# PROGRESSIVE GROWING CYCLEGAN FOR HIGH-RESOLUTION IMAGE TRANSLATION

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

This project combines CycleGAN and Progressive Growing GAN to achieve high-resolution image-to-image translation using unpaired datasets. By progressively increasing image resolution, the model is trained on paired datasets consisting of Monet, Van Gogh, CIFAR10, CelebA, and LSun. During the training process, we designed and evaluated three models: 1. Non-Progressive full size model; 2. A progressive model with only progressive training on the upsampling layers; 3. A progressive model with progressive training on both downsampling and upsampling layers. Through quantitative and qualitative evaluations using FID and Inception Score, the results demonstrate that the Non-Progressive full size model outperforms the two kinds of progressive models. Among the progressive models, the model that only train progressively on the upsampling layers showed some feasibility, while the fully progressive model did not perform up to a reasonable standard.

## 1 INTRODUCTION & RELATED WORK

The field of style transfer focuses on transforming content images into artistic representations while preserving their structural features. With the advent of deep learning, significant advancements have been achieved in this domain.

Generative Adversarial Networks (GANs), first introduced by Ian Goodfellow & Bengio (2014) has become a cornerstone for style transfer tasks. Among GAN-based architectures, StyleGANs proposed by Tero Karras & Aila (2019) nd CycleGAN proposed by Jun-Yan Zhu stand out for their ability to perform cross-domain transformations without paired training data. CycleGAN, in particular, introduced the cycle-consistency loss, which ensures bidirectional consistency in transformations and is critical for preserving content while applying styles. However, these architectures face challenges in generating high-resolution outputs, including mode collapse and training instability.

Recent advancements have addressed these challenges through improved architectures and training paradigms. The Progressive Growing GAN, introduced by Tero Karras & Lehtinen (2017) demonstrated the effectiveness of progressively increasing image resolution during training, enabling more stable learning and high-quality high-resolution outputs.

Parallel to GAN-based methods, diffusion models have gained attention for their ability to generate high-quality images. The Stable Diffusion Model, introduced by Robin Rombach (2021),iteratively refines noisy inputs through a diffusion process, producing photorealistic and style-consistent results. Diffusion models have shown strong potential for tasks requiring fine-grained control over content and style representations. Additionally, multimodal models such as DALL-E proposed by Aditya Ramesh & Sutskever (2021) and CLIP proposed by Alec Radford & et al. (2021) extend the applications of style transfer by incorporating textual conditioning.

In this context, our work aims to explore whether combining Progressive GAN and CycleGAN training can improve the quality of style transfer without paring information and additional control. Therefore, we designed and tested three models: 1. Non-Progressive full size model; 2. A progressive model with only progressive training on the upsampling layers; 3. A progressive model with progressive training on both downsampling and upsampling layers. This study evaluates the impact of progressive training on the performance of CycleGAN-like framework in style transfer tasks.

# 2 METHODOLOGY

## 2.1 DATASET

### 2.1.1 REAL IMAGE DATASET

In this project, CIFAR-10, CelebA, LSun serves as the real image datasets that will undergo style transfer to the artistic styles of Van Gogh and Monet, enabling experiments in translating realistic images into artistic representations.

a. CIFAR-10 Dataset

The CIFAR-10 dataset is a well-known dataset in the computer vision community, consisting of 60,000 32x32 color images in 10 different classes.

b. CelebA Dataset

The CelebA (CelebFaces Attributes) dataset is a large-scale face dataset containing over 200,000 celebrity images with 40 attribute annotations. Each image in the dataset has a size of 178×218 pixels.

c. LSun Dataset

The LSun (Large-scale Scene Understanding) dataset is a large-scale image dataset designed for scene understanding tasks. Each image in the dataset has a width of 256 pixels, while the height varies but is always greater than 256 pixels.

### 2.1.2 STYLE IMAGE DATASET

In this project, the Monet2Photo dataset and Van Gogh Dataset represent the artistic style domain for exploring generative models and style transfer techniques.

a. Monet2photo Dataset
The Monet2Photo dataset consists of 1,193 Monet paintings and 7,038 natural photos, each split into training and testing subsets. Each Monet painting in the dataset has a size of 256×256 pixels. In this project, only the Monet paintings are used.

b. Van Gogh Dataset
The Van Gogh dataset contains a total of 2025 unique images. It includes diverse image sizes, as the dimensions of the images are not consistent. The Van Gogh paintings represent a different artistic style domain for exploring generative models and style transfer techniques.

