Applied Intelligence (2023) 53:27001–27026
https://doi.org/10.1007/s10489-023-04807-x

How to train your pre-trained GAN models

Sung-Wook Park[1] · Jun-Yeong Kim[1] · Jun Park[1] · Se-Hoon Jung[2] · Chun-Bo Sim[1]

Accepted: 18 June 2023 / Published online: 31 August 2023

Abstract

Generative Adversarial Networks (GAN) show excellent performance in various problems of computer vision, computer graphics, and machine learning, but require large amounts of data and huge computational resources. There is also the issue of unstable training. If the generator and discriminator diverge during the training process, the GAN is subsequently difficult to converge. In order to tackle these problems, various transfer learning methods have been introduced; however, mode collapse, which is a form of overfitting, often arises. Moreover, there were limitations in learning the distribution of the training data. In this paper, we provide a comprehensive review of the latest transfer learning methods as a solution to the problem, propose the most effective method of fixing some layers of the generator and discriminator, and discuss future prospects. The model to be used for the experiment is StyleGAN, and the performance evaluation uses Fréchet Inception Distance (FID), coverage, and density. Results of the experiment revealed that the proposed method did not overfit. The model was able to learn the distribution of the training data relatively well compared to the previously proposed methods. Moreover, it outperformed existing methods at the Stanford Cars, Stanford Dogs, Oxford Flower, Caltech-256, CUB-200–2011, and Insect-30 datasets.

Keywords Deep learning · Generative adversarial networks (GAN) · Computer vision (CV) · Artificial intelligence (AI)

1 Introduction

Generative Adversarial Networks (GAN) have been successfully applied to various applications in computer vision, computer graphics, and machine learning [1–7]. However, it is not easy to apply the GAN to an actual scene, because the recently announced GAN requires large data and huge computational resources. Several methods have been proposed to solve this problem. One method transfers the knowledge of a well-trained model, while another method acquires meta knowledge for quick adaptation to the target domain [8–13]. One auxiliary task facilitates training, while another task improves the inference procedure of the suboptimal model [14–21]. A priori distributions with expressive expressions, active selection of samples to provide supervision for conditional synthesis, or active sampling of mini-batches for training can also be used [22, 23]. The a priori distribution is the distribution of parameters that are already known, while the a posteriori distribution is the distribution of the parameters changed by the sample, that is, the answer to be found.

Transfer learning is the most promising way to train models on limited data and resources [24]. The recent success of deep learning utilizes a backbone pre-trained with supervised or self-supervised learning for large datasets [25, 26]. Self-supervised learning is a machine learning method that can be considered an intermediate form between supervised learning and unsupervised learning. As a type of autonomous learning using artificial neural networks, sample data classified in advance by humans is not necessarily required, and it is achieved by training the neural network in two steps. Another way is to after successfully transmitting the classifier in the recognition task, use the well-trained backbone of the GAN for downstream synthesis. Downstream is data transmitted from the upper layer to the lower layer. For example, the convolutional layer output of a discriminator can be used as a feature extractor, and a linear model, such as a support vector machine (SVM), can be combined as a classifier. GAN frequently experience mode collapse, which

✉ Se-Hoon Jung
shjung@scnu.ac.kr

✉ Chun-Bo Sim
cbsim@scnu.ac.kr

1 Interdisciplinary Program in IT-Bio Convergence System, Sunchon National University, Suncheon 57922, Republic of Korea

2 Department of Computer Engineering, Sunchon National University, Suncheon 57922, Republic of Korea

is a form of overfitting, for a variety of reasons [27–29]. For these reasons, even the transfer learning methods proposed so far may not be robust enough to handle the problems of overfitting or small distribution shift when applied to GAN. Therefore, we comprehensively review the latest transfer learning methods as part of a solution to the problem and propose a simple yet highly effective transfer learning method for GAN.

The lower layer of the GAN using convolutional operation learns the general features of the image, while the upper layer trains how to classify whether the image is real or synthetic, based on the extracted features. The dichotomous view of feature extractor and classifier and fixing the feature extractor for fine-tuning are not new, but the effectiveness of the proposed method using various datasets is verified by comparison with the existing method.

The model uses StyleGAN pre-trained with the Flickr Faces High Quality (FFHQ) dataset [30]. The datasets are Stanford Cars, Stanford Dogs, Oxford Flower, Caltech-256, Caltech-University of California San Diego Birds (CUB)-200–2011, and Insect-30.

