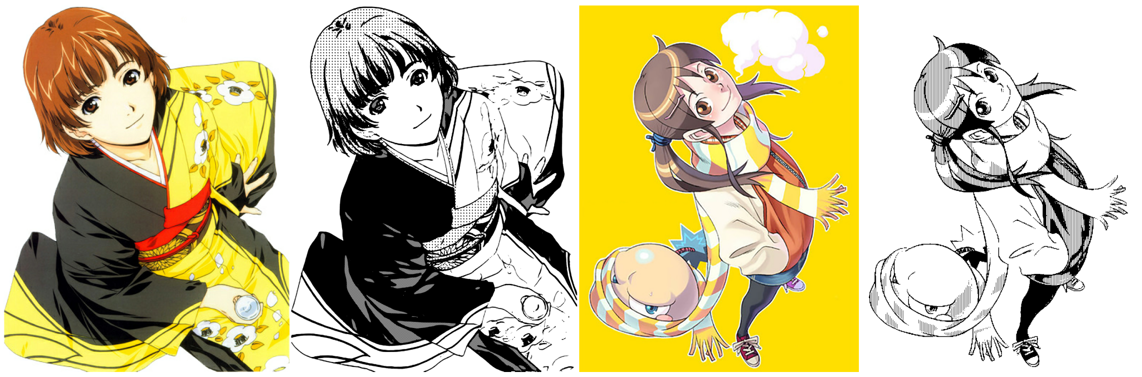


Figure 1.5: Two pairs of anime illustrations extracted from the Danbooru dataset [3] and their corresponding synthetic manga versions generated by our method. Zoom in to see the high-quality screen patterns.

1.2 Contributions

In this work, we propose a two-stage anime-to-manga translation framework that utilizes any manga book title as a target domain while preserving its screen pattern distribution and diversity. We show that breaking the anime-to-manga translation process into two stages is sufficient to generate high-quality results, without the need for complex generative models.

The first stage focuses on translating between domains while preserving the screen patterns of the target manga and the semantic content of the input anime image. This is achieved using a Generative Adversarial Network (GAN) framework [31] with three custom loss functions, a tailored training procedure to avoid local minima, and a custom discriminator architecture. The output from this stage serves as a reference for the second stage, where it is refined into a high-quality manga representation. In this stage, a classifier recommends screen patterns based on the target dataset, filling the regions where the author would likely apply a screen pattern. Finally, the structural lines of the image are overlaid, creating the final result. By imposing minimal restrictions on the screen patterns of the target dataset, our method enables the creation of results with diverse screen patterns, unlike current methods that are limited to a single pattern style. Our contributions include:

1. A two-stage unsupervised data-driven framework to generate manga representations from anime illustrations, with few restrictions on the screen patterns used in the target domain.
2. A large dataset of image pairs (anime illustrations and corresponding synthetic manga representations) that can support supervised learning approaches and other data-driven techniques that depend on paired samples for training.
3. A custom discriminator architecture to mitigate issues related to mode collapse.
4. A novel binary content loss function that helps reduce the gap between anime and manga content representation.
5. A patch-wise loss function designed to preserve the screen pattern distribution of the target dataset.
6. A customized weight initialization approach to stabilize training.
7. An user-study of the proposed method against all major image-to-manga translation methods.

1.3 Document Organization

This thesis is organized as follows:

- Chapter 2 introduces and contextualizes the fundamental concepts in computer vision and deep learning that will be used in this thesis. Readers already familiar with these topics may skip this chapter without loss of continuity.
- Chapter 3 presents a review of the literature on image-to-image translation, with focus on artistic and manga-oriented transformations.
- Chapter 4 details the method; the proposed two-stage framework for anime-to-manga translation.

- Chapter 5 describes the experiments conducted to evaluate the method. It also presents the datasets and implementation details.
- Chapter 6 discusses and presents the main results, including both quantitative and qualitative analyses. It also outlines some limitations.
- Chapter 7 concludes the thesis and proposes directions for future work.

Chapter 2

Background

This chapter presents the fundamental concepts and techniques that support the development of the methods proposed in this thesis. The topics covered include core ideas such as deep learning for computer vision, convolutional nerual networks, different architectures and its marks, training, as well as generative adversarial models. Readers who are already familiar with these concepts may choose to skip this chapter without compromising the continuity of the thesis.

2.1 Deep Learning for Computer Vision

Deep learning has fundamentally reshaped the field of computer vision by enabling the automatic extraction of hierarchical features from raw pixel data. Unlike traditional approaches that rely on hand-crafted features and separate classification pipelines, deep learning models—particularly convolutional neural networks (CNNs)—have proven capable of learning both representations and decision functions in an end-to-end manner.

However, this was not always the case. The automatic learning of visual features is the result of decades of research and incremental advances. The next subsection revisits this historical trajectory and outlines how feature extraction evolved until convolutional neural networks, the main component of this thesis, became the dominant paradigm in computer vision.

2.1.1 Motivations and Historical Context

The study of computer vision has evolved over several decades, building upon fundamental discoveries in both neuroscience and computer science. This section presents a brief overview of key milestones that have shaped the development of visual recognition systems.

A pivotal moment occurred in 1959 with the work of Hubel and Wiesel, who studied the visual cortex of cats. Using implanted electrodes, they discovered that specific neurons in the brain respond to oriented edges and simple visual patterns [32]. Their findings suggested that vision is processed hierarchically, beginning with basic features such as edges and progressively building toward more complex representations. This hierarchical processing model would later influence the design of artificial vision systems.

In 1963, Larry Roberts completed one of the first Ph.D. thesis in computer vision at MIT [33]. Inspired by Hubel and Wiesel's discoveries, Roberts developed methods for detecting edges and feature points from digital images. Despite the hardware limitations of the time, his work demonstrated the potential of using computational techniques for visual understanding.

In the 1970s, David Marr proposed that visual understanding happens in stages, starting from raw images and progressing through edge detection and depth estimation, eventually leading to full scene recognition [34, 35]. During the late 1970s and early 1980s, Brooks and Binford [36, 37] introduced the idea of representing objects using simple 3D shapes, like cylinders and cones, combined to form more complex structures.

Advances in hardware during the 1980s enabled researchers to process more realistic images. Edge detection became a dominant theme, with John Canny proposing a robust edge detection algorithm in 1986 [19]. Following this, David Lowe introduced methods for object recognition via edge matching, allowing systems to recognize objects by comparing detected edges with known templates [38, 39].

As research progressed into the 1990s, efforts increasingly focused on grouping and segmentation as essential steps for visual understanding. One influential contribution was the normalized cuts framework proposed by Shi and Malik [40]. Instead of treating segmentation as a purely local decision based on adjacent pixel similarities, they formulated the problem globally as a graph partitioning task. Normalized cuts advanced the field by demonstrating that effective grouping could facilitate recognition tasks in complex visual scenes.

The late 1990s and early 2000s saw the rise of local feature matching approaches. Notably, David Lowe proposed the Scale-Invariant Feature Transform (SIFT) [41], which introduced invariant descriptors for keypoints, allowing matching across changes in scale, rotation, and illumination.

Around the same period, Viola and Jones presented a breakthrough in face detection [42]. Their system combined simple visual features with a boosting-based learning algorithm, achieving fast and accurate face detection that was rapidly adopted in commercial applications.

With the widespread availability of the internet and digital cameras, the 2000s saw the creation of large labeled datasets, such as the PASCAL[1] Visual Object Challenge. Researchers leveraged machine learning techniques to improve recognition systems based on these datasets. But the major milestone came with the release of the ImageNet dataset [43], comprising over one million labeled images across several categories. This dataset enabled a new wave of advances in computer vision by providing the scale needed to train large neural networks. In the 2012 ImageNet Large Scale Visual Recognition Challenge (ILSVRC), AlexNet [9] achieved a substantial reduction in classification error compared to previous methods. The model used in this competition was a deep convolutional neural network (CNN), building upon earlier ideas introduced by LECUN *et al.* [8]. CNNs had already demonstrated success in tasks such as handwritten digit recognition but had not yet scaled to large datasets due to computational limitations.

From this point forward, deep learning became mainstream in computer vision. CNNs, in particular, emerged as the standard model for most visual recognition tasks, powering advances in classification, detection, segmentation, and beyond.

2.1.2 Core Building Blocks and Concepts of Neural Networks

A neural network is composed of layers that transform input data into more abstract representations. The basic components include: transformation layers, which compute weighted sums, non-linear activations, introducing non-linearity, loss functions, which measure the quality of predictions and optimizers, which adjust weights using gradients [44].

These elements are composed into deeper architectures, forming networks capable of learning complex mappings from input to output. The depth of a network allows it to learn increasingly abstract representations. For example, in image classification, lower layers may detect edges and textures, while deeper layers identify parts of objects and eventually full object categories.

Convolutional Neural Networks (CNNs) [8] are a class of deep neural networks that have proven to be particularly effective in tasks involving spatially structured data, such as images. Their architecture is inspired by the visual processing system of the human brain, especially the visual cortex, where neurons are arranged to respond to overlapping regions of the visual field.

[1]PASCAL stands for Pattern Analysis, Statistical Modelling, and Computational Learning. It was an EU Network of Excellence funded under the IST Programme of the European Union.

2.2 Convolutional Neural Networks

Unlike traditional fully connected networks, CNNs make two key assumptions about the input data: spatial locality and translational invariance. These assumptions allow CNNs to share weights across space and reduce the number of parameters drastically. The main building blocks of a CNN are convolutional layers, activation functions, pooling layers, and fully connected layers.

2.2.1 Convolutional Layers

A convolutional layer applies a set of learnable filters (also called kernels) to the input image. Each filter slides over the spatial dimensions (height and width), computing a dot product between the filter weights and the input sub-region. This process produces a feature map that encodes the presence of certain features in specific locations of the image.

The operation can be expressed as:

$$y_{i,j}^{(k)} = (x * w^{(k)})_{i,j} + b^{(k)}$$

where x is the input, $w^{(k)}$ is the k-th filter, $b^{(k)}$ is the bias term, and $*$ denotes the convolution operation. The indices i, j indicate the spatial location, and k indexes the output feature maps. Observe Figure 2.1.

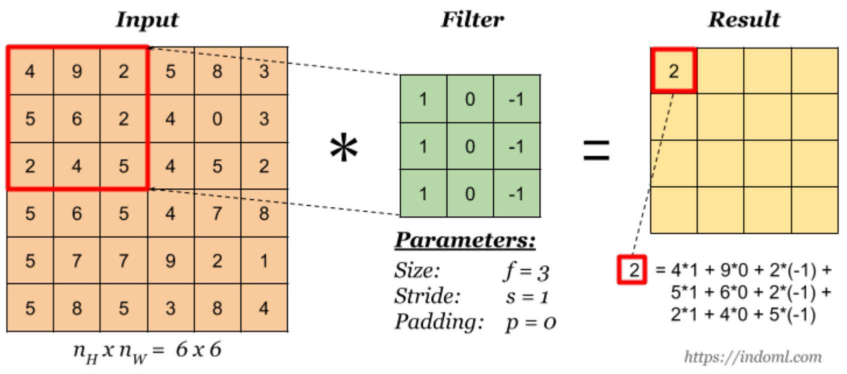


Figure 2.1: Illustration of a convolution operation applied to an input image using a learnable filter. Adapted from [4].

2.2.2 Activation Functions

After each convolutional operation, a nonlinear activation function is applied to introduce the nonlinearity required for learning complex mappings. Without this function, a network composed of multiple layers would collapse into a single linear transformation.

In the early stages of neural network research, the logistic sigmoid and hyperbolic tangent (tanh) functions were widely adopted. These functions squash the input into fixed intervals: $[0, 1]$ for sigmoid and $[-1, 1]$ for tanh; making them suitable for normalized outputs. However, both suffer from the vanishing gradient problem (when optimizing the network parameters), especially in deeper architectures, which slows down learning.

To overcome these limitations, the Rectified Linear Unit (ReLU) [45] emerged as a more effective alternative. Defined as

$$\mathrm{ReLU}(x) = \max(0, x),$$

it introduces sparsity and avoids gradient saturation for positive values, leading to faster and more stable training. ReLU gained popularity following its use in AlexNet [9], where it contributed to a significant leap in performance.

Several variants of ReLU have since been proposed to address its limitations, particularly the issue of zero gradients for negative inputs. One example is the Leaky ReLU [46], which allows a small, non-zero slope for negative values.

Figure 2.2 illustrates the behavior of these commonly used activation functions.

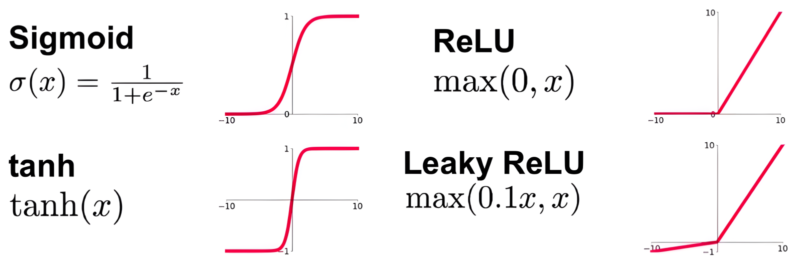


Figure 2.2: Visualization of commonly used activation functions. The Sigmoid function squashes inputs into the $(0, 1)$ range, tanh centers outputs around zero within $(-1, 1)$, ReLU allows positive activations while zeroing out negatives, and Leaky ReLU introduces a small slope for negative inputs. Image adapted from [5].

2.2.3 Pooling Layers

Pooling layers reduce the spatial dimensions of feature maps by summarizing local regions, producing more compact representations. This operation helps the network become more robust to small translations, distortions, or noise in the input. By reducing the number of activations, pooling also decreases the risk of overfitting and lowers the computational cost of deeper layers.

The most common pooling operation is max pooling, which selects the maximum value within a local region of the feature map. Alternatively, average pooling computes the mean value over the region, providing a smoother aggregation of features.

Observe Figure 2.3.

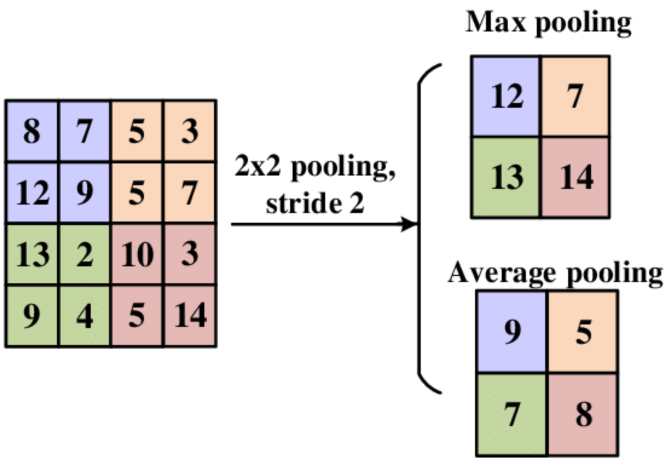


Figure 2.3: Example of max pooling and average pooling with a 2 × 2 window and stride 2. Image adapted from [6].

While max pooling has been widely adopted for its ability to preserve the most salient features, some works have explored variations such as stochastic pooling [47], where elements are selected randomly according to their activation strengths, and global average pooling [48], which aggregates features across entire spatial dimensions and is often used at the end of convolutional networks for classification tasks.

2.2.4 Fully Connected Layers

CNNs typically consist of multiple convolutional and pooling layers stacked together, forming a deep hierarchy. Features are progressively learned at different levels of abstraction. Lower layers capture simple patterns such as edges and corners, while deeper layers encode more complex structures, including object parts and entire objects [49]. The final layers are usually fully connected and are responsible for performing the classification task. Observe Figure 2.4.

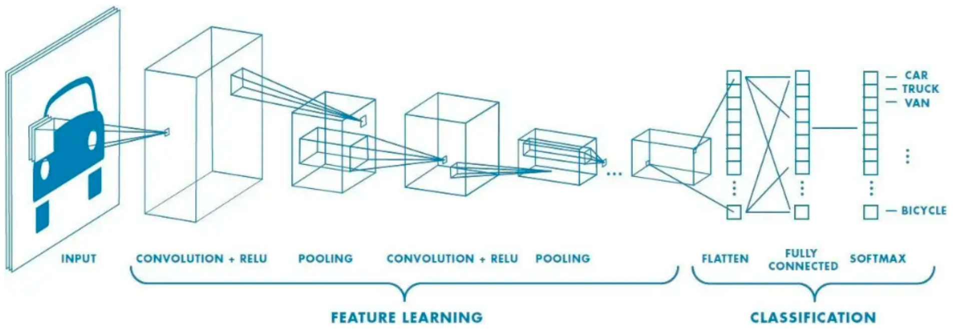


Figure 2.4: Illustration of feature learning in a convolutional neural network. Lower layers extract simple features such as edges, while deeper layers combine them into increasingly abstract representations used for classification. Source: [7].

2.2.5 CNN Architectures

This section reviews representative CNN architectures that have contributed to the development of deep learning in computer vision. Each model introduced new techniques or design choices that helped overcome limitations of previous approaches.

LeNet

LeNet-5 was proposed by LECUN *et al.* [8] for digit classification on the MNIST dataset. It introduced a simple architecture with alternating convolutional and pooling layers, followed by fully connected layers. Despite its simplicity, LeNet demonstrated that end-to-end learning with backpropagation could achieve competitive results in visual recognition tasks. Figure 2.5 shows the LeNet-5 architecture.

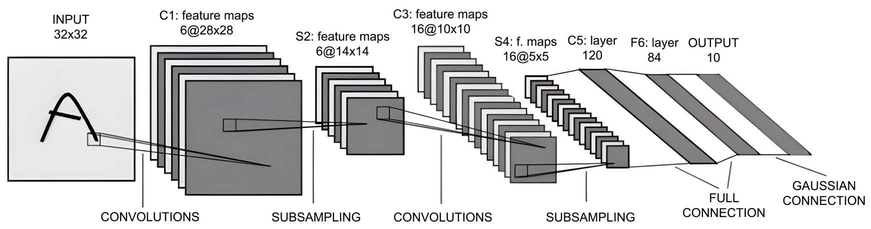


Figure 2.5: LeNet-5 architecture. Consists of two convolutional layers, two pooling layers, and three fully connected layers. Image adapted from [8].

AlexNet

AlexNet [9] achieved a breakthrough in the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) 2012. It used deeper convolutional layers and applied ReLU activations instead of traditional sigmoid functions, allowing for faster training. Local response normalization and dropout were also employed for regularization. AlexNet marked a turning point for deep learning in computer vision. Figure 2.6 illustrates the AlexNet architecture.

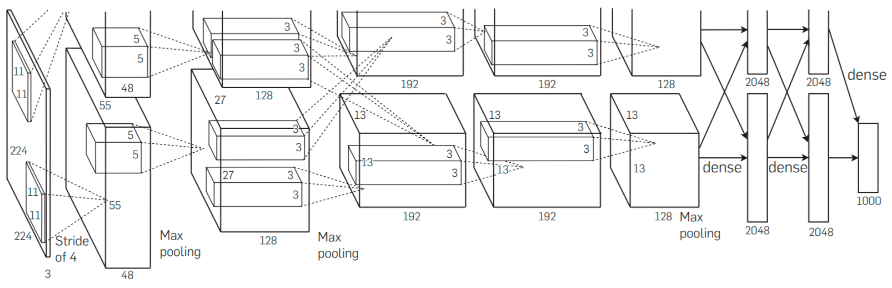


Figure 2.6: AlexNet architecture. Composed of five convolutional layers and three fully connected layers. Source: [9].

VGG

VGG networks, introduced by SIMONYAN and ZISSERMAN [50], emphasized architectural simplicity by using only 3×3 convolutions and increasing the network depth. The VGG16 and VGG19 variants became widely adopted as feature extractors due to their clean structure and performance. Figure 2.7 presents the structure of VGG-16.

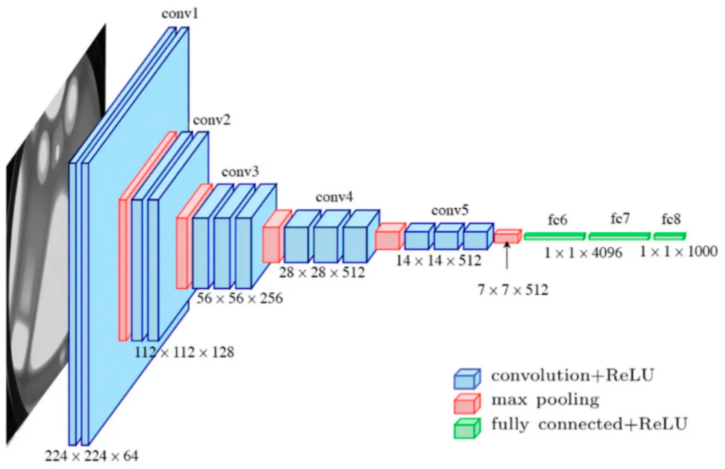


Figure 2.7: VGG-16 architecture. Deep network using repeated 3×3 convolutional filters and max-pooling layers. Figure reproduced from [10], originally sourced from ResearchGate.

GoogLeNet (Inception)

GoogLeNet [11] introduced the Inception module, which performs convolutions with different kernel sizes in parallel. This allowed the network to capture features at multiple scales without significantly increasing the number of parameters. The architecture also introduced global average pooling to replace fully connected layers, reducing overfitting. Figure 2.8 shows the Inception-based structure of GoogLeNet.

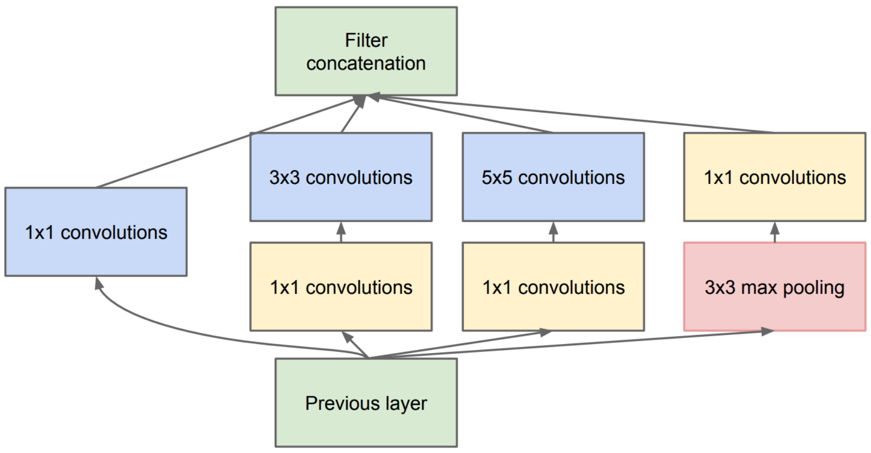


Figure 2.8: The Inception module enables multi-scale feature extraction with parallel convolutions. Source: [11].

.

U-Net

U-Net [12] was introduced for biomedical image segmentation. Its architecture features an encoder-decoder (the encoder part extracts features from images while the decoder part takes the extracted features and reconstructs the image) with skip connections that link these parts, preserving spatial information across scales. U-Net's design has been widely adopted in tasks requiring pixel-wise prediction. Figure 2.9 shows the U-Net structure.

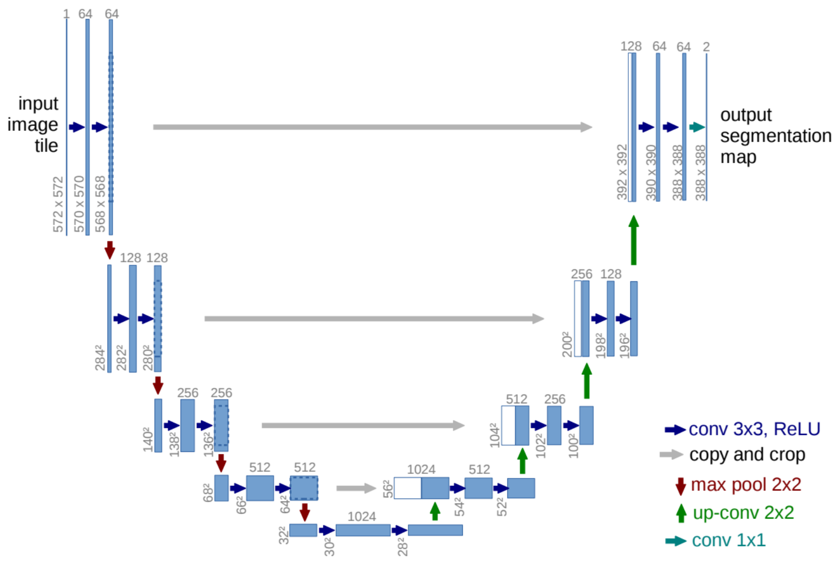
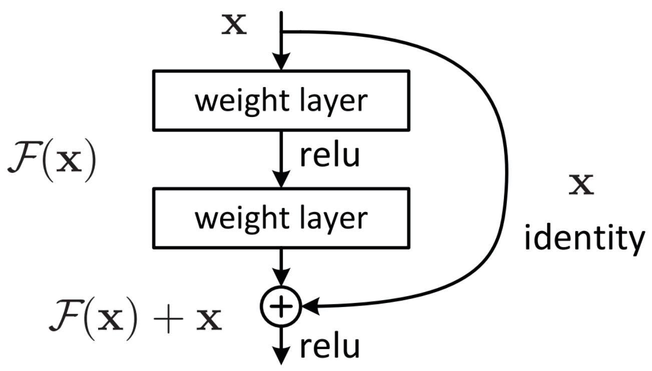


Figure 2.9: U-Net architecture. Combines an encoder-decoder structure with skip connections to preserve spatial information. Source: [12].

ResNet

ResNet [13] addressed the degradation problem in deep networks by using residual connections. These shortcut connections allow the gradient to flow directly through identity paths, enabling the training of much deeper models. Figure 2.10 illustrates the basic structure of a residual block used in ResNet.



x
weight layer
relu
$\mathcal{F}(\mathbf{x})$
weight layer
x
identity
$\mathcal{F}(\mathbf{x}) + \mathbf{x}$
relu

Figure 2.10: Residual block structure in ResNet. The identity connection helps mitigate vanishing gradients in deep networks. Source: [13].

DenseNet

DenseNet [14] proposed an architecture in which each layer receives input from all previous layers via feature map concatenation. This results in improved parameter efficiency and feature reuse. Dense connections also help gradient propagation and lead to more compact models. Figure 2.11 shows the structure of DenseNet.

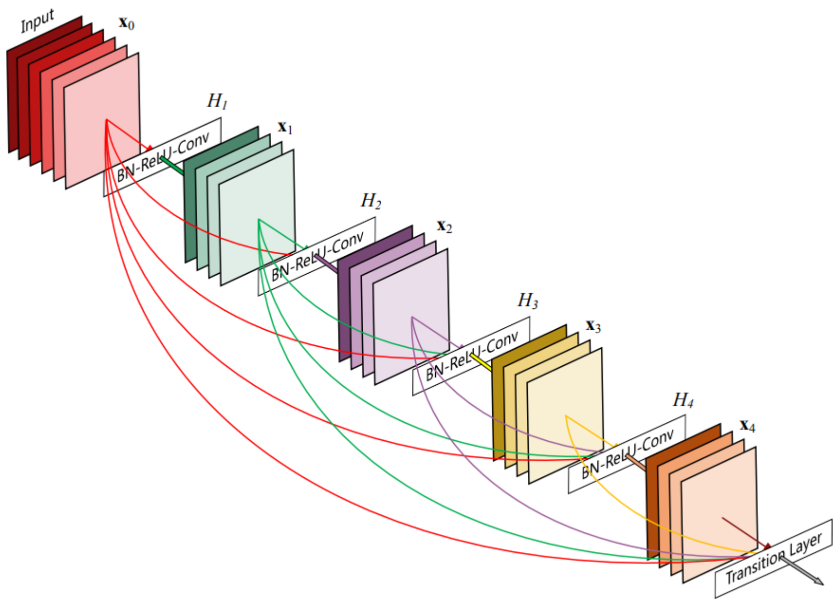


Figure 2.11: DenseNet architecture. Each layer receives as input the feature maps of all preceding layers. Source: [14].