## 2.2 MODEL ARCHITECTURE

We have largely incorporated certain architectures from cyclegan and progressive gan such that it would allow the model to extract essential information which will later be used for reconstruction. The entire model's architecture largely follows an encoder-decoder format with residual connection after downsampling such that to avoid gradient explosion and gradient vanishing. We explain both the full model architecture and various types of progressive training architectures as follows:

### 2.2.1 FULL (NON-PROGRESSIVE) MODEL ARCHITECTURE

Initially, the input image of dimensions is 3×H×W (representing an RGB image) is transformed into a feature map with dimensions 64×H×W. These features are then processed through three downsampling layers, reducing the spatial resolution progressively. The final output of the downsampling stage is a feature map of dimensions 512×H/8×W/8. The resulting features are passed through nine residual blocks, which preserve the dimensionality, producing an output feature map of size 512×H/8×W/8. Subsequently, the features are fed into three upsampling layers, which restore the spatial resolution to 64×H×W. Finally, the upsampled features pass through an output layer that maps the feature map back to a reconstructed image of size 3×H×W.

By this design, we ensure that there is a comprehensive encoding and decoding of features at different level. For input layer, downsampling layers, and residual layers, the structure of design were largely inspired by the cyclegan architectures that were used specifically to extract valuable information from the input images.

For upsampling layers and output layer, we largely utilized the structure described in the original progressive gan that normalize the output features based on the input features channel and kernel size. This ensures that the output features of convolution at each dimension will not be too large due to either a large kernel size or a larger input channel. This allows us to maintain a high-dimensional information while progressively adjusting spatial resolution.

The general structure definition for each block is defined below (please refers to code for specific hyperparameters):

**Residual layer:** $ReflectionPad2d \implies Conv2d \implies InstanceNorm2d \implies ReLU \implies ReflectionPad2d \implies Conv2d \implies InstanceNorm2d$

Notice that we use instancenorm instead of batchnorm, which allows the normalization to happen for each instance. This enables each image to have their own features normalized without damaging the distribution of other images' features.

**Input layer:** $ReflectionPad2d \implies Conv2d \implies InstanceNorm2d \implies ReLU$

**Downsampling block:** $Conv2d \implies InstanceNorm2d \implies ReLU$

**Output layer:** WSConv2d

Notice this is just convolution layer with a scale to normalize output value. The scale here is defined so that the learning rate is stablized across layer, even when the network grow deep.

**Upsampling layer:** $Upsample \implies WSConv2d \implies Leaky \implies PixelNorm \implies WSConv2d \implies Leaky \implies PixelNorm$

#### 2.2.2 PROGRESSIVE MODEL ARCHITECTURE

For progressive model's architecture, we maintain all blocks used in full model architecture but use them separately. In addition, we define output layer that maps the input features of various size including 512×H/8×W/8, 256×H/4×W/4, 128×H/2×W/2, 64×H×W to 3×H/8×W/8, 3×H/4×W/4, 3×H/2×W/2, 3×H×W.

During training, the model is trained progressively through multiple steps. At each step, the training continues until the model converges and produces reasonable results which would allow the model to move to the next step. The process begins at step 0 and proceeds sequentially to step 3, with the resolution of the output increasing at each step.

Step 0: The input feature tensor X, obtained from the residual block, is directly passed through an output layer to produce an output of size 3×H/8×W/8 . In this step, No upsampling occurs.

Step 1: The input feature tensor X is passed through the first upsampling layer, increasing the spatial resolution. This upsampled feature is then fed into an output layer to produce an output of size 3×H/4×W/4

Step 2: The input feature tensor X is processed through two upsampling layers, sequentially increasing the resolution. An output layer then generates an output of size 3×H/2×W/2.

Step 3: The input feature tensor X passes through all three upsampling layers, progressively refining the spatial resolution. Finally, the output layer produces the full-resolution output of size 3×H×W 3×H×W.

The idea of this is that progressive training approach might allow the model to focus on coarse features initially and gradually incorporate finer details as training advances, stabilizing the learning process and improving the quality of the final high-resolution output.

