CS424 – Group Project Report

Akeela Darryl Fattha
akeelaf.2022@scis.smu.edu.sg

Fadhel Erlangga Wibawanto
fadhelew.2022@scis.smu.edu.sg

Tan Zhi Rong
zhirong.tan.2022@scis.smu.edu.sg

Grace Angel Bisawan
gbisawan.2022@scis.smu.edu.sg

Lee Jia Heng
jiaheng.lee.2023@scis.smu.edu.sg

Abstract

This report provides an overview of Cycle-Consistent Adversarial Networks (CycleGAN), a technique for unpaired image-to-image translation. We explore the architecture, key innovations, applications, and limitations of CycleGAN models in computer vision tasks.

Contents

1. Task 1

1.1 Introduction

Image-to-image translation is the task of converting an image from one domain to another. CycleGAN, introduced by Zhu et al. in 2017, addresses the challenge of learning such translations without paired training data. This makes it particularly useful for applications where paired examples are difficult or impossible to obtain.

1.1.1 Configuration

1.2 Architecture

1.2.1 Discriminator

Our discriminator uses a basic sequential architecture with convolutional layers that doubles in number of channels 4 times, from 64 to 512. Each convolution output is passed to a spectral norm layer and a leaky relu activation function.

We also made use of a PatchGAN discriminator, accumulate whether these patches can be used to determine if the image is real or fake.

1.2.2 Generator

Our generator incorporates several modern attention mechanisms and architectural techniques including Convolutional Block Attention Module (CBAM), Squeeze-and-Excitation (SE) blocks, and Self-Attention mechanisms for improved feature representation, generating higher-quality images with better detail preservation, style consistency, and structural coherence.

Some general key design choices include:

- **Instance Normalization:** Used throughout the network instead of batch normalization, as it has been shown to produce better results for style transfer and image-to-image translation by normalizing each instance independently.
- **Reflection Padding:** Applied before convolutions to reduce boundary artifacts that can appear in generated images, particularly important for maintaining realistic edges.
- **Mixed Residual Blocks:** The strategic placement of different residual block types allows the network to benefit from complementary approaches to feature transformation:
  - Style-modulated blocks provide explicit control over stylistic elements
  - CBAM-enhanced blocks in the middle layers help focus on important features
  - All blocks benefit from SE attention for channel recalibration
- **Squeeze-and-Excitation Attention:** Used in Mixed Residual Blocks, it recalibrates channel-wise feature responses by explicitly modeling interdependencies between channels, allowing the network to selectively emphasize informative features.
- **Enhanced Self-Attention:** Positioned after the residual blocks, it helps ensure global coherence in the generated images by modeling long-range dependencies that convolutional operations cannot capture efficiently.

ImprovedResidualBlock: These blocks enhance the standard residual connection with both SE attention and conditional CBAM:
CBAM is applied to blocks 3-6, providing dual attention mechanisms:

- **Channel Attention:** Models interdependencies between channels using both max and average pooling operations.

- **Spatial Attention:** Focuses on important spatial regions by creating attention maps from channel-wise statistics.

StyleModulatedResBlock: Every third residual block employs style modulation through Adaptive Instance Normalization:

$$\text{AdaIN}(x) = \gamma \cdot \frac{x - \mu(x)}{\sigma(x)} + \beta \tag{1.1}$$

where γ and β are learnable style scale and bias parameters. This allows the network to better control stylistic elements in the generated images.

Enhanced Self-Attention

After the residual blocks, an enhanced self-attention module is applied:

$$\text{Attention}(Q, K, V) = \text{Softmax}\left(\frac{QK^T + P}{\sqrt{d_k}}\right)V \tag{1.2}$$

where P represents the positional bias that emphasizes the central region. This self-attention mechanism allows the network to model long-range dependencies and coherence in the image, which is particularly valuable for maintaining structural integrity in facial regions.

1.3 Data Preparation

1.3.1 Augmentations

We started with resizing images to 128x128, paired with a batch size of 16 before we hit a memory limit. We also applied no augmentations to the images, as a baseline to compare against later on when we add augmentations.

After adding various minor augmentations like affines, color jitters, flips, light gaussian blurs and posterising, we found that any combination of them resulted in worse metrics overall. From visual inspection, we found that the generated raw images were just slightly more "noisy" than without augmentations, which could contribute to the worse metrics. Furthermore, the generated cartoon images were a lot more "flat" looking than the target images, losing a lot more of the finer details and textures.

Thus, in both cases, we felt that adding the augmentations were encouraging the model to over-process the style conversions, which likely caused the reduction in metrics.

1.3.2 Training/Validation

A simple 80/20 split on training data with validation

1.4 Loss Functions

Our overall loss functions for CycleGAN are as follows:

$$L_{total} = L_{GAN} + \lambda_{cyc}L_{cyc} + \lambda_{id}L_{id} + \lambda_{edge}L_{edge} + \lambda_{color}L_{color} \tag{1.3}$$

1.4.1 Discriminator

Patch

We use a PatchGAN discriminator, which classifies whether 70x70 overlapping image patches are real or fake. This allows the model to focus on local image features and helps in generating high-quality images.

Adversarial

We use a basic BCE loss as well bruh.

1.4.2 Generator

Adversarial

Use basic BCE loss to compare between predicted & real predictions.

Cycle Consistency

Basic L1 loss to ensure that it tries to get back to where it started exactly.

Identity

We use a combination of MSE and Edge Consistency loss to determine our identity loss. We put additional weight on Edge Consistency as it is extremely important to maintain the edges of the image, especially when it comes to maintaining the structure of a "face". However, to ensure that the model keeps within the color distribution & looks of the original image, we include MSE loss as well.

Edge Consistency

As mentioned, this loss is used to determine whether the structure of the images are maintained. Firstly, we grayscaled our images as color is not our priority for this loss computation. Next, we use sobel kernels to determine the edges of the images, which will be passed to a structural similariy computation function by Piqa.

Color Consistency

1.4.3 Adaptive Loss Weighting

We also incorporate adaptive loss weighting to alter the compositions of the different loss functions according to how far along the training we are. During early training, it is more important to focus on the cycle consistency loss, as the model is still learning the basic structure of the images. As training progresses, we shift the focus towards the adversarial loss and identity loss to refine the generated images and ensure they closely resemble the target domain.

2. Task 2

2.1 Introduction

2.1.1 Configuration

2.2 Architecture

2.2.1 Discriminator

GANs consist of two networks: a generator that creates images and a discriminator that evaluates them. The two networks are trained adversarially, with the generator trying to fool the discriminator.

2.2.2 Generator

CycleGAN extends the GAN framework by using two generator-discriminator pairs, allowing translation between domains X and Y. The key innovation is the cycle-consistency loss, which ensures that translating an image to the target domain and back produces the original image.

2.3 Data Preparation

2.3.1 Pre-Processing

2.3.2 Training/Validation

A simple 80/20 split on training data with validation

2.4 Loss Functions

Our overall loss functions for CycleGAN are as follows:

$$L_{total} = L_{GAN} + \lambda_{cyc} L_{cyc} + \lambda_{id} L_{id} + \lambda_{edge} L_{edge} + \lambda_{color} L_{color} \tag{2.1}$$

2.4.1 Discriminator

Patch

Least Squares

2.4.2 Generator

Adversarial

Use basic

Cycle Consistency

Identity

Edge Consistency

Color Consistency

2.4.3 Adaptive Loss Weighting