MobileNet and EfficientNet

MobileNet [51] introduced depthwise separable convolutions to reduce computational cost and model size, making it suitable for mobile applications. EfficientNet [52] later proposed a compound scaling method to balance depth, width, and resolution in a principled way, achieving great results with fewer parameters.

2.3 Training: Hardware and Software for Deep Learning

Optimization methods are at the core of training deep neural networks. They are used to adjust the model parameters in a way that minimizes a given loss function over a dataset. As discussed in the previous subsection, network architectures have been progressively designed to facilitate optimization, addressing challenges such as vanishing gradients and enabling the training of deeper and more expressive models. However, efficient training also depends on appropriate hardware and software configurations, as they directly impact computational speed, memory usage, and overall feasibility of optimization at scale. These aspects will be examined in this section.

2.3.1 Training

Training a neural network involves finding parameters θ that minimize a loss function $L(\theta)$. This is a high-dimensional, non-convex optimization problem. Formally, the objective is:

$$\min_{\theta} L(\theta), \tag{2.1}$$

where $L(\theta)$ represents the empirical loss over a dataset.

The most widely used optimization technique in deep learning is gradient descent. It updates the model parameters in the direction of the negative gradient:

$$\theta \leftarrow \theta - \eta \nabla_{\theta} L(\theta), \tag{2.2}$$

where η is the learning rate. Variants of this method include Stochastic Gradient Descent (SGD), which updates parameters using a single training example or a mini-batch; Momentum [53], which accumulates a velocity vector to accelerate updates in consistent directions; RMSProp [54] and Adam [55], which adapt the learning rate for each parameter, improving convergence in practice. It is still the most used optimizer in practice. An overview and comparative analysis of these optimization strategies is provided by RUDER [56].

The optimization landscape of a neural network is complex. It contains many local minima, saddle points, and flat regions. Visualizations of this landscape [57] help us understand how optimization progresses. However, we do not expect to find a global minimum—our goal is to reach a "good enough" local minimum with low validation loss.

Training is typically performed using mini-batches, where each batch contributes to a gradient estimate used to update the model parameters. The choice of batch size affects both the convergence behavior and the noise present in the gradient. An epoch corresponds to one complete pass over the entire training dataset. In practice, training requires multiple epochs, often ranging from tens to hundreds, depending on the model complexity and the task. This iterative optimization procedure can be computationally demanding, particularly when working with large-scale datasets or deep architectures. This makes hardware considerations essential: efficient training depends not only on the choice of optimization algorithms but also on the availability of powerful computing resources. In the next subsection, we discuss the hardware ecosystem that supports modern deep learning workflows.

2.3.2 Hardware Ecosystem

Deep learning models, particularly CNNs, demand significant computational power. While training can technically be performed on CPUs, modern workflows rely heavily on GPUs due to their capacity to execute parallel operations over large data batches. This parallelism is particularly effective for the matrix and tensor operations that dominate deep learning.

NVIDIA has been the leading provider of GPUs for deep learning, with its CUDA platform offering libraries optimized for neural network computation. Alternative hardware, such as TPUs (Tensor Processing Units), has also emerged. TPUs are custom-designed by Google for tensor operations and are accessible through Google Cloud services. Although TPUs offer benefits in speed and scalability, they are not always a drop-in replacement for GPU-based workflows.

When local resources are limited, cloud computing platforms provide scalable infrastructure for deep learning tasks. Providers such as AWS [2], Google Cloud [3], and Microsoft Azure [4] offer access to GPUs and TPUs through virtual machines or managed services. Additional alternatives include Paperspace [5], which offers dedicated GPU machines with lower-cost options for training and inference, as well

[2]`https://aws.amazon.com/`

[3]`https://cloud.google.com/`

[4]`https://azure.microsoft.com/`

[5]`https://www.paperspace.com/`

as services like Lambda Labs [6] and RunPod [7], which specialize in providing GPU compute instances optimized for machine learning workloads. Each of the mentioned platforms offers flexibility in terms of pricing, performance, and ease of deployment, making them viable options depending on the project's computational demands.

2.3.3 Software Ecosystem

The two main software frameworks developed to simplify neural network design and training are TensorFlow [58] and PyTorch [59]. TensorFlow, developed by Google, allows for both low-level and high-level programming. It supports static computational graphs and provides tools such as TensorBoard for visualization, which is particularly useful during training. In contrast, PyTorch is popular in academic research for its dynamic computation graphs, which allow easier debugging and quick prototyping. It integrates well with Python and provides tools for automatic differentiation.

2.4 Generative Adversarial Networks

Generative Adversarial Networks (GANs) are a class of generative models, which means they are designed to synthesize new data samples that follow the same distribution as the training data. Introduced by GOODFELLOW *et al.* [31], GANs consist of two neural networks trained simultaneously: the generator and the discriminator. In the context of image-based tasks, these networks are typically implemented as CNNs. The generator attempts to produce realistic images from random noise or structured inputs, while the discriminator evaluates whether an image is real (from the training set) or fake (produced by the generator). A common analogy is to view the discriminator as an art critic or forgery expert, trained to spot inauthentic works, while the generator plays the role of the forger, constantly improving its techniques to fool the expert. This setup creates a dynamic in which the two networks are constantly competing and improving: as the discriminator becomes better at detecting fake images, the generator is forced to create increasingly convincing samples.

This adversarial relationship is formalized as a minimax optimization problem, where the generator G tries to minimize the discriminator's ability to distinguish real from fake, and the discriminator D tries to maximize it. The original loss function proposed by GOODFELLOW *et al.* [31] is:

$$\min_G \max_D \ \mathbb{E}_{x\sim p_{\text{data}}(x)}[\log D(x)] + \mathbb{E}_{z\sim p_z(z)}[\log(1 - D(G(z)))] \tag{2.3}$$

[6]`https://lambdalabs.com/`

[7]`https://www.runpod.io/`

In this framework, the generator learns to map latent vectors z (typically sampled from a Gaussian distribution) into synthetic images $G(z)$ that resemble the real data x. The discriminator, on the other hand, outputs a probability indicating whether a given image is real or fake. During training, gradients are backpropagated through both networks, guiding them to improve their respective roles. Over time, the generator ideally becomes so good at mimicking the data distribution that the discriminator can no longer distinguish between real and synthetic examples.

GANs have revolutionized computer vision by enabling tasks that were previously difficult or impossible with traditional models. Yann LeCun, one of the "godfathers of AI", said in a Quora post that GANs "(...) and the variations that are now being proposed is the most interesting idea in the last 10 years in ML" [60]. They are widely used for image-to-image translation, super-resolution, image inpainting, style transfer, and more. For instance, models like pix2pix [17] and CycleGAN [61] apply GANs to translate images between domains (e.g., sketches to photos, or summer to winter scenes). In these cases, the generator is often conditioned on an input image, leading to conditional GANs (cGANs), where the generation process is guided by additional information.

Despite their power, GANs present significant training challenges. The adversarial process can be unstable, with issues such as vanishing gradients (where gradients become too small for effective parameter updates) and mode collapse (where the generator produces limited varieties of samples). Various strategies have been proposed to address these challenges, such as improved loss functions [62], architectural changes [63], and regularization techniques [64]. Nevertheless, GANs remain one of the most influential and actively researched methods in generative modeling for computer vision.

Chapter 3

Related Works

This chapter reviews the main research efforts related to image-to-image translation, with a focus on works that are conceptually or methodologically close to the objectives of this thesis. The goal is to contextualize the proposed approach within the broader landscape of deep learning-based visual translation methods. We begin by surveying general image-to-image translation frameworks, including both paired and unpaired settings. Next, we examine methods related to visual stylization. Finally, we discuss prior works specifically targeting anime-to-manga translation.

3.1 Image-to-image translation

Recent approaches in computer vision predominantly rely on learning algorithms to support a variety of tasks. Specifically, deep generative models have shown considerable improvements in the image-to-image (I2I) translation problem. Many classic computer vision problems can be framed as an I2I translation task, where the main goal is to convert an input image from a source domain to a target domain while keeping its content and semantic representation.

An overview of the I2I works developed over the years [65] highlights some of the most used generative methods, such as generative adversarial networks (GANs) [31] and variational autoencoders (VAE) [66]. Currently, diffusion models (DM) [67] are recognized as highly effective generative models and have been successfully applied to various image synthesis tasks [68–71], although computationally expensive compared to GANs.

3.1.1 GANs and VAEs

Pix2pix [17] is a supervised I2I translation framework widely used as a baseline and inspiration for many other works. Based on a GAN framework, it can be applied to various tasks, such as synthesizing photos from semantic maps, transforming sketches into real-world photos, and generating colorized photos from black and white images. As a supervised method, pix2pix uses a set of aligned image pairs: from the source domain (the input) and the target domain (the ground truth). Unfortunately, collecting paired training data can be difficult or not well-defined in some cases.

On the other hand, CycleGAN [61] allows training without the need for paired data. Similar to pix2pix, it is a GAN framework that translates images between domains without requiring a one-to-one mapping. It uses a cycle consistency loss function and a set of four CNNs.

Complementary to these GAN-based approaches, variational autoencoders (VAEs) are also used for image-to-image translation. Many works combine a VAE with adversarial training—e.g., CVAE-GAN/BicycleGAN, UNIT, and MUNIT [72–75]—to encourage sharper and multimodal outputs.

3.1.2 Diffusion Models

Palette [76] is a diffusion-based model for image-to-image translation that produces high-quality results with stable training. Unlike GAN-based methods, which can be unstable and prone to mode collapse, Palette uses a denoising process to gradually generate realistic images from noisy inputs. It is a supervised method trained with paired images and can be applied to several tasks, such as colorization, inpainting, JPEG restoration, and uncropping. One of its main strengths is generality: the same model architecture can be used across different tasks without changes, while still preserving the semantic content of the input image. Despite the high quality of synthesized images, the generation process of these existing methods is extremely time-consuming.

3.2 Art Creation/Applications

I2I translation methods have shown a wide set of practical applications contributing to many visual tasks, such as image segmentation [77–80], realistic-looking image synthesis [81–84], image treatment procedures [85–87], colorization methods [28, 88, 89], and even artistic creation [61, 90, 90–97].

When used to create art, I2I methods usually tend to apply some artistic style to real-world photos [61, 90, 96, 97] or even translate these photos into cartoon [91, 93–97] or comic [95, 98, 99] images, which typically demand supplementary reference inputs. Moreover, those methods usually exploit the high-level features present in the real-world photos to guide the translation process, making a cartoon to comic translation more difficult.

3.3 Anime2Manga Translation Approachs

Anime illustrations can be seen as a type of cartoon image, and manga panels as a specific form of comic art, the number of dedicated image translation works targeting this domain remains limited. Existing approaches have explored a variety of techniques, including optimization-based methods [100], generative adversarial networks (GANs) [21, 101], variational autoencoders (VAEs) [20], and diffusion models (DM) [22]. While some achieve promising results, they share common limitations.

Differently from other works, our approach does not need the use of image pairs or high-resolution images, can consider any manga dataset as a target domain, and tries to not be limited in the number of possible screen patterns applied. Moreover, rather than relying on complex models like diffusion models, we use a two-stage GAN approach, demonstrating that dividing this translation process into generating a draft and refining it is sufficient to achieve high-quality results.

Our approach fits the unpaired image-to-image translation methods, specifically, the artistic creation ones. We aim to investigate how anime and manga images can be better explored within this contexts. For that, we propose a specific translation problem where anime illustrations (cartoon images) are converted to manga representations (comic images) faithfully, according to a specific target manga domain, while maintaining its screen pattern diversity.

Chapter 4

Proposed Method

We propose a two-stage anime-to-manga translation framework that utilizes any manga book title as a target domain while preserving its screen pattern distribution and diversity. The input is an image $X \in \mathbb{R}^{w \times h \times c}$, and the output is its manga representation $Y \in \mathbb{R}^{w \times h}$ according to the target domain. Here, w, h, and c represent the image's width, height, and number of channels, respectively. Three CNNs are trained:

- **Generator Network** G: Learns a mapping from the input image to an output preliminary manga representation.
- **Discriminator Network** D: Aims to distinguish real manga images from the translated ones at multiple scales, labeling them as either real or fake.
- **Encoder Network** E: Extracts and classifies feature signals from screen patterns.

Let $\mathcal{A}$ represent any anime domain and $\mathcal{M}$ any manga domain. Given an anime training dataset $S_{\text{data}}(a) = \{a_i \mid a_i \in \mathcal{A},\, i = 1, \ldots, N\}$ and a manga training dataset $S_{\text{data}}(m) = \{m_j \mid m_j \in \mathcal{M},\, j = 1, \ldots, M\}$, where N and M denote the respective number of images in each dataset. Our framework initially predicts a preliminary manga representation $G(X) = \hat{Y} \in \mathbb{R}^{w \times h \times c}$ during the first stage. In Chapter 5, we show some examples of these datasets, specifically, refer to Figure 5.6 and Figure 5.7.

In the second stage, the main goal is to generate a high-quality output Y based on the preliminary manga representation $\hat{Y}$. Alongside a proper segmentation of X, $\hat{Y}$ is used as a hint by an encoder network E to estimate a probability matrix $P \in \mathbb{R}^{K \times C}$, where K denotes the number of regions where an artist would apply a screen pattern, and C represents the number of screen pattern classes available in a predetermined set. In this thesis, we consider $C = 10$ different types of patterns, as shown in Figure 5.8, Chapter 5. Each row in P defines a set of screen pattern probabilities, indicating the most suitable patterns for each of the K regions. Based on these probabilities, the screen pattern with the highest probability is applied to the corresponding region, although our framework allows the user to select an alternative well-rated screen pattern. We refer to these steps as the screen pattern pasting procedure ψ_{pattern}. Finally, a line extraction procedure ψ_{line} is used to preserve the structural lines present in the input image. These lines typically define the boundaries and contours of the objects within a scene.

Considering $\odot$ as the Hadamard product, the output manga image can be easily composed as:

$$Y = \psi_{\text{pattern}}(\hat{Y}, X) \odot \psi_{\text{line}}(X). \tag{4.1}$$

The main inputs and outputs of the framework are shown in Figure 4.1.

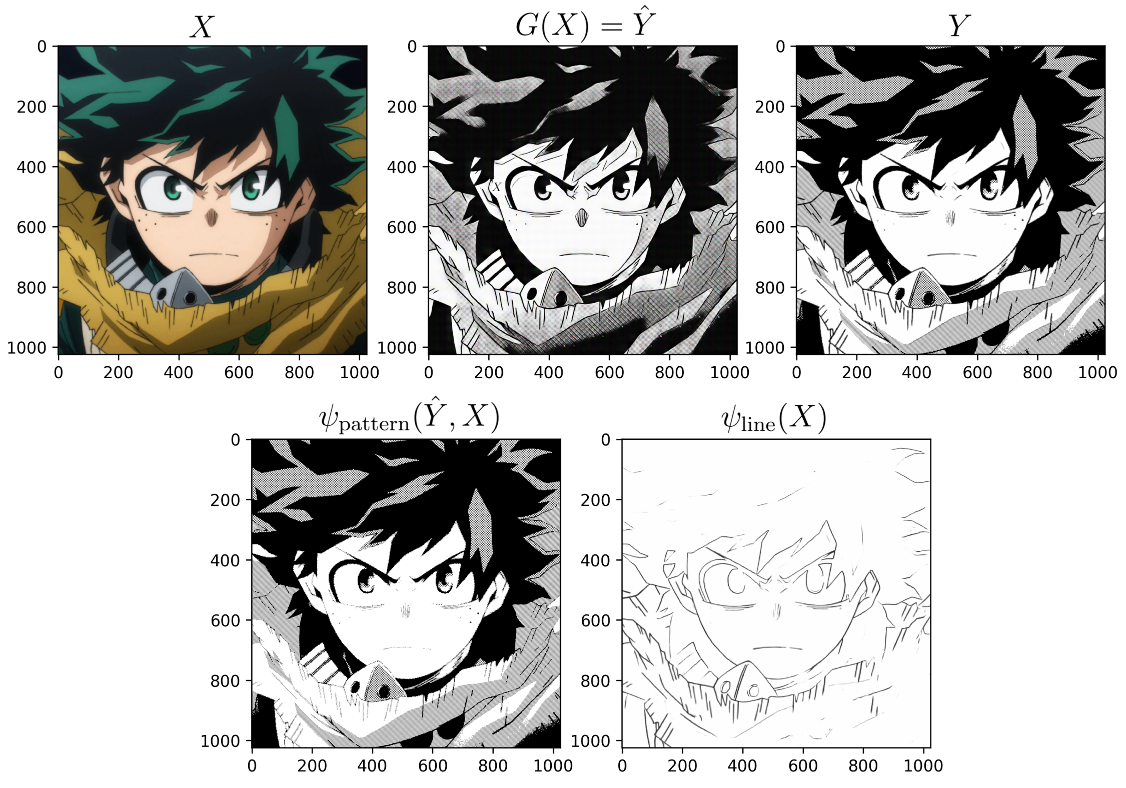


Figure 4.1: The input image X, the preliminary manga representation $\hat{Y}$, the screen pattern pasting procedure output ψ_{pattern}, the line extraction procedure output ψ_{line} and the final output Y.

4.1 First Stage: Preliminary Prediction

In the first stage, a CNN learns a mapping from an input image to an output preliminary manga image. We consider this prediction preliminary because achieving a high-quality manga representation, where screen patterns are perfectly applied to the appropriate regions, is challenging. Initially, our goal is to translate an anime image into a draft manga representation while preserving the global screen pattern composition of the target manga dataset $S_{\text{data}}(m)$ and the semantic content of the input image X.

Following a GAN framework [31], the discriminator network D is trained to distinguish real manga images from generated ones and to provide feedback to the generator network G. This feedback helps G learn how to transform anime images into manga-style outputs. As training progresses, D becomes increasingly proficient at detecting synthetic images, while G is simultaneously pushed to generate outputs that are more indistinguishable from real manga illustrations.

Let $\mathcal{L}_{full}(G, D)$ be the full loss function involving both networks, where G and D denote their parameters. The training process aims to solve the following min-max optimization problem:

$$(G^*, D^*) = \arg\min_{G}\max_{D} \mathcal{L}_{full}(G, D) \tag{4.2}$$

After convergence, the trained generator G^* is retained for inference, while the discriminator D^* is no longer required, as its role was solely to guide the generator during the adversarial learning process. The next subsections describe how we define the $\mathcal{L}_{\text{full}}(G, D)$ function. Technical details of this training dynamics will be discussed in Chapter 5.

4.1.1 Learning from the Style Representation

Anime images are essentially composed of several low-level features, such as edges, color blocks, simple textures, simple shapes, and variations in brightness and contrast. Additionally, we observe that this kind of art has a significant correlation between its color segments and the screen patterns present in manga images. These segments often correspond to visual elements like clothing, shading, background and skin.

Figure 4.2 illustrates this correspondence with an example pair of anime and manga images. Observe how the distinct color regions in the anime image are translated into specific screen pattern types in the corresponding manga version. Darker colors, such as the dark blue of the outfit or the character's hair, are predominantly mapped to plain black screen pattern. Lighter color areas, including the character's clothing highlights, are mapped to plain white pattern. Additionally, mid-tone colors used for shading, such as the blue in the shaded folds of the outfit, are rendered using a chess-like screen pattern.

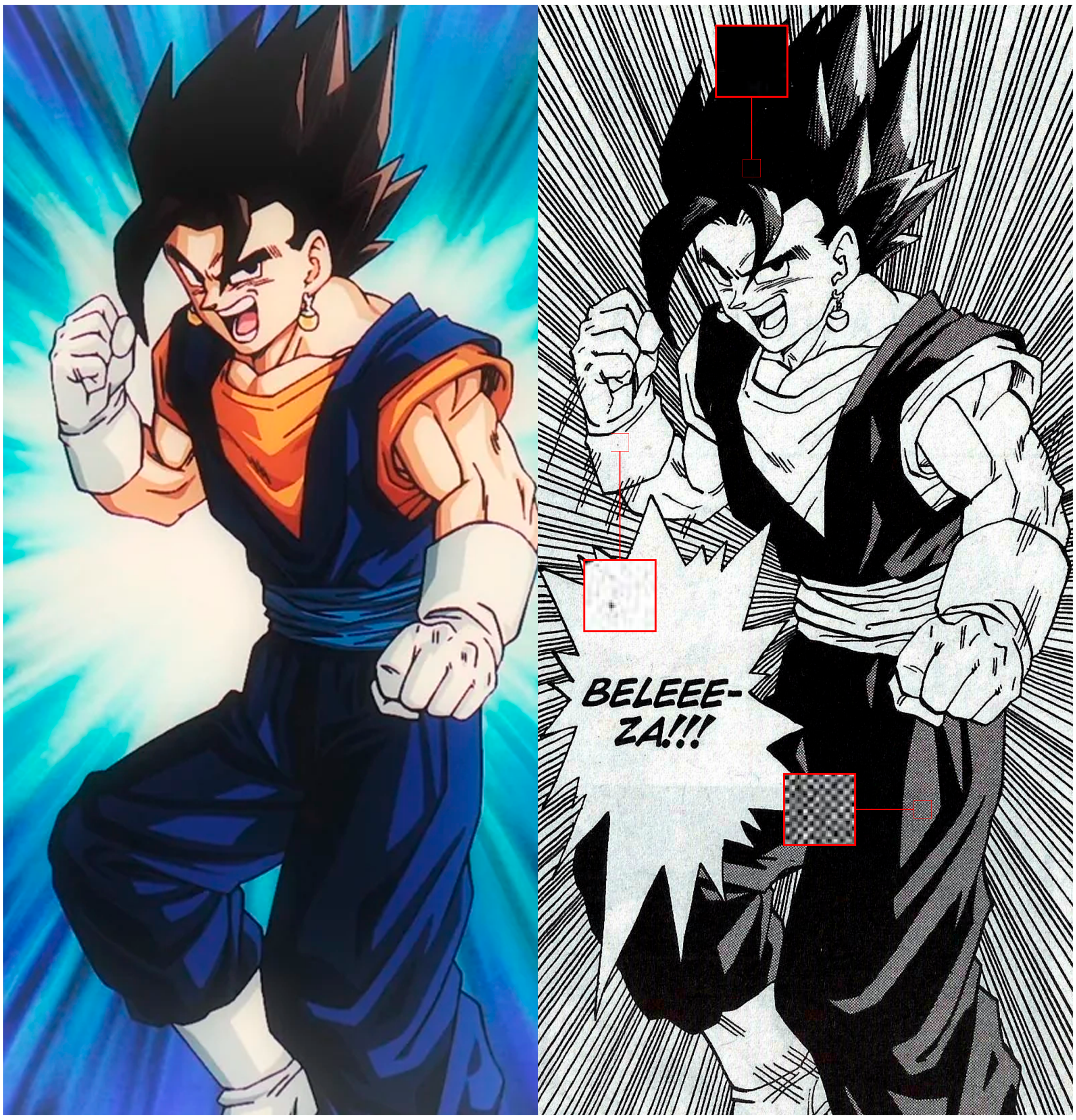


Figure 4.2: Example of a anime frame (left) [2] and its corresponding manga panel (right) [1] from the Dragon Ball manga series and Dragon Ball Z anime. Distinct color regions in the anime frame map to screen patterns in the manga version.

Another example is shown in Figure 4.3. In this case, the background in the anime image is a solid blue tone and is mapped to a dotted screen pattern in its manga version.

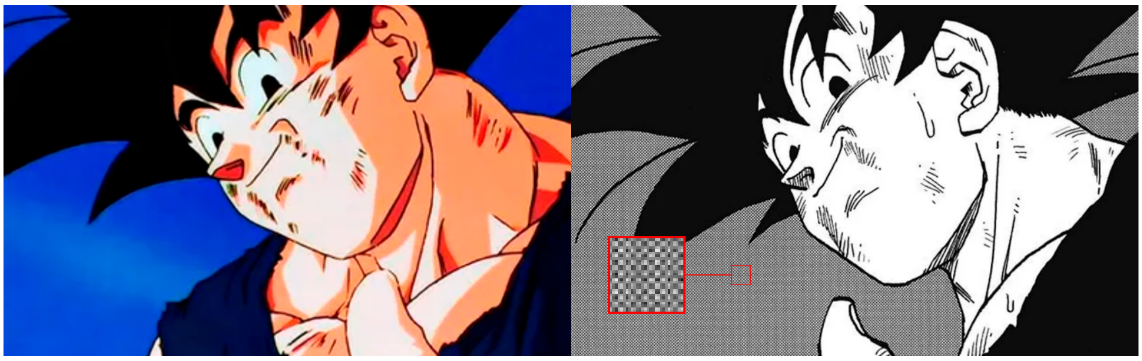


Figure 4.3: Example of an anime image (left) and its corresponding manga version (right). The blue background in the anime frame is represented by a dotted screen pattern in the manga version. Anime frame from Dragon Ball Z [2]; manga frame from Dragon Ball [1].

A third example is shown in Figure 4.4. In this case, the orange color of the dragon in the anime image is translated into a gray wave-like screen pattern in the manga version.

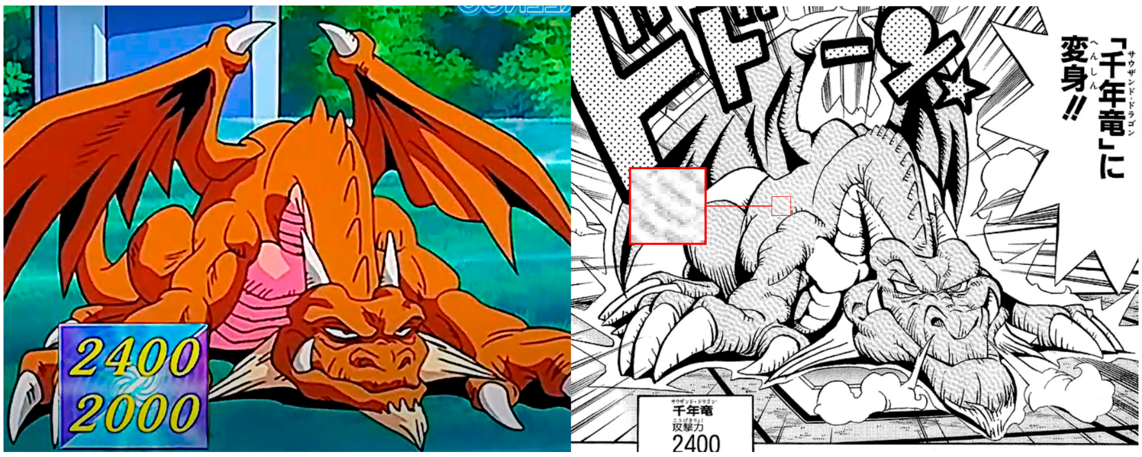


Figure 4.4: Example of an anime image (left) and its corresponding manga version (right), showing the Thousand Dragon character. The orange area of the anime dragon is mapped to a gray wave-like screen pattern in the manga version. Anime frame from Yu-Gi-Oh! Duel Monsters [15] and manga panel from Yu-Gi-Oh! [16].