In training, we also apply alpha smoothing which is also introduced in the original progressive GAN's architecture that blend the upscaled output from last step and the output of current step. Alpha controls the degree of blending such that only the output of current step is considered when

alpha = 1 and only the upscaled output of the last step is considered when alpha = 0. The idea is to introduce each new layer smoothingly and not disturb the distribution of already trained layer too much.

#### 2.2.3 TWO-END PROGRESSIVE MODEL ARCHITECTURE

In addition to the progressive model architecture, we define several input layer that maps the features image of size 3×H×W to various size including 64×H×W, 128×H/2×W/2, 256×H/4×W/4, 512×H/8×W/8.

At step 3, the input image of size 3×H×W passes through only the input layer that maps it to size 512×H/8×W/8. This feature then goes directly into residual blocks.

At step 2, the input image of size 3×H×W passes through the input layer that maps it to size 256×H/4×W/4. This feature then goes through downsampling layer that maps it to size 512×H/8×W/8 and passes it into residual blocks.

At step 1, the input image of size 3×H×W passes through the input layer that maps it to size 128×H/2×W/2. This feature then goes through 2 downsampling layers that map it to size 512×H/8×W/8 and passes it into residual blocks.

At step 1, the input image of size 3×H×W passes through the input layer that maps it to size 64×H×W. This feature then goes through all 3 downsampling layers that map it to size 512×H/8×W/8 and passes it into residual blocks.

The general idea of this is to allow progressive growing of both downsampling and upsampling layers such that at first, only the middle portion of the model is trained, then we gradually add more layers to the sides.

## 3 EXPERIMENT

### 3.1 TRAINING ENVIRONMENT

All our models were run on an NVIDIA T4 GPU using Google Colab.

### 3.2 DATA PREPROCESSING

We paired the 2 style datasets (Van Gogh and Monet) with the 3 real image datasets (CIFAR10, CelebA, LSun), resulting in 6 dataset pairs. Each pair was used to train both the non-progressive model and the progressive model for comparison.
During training, CIFAR10 and style datasets (Van Gogh and Monet) were resized to 64×64, while all other dataset and style dataset pairings were resized to 256×256.
For all datasets, the first 250 images were used as the training dataset, and the subsequent 50 images (indices 250-300) were used as the testing dataset.

#### 3.2.1 DIFFERENT DATASET SETUP

a. Van Gogh Dataset: Due to the diverse categories and uneven distribution of data within the dataset, we selected only the “Drawings” category for our experiments. Specifically, 1000 images from this category were chosen for both training and testing purposes.

b. LSun Dataset: Given the large size of the dataset (30GB) and its multiple categories stored in mdb format, we limited our selection to the “Bedroom” category. The first 1000 images were extracted, converted to jpg format, and used for training and testing.

c. Monet Dataset: The Monet2Photo dataset contains two categories: “Monet” and “Photo”. For our experiments, only the Monet images were used.

d. CIFAR10 and CelebA: The CIFAR10 dataset was directly downloaded from the PyTorch library. And CelebA dataset was obtained from Kaggle.

### 3.3 TRAINING

For all six pair of training datasets, we train both the progressive model (only train progressively on upsampling layers) and non-progressive model. The training structure largely follows the original setup in Jun-Yan Zhu. where The objective is defined as :

$$\mathcal{L}(G, F, D_X, D_Y) = \mathcal{L}_{\text{GAN}}(G, D_Y, X, Y) + \mathcal{L}_{\text{GAN}}(F, D_X, Y, X) + \lambda \mathcal{L}_{\text{cyc}}(G, F) + \gamma \mathcal{L}_{\text{id}}(G, F)$$

Where G and F are functions mapping from X to Y and Y to X. The cycle loss here is defined as the L1 loss of $|G(F(Y)) - Y|$ or $|F(G(X)) - X|$. This means the mapping of real image to artistic domain and back to real domain should give us the same image and vice versa. The identity loss is defined as the L1 loss of $|F(X) - X|$ or $|G(Y) - Y|$. Intuitively, this means that since X is already in real domain, F(X) maps X to real domain should give us the same image. The GAN loss is just the standard GAN loss in this case.