The contributions of this paper are as follows:

- It is possible to check how various transfer learning methods affect the GAN model training results.
- By providing experimental results of various datasets, service improvement and performance in related fields can be achieved.
- We analyzed the issues with major GAN evaluation metrics and suggested careful consideration in the selection of evaluation metrics as an appropriate solution.

A comprehensive review of the latest transfer learning methods can provide valuable insights for future research and identify areas for further investigation, making a significant contribution to the academic community. While the proposed methods, such as selecting the appropriate combination of Freezing for the generator and discriminator, may vary depending on the architecture and dataset characteristics, we offer insights that can minimize experimentation and trial-and-error to determine the most effective method in a given scenario.

Section 1 of this paper describes the problems and solutions of GAN. Section 2, we introduced the latest transfer learning methods of GAN, such as fine-tuning, freezing, scale/shift, Generative Latent Optimization (GLO), MineGAN, L2-Starting Point (SP), and Feature Distillation (FD). Section 3 describes the dataset and experimental environment configuration. Section 4 compares the experimental results to see how the proposed method differs from the previously published method. Section 5 provides a final summary of what has been described above, and the prospects for future work. The last section includes the results of the ablation study of the proposed method, and images synthesized by various transfer learning methods.

2 Methods

Transfer Learning is a method that takes trained weights from a specific dataset of tens of thousands, and uses them in a user's project; it is mainly useful when the size of the dataset is small. Currently, in various fields, such as computer vision and natural language processing, the prediction rate is increasing with transfer learning methods.

In the case of Convolutional Neural Network (CNN), training starts by identifying which pixel combination is a line, and which type of group becomes a plane. If image discrimination starts without any information, training takes some time, so the transfer learning method is used. Transfer learning first loads an existing network trained on a large dataset. After that, the front side of the CNN is filled with the loaded network, and connected with the user project in the back layer. Finally, the above two networks are tuned, so that they mesh well. GAN is also like CNN except that it has two neural networks, and the mechanism is shown in Fig. 1. We assume that the pre-trained generator and discriminator can be fully utilized and compare it with the previously proposed method.

Fine-tuning: First, fine-tuning is the simplest and most effective way to transfer trained knowledge and initializes the parameters of the target model with the pre-trained weights of the source model. The fine-tuning method retrains the weights of all layers of the neural network. That is, the method sets the weights of the pre-trained model as initial values, and retrains from the beginning. At this time, it is important to set the learning rate low. If the learning rate is high, it is difficult to expect good performance, because the existing weights of the model are greatly damaged. Because the learning rate is low and the target to be learned is changed, the fine-tuning method takes more time to train the parameters of the output layer than the freezing method. The fine-tuning method can be implemented by passing the variable list of all layers to the optimizer. Note that it often suffers from overfitting, and requires regularization. Regularization is a technique for solving an ill-posed problem or preventing overfitting in machine learning; and an ill-posed problem means a rogue condition problem, in which there is no single correct answer. In this paper, we compare the performance when the entire layer is fine-tuned, and when only a part of the layer is fine-tuned; and propose a method of fine-tuning only some layers. In general, the deeper the layer, the more specialized the CNN.

The first few layers learn simple, general features that can be considered in any type of image, and the higher the layer,