Based on these observations, for any input image X, we obtain the feed-forward prediction $G(X)$ as $\hat{Y}$, where G is a U-Net-based architecture [12] network, following the approach proposed by ISOLA *et al.* [17]. The skip connections present in the U-Net architecture facilitate the effective propagation of low-level features and help preserve spatial information, which is particularly important for maintaining the structural details of anime images during translation. Refer to Figure 4.5, where the skip connections between encoder and decoder blocks—namely, `sequential[1][0]`, `sequential_1[1][0]`, `sequential_2[1][0]`, `sequential_3[1][0]`, `sequential_-4[1][0]`, and `sequential_5[1][0]`— transfer features from early layers to their corresponding decoding stages.

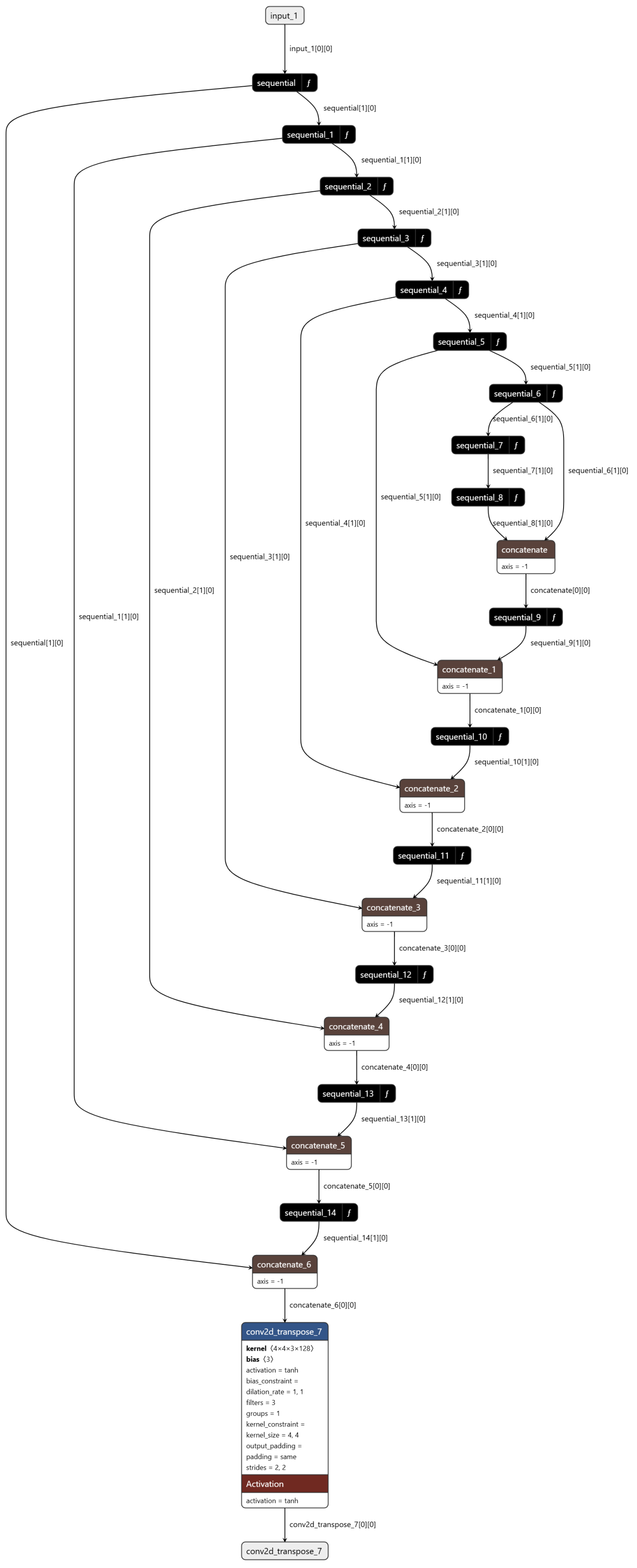


Figure 4.5: Generator architecture based on U-Net [12], following [17].

Regarding the discriminator D, we initially designed it as the traditional convolutional PatchGAN classifier [18], which penalizes structures only at the scale of image patches. It outputs a matrix of values with each value representing the decision of discriminator on a patch or small region on the input image. Instead of classify an entire image as fake or real, it makes that decision on patches on the image. Observe Figure 4.6.

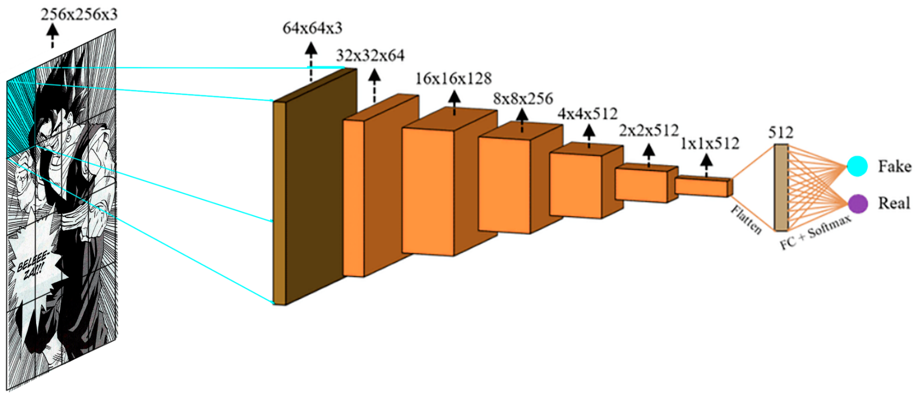


Figure 4.6: Illustration of the PatchGAN discriminator [18]. Instead of computing a single score for the entire input image, PatchGAN evaluates smaller patches independently. Each patch is classified as real or fake, and the final decision aggregates these local evaluations.

However, we observed that discriminating between model outputs and reference manga images in this way is insufficient for effective translation between anime and manga images. Since manga patches typically consist of both black and white patterns, the discriminator may accept arbitrary monochromatic images, causing the generator to make no further improvements.

To address this issue, we propose discriminating between real and generated data at multiple scales during training, allowing the discriminator to more effectively capture both mid-level and low-level characteristics of manga images. Given a generated output $G(a_i)$, where $a_i \in S_{\text{data}}(a)$ is an instance from the anime training dataset, and a real manga instance $m_j \in S_{\text{data}}(m)$, we modify the discriminator to accept as input features extracted from multiple layers of the VGG16 network [50], pre-trained on ImageNet [102]. We denote the set of feature maps extracted from specific layers of VGG16 as $\mathcal{F}(x)$ for an arbitrary image x, where:

$$\mathcal{F}(x) = \{\mathcal{F}_{12}(x), \mathcal{F}_{22}(x), \mathcal{F}_{33}(x), \mathcal{F}_{43}(x)\}. \tag{4.3}$$

Here, $\mathcal{F}_{ij}(x)$ represents the feature map extracted from the ij-th convolutional layer of VGG16. In our implementation, we specifically use the layers `conv1_2`, `conv2_2`, `conv3_3`, and `conv4_3`, which correspond to $\mathcal{F}_{12}(x)$, $\mathcal{F}_{22}(x)$, $\mathcal{F}_{33}(x)$, and $\mathcal{F}_{43}(x)$, respectively. Observe the architecture shown in Figure 4.7, where these feature maps are being fed to the network.

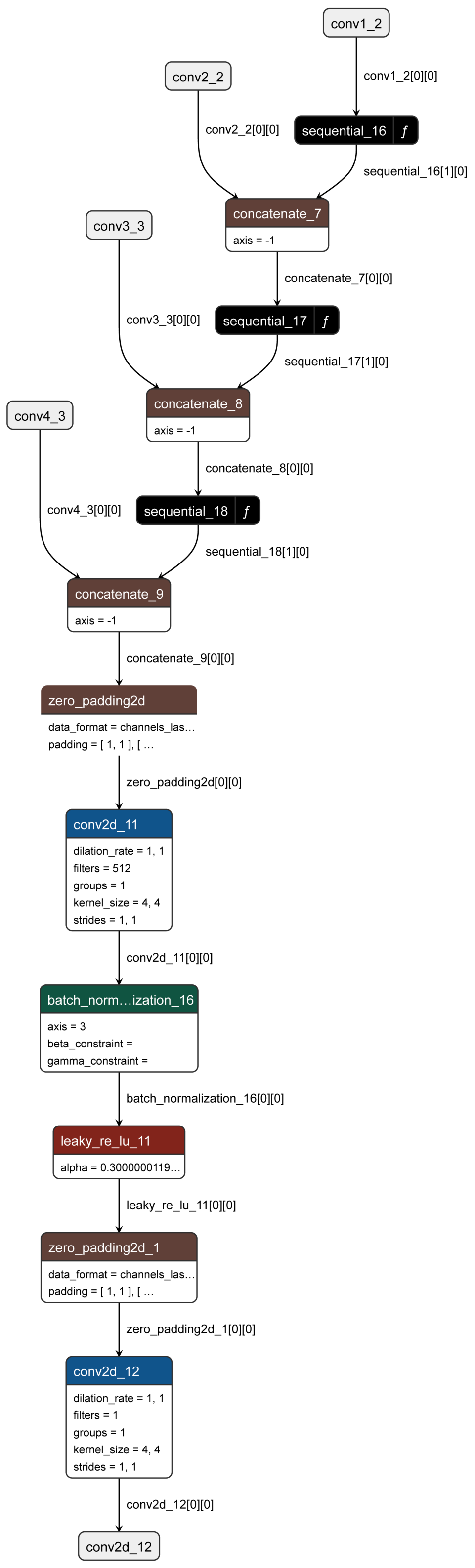


Figure 4.7: Custom PatchGAN discriminator architecture.

The selection of VGG16 layers is based on their ability to capture visual features at different levels of abstraction [49]. Lower layers, such as $\mathcal{F}_{12}$ and $\mathcal{F}_{22}$, respond mostly to low-level features such as edges, contours, and basic textures—elements relevant to manga generation. In contrast, intermediate layers like $\mathcal{F}_{33}$ and $\mathcal{F}_{43}$ are more sensitive to mid-level features, such as regional patterns and shape structures, which are related to the use of screen patterns.

This combination was empirically selected after preliminary experiments revealed that deeper layers beyond $\mathcal{F}_{43}$ contributed less meaningfully to the discriminator's judgment in this specific task. These deeper layers tend to capture more high-level features, like semantic and object-level information, which, while beneficial in natural image domains, showed limited impact in our scenario where semantic fidelity is often abstracted through repeated black-and-white color blocks.

Using this notation, we formulate the adversarial style loss $\mathcal{L}_{\text{adversarial}}$ as

$$\mathcal{L}_{\text{adversarial}} = \mathbb{E}_{m_j \sim S_{\text{data}}(m)}\big[\log D(\mathcal{F}(m_j))\big] + \mathbb{E}_{a_i \sim S_{\text{data}}(a)}\big[\log\big(1 - D\big(\mathcal{F}(G(a_i))\big)\big)\big]. \quad (4.4)$$

As defined by the GAN setup (4.2), the corresponding discriminator loss is given by $\mathcal{L}_D(G, D) = -\mathcal{L}_{\text{adversarial}}(G, D)$, which reflects the discriminator's goal of maximizing the adversarial objective.

4.1.2 Maintaining the Input Content

Originally introduced in the CartoonGAN framework [91], a content loss was designed to preserve the semantic content of a real-world photo while translating it to a cartoon style. The idea is to minimize the ℓ_1 distance between the high-level feature maps extracted by a VGG network from both the input and the generated images. Formally, the loss is defined as

$$\mathcal{L}_{\text{con}} = \mathbb{E}_{p_i \sim S_{\text{data}}(p)} \left[\big\| \text{VGG}_l\big(G(p_i)\big) - \text{VGG}_l(p_i) \big\|_1 \right], \quad (4.5)$$

where p_i is an input photo, $G(p_i)$ is the generated cartoon image, and $\text{VGG}_l(\cdot)$ denotes the feature map output from the `conv4_4` layer of a VGG network pretrained on ImageNet [102]. Although these high-level feature maps in the VGG16 network have demonstrated good object preservation capabilities, in the context of image-to-image translation, local color and texture information is often lost [103, 104]. Specifically, in the case of anime-to-manga translation, local color and texture information is what defines the input content.

The nature of anime and manga images differs from that of real-world photos. While CartoonGAN focuses on simplifying complex photo textures into clean cartoon lines, anime and manga images already share a similar and simpler visual abstraction. Thus, applying a content loss based on high-level features extracted from a VGG network trained on photographic data, such as ImageNet [102], is not sufficient to preserve content.

We observed that the content of anime or manga images is essentially their structural lines (lines that define the shapes, forms, and contours of characters, objects, and scenes) alongside the black and white color blocks. This is enough to understand the image semantically. Figure 4.8 shows some examples. Observe that the binarized images still keep the main content of the originals. Even with only black and white, it is possible to understand the shapes and structure of the scene. Different colors in anime images or screen patterns in manga images typically serve an aesthetic purpose rather than contributing to the image content itself.

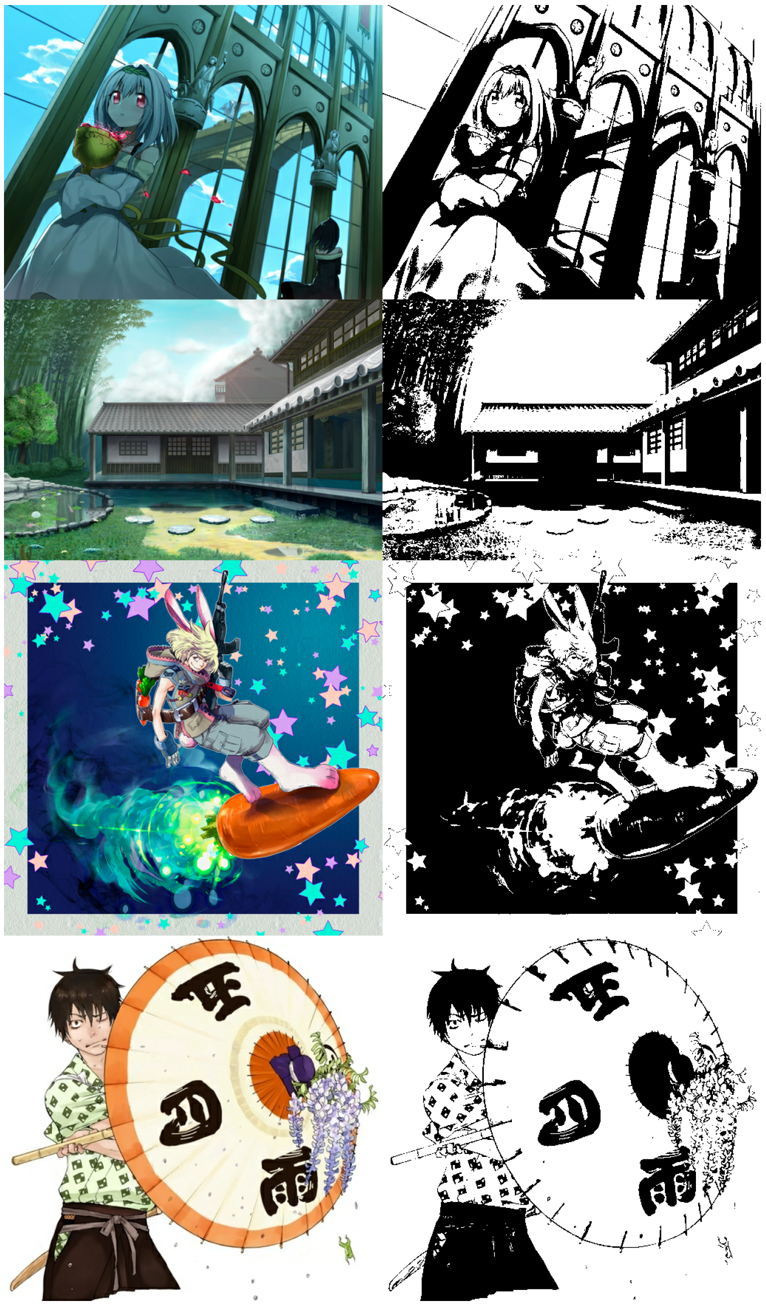


Figure 4.8: Examples of input anime images (left) and their corresponding binarized forms (right).

With that in mind, a binary representation can be viewed as an intermediate representation between anime and manga images where only black and white colors or screen patterns (considering plain black and plain white as screen patterns) are applied. To ensure that the resulting manga image retains the semantic content of the input anime image, we introduce the binary content loss $\mathcal{L}_{content}$. It is defined as the ℓ_1 sparse regularization between the preliminary manga image $G(a_i)$ and the input binary representation $\phi(a_i)$.

$$\mathcal{L}_{content} = \mathbb{E}_{a_i \sim S_{\text{data}}(a)} \left[\| G(a_i) - \phi(a_i) \|_1 \right], \tag{4.6}$$

where ϕ represents a binarization function that segments and maps pixel intensities into two clusters: black and white. Details on the binarization function are discussed in the following paragraphs.

Alternative Binarization Strategies

The function $\phi(\cdot)$ is responsible for generating high-contrast black-and-white representations from anime images. Since this step directly affects the semantic alignment between the input and output, we evaluated several binarization strategies with the goal of preserving structural elements such as contours and edges while minimizing noise and irrelevant detail.

We considered four candidates for $\phi(\cdot)$: Otsu's global thresholding [105], adaptive Gaussian thresholding [106, 107], Laplacian-based edge-enhancing preprocessing [108], and clustering-based binarization via k-means. Otsu's method selects a threshold that maximizes between-class variance; it is efficient and unsupervised but struggles with gradual tone transitions. Adaptive Gaussian thresholding computes local thresholds from neighborhood statistics and handles non-uniform illumination, yet it can amplify texture and fragment regions in low-contrast areas. Laplacian preprocessing emphasizes edges prior to thresholding but is sensitive to background texture. Finally, the k-means approach clusters pixel intensities into two groups, mapping the brighter cluster to white and the darker to black.

As shown in Figure 4.9, Otsu and k-means yield visually satisfactory results. Laplacian preprocessing also performs well but loses some structural line details, whereas the adaptive method amplifies texture and introduces noise. Given these outcomes, and to keep preprocessing simple and reproducible, we set $\phi(\cdot)$ to k-means for all experiments; Otsu performs comparably and remains a viable alternative.

Segmentation models based on convolutional networks, such as those proposed by RONNEBERGER *et al.* [12], BADRINARAYANAN *et al.* [109], could also be used. However, since Otsu's method and k-means clustering produce satisfactory results, their simplicity was preferred over more complex data-driven approaches. We selected k-means clustering for $\phi(\cdot)$ in our experiments, acknowledging that Otsu's method performed comparably and remains a viable alternative.

4.1.3 Patch-wise Pattern Consistency

Given the set of manga images $S_{\text{data}}(m)$ as the target dataset, we first calculate the screen pattern distribution, denoted as P_{target}, using a patch-based approach. Specifically, we divide the images into non-overlapping patches of size 32×32, and then determine the distribution P_{target} from the patterns observed in these patches.

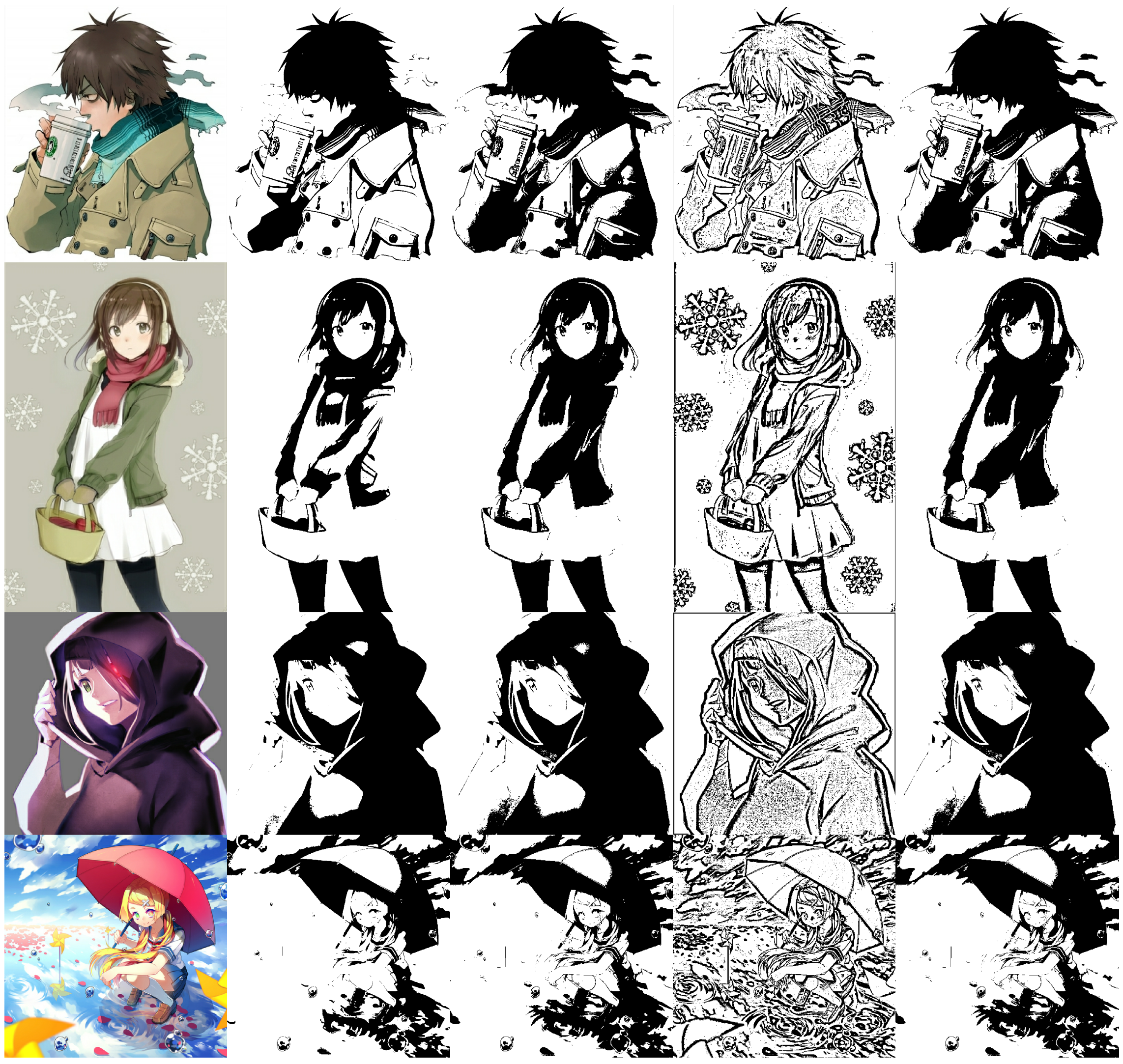


Figure 4.9: Comparison of binarization methods applied to a sample anime image: (**a**) Original input, (**b**) Laplacian-enhanced preprocessing, (**c**) Otsu's thresholding, (**d**) Adaptive Gaussian thresholding, (**e**) k-means clustering.

Similarly, the preliminary manga representation $\hat{Y}$ is divided into non-overlapping patches, denoted as $\{\hat{Y}_i\}_{i=1}^{L}$, where L is the total number of patches. Each patch $\hat{Y}_i$ is then classified by E to predict the pattern distribution for that patch, denoted as $P_{\text{pred}}(\hat{Y}_i)$.

The pattern consistency loss, $\mathcal{L}_{\text{pattern}}$, is defined as the average Kullback-Leibler (KL) divergence [110] between the target pattern distribution P_{target} and the predicted pattern distribution for each patch $P_{\text{pred}}(\hat{Y}_i)$:

$$\mathcal{L}_{\text{pattern}} = \frac{1}{L}\sum_{i=1}^{L} D_{\text{KL}}\left(P_{\text{target}} \parallel P_{\text{pred}}(\hat{Y}_i)\right), \tag{4.7}$$

where D_{KL} denotes the KL divergence, which is defined as:

$$D_{\text{KL}}(P \parallel Q) = \sum_{j} P(j) \log\left(\frac{P(j)}{Q(j)}\right). \tag{4.8}$$

Here, $P(j)$ and $Q(j)$ represent the probability distributions over the screen pattern classes for the target and predicted distributions, respectively. This loss encourages the generated image $\hat{Y}$ to maintain a pattern distribution across its patches that closely matches the target distribution P_{target}. Furthermore, it enforces the generation of patterns that are more easily recognized by the classifier E, resulting in images that can be better classified in the second stage of the framework.

To better illustrate the target distribution P_{target}, consider that we are working with a set of $C = 10$ available screen patterns (as discussed in Chapter 5 and shown in Figure 5.8). Figure 4.10 shows the frequency of each screen pattern class computed over all patches in the target manga dataset. As observed, the plain white pattern appears most frequently, accounting for more than 50% of the patches - the common use of white regions in manga panels. Other patterns, such as plain black, dots, and vertical lines, are also well represented, while some like checked or diagonal left appear less often.

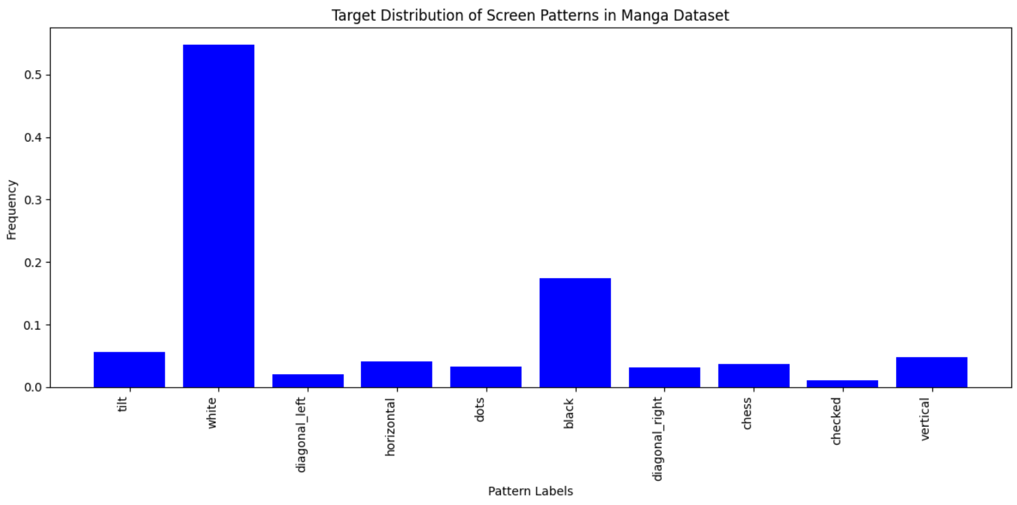


Figure 4.10: Target distribution of screen patterns in the manga dataset, computed over all patches. The horizontal axis denotes each pattern label, while the vertical axis shows its relative frequency.

The Patch Classifier

The encoder network E is used to classify screen patterns in both training and inference stages. It is used in the first stage to evaluate local predictions during generation and helps define which screen patterns should be applied in the second stage of the framework.

Screen patterns are visually similar, often bitonal and defined by regular structures. This makes classification challenging, especially when the number of available patterns increases. Figure 4.11 illustrates several different screen patterns.