The training happens in cycle as defined in CycleGAN that 2 generators are train with 2 discriminators, we aggregate the loss of mapping real X to domain Y and back to X and also the loss of mapping real Y to domain X and back to Y.

For training of non-progressive model, we train the model for a total of 10 epochs. For training of progressive model, we train the model for 8 epochs for step = 0,1,2 and train the model for 10 epochs when step = 3. When introduce new layers, we apply alpha smoothing in the first 4 epochs (alpha= 0.25, 0.5, 0.75, 1).

### 3.4 TWO-ENDED PROGRESSIVE TRAINING

For training of the two-ended progressive model (training both the 2 downsampling and upsampling layers progressively), the generators did not performing well and only outputs purely black or green images. The underlying reason might be due to the fact that the depth of the model, especially in encoder or downsampling layers, is supposed to make the model learn high level features that it can later applies to downstream tasks. However, the progressive training of downsampling layers forces the 3rd or 4th layers to still consume input latent features with low level features. In addition, the input features number drops dramatically for 3rd and 4th layers which contributes to the loss of information. All of this could be problematic for model's reconstruction and comprehension.

## 4 RESULT ANALYSIS

### 4.1 QUALITATIVE ANALYSIS

According to the quantitative result (see image attached below), we found that non-progressive training often worked better than progressive training. (Below are images comparing non progressive training and progressive training, the images from top to bottom follows: real image in artistic domain A, generated image in real domain B, real image in domain B, generated image in artistic domain A)

On one hand, in the context of progressive training, the primary goal is to generate higher-resolution images. This means that a low-resolution input image is expected to produce a high-resolution output, such as converting an input of size 256×256 into an output of size 512×512. However, in this task, the objective is to perform style transfer which means the input and output images still have the same resolution. For instance, in the case of Monet to CelebA translation, the input size is 256×256, and the output is required to reconstruct the same resolution.

To incorporate four stages of progressive training, we first downsample the image three times, suppose the input image resolution is 256×256, we are reducing the resolution from 256×256 $\rightarrow$ 128×128 $\rightarrow$ 64×64 $\rightarrow$ 32×32. Only in this form we would be able to output image of size 32×32. This introduces unnecessary information compression during downsampling, which negatively impacts the feature representation. In experimentation, we find that the original implementation of

CycleGAN model with only 2 downsampling layers are able to output similar images compare to our full sized model. Meaning the depth of model did not contribute to a better performance.

In addition, models trained with progressive training tend to underperform compared to full-size model, as the latter avoids these resolution-induced bottlenecks. This is intuitive to think as in a conventional neural network, full-size training ensures that deeper layers capture increasingly abstract and high-level features, which are then combined to reconstruct lower-level information as the network progresses. However, in progressive training tasks, we deliberately enforce the deeper layers to represent low-resolution information, while the higher layers are tasked with encoding high-resolution image details. This forced compression of information across resolutions may not always result in meaningful or efficient feature representation.