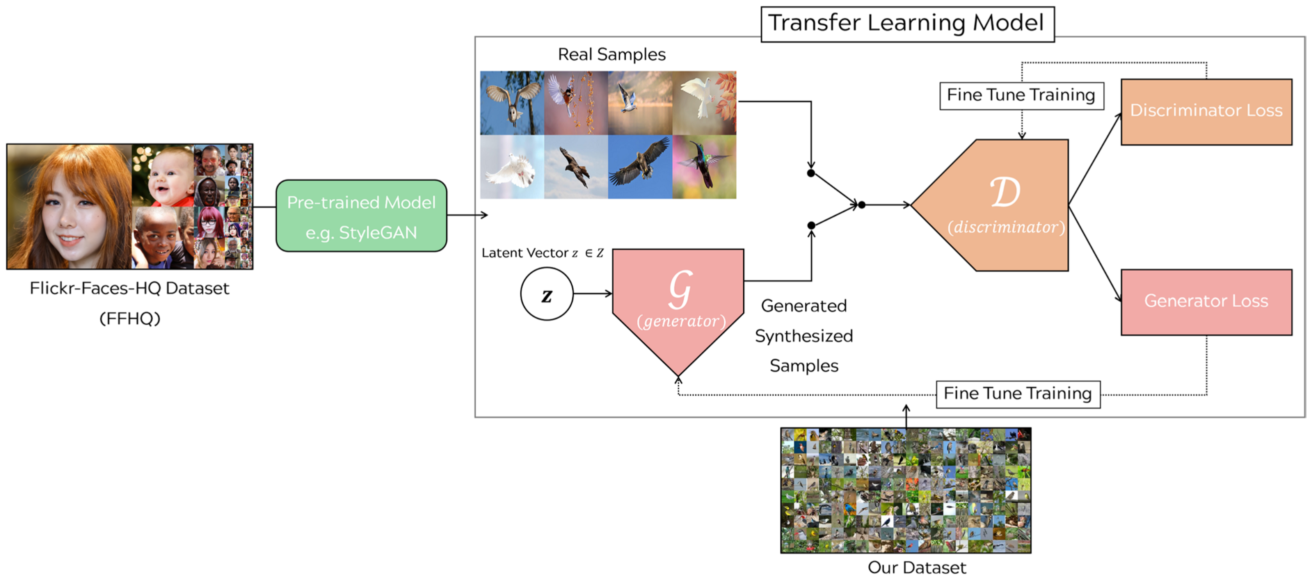
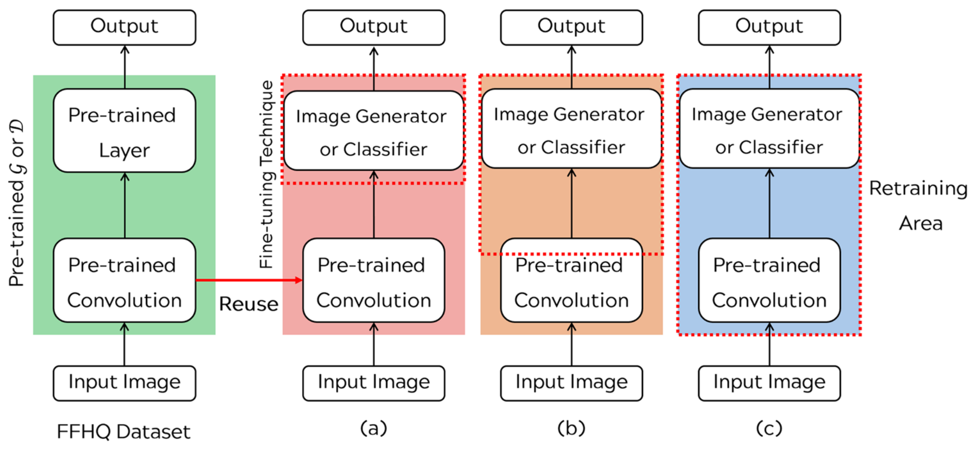

Transfer Learning Model
Real Samples
Fine Tune Training
Discriminator Loss
$\mathcal{D}$ (discriminator)
Pre-trained Model e.g. StyleGAN
Flickr-Faces-HQ Dataset (FFHQ)
Latent Vector $z \in Z$
z
$\mathcal{G}$ (generator)
Generated Synthesized Samples
Generator Loss
Fine Tune Training
Our Dataset

Fig. 1 Architecture and the principle of transfer learning

the more specific the unique features found in the training dataset. The goal of fine-tuning is to tune these unique features so that they work well on new datasets, without being overwritten by general features. Since the weights within the layers are set at random, trying to train all layers will cause the gradient updates to become too large, and the pre-trained model may forget what it has learned. Therefore, to reduce the possibility of learning failure, we propose a method of fine-tuning only some layers. The fine-tuning method is shown in Fig. 2.