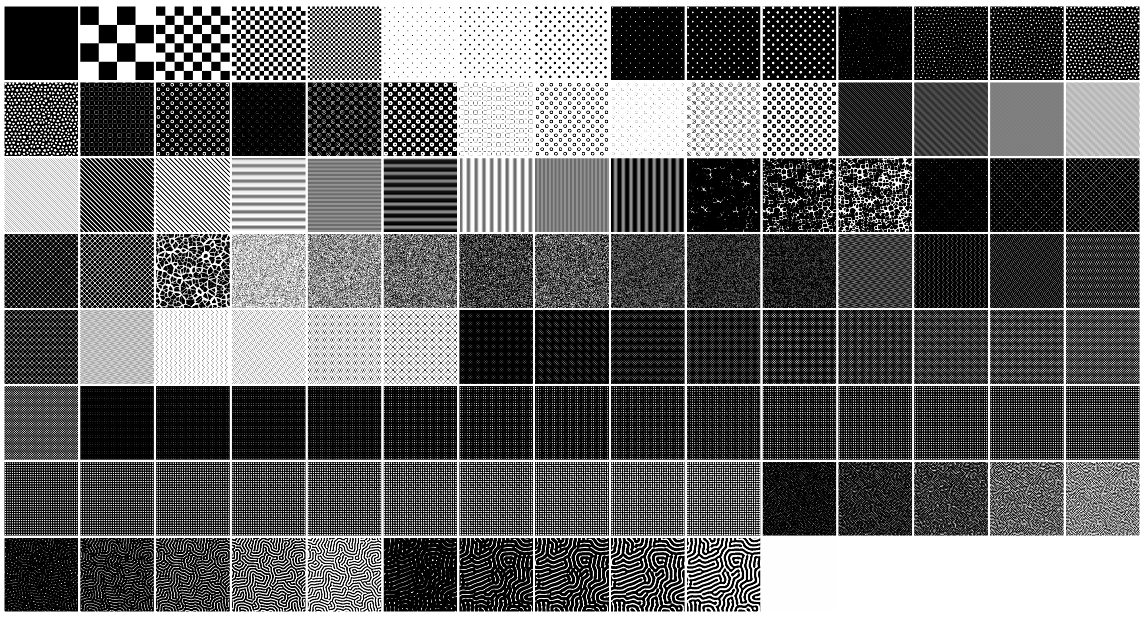


Figure 4.11: Examples of screen patterns demonstrating their bitonal nature and spatial regularity.

Training the network directly with a cross-entropy objective often leads to poor separation in feature space, as similar patterns may lie close to decision boundaries. To address this, we apply supervised contrastive learning [111], which improves class separation by encouraging samples of the same class to cluster together in the embedding space, while pushing apart those from different classes. This approach works well for multi-class problems with subtle visual differences and no fixed limit on the number of classes.

The encoder is based on the ResNet50 architecture [13] and is trained in two phases. In the first phase, the contrastive loss is used to structure the embedding space. In the second, a softmax classifier is trained on top of the frozen encoder using a cross-entropy objective. After training, the model achieved 84.3% accuracy on the test set. As a baseline, end-to-end training with cross-entropy only attains 81.5%.

We consider a fixed set of 10 screen pattern categories in this thesis, as shown in Figure 5.8, although this number can be expanded or reduced depending on user needs. All input patches used for training the encoder have a resolution of 32×32 pixels. More details on the training dataset are provided in Chapter 5.

4.1.4 Full Loss

The loss function is used to jointly optimize the GAN-based framework. It consists of three parts: (1) the adversarial style loss $\mathcal{L}_{\text{adversarial}}$ which drives the generator network to achieve the stylization, (2) the binary content loss $\mathcal{L}_{\text{content}}$ which preserves the image content during the transformation, and (3) the pattern consistency loss $\mathcal{L}_{\text{pattern}}$ which enforces the generation of screen patterns present in the target dataset. The hyper-parameters α, β, and γ are used to ensure a balance between stylization, content preservation, and patch-wise pattern consistency.

$$\mathcal{L}_{\text{full}} = \alpha\mathcal{L}_{\text{content}} + \beta\mathcal{L}_{\text{adversarial}} + \gamma\mathcal{L}_{\text{pattern}}. \tag{4.9}$$

We empirically tested multiple configurations of the weighting coefficients and found that $\alpha = 1$, $\beta = 10$, and $\gamma = 5$ provided a balanced trade-off between preserving semantic content and achieving visual stylization. As is common in GAN training, evaluation is mainly qualitative, given the lack of standardized metrics for measuring visual quality in image translation tasks. These weights also help stabilize adversarial training and prevent issues such as mode collapse, as discussed in Chapter 5.

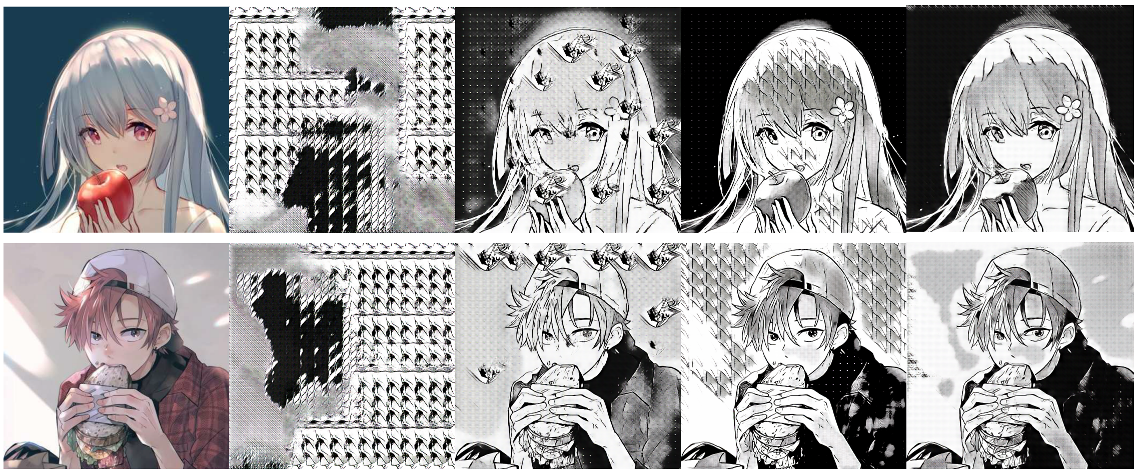

(a) (b) (c) (d) (e)

Figure 4.12: Ablation study conducted by systematically removing each component, focusing on results from the first stage aimed at creating the intermediate result: (**a**) Input image. (**b**) With traditional discriminator (PatchGAN). (**c**) With traditional content loss (CartoonGAN). (**d**) W/O pattern loss. (**e**) Full model. Zoom in to see the details.

4.1.5 Ablative Study

We present the results of our ablation study in Figure 5.5. This study aims to analyze the impact of each component in the first stage of our framework, where an intermediate result is created.

Replacing the custom discriminator with a traditional PatchGAN architecture [18] causes the GAN to quickly fall into mode collapse (Figure 5.5-(b)). We believe that a more appropriate discriminative capability for this kind of image, helps maintain stable training.

Additionally, substituting the binary content loss (4.6) with a traditional content loss (4.5), as discussed in Section 4.1.2, can cause the generated images to have inaccurate intensity fidelity and artifacts. This is shown in Figure 5.5-(c).

Lastly, omitting the pattern loss function (Figure 5.5-(d)) can hinder the overall process since a lot of artifacts are created. In the second stage, where a classifier evaluates the results and determines the appropriate screen pattern for each region, it is important to ensure that the generated image possesses features that will facilitate the classifier's task.

The final result of the first stage, the draft manga representation, incorporating all components, is presented in Figure 5.5-(e).

4.2 Second Stage: Final Image Generation

The manga draft predicted by the generator network, although considered a good sample from the manga dataset by the discriminator network, is not yet satisfactory from an artist's perspective. This is because a good manga representation has regions presenting the same screen pattern perfectly and uniformly distributed. Therefore, an output manga image without a clear screen pattern pasted into each region yet with correct shading is likely to confuse the discriminator. As a result, the generator may stagnate and no longer improve.

To circumvent this problem, we aim to use the manga draft as a guide to select and paste a similar and proper screen pattern based on the possible regions where one would be applied by an artist. Moreover, we want to provide the user with control over the regions and screen patterns that would best compose the final image. Our framework is designed to detect and suggest proper screen patterns for each region an artist would apply.

Unlike the first stage, which is learning-based and relies on adversarial training to generate a preliminary manga representation $\hat{Y}$, the second stage is more application-driven. It operates as a post-processing step designed to enhance $\hat{Y}$ by adding consistent and semantically appropriate screen patterns, as well as structural lines that define the final composition. As such, it involves several user-defined parameters and design choices, allowing for flexibility and control.

4.2.1 The Screen Pattern Pasting Procedure

In real manga images, patches from the same region have a highly homogeneous semantic relationship. For instance, in Figure 4.13, patches of the character's hair have similar semantic information as they represent the same screen pattern applied to that region. The same can be said about the character's face, pants, eyes and shirt. Since the colors in the anime image usually define those regions, we observe that in the problem of translating an anime to a corresponding manga version, those regions should have the same screen pattern.

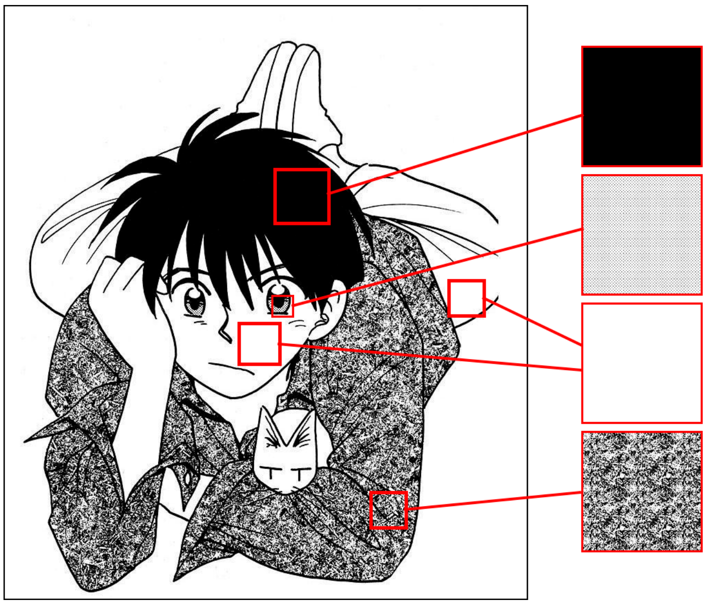


Figure 4.13: Patches within the same region exhibiting strong semantic consistency, indicative of the same screen pattern.

Given the input image X, we first segment it into regions where the same screen pattern is likely to be applied. This segmentation is based on RGB similarity using k-means clustering with a fixed number of 6 clusters. Other clustering methods, such as Mean Shift [112], could also be used, especially if the user does not want to specify the number of clusters. The segmentation produces a set of non-overlapping regions (masks) $\{R_k\}_{k=1}^{K}$ that partition the image:

$$\forall i \neq j, \quad R_i \cap R_j = \emptyset, \quad \text{and} \quad \bigcup_{k=1}^{K} R_k = X \tag{4.10}$$

An example of this segmentation is shown in Figure 4.14.

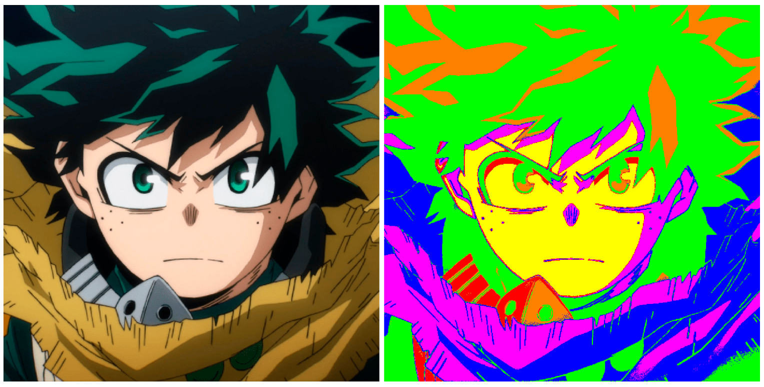


Figure 4.14: On the left is the input image X, and on the right is the result of the segmentation into six regions, each indicated by a different color. These regions (masks) are expected to receive the same screen pattern.

For each mask $\{R_k\}_{k=1}^{K}$, we compute a corresponding masked image $\{\hat{R}_k\}_{k=1}^{K}$ to be used as input to the encoder. Each $\hat{R}_k$ is obtained by masking the generated image $\hat{Y}$ and filling the complement with mid-gray according to the following operation:

$$\hat{R}_k = R_k \circledast \hat{Y} \tag{4.11}$$

Here, $\circledast$ denotes a pixel-wise masking operation. For each pixel p,

$$\hat{R}_k(p) = \begin{cases} \hat{Y}(p), & \text{if } p \in R_k \\ (127, 127, 127), & \text{otherwise} \end{cases}$$

In this formulation, pixels outside the mask R_k are assigned a fixed gray value (127,127,127), which visually separates them from the region of interest. An illustration of this masking procedure is presented in Figure 4.15.

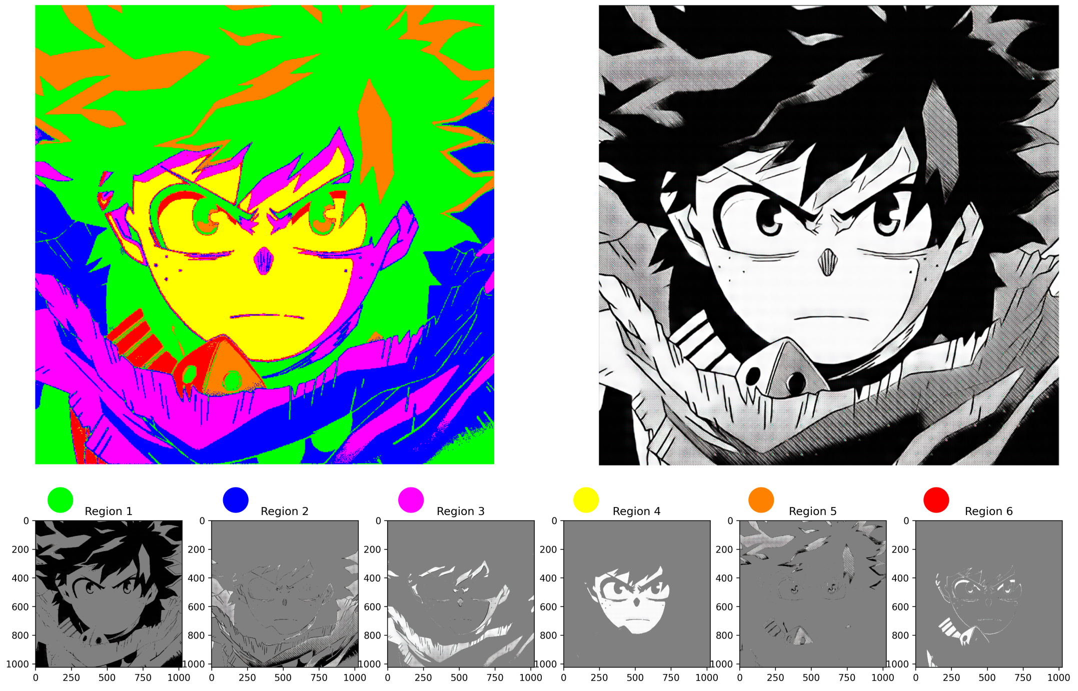


Figure 4.15: Region masks R_k (top left), preliminary output $\hat{Y}$ (top right), and masked images $\hat{R}_k$ (bottom).

For each masked image $\hat{R}_k$, the framework extracts a set of patches of size 32×32 centered on representative locations. To select these locations, we apply k-means clustering over the pixel coordinates inside $\hat{R}_k$, using a large and fixed number of centroids (in our experiments, 20). This produces, for each region, a set of candidate patches $\mathcal{P}_k = \{\pi_k^{(n)}\}_{n=1}^{20}$ for each region. These patches are then ordered in descending order based on their intersection with R_k, meaning that patches with a higher overlap appear earlier in the list. Figure 4.16 shows the extracted patch locations.

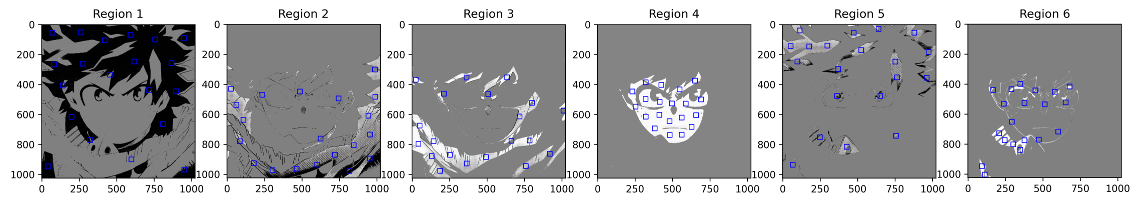


Figure 4.16: Patch locations selected by clustering for each masked region.

Next, the encoder selects the top $T = 3$ patches from $\mathcal{P}_k$, which better represent the region due to their high overlap and lower background noise, and classifies each one into one of the C screen pattern classes. Let $s_k^{(t)} \in \mathbb{R}^C$ be the predicted softmax vector for the t-th selected patch. The final screen pattern score for the masked region $\hat{R}_k$ is computed as:

$$S_k = \sum_{t=1}^{T} s_k^{(t)} \tag{4.12}$$

The screen pattern assigned to R_k is the one corresponding to the highest score in S_k. Figure 4.17 shows, for each region, the selected patches and the final screen pattern chosen.

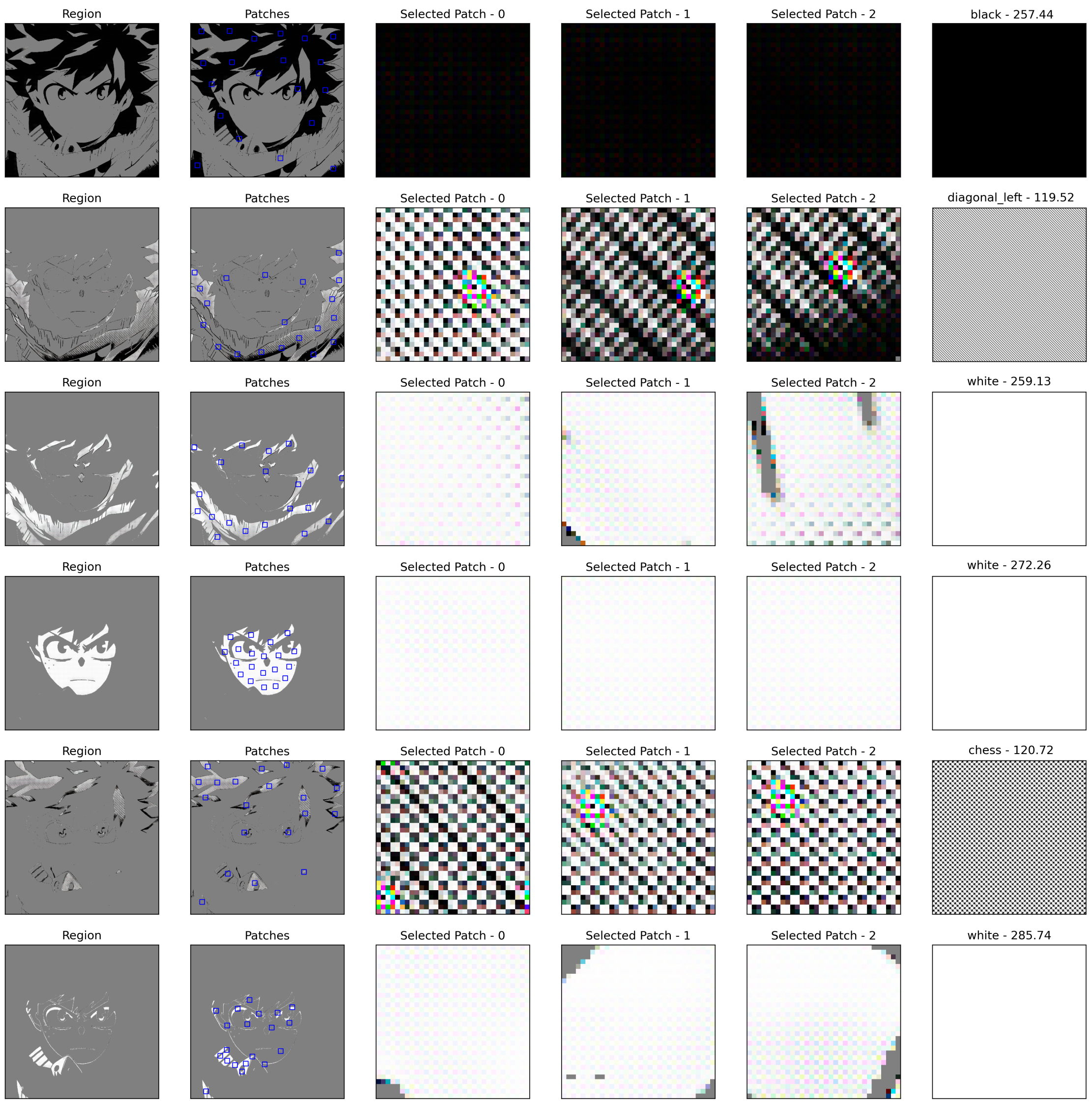


Figure 4.17: For each region, we show (from left to right): the region mask, the candidate patch locations (blue squares), the top 3 selected patches, and the screen pattern selected by the voting system.

In particular, for Region 1, the top patches indicate that the pattern plain black should be applied. For Region 2, the selected pattern is diagonal left. Regions 3, 4, and 6 all receive the plain white pattern. Region 5 is assigned the chess pattern. Figure 4.18 shows how each selected pattern is applied to its respective region while Figure 4.19 shows the final result.

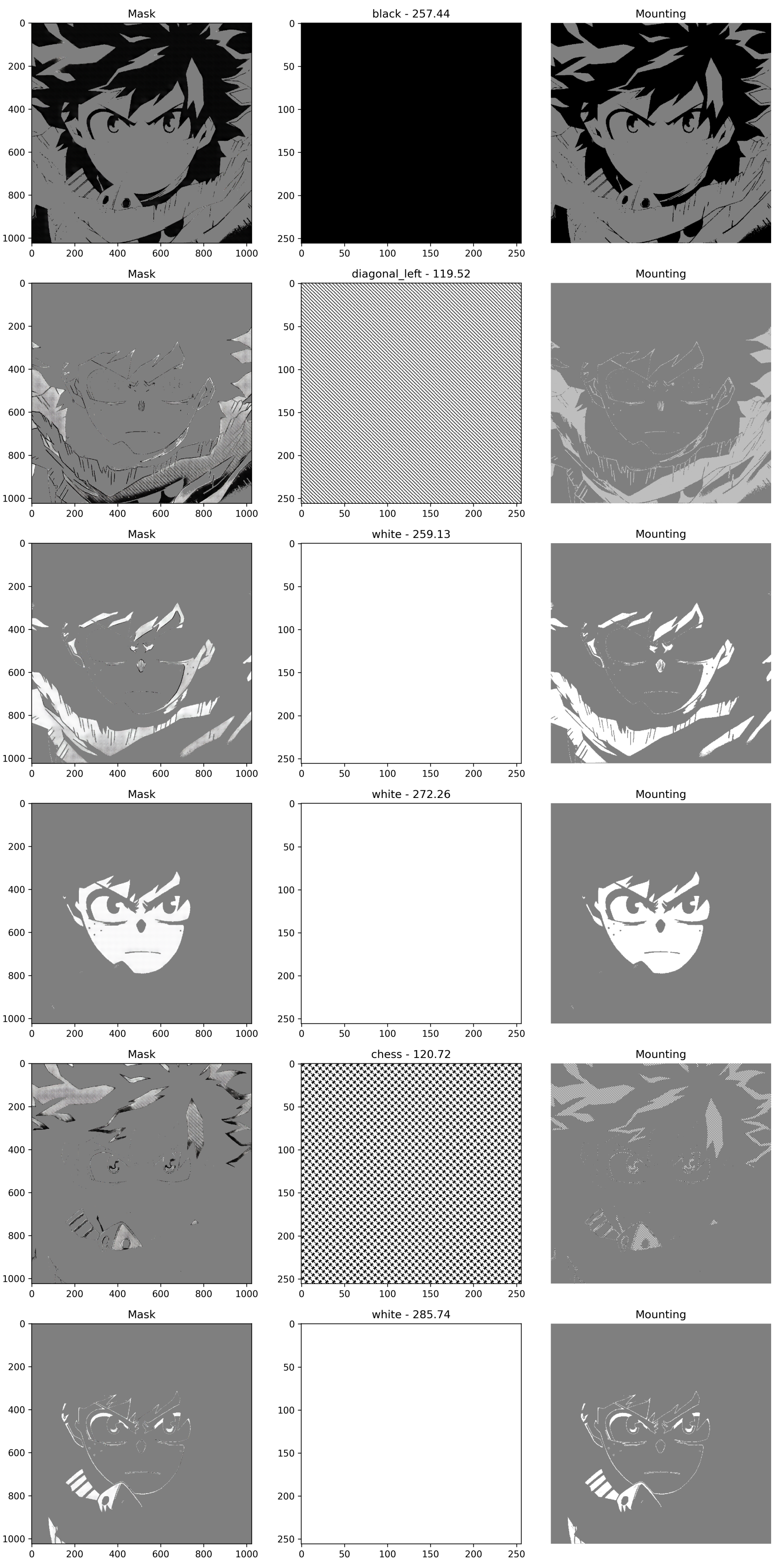


Figure 4.18: Application of selected screen patterns to each region. The image illustrates how the top-voted pattern for each region is mapped onto the corresponding masked area, based on representative patch locations.

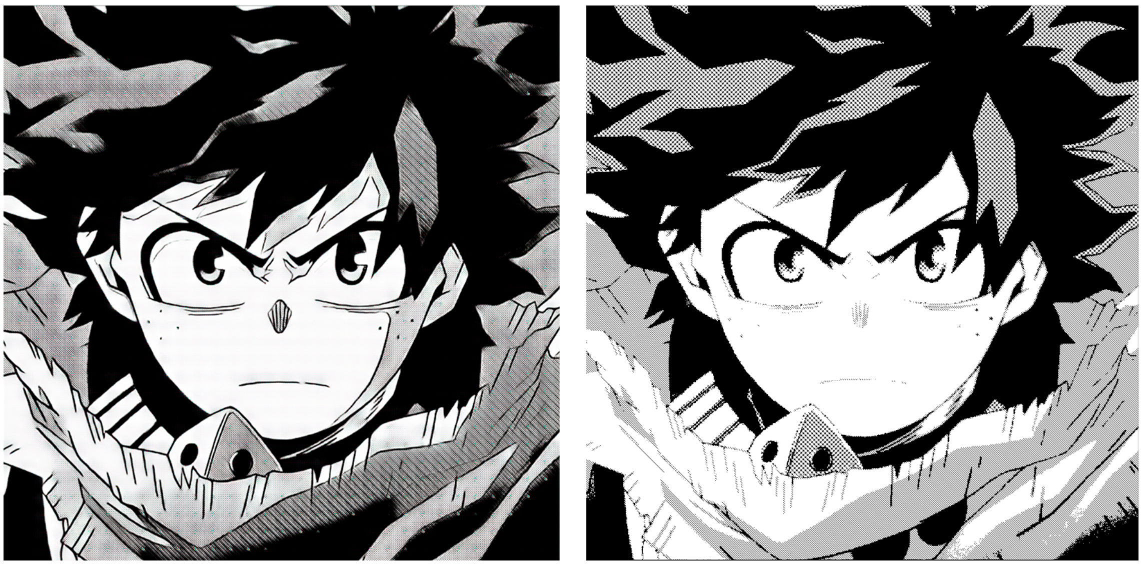


Figure 4.19: Final output after applying the selected screen patterns to each region based on the top-voted patches. The result combines all regions into a manga-style composition, with patterns reflecting the semantic context of each area.