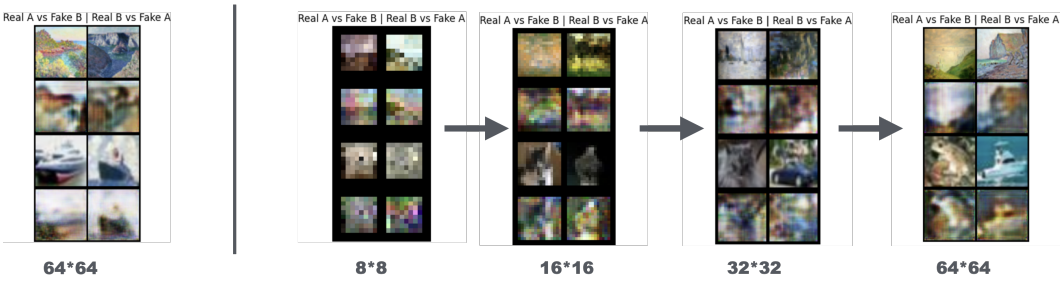


Figure 1: Monet2CIFAR10:None Progressive and Progressive Training

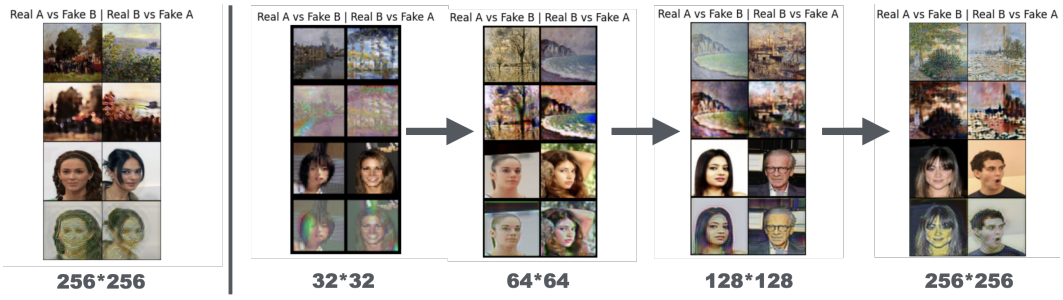


Figure 2: Monet2CelebA:None Progressive and Progressive Training

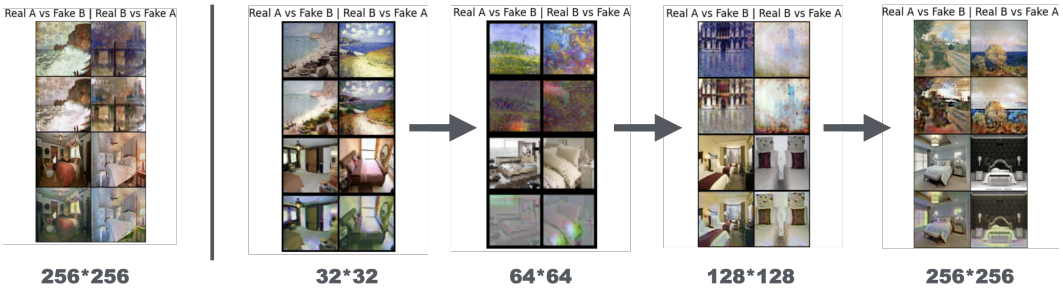


Figure 3: Monet2Lsun:None Progressive and Progressive Training

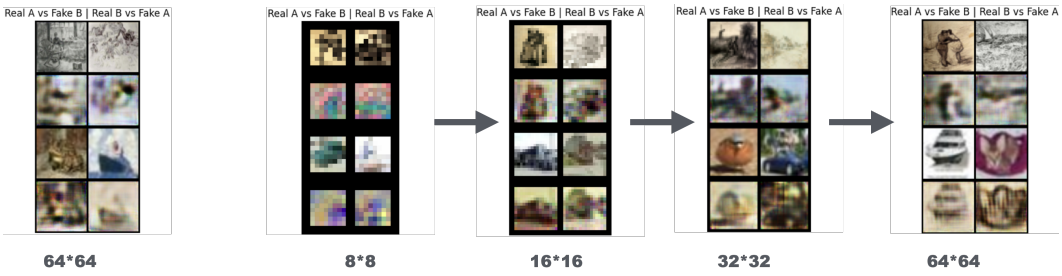


Figure 4: VanGogh2CIFAR10:None Progressive and Progressive Training

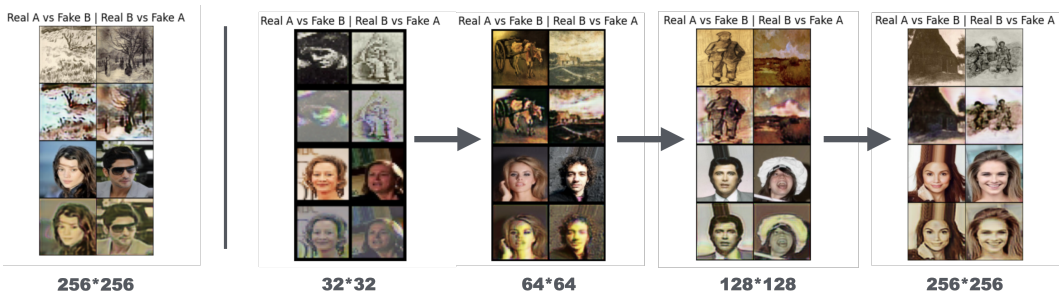


Figure 5: VanGogh2celebA:None Progressive and Progressive Training

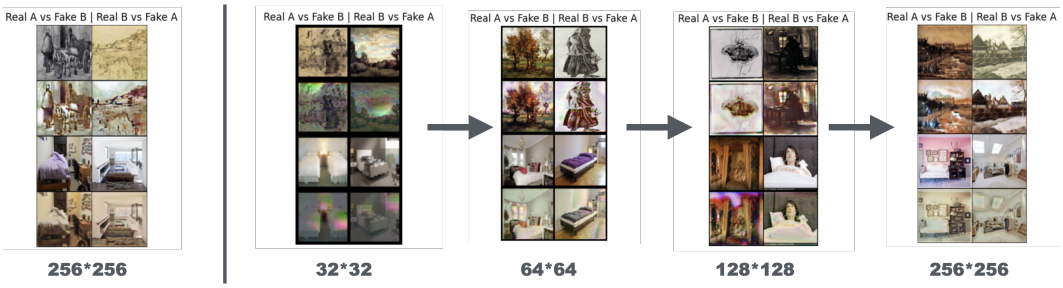


Figure 6: VanGogh2Lsun:None Progressive and Progressive Training

### 4.2 Quantitative Analysis

By analysing FID and Inception Score, we can find that non-progressive training often performs better than progressive training. However, the overall FID score for both models are not good.

We believe that the reason is as follows: due to the limitation of computational resources, the two datasets of each pair have only a few hundreds images selected. And FID measures the similarity of distributions. The smaller dataset cannot represent the real sample distribution. And the smaller dataset provides fewer samples will lead to inaccurate estimation of mean and covariance, thus increasing the instability of FID calculation. For the same reason, Inception Score are generally lower due to unstable statistical estimates and underrepresentative samples for small samples.

Table 1: FID (Fréchet Inception Distance)

| Style | Dataset | Non-Progressive | Progressive |
|---|---|---|---|
| Monet | CIFAR10 | 152.03 | 175.25 |
| | CelebA | 442.81 | 440.92 |
| | LSun | 272.53 | 305.44 |