In general, if a dataset is large and the similarity to the pre-trained model is small, Fig. 2c is used. Due to the large dataset size, retraining the entire model would be a good strategy. If the dataset is large and the similarity to the pre-trained model is large, Fig. 2b is used. Figure 2b describes the case of training the rear part of the convolutional layer and the image generator or classifier. Since the dataset is similar, it is possible to achieve an optimal performance even if only the latter part of the convolutional layer, which shows strong features, and the image generator or classifier is newly trained rather than training the whole. Figure 2b is also used when the dataset is small and the similarity to the pre-trained model is small. Because of the small amount of data, applying a fine-tuning technique to some layers might not be effective. Therefore, it is necessary to appropriately set the level of the convolutional layer to be newly trained. If the dataset is small and the similarity to the pre-trained model is large, Fig. 2a is used. Figure 2a describes the case where only the image generator or classifier is trained. Since the dataset is small, applying the fine-tuning technique to many layers can lead to overfitting. In this paper, the performance of the Fig. 2b method was the best. In the case of fine-tuning, if a large change is made to a parameter during the parameter update process,

Fig. 2 Types of fine-tuning techniques



an overfitting problem might occur. Thus, a sophisticated and fine parameter update is required.

Freezing The freezing method is also widely used, and the weights up to the output layer are fixed and reused. The convolutional layer responsible for feature extraction of the existing pre-trained model is trained to detect low-level features in the image, so it is useful for other image classification and synthesizing tasks. Therefore, the weights of the corresponding layer are fixed to their initial values, and then reused. This method is the freezing method. If the freezing method is used, it is easy to train the weights of the output layer, because the object to be learned does not change. At this time, the output layer of the original model may be different from the new work, and the number of output units is not the same, so it is usually added after deletion. The freezing method can be implemented by passing the list of variables of the output layer to be trained to the optimizer, excluding the variables of the corresponding layer.

Scale/shift Scale/shift is a method that updates the normalization layer only when the weights are frozen. It uses the knowledge of the pre-trained network to learn small datasets in other domains. As a representative normalization layer, there is batch normalization. The model synthesizes images using knowledge that cannot be acquired from small datasets. This method focuses on batch statistics of the network hidden layer, scale, and shift parameters. The network is trained stably while updating parameters in a supervised learning method. Even with small datasets, high-quality images can be synthesized and classes or domains can be added to the pre-trained network while maintaining the performance of the source domain. This means that the diversity of filters obtained in the pre-trained network is important for the performance of the target domain. However, if movement between the source and target distribution is frequent, inferior results can occur due to restrictions.

Generative Latent Optimization (GLO) GLO defines the loss as the sum of the $L2$ loss and the perceptual, fine-tuning the generator with a supervised learning method, and optimizing the generator and the latent code together to avoid overfitting [31]. The perceptual loss is defined using the feature reconstruction loss and the style reconstruction loss [32]. A pre-trained Visual Geometry Group (VGG)-16 model is used to obtain the feature reconstruction loss and the style reconstruction loss. The purpose of the feature reconstruction loss is to make the same content and the style reconstruction loss the same style. The feature reconstruction loss is calculated by comparing the activation map passed through the activation function existing for each layer. At this time, by making the low-dimensional expression synthesized by the convolution operation similar, we intend to synthesize a perceptually similar image that looks similar when we accept it, even if it is not exactly the same. Style reconstruction can identify which features are active at the same time as each other. GLO can perform the linear arithmetic operations with latent vector z without adversarial optimization. Setting the latent vector z value during training is important because it tracks the correspondence between the sample and its representatives. Since the image synthesized with a meaningful latent code in GLO matches the real image, the generator can generalize the image by interpolation. However, since there is no prior knowledge of adversarial loss and source discriminator, blurry and ambiguous images can be synthesized. For example, with the Large-scale Scene UNderstanding (LSUN) bedroom dataset, the adversarial optimization method performed better than the GLO. Of course, the visual quality can be improved by changing the loss function, architecture, progressive synthesis, post-training sampling method, and so on. This characteristic of GLO can be a strength in terms of possibility. However, it can also be interpreted as being sensitive to hyperparameter settings. Thus, it can also be considered a weakness.

MineGAN MineGAN can mine the most meaningful knowledge of a specific target domain from single or multiple pre-trained GAN. The task is performed using a minor network that identifies which parts of the pre-trained GAN synthesis distribution output the closest samples in the target domain. That is, it is based on a mining task that identifies regions of the GAN manifold that have been trained closer to a given target domain. Mining adjusts the GAN sampling to an appropriate region of latent space to facilitate post-mortem fine-tuning and avoid problems such as mode collapse and training instability. For this reason, mining can make efficient fine-tuning even when there are few images of the target domain. The minor network consists only of fully connected layers. In the paper, it is optimal when four layers are stacked. MineGAN can be effective when the source and target distributions share support. However, it can be difficult to generalize when the source and target distributions do not share support. Support is a support set whose function is a closure of a set of non-zero points, and a closure is the smallest closed set containing a subset of a given topological space. Here, the phase space is a space that does not contain information about the distance, area, volume, etc. between points, and a closed set is a concept that contradicts an open set, a subset of the phase space, that does not contain any boundaries of its own.