The reader might have noticed that the screen pattern pasting procedure involves several parameters that affect both the quality and consistency of the final output. Essentially, each of these components helps define how screen patterns are applied. The number of regions controls how many separate areas will receive different patterns; the patch size sets the level of detail used to classify each region; and the voting method defines the criteria for the final pattern choice per region. Table 4.1 summarizes these parameters, which can be adjusted or replaced depending on specific goals.

Table 4.1: Configurable parameters in the screen pattern pasting procedure.

Component	**Description**
Region segmentation algorithm	k-means clustering applied over the RGB channels of the input image. Alternatives like Mean Shift are also viable.
Number of regions	Fixed to 6 in our implementation. Can be adjusted according to image complexity.
Patch size	32×32 pixels. Chosen to match the input size expected by the encoder E.
Patches per region	20 patches sampled per region. Helps capture intra-region variation.
Patch sampling strategy	k-means clustering over pixel positions to distribute samples evenly across each region.
Pattern voting method	Class probabilities are aggregated across T sampled patches using a summation strategy; $T = 3$.
Screen pattern classifier	Encoder E, trained on synthetic screen pattern dataset. May be replaced with any other classifier trained on the same target patterns.

4.2.2 The Structural Line Procedure

Structural lines play an important role in anime and cartoon art styles as they define the boundaries and contours of objects, characters, and their various components such as facial features, clothing, and props. However, not all lines in anime or cartoon images are structural lines. There may be additional lines, such as shading lines, decorative lines, or texture lines, which serve different purposes and convey specific artistic details.

General line detection methods [19, 113, 114] rely on detecting high contrast edges or changes in intensity values to identify lines. As anime images are usually composed of color blocks, a transition between adjacent color blocks that are not part of structural lines may lead to misdetection. Instances of such misclassifications are highlighted by the red circles in Figure 4.20 when applying Canny Edge Detection Technique [19].

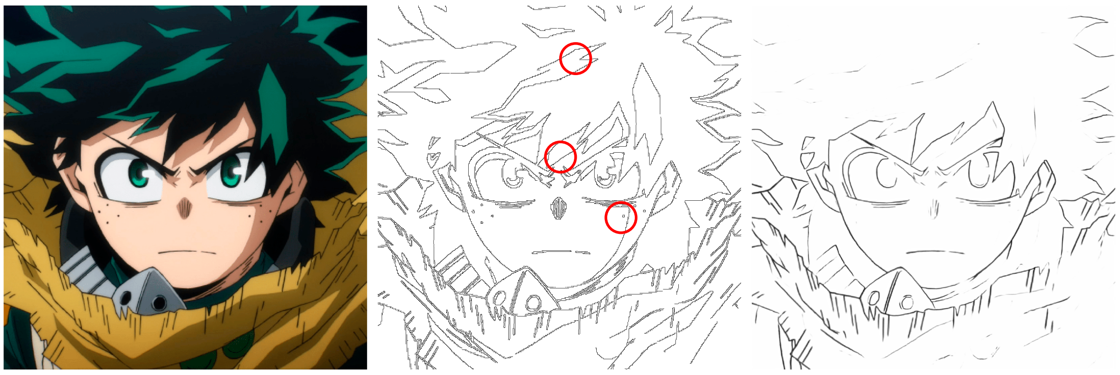


Figure 4.20: On the right, the input image used for comparison. In the middle, the result of the Canny Edge Detection Technique [19], with misclassified lines highlighted by red circles. On the left, structural lines extracted by our approach.

Alternative approaches such as deep learning-based methods or sketch simplification techniques can be used. These methods can learn the patterns and characteristics specific to anime images, allowing them to better capture structural lines. We apply the learning-based line extractor algorithm proposed by LI *et al.* [115]. Although this method is used to extract lines from manga images, we notice that it has shown reasonable results in extracting structural lines from colored images. We observe that the method maps the difficult boundaries (highlighted by the red circles in Figure 4.20) to high-intensity values, whereas true structural lines are mapped to low-intensity values.

Consider $\psi_{\text{line}}(X)$ as the procedure used to extract structural lines from the input image. We first extract the lines through the learning-based method [115], and then apply a thresholding step to remove noise. Specifically, pixels with intensity values above a threshold τ are converted to white.

$$\psi_{\text{line}}(X)_{i,j} = \begin{cases} 255, & \text{if } \omega(X(i,j)) > \tau, \\ X(i,j), & \text{otherwise}, \end{cases} \tag{4.13}$$

where X is the input image, $\omega(X)$ is the normalized line-response map (values in $[0, 1]$), and $X(i, j)$ denotes the pixel intensity at location (i, j).

Figure 4.21 illustrates the effect of the threshold parameter on the structural line extraction step. We adjusted the intensity threshold from 0.1 to 1.0 in increments of 0.1. Lower threshold values tend to remove unwanted noise and faint lines, but can also erase true structural lines. Higher values preserve more lines, but can include those that may not correspond to actual structure. In our experiments, we set $\tau = 0.7$.

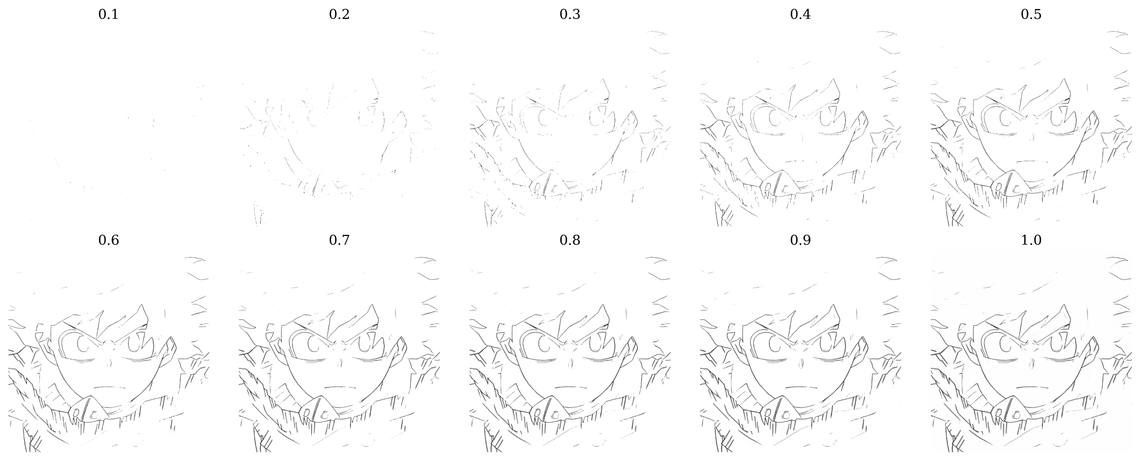


Figure 4.21: Effect of varying the threshold value from 0.1 to 1.0 (in steps of 0.1) on the structural line extraction.

After completing both steps, we invite the reader to revisit Figure 4.1, where the outputs of each component of the second stage—$\psi_{\text{pattern}}(\hat{Y}, X)$ and $\psi_{\text{line}}(X)$—are shown, along with the final result Y. Additionally, a complete overview of the framework is shown in Figure 4.22.

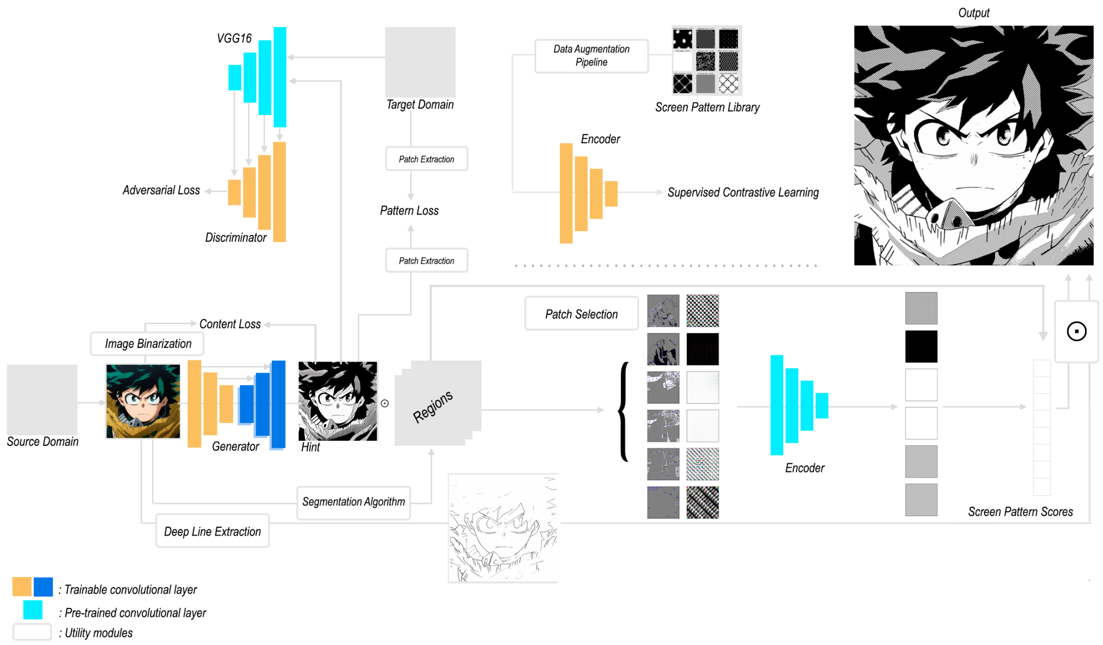


Figure 4.22: Two-stage anime-to-manga translation framework. First Stage generates a preliminary manga representation using a generator, a discriminator, and an encoder to classify screen patterns. Second Stage refines the preliminary output by segmenting the input image, applying screen patterns based on probabilities (regions and pattern classes), and preserving structural lines using a line extraction procedure.

Chapter 5

Experiments, Data and Architectures Details

This chapter describes the technical and experimental procedures used to evaluate the proposed method. It begins with the implementation details and training strategy, followed by a description of the datasets used to train each component of the framework. The chapter also presents the network architectures in more detail and the hyperparameter values adopted in the experiments.

5.1 Experimental Setup

5.1.1 Implementation and Hyperparameters

We implemented our framework in TensorFlow [58] and Python language. We aim to make the trained models used in our experiments available to facilitate evaluation of future methods. All experiments were performed on a NVIDIA Quadro P5000 GPU, which provides the parallel computing capability necessary for efficient training of convolutional networks. The networks were trained using the Adam [55] algorithm as the optimizer, with the learning rate and batch size set to 2×10^{-4} and 8, respectively. The development and experimentation were carried out on Paperspace[1], through a virtual machine with GPU support optimized for deep learning tasks. Table 5.1 summarizes the main hyperparameters used during training.

5.1.2 Training Procedure

Training GANs is known to be challenging, often affected by issues such as mode collapse, non-convergence, and instability [116]. Mode collapse occurs when the generator produces samples from only a few modes of the data distribution, leading

[1]`https://www.paperspace.com/`

to low diversity in the generated outputs. On the other hand, non-convergence and instability refer to the failure of the training process to reach a stable equilibrium. This typically happens when the generator and discriminator are imbalanced in their learning dynamics—e.g., if the discriminator becomes too strong, it provides vanishing gradients to the generator, preventing effective learning.

To mitigate these GAN training problems, several solutions have been proposed over time. These solutions are essentially based on three main techniques: introducing new loss functions [117], reengineering the networks architectures [118], and using alternative optimization algorithms [62] or training techniques [91].

In our specific case, mode collapse, non-convergence, and instability could lead to loss of content structure or repetitive screen pattern assignments in the manga outputs. To circumvent these issues, the binary content loss and the pattern consistency loss help maintain stable training. In addition, the custom discriminator architecture provides richer and more informative gradients to the generator throughout training. Finally, we propose a training procedure designed to better control the anime-to-manga translation process.

Given these circumstances, we propose a sequential initialization of the α, β, and γ hyper-parameters. First, we would like to enforce effective content preservation before proceeding with stylization. Then, since the pattern consistency promotes the generation of consistent patterns, it is important to consider this aspect only after some stylization has been applied.

With that in mind, we first pretrain the network to generate binary content representations, considering only the binary content loss, setting $\alpha = 1$ and $\beta = \gamma = 0$ during the first 20 epochs. We then proceed to train the network for the next 20 epochs, considering two losses: the adversarial loss and the binary content loss, setting $\alpha = 1$, $\beta = 10$, and $\gamma = 0$. We want to apply the style only after preserving the input content. Finally, all three loss functions are used in the training process until the end, which consists of a total of 180 epochs, after which no further improvements were observed. We set the parameters as $\alpha = 1$, $\beta = 10$, and $\gamma = 5$.

Although the evaluation of GANs is still mostly qualitative — based on human visual inspection — we present here the loss function graph to illustrate the training dynamics. The Figures 5.1–5.4 present the values of the generator's adversarial loss (along with the corresponding discriminator loss), the binary content loss, and the pattern consistency loss over the 180 training epochs.

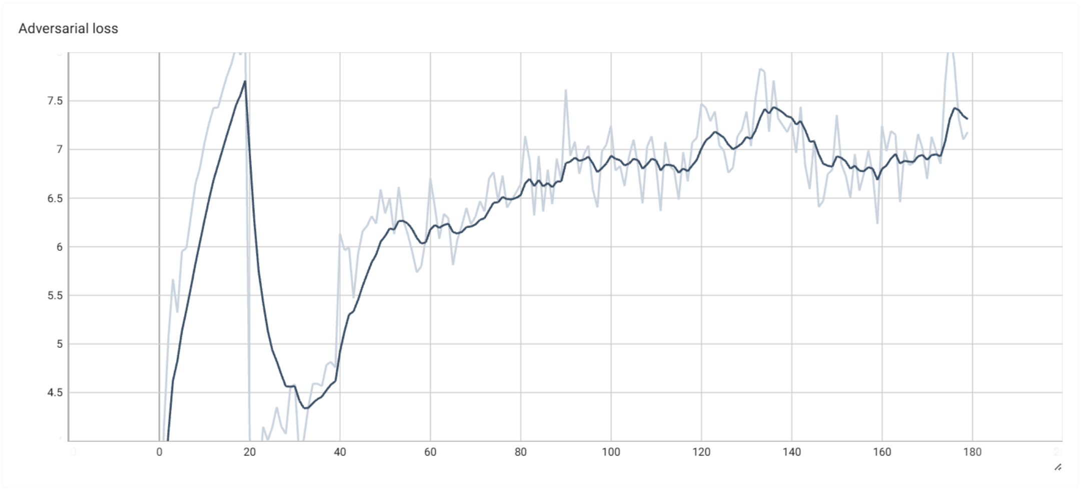


Figure 5.1: Adversarial loss function.

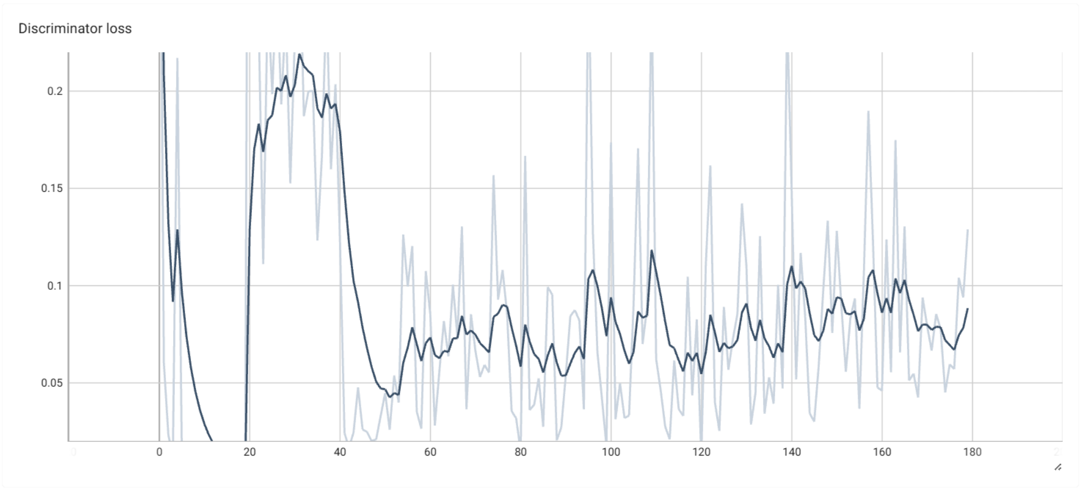


Figure 5.2: Discriminator loss function.

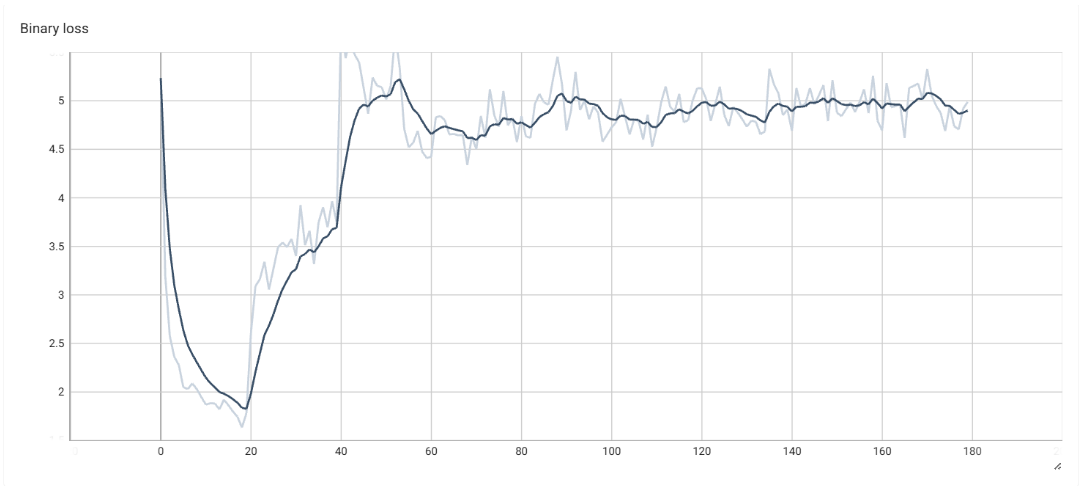


Figure 5.3: Binary Content loss function.

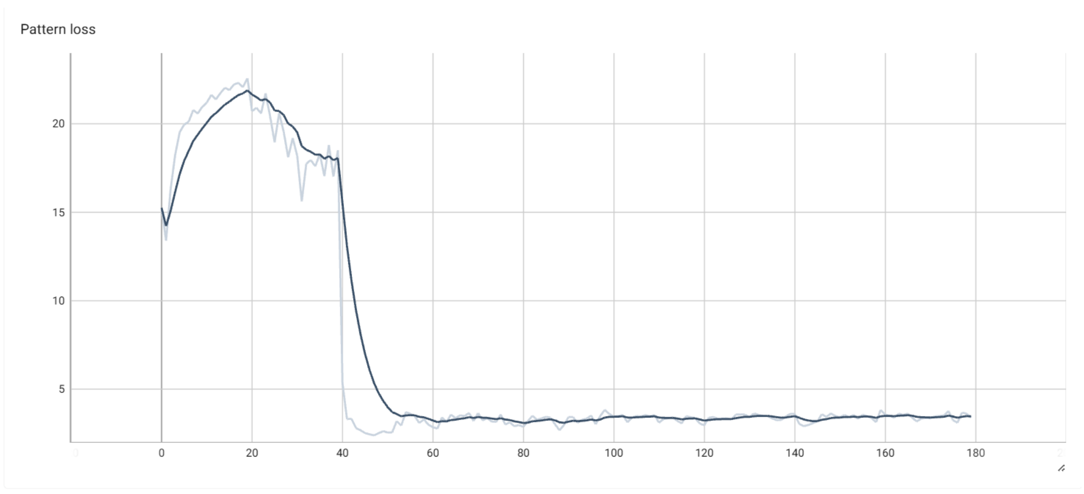


Figure 5.4: Pattern loss function.

In the first 20 epochs, only the binary content loss $\mathcal{L}_{\text{content}}$ was used. As shown in Figure 5.3, its value drops quickly and becomes stable, which means the model learned to extract the binary content structure of the input images before starting the stylization.

From epoch 20 to epoch 40, the adversarial training occurs, as shown in Figure 5.1 and Figure 5.2, respectively. The generator and discriminator losses stayed in balance, with no sign that one was overpowering the other. If the discriminator loss dropped to zero, it would indicate that it was easily detecting fake images. If the generator loss became too low, it could suggest overfitting or lack of diversity. In this case, both losses remained within a normal range, indicating that the GAN did not collapse or overfit.

After epoch 40, the pattern loss $\mathcal{L}_{\text{pattern}}$ was added. Figure 5.4 shows a clear drop in this loss, which means the generator started to match the expected pattern distribution. Thereafter, with all loss terms enabled, training remains stable.

An ablation study was conducted to evaluate the importance of the training procedure adopted. Without it, the GAN suffers from mode collapse, producing poor results. See Figure 5.5 for a comparison.

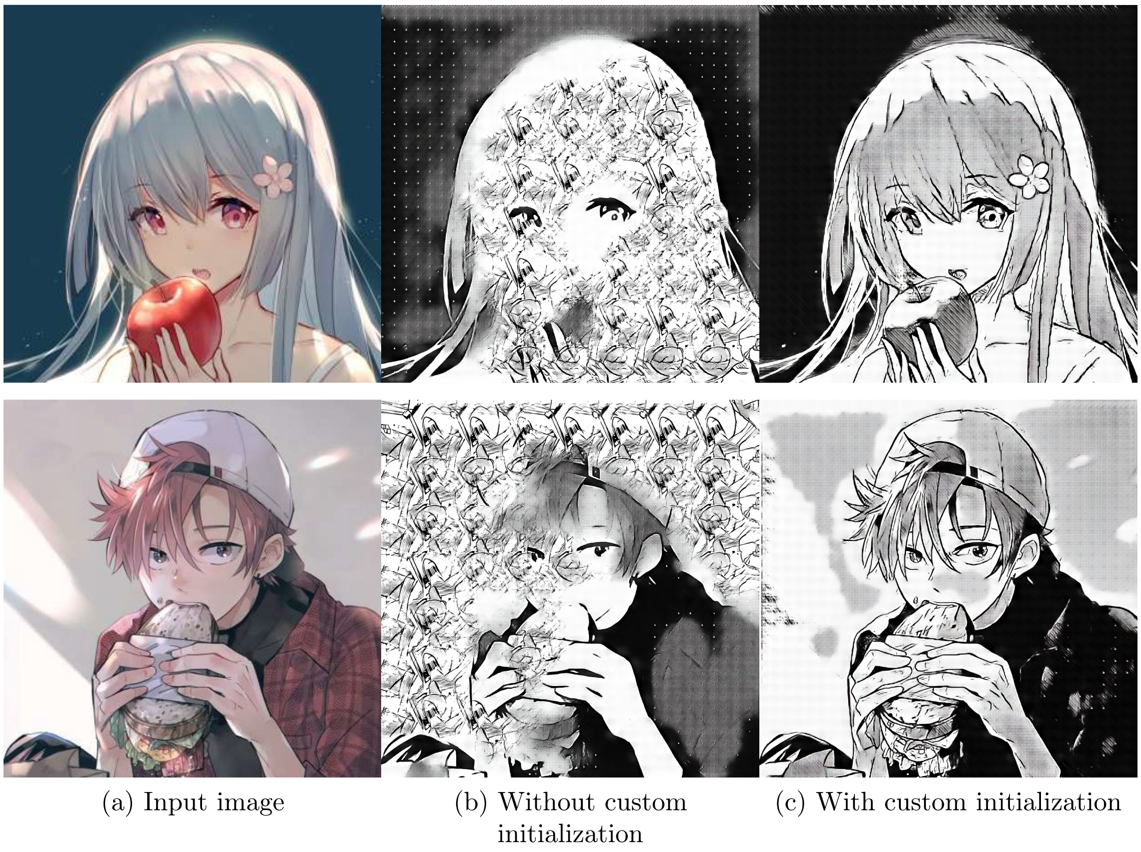

(a) Input image
(b) Without custom initialization
(c) With custom initialization

Figure 5.5: Ablation study illustrating the effect of the training procedure on the intermediate result produced by the first stage: (**a**) Input image. (**b**) Output without custom initialization. (**c**) Output with custom initialization.

Table 5.1: Summary of hyperparameters used in the training process.

Parameter	**Value**
Learning Rate	2×10^{-4}
Batch Size	8
Optimizer	Adam [55]
Number of Epochs	180
α (Adversarial Loss Weight)	1
β (Content Loss Weight)	10
γ (Pattern Consistency Loss Weight)	5
Loss Scheduling	Progressive, based on stability

5.1.3 Dataset

The training process of our framework is organized according to the three main components of the architecture: the generator G, the discriminator D, and the encoder E. The generator and discriminator are jointly trained following a standard adversarial learning scheme, where the generator learns to produce preliminary manga representations from anime inputs, and the discriminator evaluates the realism of these outputs by comparing them to true manga samples. In contrast, the encoder E, responsible for classifying screen patterns in the second stage of the framework, is trained independently using a augmented dataset of screen patterns. The following subsections provide details about the datasets used for training these components, along with relevant preprocessing and augmentation strategies.

The generator and discriminator are trained using independent datasets composed of anime and manga images. On the other hand, the encoder is trained on a different augmented dataset built specifically from screen patterns. We aim to make it publicly available to support reproducibility.

Generator and Discriminator To obtain the training data as the source domain, a set of anime images was collected from the Danbooru [3] dataset, a widely recognized repository of anime illustrations frequently used as a benchmark in data-driven tasks. We extracted and selected 1957 images, cropping them to a size of 286×286 pixels. Since the dataset is compiled from millions of images, we manually selected a subset to ensure a rich variety of content. This selection includes everything from individual characters to scenes with multiple characters and diverse landscapes, as shown in Figure 5.6.

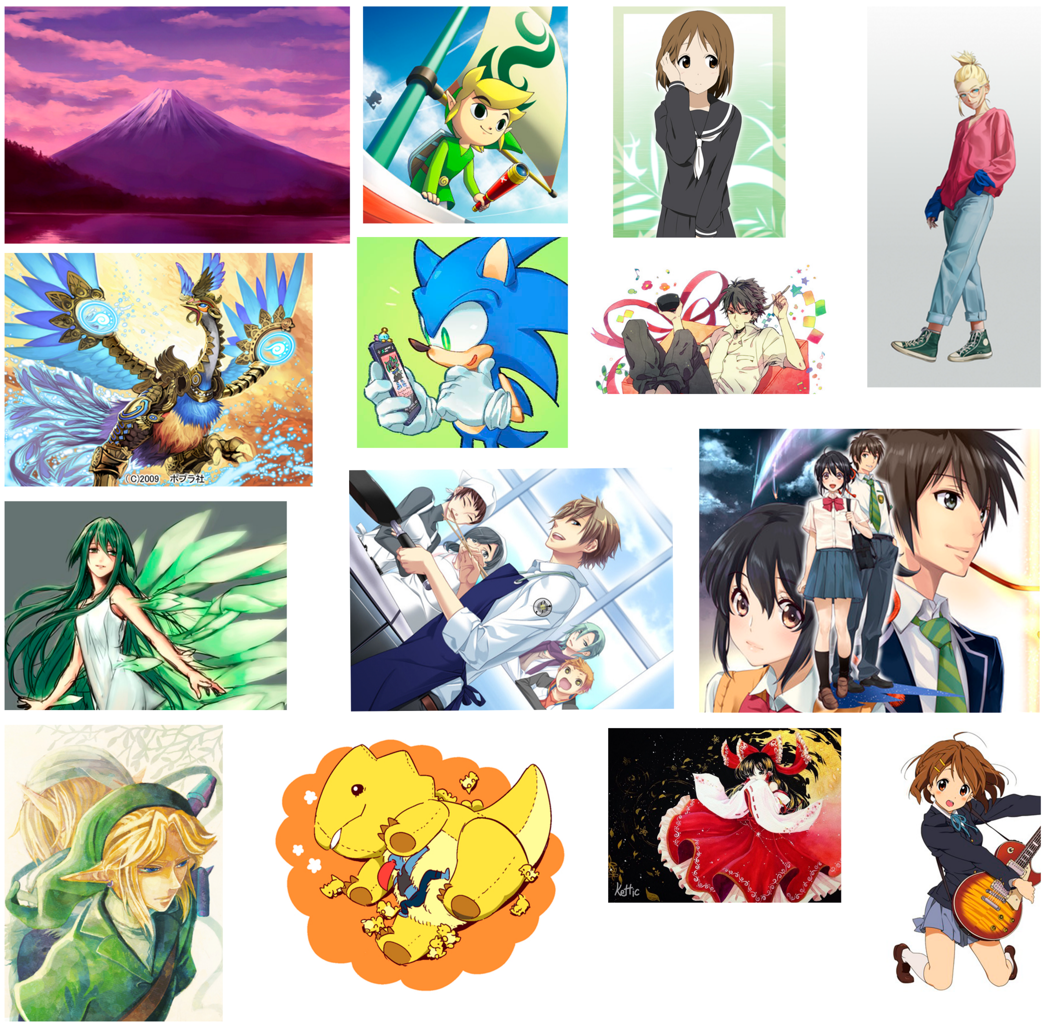


Figure 5.6: Danbooru dataset samples.