| Van Gogh | CIFAR10 | 198.12 | 204.66 |
| | CelebA | 409.37 | 418.93 |
| | LSun | 434.75 | 432.81 |

Table 2: Inception Score

| Style | Dataset | Non-Progressive | Progressive |
|---|---|---|---|
| Monet | CIFAR10 | 1.71 | 1.70 |
| | CelebA | 2.03 | 2.32 |
| | LSun | 1.94 | 1.75 |
| Van Gogh | CIFAR10 | 1.63 | 1.59 |
| | CelebA | 1.83 | 1.60 |
| | LSun | 2.30 | 2.11 |

## REFERENCES

Gabriel Goh Scott Gray Chelsea Voss Alec Radford Mark Chen Aditya Ramesh, Mikhail Pavlov and Ilya Sutskever. Dall·e: Creating images from text. `https://openai.com/dall-e`, 2021.

Chris Hallacy Aditya Ramesh Gabriel Goh Sandhini Agarwal Girish Sastry Amanda Askell Pamela Mishkin Alec Radford, Jong Wook Kim and Jack Clark et al. Learning transferable visual models from natural language supervision. *arXiv preprint arXiv:2103.00020*, 2021.

Mehdi Mirza Bing Xu David Warde-Farley Sherjil Ozair Aaron Courville Ian Goodfellow, Jean Pouget-Abadie and Yoshua Bengio. Generative adversarial networks. *Communications of the ACM*, 63(11):139–144, 2014.

Phillip Isola Alexei A. Efros Yee Whye Jun-Yan Zhu, Taesung Park. Unpaired image-to-image translation using cycle-consistent adversarial networks.

Dominik Lorenz Patrick Esser Björn Ommer Robin Rombach, Andreas Blattmann. High-resolution image synthesis with latent diffusion models. *arXiv preprint arXiv:2112.10752*, 2021.

Samuli Laine Tero Karras and Timo Aila. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4401–4410, 2019.

Samuli Laine Tero Karras, Timo Aila and Jaakko Lehtinen. Progressive growing of gans for improved quality, stability, and variation. *arXiv preprint arXiv:1710.10196*, 2017. URL `https://arxiv.org/pdf/1710.10196`.