L2-Starting Point (SP) L2-SP is a mechanism to keep features learned from the source dataset well. The target model is regularized so that it does not stray too far from the source model [33]. The source model and target model parameters use $L2$-norm regularization. The L2-SP grants an $L2$ penalty based on the starting point, which is more effective in learning than a typical $L2$ penalty. It also has the advantage of possessing an explicit inductive bias since it can explicitly promote similarity between the initial model and the final solution in this way. Inductive bias aims to construct an algorithm that can learn how to predict a specific target output. It is noteworthy that the early stopping algorithm may work before convergence. However, the L2-SP is robust against such problems. Furthermore, it is easier to implement than freezing the initial layer in the network. We tried L2-SP as a generator, a discriminator, and both, but the results were not good. However, since freezing layers can be seen as providing an infinite weight of L2-SP for the selected layer and weight of '0' for the unselected layer, it is considered that the appropriate weight for each layer can perform better.

Feature Distillation (FD) FD is also one of the most widely used methods for the transfer learning of classifiers [34, 35]. We distill the activation of the source model and the target model, and Fig. 3 shows the operation principle of vanilla FD.

In Fig. 3, the teacher model is a large and deep model with high prediction accuracy, while the student model is a small and shallow model that will receive the features of the teacher model. L_{Soft} in Fig. 3 can be defined by Eq. (1), and L_{Task} can be calculated by Eq. (2)

$$L_{Soft} = \sum_{x_i \in X} KL\left(softmax\left(\frac{f_T(x_i)}{\tau} \right), softmax\left(\frac{f_S(x_i)}{\tau} \right) \right) \tag{1}$$

$$L_{Task} = CrossEntropy(softmax(f_S(x_i)), y_{true}) \tag{2}$$

In Eq. (1), $f_T(x_i)$ is the logit value of the teacher model, and $f_S(x_i)$ is the logit value of the student model, while τ is a hyperparameter that plays a scaling role. As τ is larger, it has a softer probability distribution, and if it is 1, it is the same as the existing softmax function. The modified softmax function is equal to Eq. (3)

$$Softmax(z_i) = \frac{\exp(z_i/\tau)}{\sum_j \exp(z_i/\tau)} \tag{3}$$

That is, the method of distilling the features of a large model into a small model is to designate the final softmax output of the large model as a soft target and use it in the training process of the small model. As a result, the small model has both the soft target, which is the output of the large model, and the hard target, which is the existing label value. The soft target means the probability for each class. The hard target is the result of one-hot encoding the probability of each class. Therefore, FD is applicable to the discriminator model. Training proceeds by calculating the loss of both targets. A soft target is actually a value output by the model through training, and a value that the model can output sufficiently, so when training a new model, it can be adopted as a realistic target. Specifically, a large model is