For the target domain, we use manga volumes from the Manga109 dataset [119], a public collection of professionally drawn manga commonly used in vision-based research. Examples of this dataset are shown in Figure 5.7. It is important to note that manga pages typically consist of multiple panels, lines separating them, speech balloons, and abundant text. To ensure the quality of our training data, we conducted a preprocessing step where we sampled individual manga panels and cropped the images to a size of 286 $\times$ 286 pixels. We then only included crops without any text characters or separating lines. This preprocessing is important because, otherwise, the model would learn to reproduce these artifacts. Finally, the total number of samples in the target domain dataset was 531.

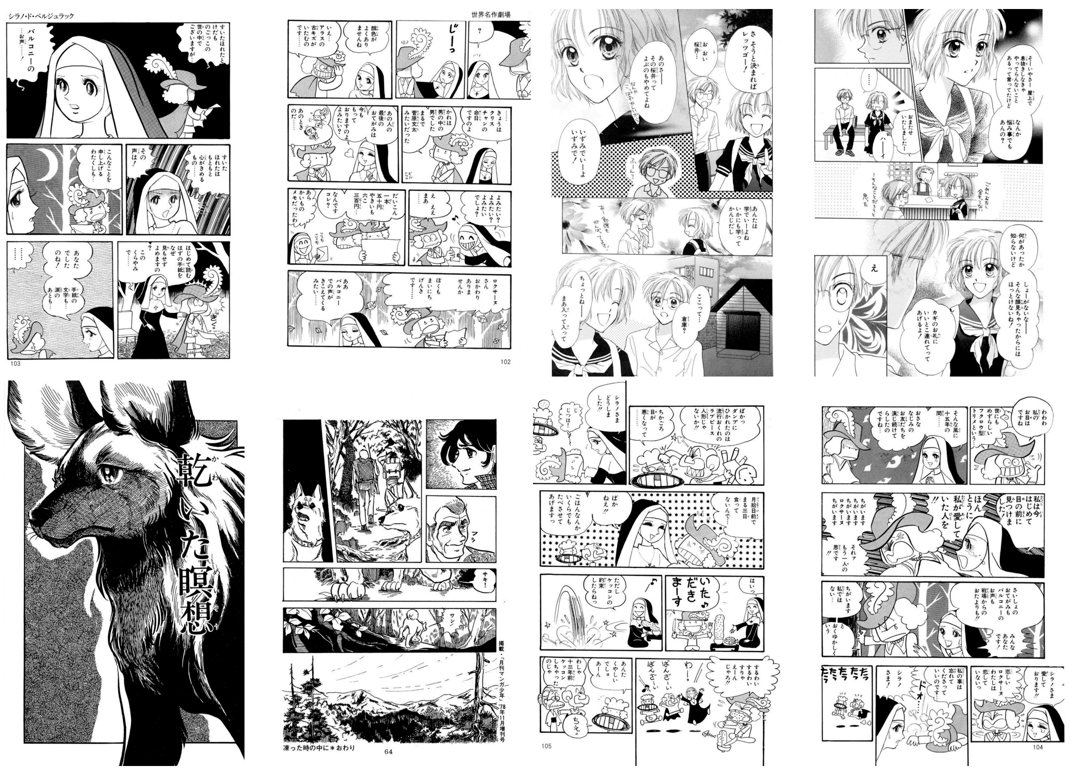


Figure 5.7: Manga19 dataset samples.

During training, we apply a set of data augmentation techniques to enhance the robustness of the model against overfitting and to introduce variability in the training data. The applied operations include random rotations, zoom-based cropping, and horizontal/vertical flipping.

Encoder The dataset used to train the encoder was created from the widely-used DELETER screen pattern standard [23]. We selected a set of 10 commonly used screen patterns, each representing a distinct class. Examples of screen pattern classes include plain black, chess-like, checked squares, diagonal lines (left and right), sparse dots, horizontal lines, tilt, vertical lines, and plain white, as shown in Figure 5.8. All of these patterns are present in the manga books from the Manga109 [119] dataset.

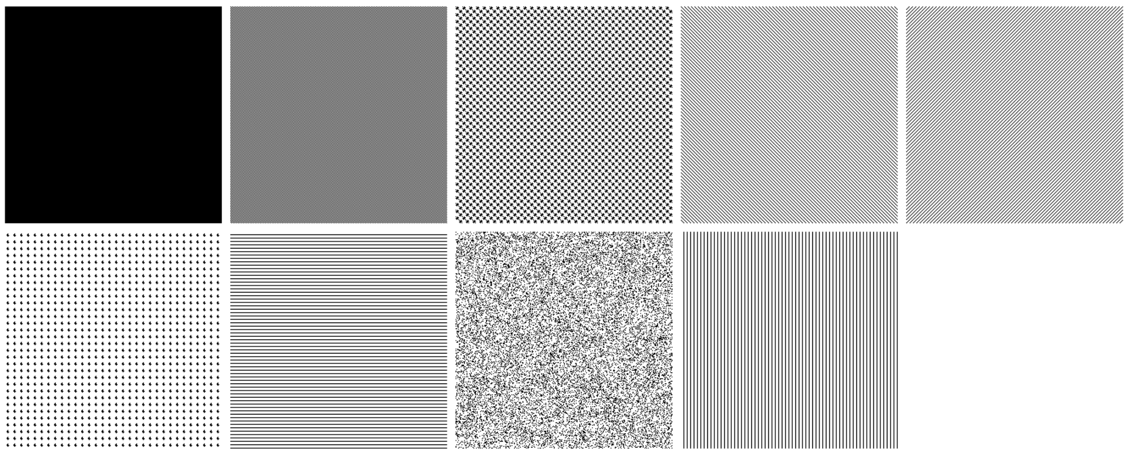


Figure 5.8: Examples of screen patterns demonstrating their bitonal nature and spatial regularity.

For training purposes, we applied a data-augmentation pipeline to increase per-class samples and to simulate distortions introduced by the generator during translation. The pipeline performs random brightness adjustments (max change 0.005) and contrast scaling (range $[0.5, 2.0]$); Gaussian blur (kernel 3×3, $\sigma = 0.5$) with probability 0.9; random crops retaining 90% of the spatial dimensions followed by resizing to the original size; small rotations (uniform in $[-0.05, 0.05]$ rad) and shear (uniform in $[-0.1, 0.1]$). Structural lines extracted from the anime dataset are overlaid via a blend operation with probability 0.99 to simulate noise. Finally, images are cropped to 32×32 pixels. The pipeline yields 80,000 training images and 20,000 test images, evenly distributed across 10 classes. Representative samples (before cropping) are shown below.

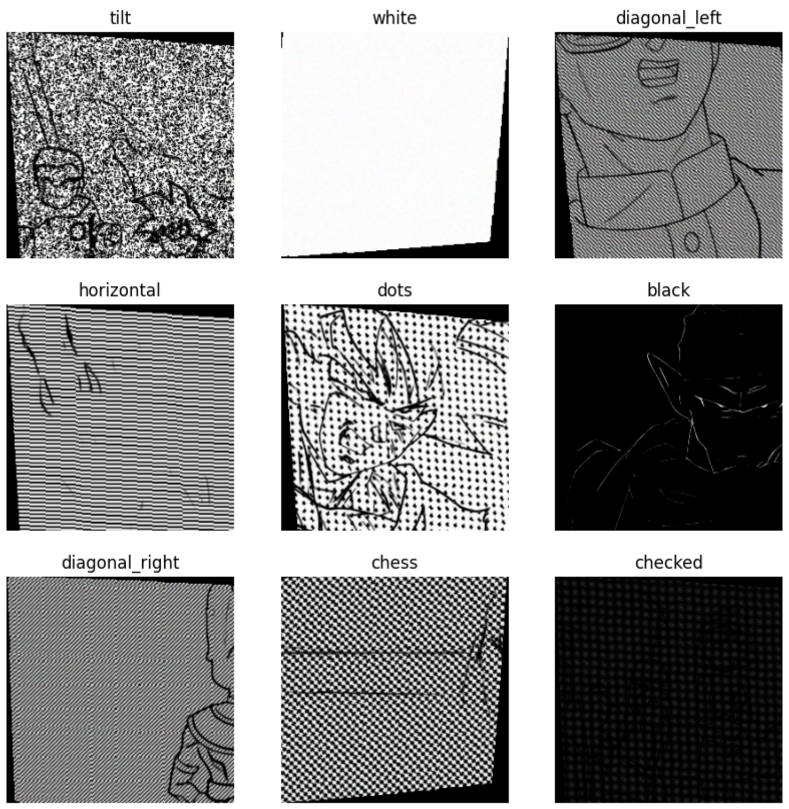


Figure 5.9: Samples from the data augmentation pipeline. The images shown were captured before cropping.

5.1.4 Architectures Details

This section presents the detailed architectures of the main networks that compose the proposed framework. Tables 5.2 and 5.3 summarize the layers, output shapes, number of parameters, and connectivity for the generator and discriminator, respectively, complementing the visual diagrams shown in Figures 4.5 and 4.7. As for the encoder, we adopt the standard ResNet50 architecture [13] without modifications.

Table 5.2: Architecture summary of the generator model.

Layer (type)	**Output Shape**	**# Parameters**	**Connected to**
input_1 (InputLayer)	(256,256,3)	0	–
sequential (Sequential)	(128,128,64)	3,072	input_1
sequential_1 (Sequential)	(64,64,128)	131,584	sequential
sequential_2 (Sequential)	(32,32,256)	525,312	sequential_1

Continues on next page

Layer (type)	**Output Shape**	**# Parameters**	**Connected to**
sequential_3 (Sequential)	(16,16,512)	2,099,200	sequential_2
sequential_4 (Sequential)	(8,8,512)	4,196,352	sequential_3
sequential_5 (Sequential)	(4,4,512)	4,196,352	sequential_4
sequential_6 (Sequential)	(2,2,512)	4,196,352	sequential_5
sequential_7 (Sequential)	(1,1,512)	4,196,352	sequential_6
sequential_8 (Sequential)	(2,2,512)	4,196,352	sequential_7
concatenate (Concatenate)	(2,2,1024)	0	sequential_8, sequential_6
sequential_9 (Sequential)	(4,4,512)	8,390,656	concatenate
concatenate_1 (Concatenate)	(4,4,1024)	0	sequential_9, sequential_5
sequential_10 (Sequential)	(8,8,512)	8,390,656	concatenate_1
concatenate_2 (Concatenate)	(8,8,1024)	0	sequential_10, sequential_4
sequential_11 (Sequential)	(16,16,512)	8,390,656	concatenate_2
concatenate_3 (Concatenate)	(16,16,1024)	0	sequential_11, sequential_3
sequential_12 (Sequential)	(32,32,256)	4,195,328	concatenate_3
concatenate_4 (Concatenate)	(32,32,512)	0	sequential_12, sequential_2
sequential_13 (Sequential)	(64,64,128)	1,049,088	concatenate_4
concatenate_5 (Concatenate)	(64,64,256)	0	sequential_13, sequential_1
sequential_14 (Sequential)	(128,128,64)	262,400	concatenate_5

Continues on next page

Layer (type)	**Output Shape**	**# Parameters**	**Connected to**
concatenate_6 (Concatenate)	(128,128,128)	0	sequential_14, sequential
conv2d_-transpose_7 (Conv2DTranspose)	(256,256,3)	6,147	concatenate_6
Total Parameters		**54,425,859**	Trainable: 54,414,979 Non-trainable: 10,880

Table 5.3: Architecture summary of the discriminator model.

Layer (type)	**Output Shape**	**# Parameters**	**Connected to**
conv1_2 (InputLayer)	(256,256,64)	0	–
sequential_16 (Sequential)	(128,128,64)	65,536	conv1_2
conv2_2 (InputLayer)	(128,128,128)	0	–
concatenate_7 (Concatenate)	(128,128,192)	0	sequential_16, conv2_2
sequential_17 (Sequential)	(64,64,128)	393,728	concatenate_7
conv3_3 (InputLayer)	(64,64,256)	0	–
concatenate_8 (Concatenate)	(64,64,384)	0	sequential_17, conv3_3
sequential_18 (Sequential)	(32,32,256)	1,573,888	concatenate_8
conv4_3 (InputLayer)	(32,32,512)	0	–
concatenate_9 (Concatenate)	(32,32,768)	0	sequential_18, conv4_3
zero_padding2d	(34,34,768)	0	concatenate_9
conv2d_11 (Conv2D)	(31,31,512)	6,291,456	zero_padding2d
batch_norm_16 (BatchNorm)	(31,31,512)	2,048	conv2d_11
leaky_relu_11 (LeakyReLU)	(31,31,512)	0	batch_norm_16
zero_padding2d_1	(33,33,512)	0	leaky_relu_11
conv2d_12 (Conv2D)	(30,30,1)	8,193	zero_padding2d_1
Total Parameters		**8,334,849**	Trainable: 8,333,057 Non-trainable: 1,792

Chapter 6

Results

In this chapter, we present a comparative evaluation of the proposed method. We begin by presenting the baseline approaches used for comparison. Next, we report the results of a qualitative analysis and a user study. We then discuss the main limitations observed. Finally, we consolidate some of the experimental results and visual outputs.

6.1 Previous Methods and Evaluation Metrics

We compare our method with three algorithms that represent the state-of-the-art in the illustration-to-manga translation problem: ScreenVAE [20], a screen pattern encoder to enable the transfer between color illustrations and manga, Mimicking [21], a supervised approach that uses screen pattern segmentation maps manually created to mimic the manga creation workflow, and Sketch2Manga [22] (S2M), which uses a diffusion model to generate manga images with shaded high-frequency screen patterns.

6.1.1 Qualitative Comparison

The qualitative comparison in Figure 6.1 illustrates the results of different methods. The input images (Figure 6.1-(a)), displayed in the first column, serve as baseline. The grayscale conversion (Figure 6.1-(b)) reduces the image to shades of gray, but fails to reproduce the characteristic screen patterns and structural details of manga.

For the ScreenVAE approach (Figure 6.1-(c)), we employed the model weights provided by the authors without additional training on our own dataset, since the process requires a high-resolution training set. While the ScreenVAE performed well in maintaining diverse screen patterning, some noise and artifacts are visible, particularly in the shaded regions.

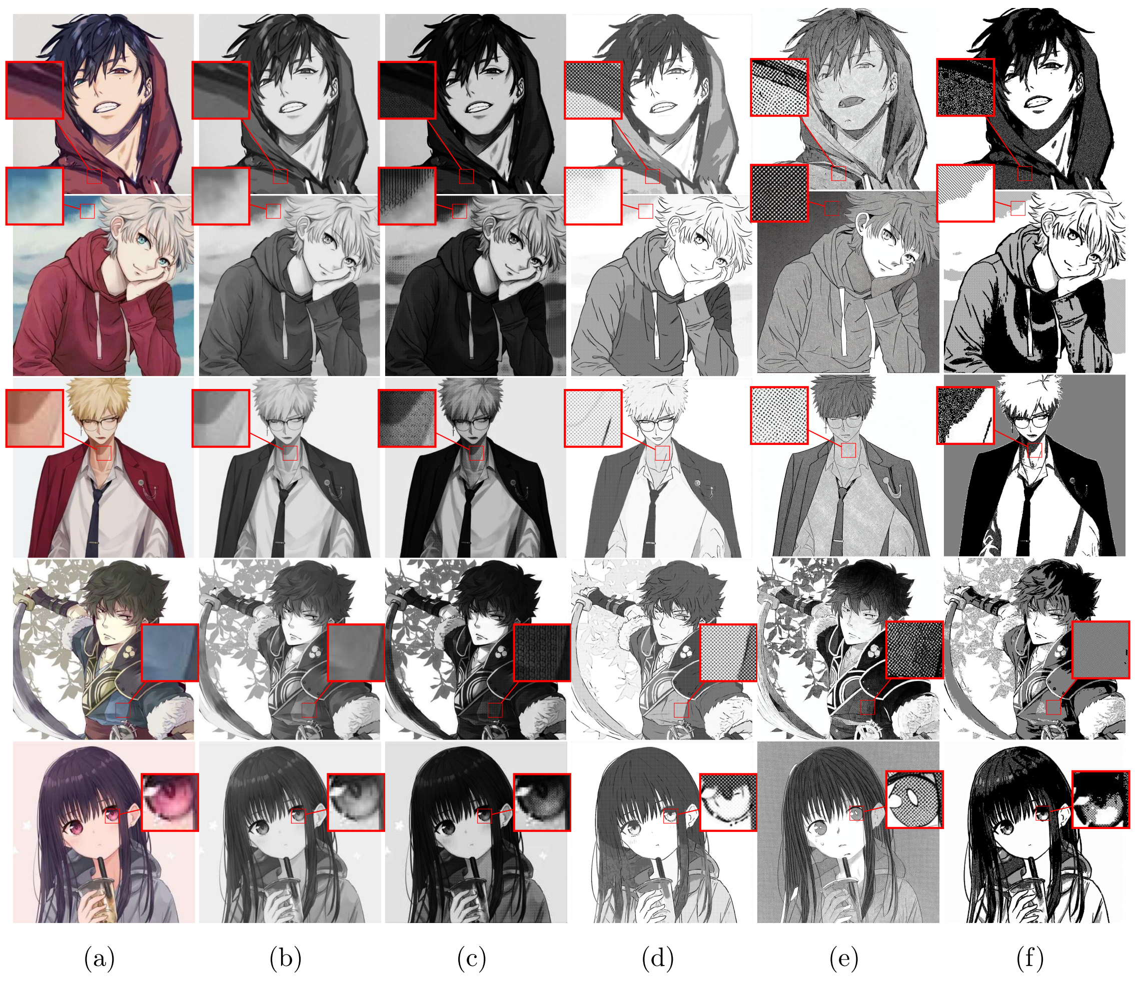


Figure 6.1: Qualitative comparison of results generated by our method against alternative anime-to-manga translation approaches: (**a**) Input image. (**b**) Greyscale. (**c**) ScreenVAE [20]. (**d**) Mimicking [21]. (**e**) S2M [22]. (**f**) Our method. Zoom-in regions were selected to show differences in screen pattern application and shading.

As shown in Figure 6.1-(d), the Mimicking approach produces high-quality results. It generates clean and professional manga outputs with well-placed screen pattern. However, as a supervised method, it relies on a manually prepared dataset containing 1502 tuples, including illustrations, line drawing maps, regular screen pattern segmentation maps, and irregular texture masks, all at a resolution of 1024 pixels. This dependency restricts the method to the 8 screen pattern types provided in their dataset, which are predominantly variations of a single class, "dots", with different intensities; zoom in to the details. Adding a new screen pattern would require the creation of a new dataset with over 1000 manually annotated samples, making the process both time-consuming and resource-intensive.

The S2M approach (Figure 6.1-(e)) generates high-quality manga with shaded, high-frequency screen patterns. However, this method relies on a diffusion model, which is computationally expensive and requires both an extensive high-resolution training dataset and high-performance hardware to produce a single result. Additionally, it does not maintain a consistent regular distribution of screen patterns and is limited to a single pattern style too, as can be observed by zooming into the results.

In contrast, our proposed method (Figure 6.1-(e)) produces results with well-positioned and diverse screen patterns. Unlike the other approaches, which apply a single pattern style throughout the image, our method demonstrates flexibility in applying multiple patterns across different regions.

Additionally, our framework allows for easy adaptation to new screen patterns. By simply expanding the screen pattern set and retraining the models, results with additional patterns can be generated. Our method is based on a GAN framework that does not require an extensive dataset, high-resolution images, or paired image data, making it a practical and accessible approach for generating high-quality manga images with customized screen patterns.

6.1.2 User Study

The user study evaluates on the synthetic manga images, where 20 participants, avid manga readers, were asked to rate a total of 100 images based on three criteria: screen pattern diversity, screen pattern regularity, and overall manga quality. Ratings were given on a scale from 1 (poor) to 5 (excellent). For the images, 25 samples were randomly selected from the Danbooru dataset [3]. Each sample was processed using the four approaches—ScreenVAE, Mimicking, S2M, and our method—resulting in a total of 100 generated images.

The results are shown in Table 6.1. While Mimicking was rated the highest in overall manga quality, with our method following closely, our approach performed

Table 6.1: Average user ratings for each method. Top score for each criterion is marked in blue.

Method	**Screen Pattern Diversity**	**Screen Pattern Regularity**	**Overall Manga Quality**
Our	4.33 ± 0.70	3.97 ± 0.97	4.08 ± 0.87
Mimicking	3.14 ± 1.36	3.83 ± 1.06	4.31 ± 0.71
ScreenVAE	3.95 ± 0.94	3.21 ± 1.22	3.09 ± 1.29
S2M	3.07 ± 1.32	3.54 ± 1.17	3.93 ± 1.02

better in screen pattern diversity and screen pattern regularity. ScreenVAE also performed well in screen pattern diversity, while Mimicking and S2M ranked lower in this category. These results indicate that our method produces varied and consistent screen patterns, while maintaining competitive overall quality compared to state-of-the-art approaches.

6.2 Limitations

Highly complex or irregular screen patterns, usually associate with some kind of visual effect, could prove more challenging for our framework, since classification occurs at the patch level and may miss global context. Observe Figure 6.2.

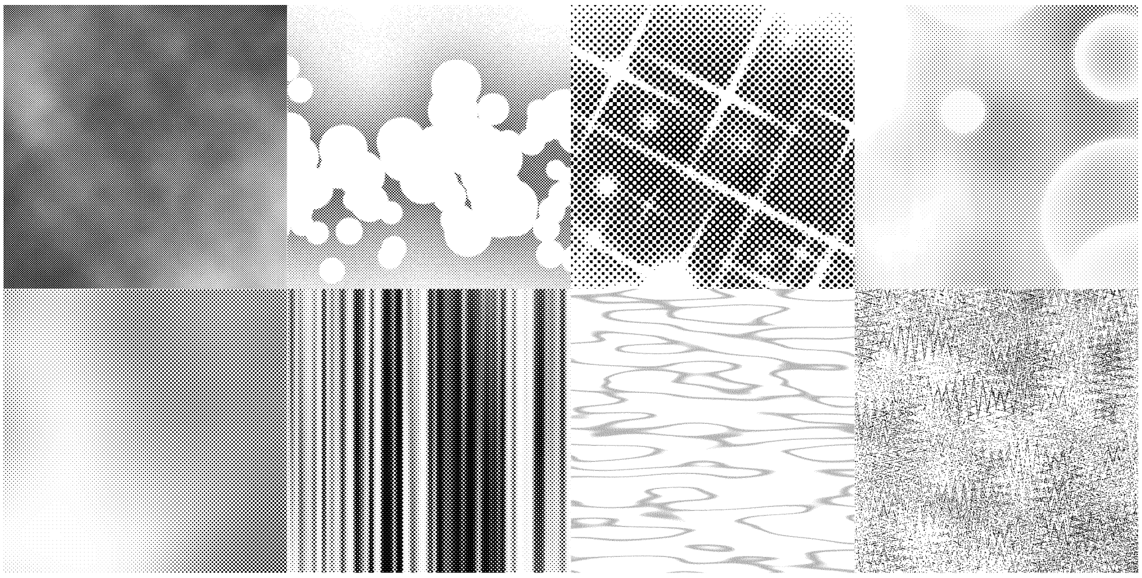


Figure 6.2: Examples from the DELETER screen pattern standard [23] with complex and irregular screen patterns. These cases can be a challenge for the classification step.

Additionally, due to the clustering nature of the segmentation step, if neighboring regions end up receiving the same screen pattern, elements can visually merge.

Figure 6.3 illustrates such a case, where the cloud and the background were placed in different regions but both received the same plain white pattern.

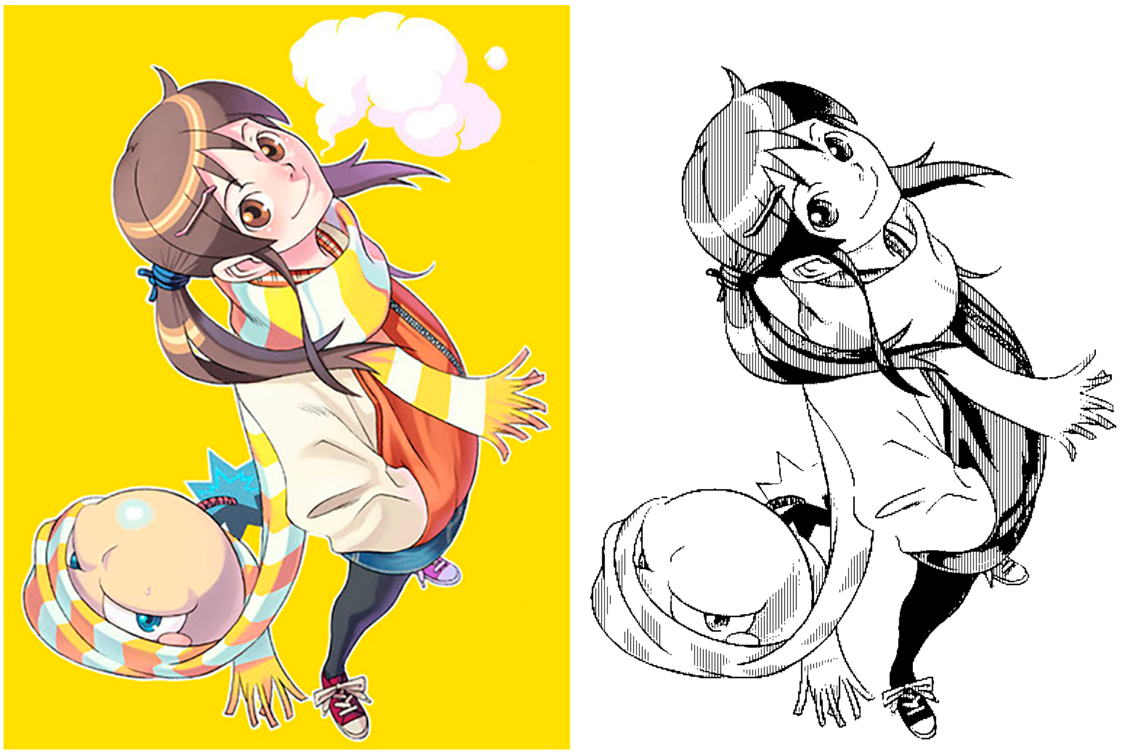


Figure 6.3: Visual merging of cloud and background due to identical screen patterns.