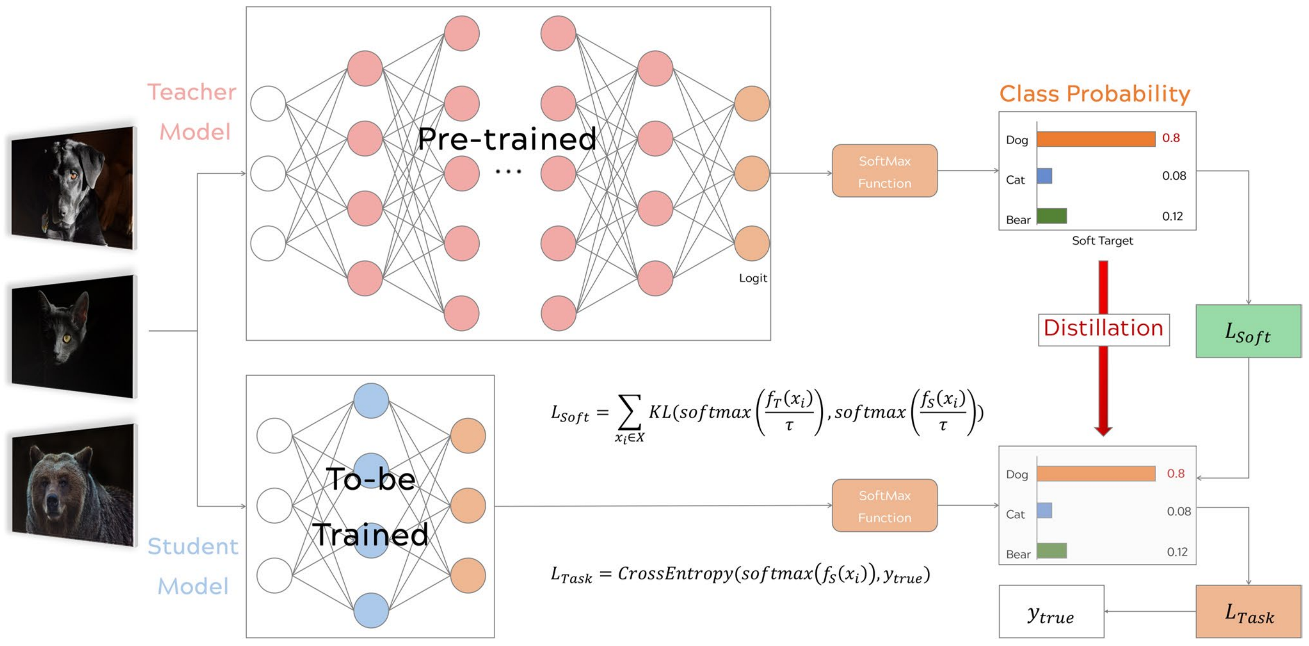


Fig. 3 The architecture and principle of vanilla feature distillation (FD)

trained using the training dataset, and after the large model has been sufficiently trained, a transfer dataset with the output as a soft target is synthesized.

Then, a small model is trained using the transfer dataset and the training dataset. Each loss function uses cross-entropy, and as a result, the final loss function of the smaller model is equal to Eq. (4) multiplied by the weight λ of L_{Soft}; λ is used as a form of feedback control. Feedback control is a control function that compares the output result with the target value in automatic control, returns it to the previous step, and corrects it. FD showed similar results to the proposed method, but the calculation took one more time.

$$L_{total} = L_{Task} + \lambda \bullet L_{Soft} \tag{4}$$

3 Experimental setup

3.1 Training and test environment

For hardware specification, the Central Processing Unit (CPU) used was Intel Core i9 11900 K Rocket LakeS; the graphics card was NVIDIA GeForce RTX 3090 24 GB, the RAM was G.Skill DDR4 64 GB, and the Solid State Drive (SSD) was FireCuda 530 Gaming Peripheral Component Interconnect express (PCIe) 4.0 Non Volatile Memory express (NVMe) 1 TB. Table 1 shows the hardware specifications for the experiment. We found that there was a limit in generating images with a resolution higher than 256×256 due to hardware limitations.

For software specification, the operating system was ubuntu 20.04.3 Long Term Support (LTS); the Compute Unified Device Architecture (CUDA) was 11.2.67, the cuda Deep Neural Network library (cuDNN) was 8.1.0, Torch was 1.8.2, and python was 3.8.10. Torch is a framework for machine learning and deep learning. Table 2 presents the software specifications for the experiment:

3.2 Model architecture

StyleGAN is the first model to use a combination of Progressive Growing of Generative Adversarial Networks (PGGAN)

Table 1 Hardware specifications

Hardware	Specifications
CPU	Intel Core i9 11900 K
Graphics card	NVIDIA RTX 3090 24 GB
RAM	G.Skill DDR4 64 GB
SSD	FireCuda 530 Gaming PCIe4.0 NVMe 1 TB

Table 2 Software versions

Software	Version
Operating system	Ubuntu Linux 20.04.3 LTS
Programming language	Python 3.8.10
GPGPU	CUDA 11.2.67
Deep neural network library	cuDNN 8.1.0
Deep learning framework	Torch 1.8.2

and neural style transfer technology [36, 37]. Its solution has been recognized and widely used. Of course, there are currently several applied models [38, 39]. However, in the case of the initial model, the number of layers that can be frozen in the generator and the discriminator is the same. Thus, an accurate comparative analysis is possible. For this reason, in this paper, StyleGAN was selected as the transfer learning method comparison model. StyleGAN drew attention for generating full high-definition quality results with few steps of control from detail to overall image. Figure 4 shows the generator architecture of StyleGAN:

A in Fig. 4 is a learned affine transformation. StyleGAN proposed a method called Adaptive Instance Normalization (AdaIN), and AdaIN uses reference style bias $y_{b,i}$ and scale $y_{s,i}$. The mean and variance of the feature map x_i output from the layers in the synthesis network are adjusted using $y_{b,i}$ and $y_{s,i}$, respectively. AdaIN is given by Eq. (5):

$$AdaIN\left(x_i, y\right) = y_{s,i}\frac{x_i - \mu(x_i)}{\delta(x_i)} + y_{b,i} \tag{5}$$

A latent vector z is passed through the mapping network f to compute the style parameters, and an intermediate vector w is generated. Then, it passes through the fully connected layer, and synthesizes $y_{b,i}$ and $y_{s,i}$ vectors of length n. This is to separate the image style selection process. AdaIN prevents style information from being lost between layers. The style vector added to each layer does not affect the feature of other layers. This latent vector w is better than the original vector z.

Using AdaIN, StyleGAN learns about interpretable disentangled representations by solving the problem of entanglement in latent space. Generative models aim to capture generative factors in the training data. A disentangled representation is associated with a symmetry transformation in which some properties are preserved, while other properties are changed. Symmetry transformation transforms certain properties, but preserves others. To realize a symmetry transformation in a neural network, neurons must have no connections with other neurons. That is, each neuron is in an isolated state. The concept of symmetry is broader than the scope of geometry and is mainly used in quantum mechanics.

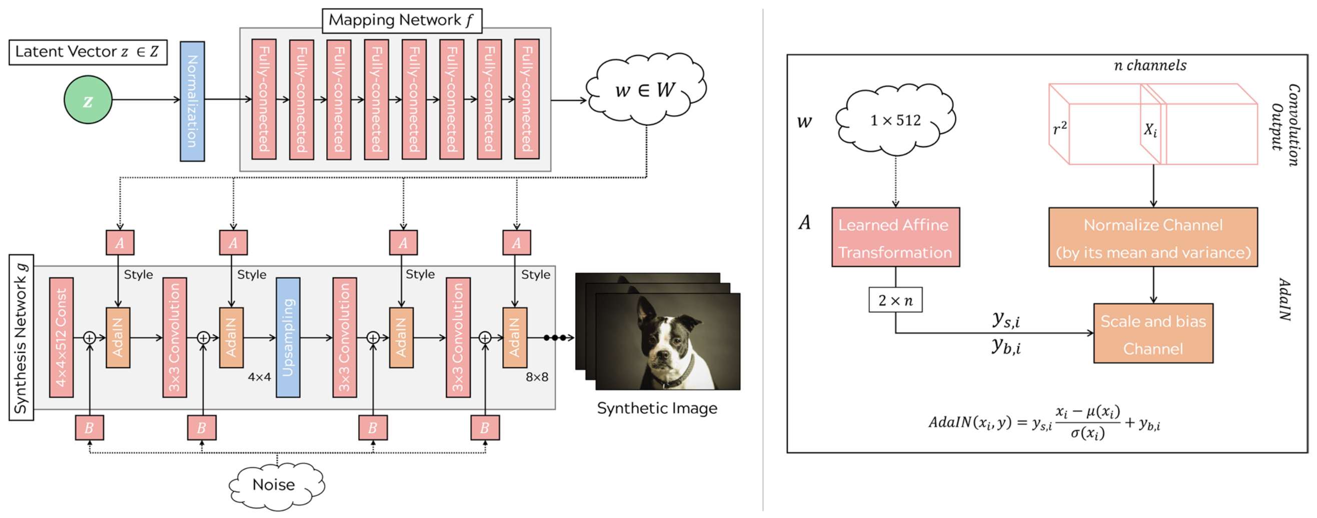


Fig. 4 Generator *G* architecture of StyleGAN and Adaptive Instance Normalization (AdaIN) layer mechanism

The disentangled representation is the process of learning symmetry through training, and becoming disentangled, even starting from the fully connected layer. This means that the latent unit is sensitive to changes in the generative factor. From the point of view of information theory, disentangled representation is highly useful. Because it compresses information, it is more efficient than other algorithms, and this is because small things can be increased into many things. However, disentangled representation is only effective for latent vectors.