Another example is shown in Figure 6.4, where visual merging of the ears and hat occurs due to uniform pattern assignment. Both the background and the hat received the same plain white screen pattern, and no structural lines were recognized to mark a boundary between these regions.

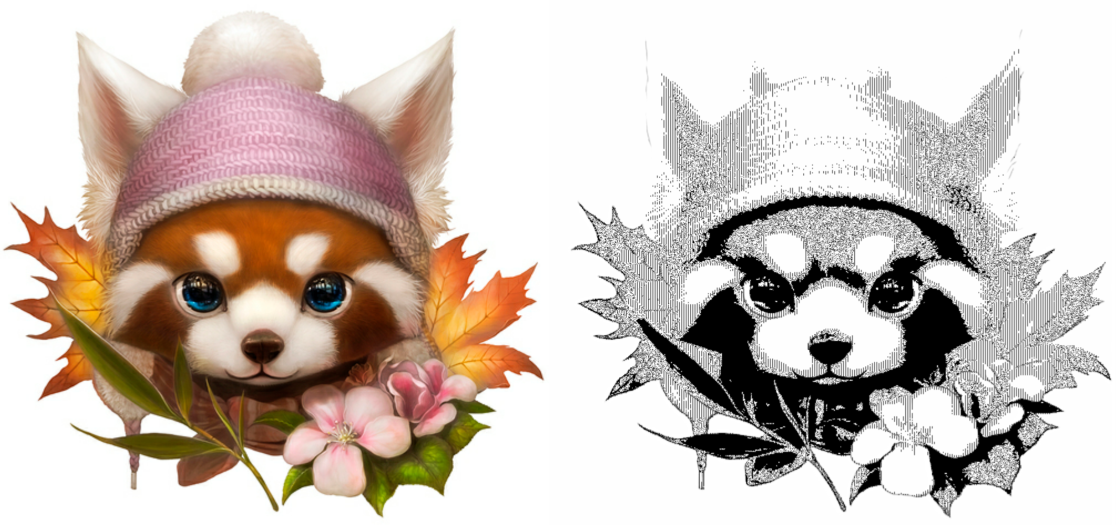


Figure 6.4: Visual merging of the background and hat caused by uniform pattern assignment.

Although this may occur, our framework allows users to control these effects by adjusting the patch size and manually selecting screen patterns based on the computed scores for each region. This helps prevent neighboring regions from receiving the same screen pattern. A second strategy to reduce this effect involves adjusting the threshold parameter used in the line extraction procedure, which influences the separation between regions. Since this threshold determines which pixels are preserved as structural lines, users can tune it to retain more line details when needed. Refer to Subsection 4.2.2 for details on the structural line extraction process.

6.3 Results

We constructed a synthetic dataset comprising paired anime and manga-style images using our two-stage framework. The anime samples were sourced mainly from the Danbooru dataset [3]. Additionally, we created some outputs from famous anime frames and from digital art generated by the author of this thesis.

Some of these results are shown throughout this work. Figure 1.5 illustrates representative outputs obtained with the final version of our model. Further examples are included in the qualitative comparison (Figure 6.1) and ablation study (Figure 5.5). A broader set of generated samples is presented here, including visualizations for various combinations of screen patterns, input conditions, and content types. Observe the following figures.

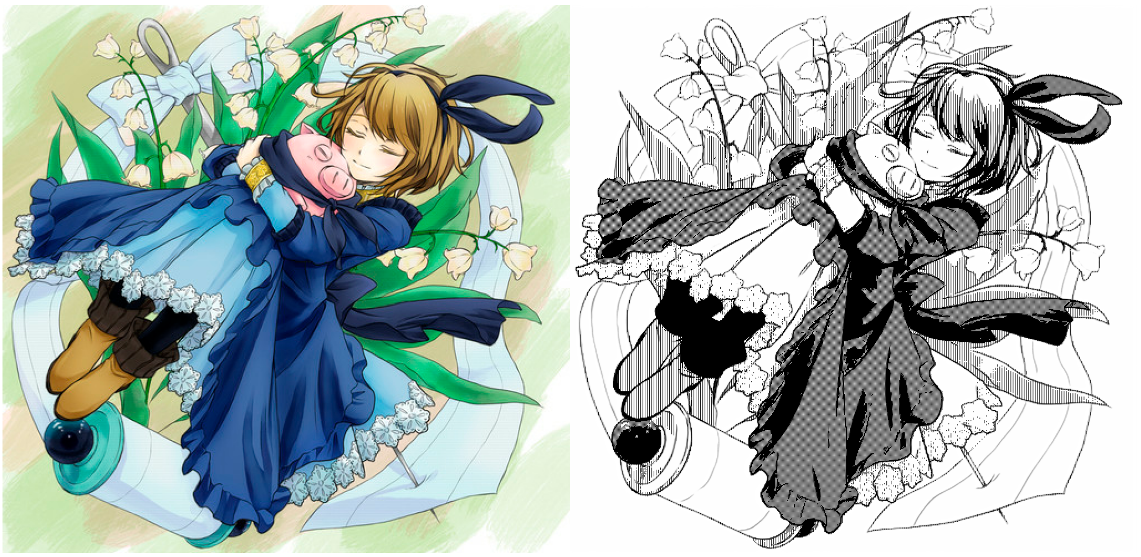


Figure 6.5: The original input is shown on the left, while the result is displayed on the right. Zoom in to see the details.

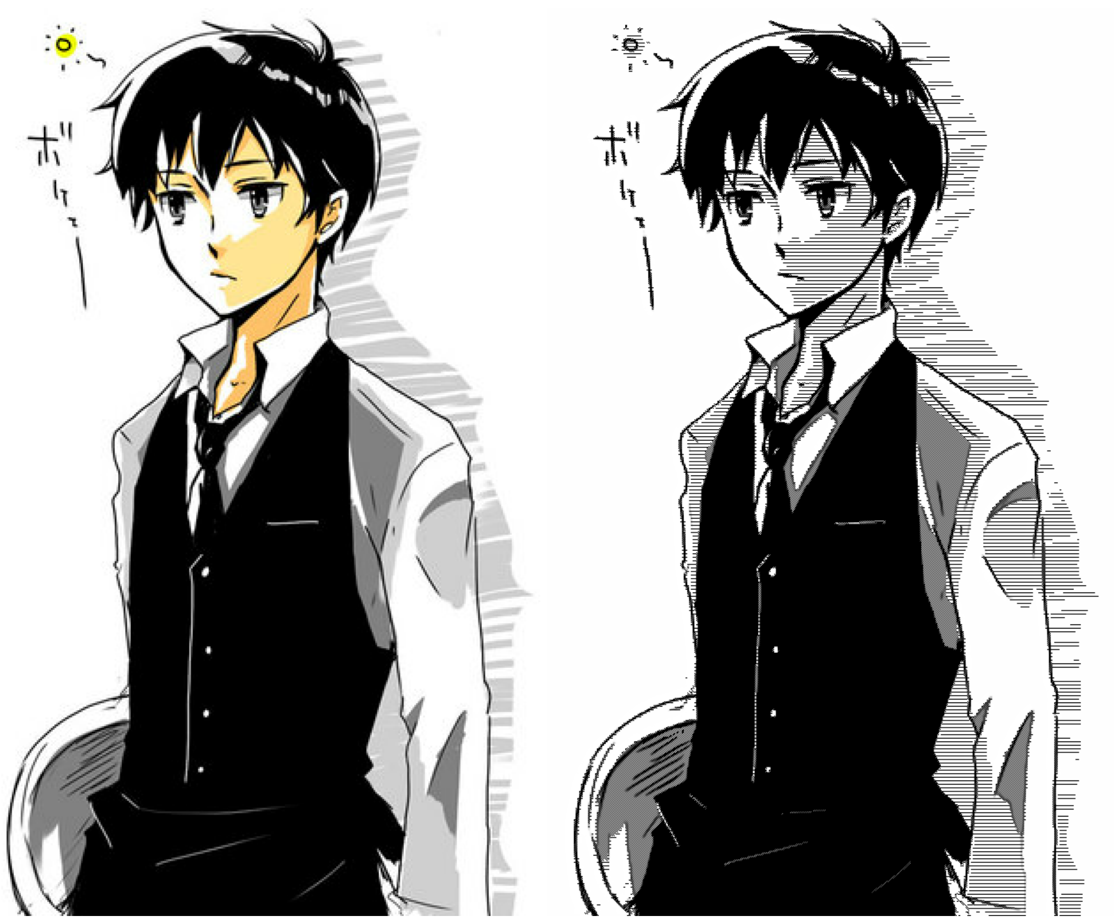


Figure 6.6: The original input is shown on the left, while the result is displayed on the right. Zoom in to see the details.

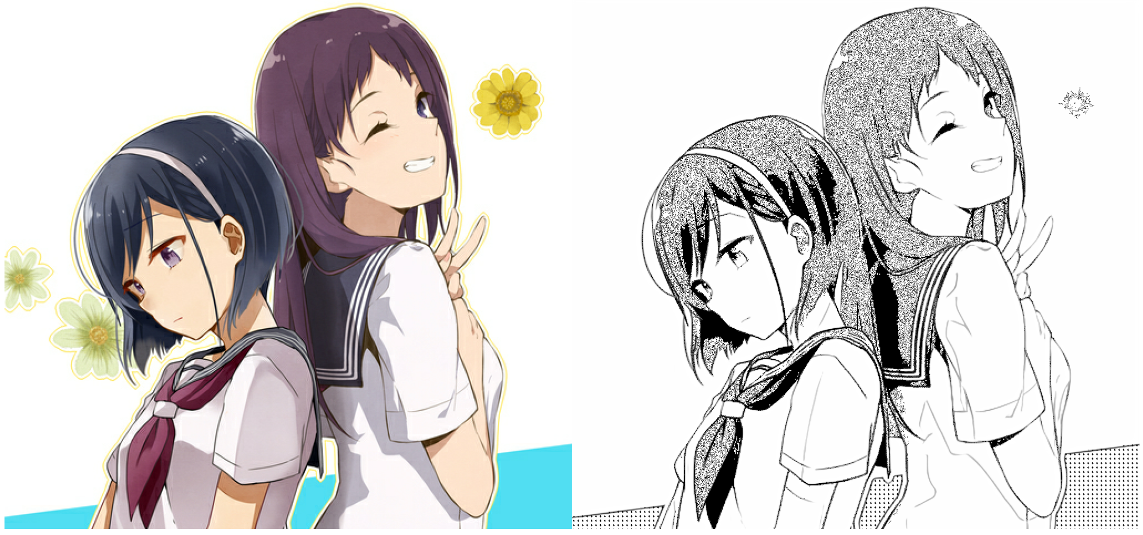


Figure 6.7: The original input is shown on the left, while the result is displayed on the right. Zoom in to see the details.

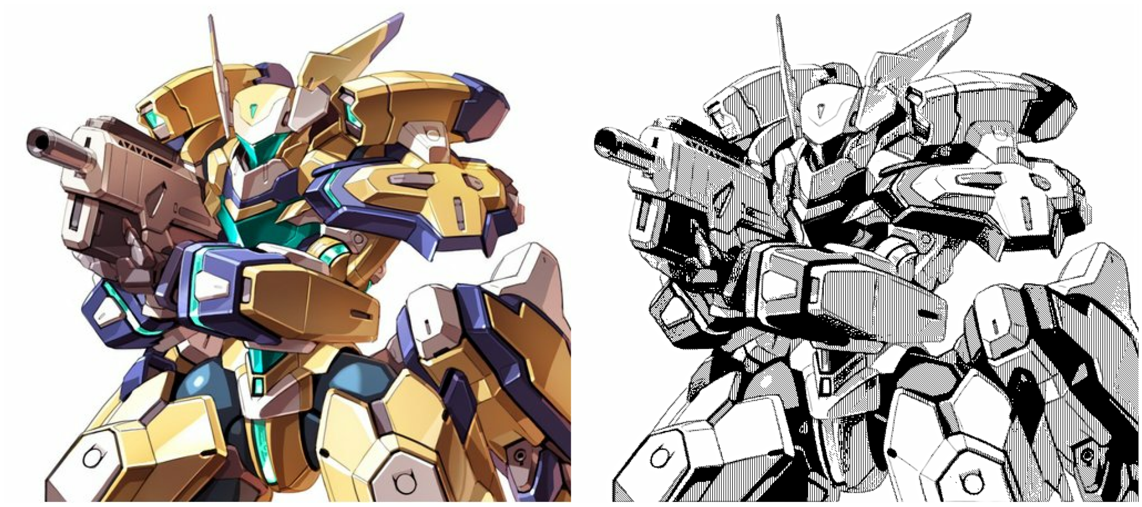


Figure 6.8: The original input is shown on the left, while the result is displayed on the right. Zoom in to see the details.

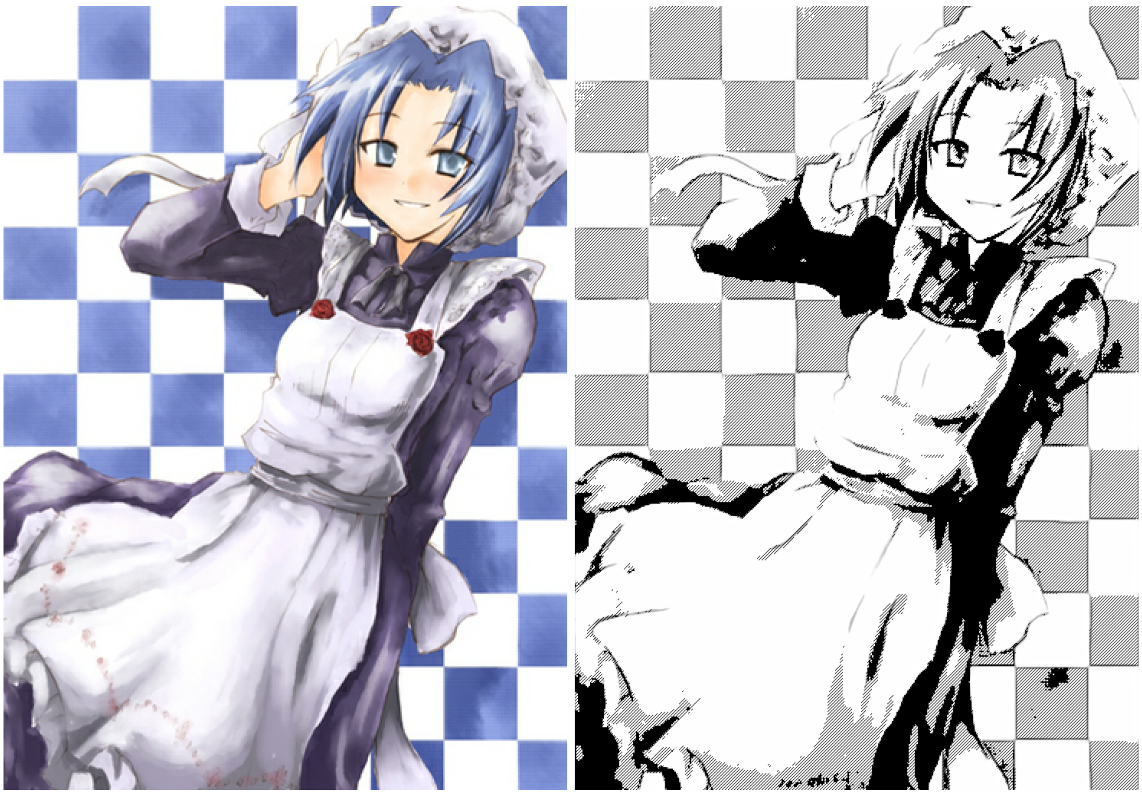


Figure 6.9: The original input is shown on the left, while the result is displayed on the right. Zoom in to see the details.

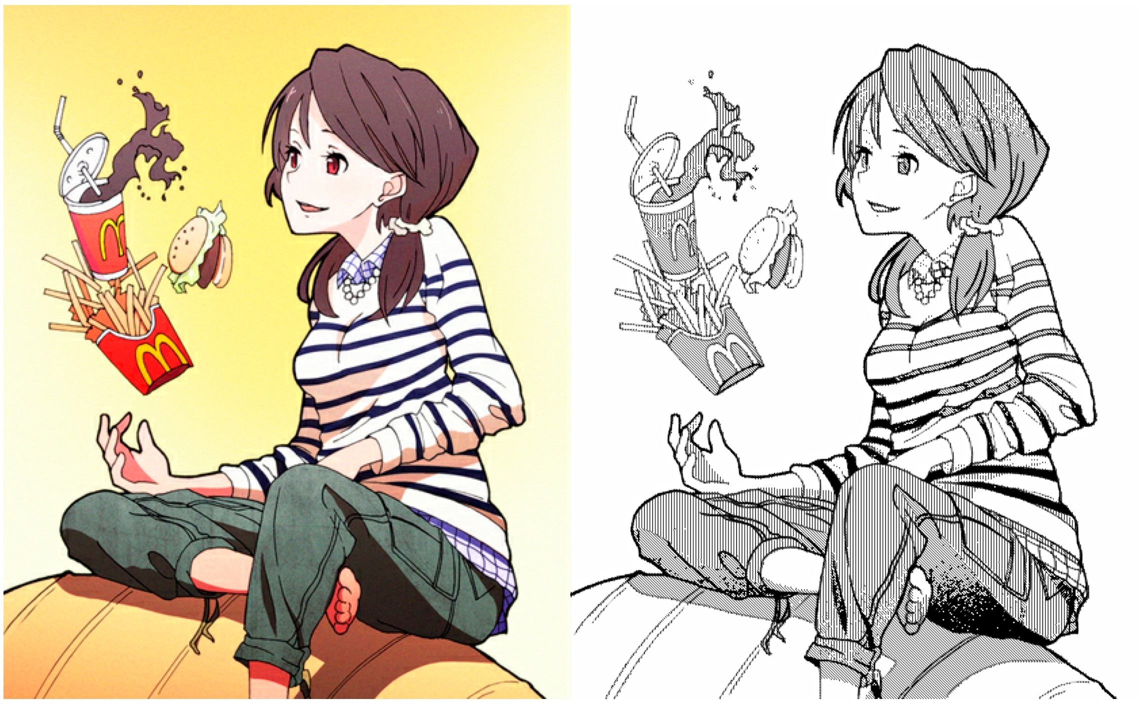


Figure 6.10: The original input is shown on the left, while the result is displayed on the right. Zoom in to see the details.

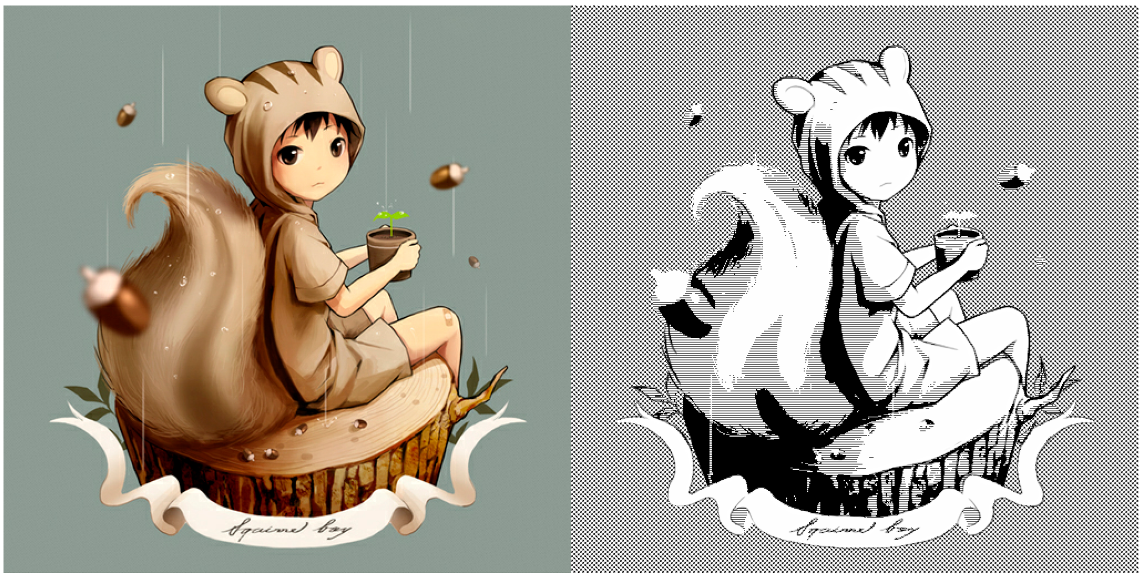


Figure 6.11: The original input is shown on the left, while the result is displayed on the right. Zoom in to see the details.

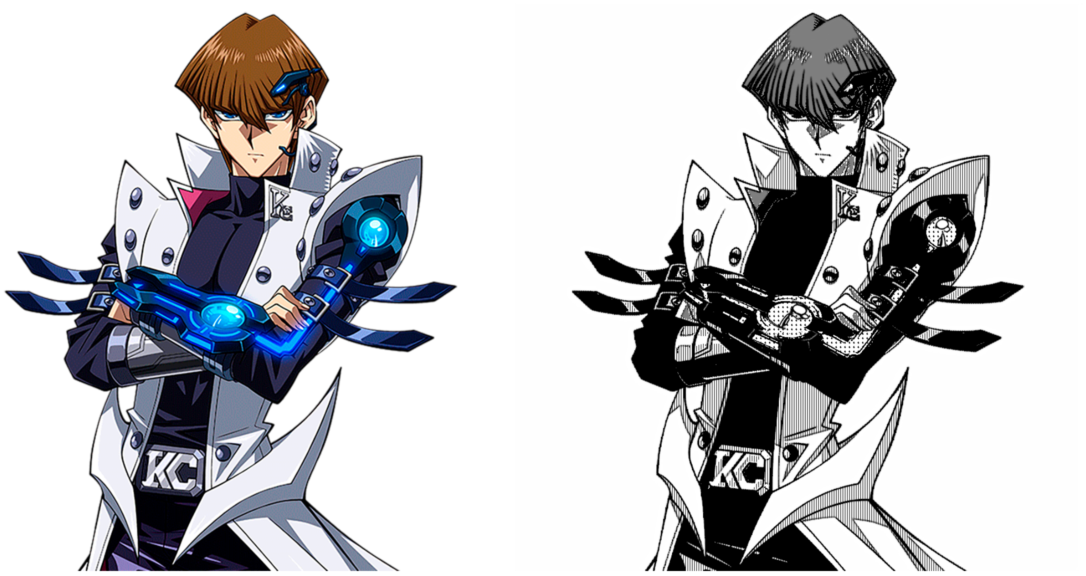


Figure 6.12: The original input is shown on the left, while the result is displayed on the right. Zoom in to see the details.

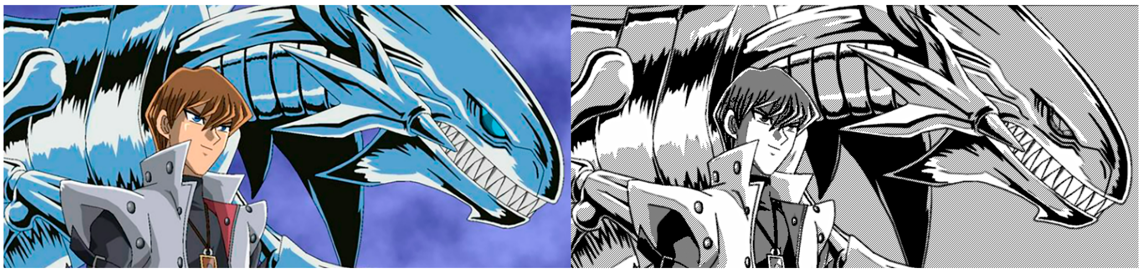


Figure 6.13: The original input is shown on the left, while the result is displayed on the right. Zoom in to see the details.

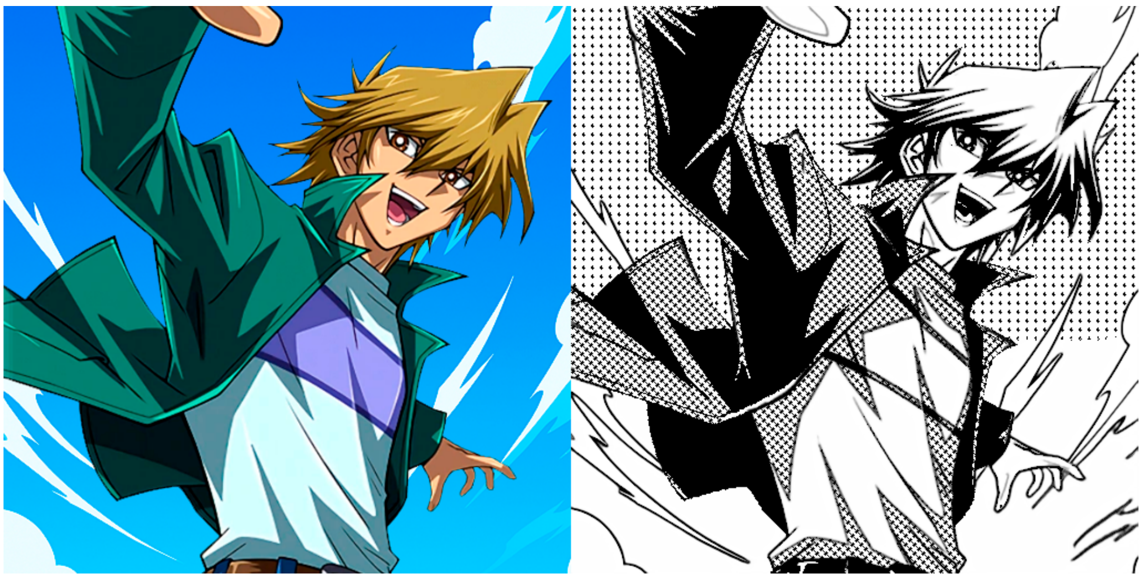


Figure 6.14: The original input is shown on the left, while the result is displayed on the right. Zoom in to see the details.

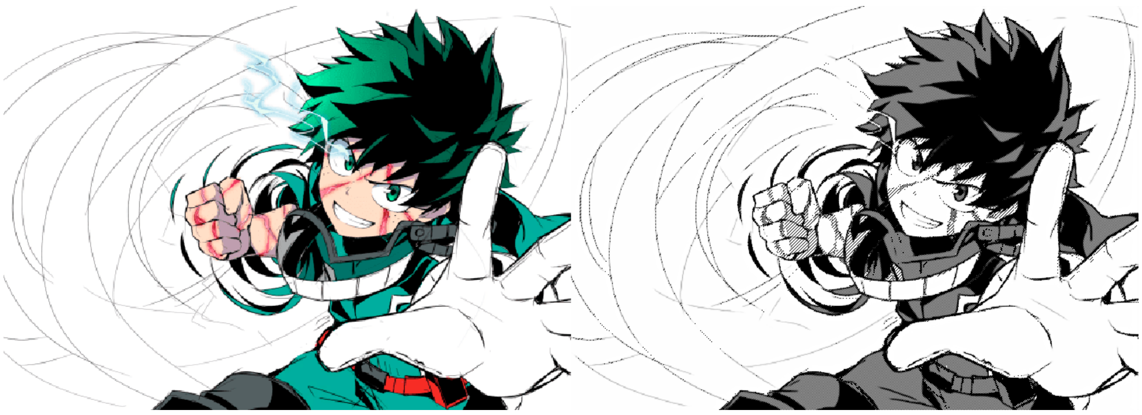


Figure 6.15: The original input is shown on the left, while the result is displayed on the right. Zoom in to see the details.

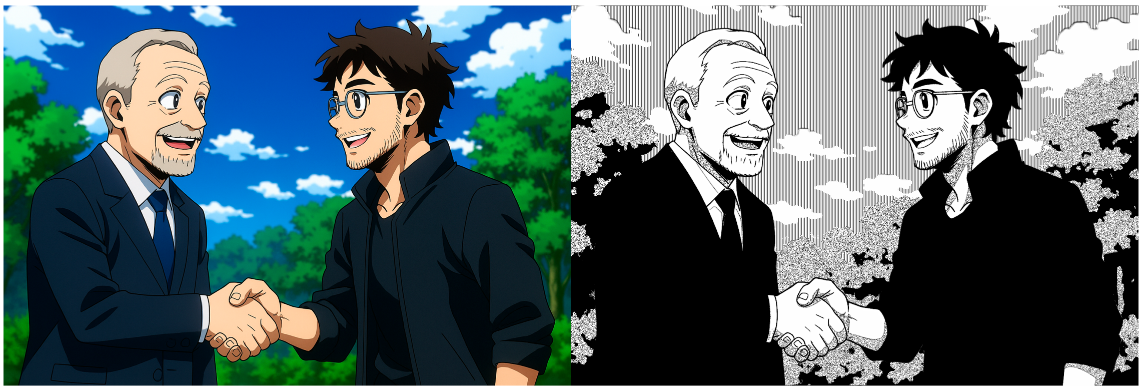


Figure 6.16: The original input is shown on the left, while the result is displayed on the right. Zoom in to see the details.

Chapter 7

Conclusion

7.1 Conclusion

This thesis propose a two-stage unsupervised approach for translating anime-style illustrations into manga images. We address the lack of paired datasets for supervised learning methods in manga colorization and synthesis, while also providing a toll to support the manga creation process. By dividing the translation into two distinct stages, our framework enables control over the stylization procedure.

We developed a pipeline that begins with an unsupervised image translation model to convert colored anime frames into preliminary manga counterparts. This is achieved through a GAN-based approach composed of three CNNs. To this end, a novel custom discriminator architecture was designed for this task— to the best of our knowledge—along with three loss functions specifically adapted to the artistic style involved in this work. Specifically, we aimed to better understand what it means to preserve the content of an illustration when translating it into a manga version, as well as how to maintain the distribution of screen patterns that compose the dataset used as the target domain. We investigated which types of features are relevant for these types of illustrations and how they can be effectively explored by the CNN structure.

For stable training, we also proposed a progressive loss function initialization method. We plotted the behavior of each individual curve to show the effect of each loss function. Additionally, these curves help illustrate that the training remains stable throughout the adversarial dynamics.

In the second stage, a classifier predicts screen patterns and applies them based on the preliminary image produced in the first stage. An adapted method for extracting structural lines is also proposed and used to preserve the source image's structural lines while avoiding inaccurate ones in the final result. Both the screen patterns and structural lines are then used to generate the final output. As this stage works as a post-processing step, we emphasize the main parameters involved and discuss how they can be adjusted to meet specific requirements in the translation process.

In a more practical and implementation-focused perspective, we presented the hardware and software ecosystem used in the experiments of this thesis. We also described the datasets used, the data augmentation procedures applied, as well as architectural details of the CNNs used in the proposed framework.

We conducted a qualitative and user study, comparing our approach with all state-of-the-art methods. The results demonstrated that our method produces outputs with diverse screen patterns without sacrificing image quality or structural fidelity, while not requiring training on high-resolution datasets, maintaining its unsupervised nature, and allowing for a high degree of customization. We also discussed the main limitations of our method, especially those related to highly complex screen patterns and the merge region problem, where two adjacent regions receive the same screen pattern without a clear division by structural lines. Nevertheless, we proposed some strategies to mitigate these issues.

We concluded by presenting a series of images translated by our framework, sourced from the publicly available Danbooru [3] dataset, from famous anime frames, and from digital art generated by the author of this work.

7.2 Final Remarks

Although focused on a specific niche, this work has practical relevance in the context of industrial tools for automating manga production. The proposed framework contributes to the field of image-to-image translation by providing a flexible and modular approach to manga generation, with applications in both academic research and industry.

We also believe that this study contributes to the broader field of image-to-image stylization by addressing the task without resorting to highly complex models. Our results indicate that decomposing this kind of problem into a well-structured pipeline can deliver high-quality outputs while relying on more computationally efficient tools.

7.3 Future Work

There are several directions for future research. First, considering other screen patterns sets, we would like to train the model using a full anime and manga dataset from the same series — for instance, Dragon Ball, My Hero Academia, or Yu-Gi-Oh! — in order to produce a complete dataset of translated frames. The idea is to evaluate the model's consistency across a sequence of scenes.

Second,we would like to explore how short video clips can be converted into animated manga sequences. This would require enforcing temporal consistency between consecutive frames to simulate a stylized manga animation, potentially enabling new formats of multimedia storytelling that blend static and dynamic representations.

Lastly, while this work focused on anime-style illustrations, we would like to investigate how the same pipeline could be extended to other domains involving stylized-to-stylized translation. We believe the modularity and unsupervised nature of the method provide a strong foundation for further exploration in cross-domain artistic translation tasks.

References

[1] TORIYAMA, A. *Dragon Ball*. Shueisha, 1994. Selected pages and cover image used for illustrative purposes.

[2] TOEI ANIMATION. "Dragon Ball Z". 1995. Anime frames used for illustrative and academic purposes only.

[3] ANONYMOUS, COMMUNITY, D., BRANWEN, G. "Danbooru2021: A Large-Scale Crowdsourced Tagged Anime Illustration Dataset". `https://gwern.net/danbooru2021`, January 2022. Available at: <`https://gwern.net/danbooru2021`>. Accessed: 2024.

[4] INDOML. "Student Notes: Convolutional Neural Networks (CNN) Introduction". 2018. Available at: <`https://indoml.com/2018/03/07/student-notes-convolutional-neural-networks-cnn-introduction/`>. Accessed: 2025-05-11.

[5] JADON, S. "Introduction to different activation functions for deep learning", *Medium, Augmenting Humanity*, v. 16, 2018.

[6] YINGGE, H., ALI, I. "Deep Neural Networks on Chip - A Survey". pp. 589–592, 02 2020. doi: 10.1109/BigComp48618.2020.00016.

[7] SAHA, S. "A Comprehensive Guide to Convolutional Neural Networks (The ELI5 Way)". `https://saturncloud.io/blog/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way/`, 2023. Accessed: April 27, 2025. Originally published: December 15, 2018. Last updated: November 20, 2023.

[8] LECUN, Y., BOTTOU, L., BENGIO, Y., et al. "Gradient-based learning applied to document recognition", *Proceedings of the IEEE*, v. 86, n. 11, pp. 2278–2324, 1998. doi: 10.1109/5.726791.

[9] KRIZHEVSKY, A., SUTSKEVER, I., HINTON, G. E. "Imagenet classification with deep convolutional neural networks", *Advances in neural information processing systems*, v. 25, 2012.

[10] YEN, K. "An Overview of VGG16 and NIN Models". `https://lekhuyen.medium.com/an-overview-of-vgg16-and-nin-models-96e4bf398484`, 2021. Accessed: 2025-04-27.

[11] SZEGEDY, C., LIU, W., JIA, Y., et al. "Going deeper with convolutions". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1–9, 2015.

[12] RONNEBERGER, O., FISCHER, P., BROX, T. "U-net: Convolutional networks for biomedical image segmentation". In: *Medical image computing and computer-assisted intervention–MICCAI 2015: 18th international conference, Munich, Germany, October 5-9, 2015, proceedings, part III 18*, pp. 234–241. Springer, 2015.

[13] HE, K., ZHANG, X., REN, S., et al. "Deep residual learning for image recognition". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

[14] HUANG, G., LIU, Z., VAN DER MAATEN, L., et al. "Densely connected convolutional networks". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4700–4708, 2017.

[15] STUDIO GALLOP. "Yu-Gi-Oh! Duel Monsters". Television series, 2000s. Anime frame featuring Thousand Dragon.

[16] TAKAHASHI, K. "Yu-Gi-Oh!" 1990s. Manga panel featuring Thousand Dragon.

[17] ISOLA, P., ZHU, J.-Y., ZHOU, T., et al. "Image-to-image translation with conditional adversarial networks". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1125–1134, 2017.

[18] LI, C., WAND, M. "Precomputed real-time texture synthesis with markovian generative adversarial networks". In: *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part III 14*, pp. 702–716. Springer, 2016.

[19] CANNY, J. "A computational approach to edge detection", *IEEE Transactions on pattern analysis and machine intelligence*, , n. 6, pp. 679–698, 1986.

[20] XIE, M., LI, C., LIU, X., et al. "Manga filling style conversion with screentone variational autoencoder", *ACM Transactions on Graphics (TOG)*, v. 39, n. 6, pp. 1–15, 2020.

[21] ZHANG, L., WANG, X., FAN, Q., et al. "Generating manga from illustrations via mimicking manga creation workflow". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 5642–5651, 2021.

[22] LIN, J., LIU, X., LI, C., et al. "SKETCH2MANGA: Shaded Manga Screening from Sketch with Diffusion Models". In: *2024 IEEE International Conference on Image Processing (ICIP)*, pp. 2389–2395. IEEE, 2024.

[23] DELETER. "DELETER OFFICIAL SHOPPING SITE". `http://deleter-mangashop.com/`, 2020. Accessed: 2024.

[24] FURUSAWA, C., HIROSHIBA, K., OGAKI, K., et al. "Comicolorization: semi-automatic manga colorization". In: *SIGGRAPH Asia 2017 Technical Briefs*, pp. 1–4, 2017.

[25] HENSMAN, P., AIZAWA, K. "cGAN-based manga colorization using a single training image". In: *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*, v. 3, pp. 72–77. IEEE, 2017.

[26] ZHANG, L., LI, C., WONG, T.-T., et al. "Two-stage sketch colorization", *ACM Transactions on Graphics (TOG)*, v. 37, n. 6, pp. 1–14, 2018.

[27] CI, Y., MA, X., WANG, Z., et al. "User-guided deep anime line art colorization with conditional adversarial networks". In: *Proceedings of the 26th ACM international conference on Multimedia*, pp. 1536–1544, 2018.

[28] LEE, J., KIM, E., LEE, Y., et al. "Reference-based sketch image colorization using augmented-self reference and dense semantic correspondence". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 5801–5810, 2020.

[29] ADOBE. "Adobe After Effects". `https://www.adobe.com/products/aftereffects.html`, 2025. Software.

[30] CLIP STUDIO. "Clip Studio Paint". `https://www.clipstudio.net/en/`, 2025. Software.

[31] GOODFELLOW, I., POUGET-ABADIE, J., MIRZA, M., et al. "Generative adversarial nets", *Advances in neural information processing systems*, v. 27, 2014.

[32] HUBEL, D. H., WIESEL, T. N., OTHERS. "Receptive fields of single neurones in the cat's striate cortex", *J physiol*, v. 148, n. 3, pp. 574–591, 1959.

[33] ROBERTS, L. *Machine Perception of Three-Dimensional Solids.* 01 1963. ISBN: 0-8240-4427-4.

[34] MARR, D. "Analyzing natural images: A computational theory of texture vision". In: *Cold Spring Harbor symposia on quantitative biology*, v. 40, pp. 647–662. Cold Spring Harbor Laboratory Press, 1976.

[35] MARR, D. "Early processing of visual information", *Philosophical Transactions of the Royal Society of London. B, Biological Sciences*, v. 275, n. 942, pp. 483–519, 1976.

[36] BROOKS, R. A., CREINER, R., BINFORD, T. O. "The ACRONYM model-based vision system". In: *Proceedings of the 6th international joint conference on Artificial intelligence-Volume 1*, pp. 105–113, 1979.

[37] BROOKS, R. A. "Symbolic reasoning among 3-D models and 2-D images", *Artificial intelligence*, v. 17, n. 1-3, pp. 285–348, 1981.

[38] LOWE, D. G. "The viewpoint consistency constraint", *International Journal of Computer Vision*, v. 1, n. 1, pp. 57–72, 1987.

[39] LOWE, D. G. "Three-dimensional object recognition from single two-dimensional images", *Artificial intelligence*, v. 31, n. 3, pp. 355–395, 1987.

[40] SHI, J., MALIK, J. "Normalized cuts and image segmentation", *IEEE Transactions on pattern analysis and machine intelligence*, v. 22, n. 8, pp. 888–905, 2000.

[41] LOWE, D. G. "Object recognition from local scale-invariant features". In: *Proceedings of the seventh IEEE international conference on computer vision*, v. 2, pp. 1150–1157. Ieee, 1999.

[42] VIOLA, P., JONES, M. "Rapid object detection using a boosted cascade of simple features". In: *Proceedings of the 2001 IEEE computer society conference on computer vision and pattern recognition. CVPR 2001*, v. 1, pp. I–I. Ieee, 2001.

[43] DENG, J., DONG, W., SOCHER, R., et al. "ImageNet: A large-scale hierarchical image database". In: *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 248–255, 2009. doi: 10.1109/CVPR.2009.5206848.

[44] GOODFELLOW, I., BENGIO, Y., COURVILLE, A. *Deep Learning.* MIT Press, 2016. `http://www.deeplearningbook.org`.

[45] NAIR, V., HINTON, G. E. “Rectified linear units improve restricted boltzmann machines”. In: *Proceedings of the 27th international conference on machine learning (ICML-10)*, pp. 807–814, 2010.

[46] MAAS, A. L., HANNUN, A. Y., NG, A. Y., et al. “Rectifier nonlinearities improve neural network acoustic models”. In: *Proc. icml*, v. 30, p. 3. Atlanta, GA, 2013.

[47] ZEILER, M. D., FERGUS, R. “Stochastic pooling for regularization of deep convolutional neural networks”, *arXiv preprint arXiv:1301.3557*, 2013.

[48] LIN, M., CHEN, Q., YAN, S. “Network in network”, *arXiv preprint arXiv:1312.4400*, 2013.

[49] ZEILER, M. D., FERGUS, R. “Visualizing and Understanding Convolutional Networks”. In: Fleet, D., Pajdla, T., Schiele, B., et al. (Eds.), *Computer Vision – ECCV 2014*, pp. 818–833, Cham, 2014. Springer International Publishing. ISBN: 978-3-319-10590-1.

[50] SIMONYAN, K., ZISSERMAN, A. “Very deep convolutional networks for large-scale image recognition”, *arXiv preprint arXiv:1409.1556*, 2014.

[51] HOWARD, A. G. “Mobilenets: Efficient convolutional neural networks for mobile vision applications”, *arXiv preprint arXiv:1704.04861*, 2017.

[52] TAN, M., LE, Q. “Efficientnet: Rethinking model scaling for convolutional neural networks”. In: *International conference on machine learning*, pp. 6105–6114. PMLR, 2019.

[53] QIAN, N. “On the momentum term in gradient descent learning algorithms”, *Neural networks*, v. 12, n. 1, pp. 145–151, 1999.

[54] TIELEMAN, T. “Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude”, *COURSERA: Neural networks for machine learning*, v. 4, n. 2, pp. 26, 2012.

[55] KINGMA, D. P., BA, J. “Adam: A method for stochastic optimization”, *arXiv preprint arXiv:1412.6980*, 2014.

[56] RUDER, S. “An overview of gradient descent optimization algorithms”, *CoRR*, v. abs/1609.04747, 2016. Available at: <http://arxiv.org/abs/1609.04747>.

[57] LI, H., XU, Z., TAYLOR, G., et al. "Visualizing the Loss Landscape of Neural Nets". In: Bengio, S., Wallach, H., Larochelle, H., et al. (Eds.), *Advances in Neural Information Processing Systems*, v. 31. Curran Associates, Inc., 2018. Available at: <`https://proceedings.neurips.cc/paper_files/paper/2018/file/a41b3bb3e6b050b6c9067c67f663b915-Paper.pdf`>.

[58] ABADI, M., BARHAM, P., CHEN, J., et al. "{TensorFlow}: a system for {Large-Scale} machine learning". In: *12th USENIX symposium on operating systems design and implementation (OSDI 16)*, pp. 265–283, 2016.

[59] PASZKE, A. "Pytorch: An imperative style, high-performance deep learning library", *arXiv preprint arXiv:1912.01703*, 2019.

[60] LECUN, Y. "What are some recent and potentially upcoming breakthroughs in deep learning?" `https://www.quora.com/What-are-some-recent-and-potentially-upcoming-breakthroughs-in-deep-learni` 2017. Accessed: June 2025.

[61] ZHU, J.-Y., PARK, T., ISOLA, P., et al. "Unpaired image-to-image translation using cycle-consistent adversarial networks". In: *Proceedings of the IEEE international conference on computer vision*, pp. 2223–2232, 2017.

[62] ARJOVSKY, M., CHINTALA, S., BOTTOU, L. "Wasserstein generative adversarial networks". In: *International conference on machine learning*, pp. 214–223. PMLR, 2017.

[63] RADFORD, A., METZ, L., CHINTALA, S. "Unsupervised representation learning with deep convolutional generative adversarial networks", *arXiv preprint arXiv:1511.06434*, 2015.

[64] GULRAJANI, I., AHMED, F., ARJOVSKY, M., et al. "Improved training of wasserstein gans", *Advances in neural information processing systems*, v. 30, 2017.

[65] PANG, Y., LIN, J., QIN, T., et al. "Image-to-image translation: Methods and applications", *IEEE Transactions on Multimedia*, v. 24, pp. 3859–3881, 2021.

[66] NEAL, R. M., DAYAN, P. "Factor analysis using delta-rule wake-sleep learning", *Neural computation*, v. 9, n. 8, pp. 1781–1803, 1997.

[67] YANG, L., ZHANG, Z., SONG, Y., et al. "Diffusion models: A comprehensive survey of methods and applications", *ACM Computing Surveys*, v. 56, n. 4, pp. 1–39, 2023.

[68] DHARIWAL, P., NICHOL, A. “Diffusion models beat gans on image synthesis”, *Advances in neural information processing systems*, v. 34, pp. 8780–8794, 2021.

[69] HO, J., JAIN, A., ABBEEL, P. “Denoising diffusion probabilistic models”, *Advances in neural information processing systems*, v. 33, pp. 6840–6851, 2020.

[70] SONG, Y., ERMON, S. “Generative modeling by estimating gradients of the data distribution”, *Advances in neural information processing systems*, v. 32, 2019.

[71] SONG, Y., SOHL-DICKSTEIN, J., KINGMA, D. P., et al. “Score-based generative modeling through stochastic differential equations”, *arXiv preprint arXiv:2011.13456*, 2020.

[72] BAO, J., CHEN, D., WEN, F., et al. “CVAE-GAN: fine-grained image generation through asymmetric training”. In: *Proceedings of the IEEE international conference on computer vision*, pp. 2745–2754, 2017.

[73] ZHU, J.-Y., ZHANG, R., PATHAK, D., et al. “Toward multimodal image-to-image translation”, *Advances in neural information processing systems*, v. 30, 2017.

[74] LIU, M.-Y., BREUEL, T., KAUTZ, J. “Unsupervised image-to-image translation networks”, *Advances in neural information processing systems*, v. 30, 2017.

[75] HUANG, X., LIU, M.-Y., BELONGIE, S., et al. “Multimodal unsupervised image-to-image translation”. In: *Proceedings of the European conference on computer vision (ECCV)*, pp. 172–189, 2018.

[76] SAHARIA, C., CHAN, W., CHANG, H., et al. “Palette: Image-to-image diffusion models”. In: *ACM SIGGRAPH 2022 conference proceedings*, pp. 1–10, 2022.

[77] WANG, L., LI, D., ZHU, Y., et al. “Dual super-resolution learning for semantic segmentation”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 3774–3783, 2020.

[78] LIU, Y., ZHANG, W., WANG, J. “Source-free domain adaptation for semantic segmentation”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 1215–1224, 2021.

[79] BARANCHUK, D., RUBACHEV, I., VOYNOV, A., et al. "Label-efficient semantic segmentation with diffusion models", *arXiv preprint arXiv:2112.03126*, 2021.

[80] GRAIKOS, A., MALKIN, N., JOJIC, N., et al. "Diffusion models as plug-and-play priors", *Advances in Neural Information Processing Systems*, v. 35, pp. 14715–14728, 2022.

[81] ZHANG, X., CHENG, Z., ZHANG, X., et al. "Posterior promoted GAN with Distribution discriminator for unsupervised image synthesis". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 6519–6528, 2021.

[82] LIN, J., ZHANG, R., GANZ, F., et al. "Anycost gans for interactive image synthesis and editing". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 14986–14996, 2021.

[83] ROMBACH, R., BLATTMANN, A., LORENZ, D., et al. "High-Resolution Image Synthesis with Latent Diffusion Models". 2021.

[84] SAHARIA, C., CHAN, W., SAXENA, S., et al. "Photorealistic text-to-image diffusion models with deep language understanding", *Advances in neural information processing systems*, v. 35, pp. 36479–36494, 2022.

[85] SHAO, Y., LI, L., REN, W., et al. "Domain adaptation for image dehazing". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 2808–2817, 2020.

[86] CHEN, L., ZHANG, J., PAN, J., et al. "Learning a Non-Blind Deblurring Network for Night Blurry Images". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 10542–10550, June 2021.

[87] CHEN, Y., WANG, O., ZHANG, R., et al. "Image Neural Field Diffusion Models". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8007–8017, 2024.

[88] XU, Z., WANG, T., FANG, F., et al. "Stylization-based architecture for fast deep exemplar colorization". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9363–9372, 2020.

[89] LIANG, Z., LI, Z., ZHOU, S., et al. "Control Color: Multimodal Diffusion-based Interactive Image Colorization", *arXiv preprint arXiv:2402.10855*, 2024.

[90] YI, Z., ZHANG, H., TAN, P., et al. "Dualgan: Unsupervised dual learning for image-to-image translation". In: *Proceedings of the IEEE international conference on computer vision*, pp. 2849–2857, 2017.

[91] CHEN, Y., LAI, Y.-K., LIU, Y.-J. "Cartoongan: Generative adversarial networks for photo cartoonization". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 9465–9474, 2018.

[92] WANG, X., YU, J. "Learning to cartoonize using white-box cartoon representations". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 8090–8099, 2020.

[93] HICSONMEZ, S., SAMET, N., AKBAS, E., et al. "GANILLA: Generative adversarial networks for image to illustration translation", *Image and Vision Computing*, v. 95, pp. 103886, 2020.

[94] ZHENG, Z., LIU, H., YANG, F., et al. "Representation-guided generative adversarial network for unpaired photo-to-caricature translation", *Computers & Electrical Engineering*, v. 90, pp. 106999, 2021.

[95] SU, H., NIU, J., LIU, X., et al. "Mangagan: Unpaired photo-to-manga translation based on the methodology of manga drawing". In: *Proceedings of the AAAI Conference on Artificial Intelligence*, v. 35, pp. 2611–2619, 2021.

[96] ZHANG, Y., HUANG, N., TANG, F., et al. "Inversion-based style transfer with diffusion models". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 10146–10156, 2023.

[97] HUANG, N., ZHANG, Y., TANG, F., et al. "Diffstyler: Controllable dual diffusion for text-driven image stylization", *IEEE Transactions on Neural Networks and Learning Systems*, 2024.

[98] WU, H., MA, Z., WU, W., et al. "Shading-guided manga screening from reference", *IEEE Transactions on Visualization and Computer Graphics*, 2023.

[99] LI, Z., ZHAO, N., WU, Z., et al. "Reference-based Screentone Transfer via Pattern Correspondence and Regularization". In: *Computer Graphics Forum*, v. 42, p. e14800. Wiley Online Library, 2023.

[100] QU, Y., PANG, W.-M., WONG, T.-T., et al. "Richness-preserving manga screening", *ACM Transactions on Graphics (TOG)*, v. 27, n. 5, pp. 1–8, 2008.

[101] XIE, M., LI, C., WONG, T.-T. "Manga Rescreening with Interpretable Screentone Representation", *arXiv preprint arXiv:2306.04114*, 2023.

[102] RUSSAKOVSKY, O., DENG, J., SU, H., et al. "Imagenet large scale visual recognition challenge", *International journal of computer vision*, v. 115, pp. 211–252, 2015.

[103] JOHNSON, J., ALAHI, A., FEI-FEI, L. "Perceptual losses for real-time style transfer and super-resolution". In: *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part II 14*, pp. 694–711. Springer, 2016.

[104] MAHENDRAN, A., VEDALDI, A. "Understanding deep image representations by inverting them". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 5188–5196, 2015.

[105] OTSU, N. "A threshold selection method from gray-level histograms", *IEEE transactions on systems, man, and cybernetics*, v. 9, n. 1, pp. 62–66, 1979.

[106] BRADSKI, G. "The OpenCV Library". In: *Dr. Dobb's Journal of Software Tools*, 2000.

[107] GONZALEZ, R. C., WOODS, R. E. "Digital Image Processing", *Prentice Hall*, 2002. Section on adaptive thresholding.

[108] GONZALEZ, R. C., WOODS, R. E. *Digital Image Processing (3rd Edition)*. USA, Prentice-Hall, Inc., 2006. ISBN: 013168728X.

[109] BADRINARAYANAN, V., KENDALL, A., CIPOLLA, R. "SegNet: A deep convolutional encoder-decoder architecture for image segmentation", *IEEE transactions on pattern analysis and machine intelligence*, v. 39, n. 12, pp. 2481–2495, 2017.

[110] KULLBACK, S., LEIBLER, R. A. "On information and sufficiency", *The annals of mathematical statistics*, v. 22, n. 1, pp. 79–86, 1951.

[111] KHOSLA, P., TETERWAK, P., WANG, C., et al. "Supervised contrastive learning", *Advances in neural information processing systems*, v. 33, pp. 18661–18673, 2020.

[112] COMANICIU, D., MEER, P. "Mean shift: A robust approach toward feature space analysis", *IEEE Transactions on pattern analysis and machine intelligence*, v. 24, n. 5, pp. 603–619, 2002.

[113] ITO, K., MATSUI, Y., YAMASAKI, T., et al. "Separation of Manga Line Drawings and Screentones." In: *Eurographics (Short Papers)*, pp. 73–76, 2015.

[114] KANG, H., LEE, S., CHUI, C. K. “Coherent line drawing”. In: *Proceedings of the 5th international symposium on Non-photorealistic animation and rendering*, pp. 43–50, 2007.

[115] LI, C., LIU, X., WONG, T.-T. “Deep extraction of manga structural lines”, *ACM Transactions on Graphics (TOG)*, v. 36, n. 4, pp. 1–12, 2017.

[116] SAXENA, D., CAO, J. “Generative Adversarial Networks (GANs): Challenges, Solutions, and Future Directions”, *ACM Computing Surveys*, v. 54, pp. 1–42, 05 2021. doi: 10.1145/3446374.

[117] QI, G.-J. “Loss-sensitive generative adversarial networks on lipschitz densities”, *International Journal of Computer Vision*, v. 128, n. 5, pp. 1118–1140, 2020.

[118] MAO, X., LI, Q., XIE, H., et al. “Least squares generative adversarial networks”. In: *Proceedings of the IEEE international conference on computer vision*, pp. 2794–2802, 2017.

[119] MATSUI, Y., ITO, K., ARAMAKI, Y., et al. “Sketch-based manga retrieval using manga109 dataset”, *Multimedia tools and applications*, v. 76, pp. 21811–21838, 2017.