The synthesis network was designed with inspiration from PGGAN. The more the style vector of the synthesis network is in the front layer, the larger the feature is. StyleGAN completely controlled the synthesized image using the latent vector *w*, And by changing the position of the *w* vector in the synthesis network, different levels of style were synthesized.

StyleGAN passes the *w* vector of *I* through the synthesis network to combine different images *I* and *I*′, and turns it into an *I*′ vector at a specific point. *I* and *I*′ were synthesized with different vectors. When the transformation occurs in the early stages, the posture, appearance, and styles, such as glasses, are transmitted to the *I*. If the transformation occurs later, styles, such as the color and micro-shape of a face, are transferred to *I*′. All *I* image features are maintained. StyleGAN adds noise behind each convolutional layer to capture parts such as the position of the hair or the background of the face. The noise injection location determines the fineness and coarseness of the image.

3.3 Dataset description

We used the pre-trained StyleGAN model with the FFHQ dataset, and fine-tuned our six datasets: Stanford Cars, Stanford Dogs, Oxford Flower, Caltech-256, CUB-200–2011, and Insect-30. There are 196 classes in the Stanford Cars dataset, with a total of 8,144 images, while the Stanford Dogs dataset has 120 classes and a total of 20,580 images. There are 102 classes in the Oxford Flower dataset, a total of 8,189 images, and 256 classes in the Caltech-256 dataset, with a total of 30,609 images embedded. There are 200 classes in the CUB-200–2011 dataset, a total of 11,788 images, while the Insect-30 dataset has 30 classes, a total of 28,896 images.

In the case of the Insect-30 dataset, 30 species of forest insects that are commonly observable were selected. Five types of images from the ImageNet and 25 types of images through Screen Scraping were collected and organized into datasets through a separate screening process. Screen scraping is a program designed to extract only necessary data from data displayed on the internet screen. The image of the insect dataset was cropped around the insect in the image, as Fig. 5 shows:

The model used was trained on 256 × 256 images, and the iteration was maintained at 50,000. Learning was

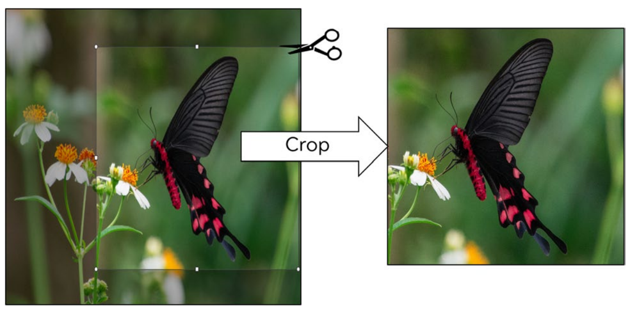


Fig. 5 Example of a cropped insect image

successful even without progressive training. Progressive training is a method of synthesizing high-quality and high-resolution images by adding a new layer in the training process of the generator and discriminator. We train the generator and discriminator from a 4×4 pixel low-resolution image. After that, the resolution is increased by adding a layer, and the added layer is continuously trained without being frozen.

3.4 Implementation details

When injecting label information into the StyleGAN architecture generator, a conditional version of AdaIN was used. When injecting label information into the discriminator, the PGGAN projection discriminator was used. The loss function used logistics and the activation function used softplus. The softplus function is a variant of the Rectified Linear Unit (ReLU) that can relax the criteria for creating zero. It can be differentiated across all intervals. Additionally, exponential moving average (EMA) was used for generator updates.

When using the L2-SP or FD method, the loss was defined as the sum of the existing loss and the Mean Squared Error (MSE). The supervised loss was defined as the sum of $L2$ loss and perceptual, with the perceptual and embedding scales set to be 0.1, the regularizer scale set to be 0.02, and the image and perceptual normalization set to be True.

For the optimizer, Adaptive moment estimation (Adam) was used. Adam can perform deflection correction of hyperparameters by fusion of momentum and Adaptive Gradient algorithm (AdaGrad). The initial learning rate was set to 0.002. The coefficient for primary momentum β_1 was set to 0.0. The coefficient for secondary momentum β_2 was set to 0.99. Epsilon was set to 1e-08. Weight decay was set to 0. AMSGrad was set to False. Foreach was None. The maximum was set to False. The capability was also set to False. For weight initialization, the linear layer used the Xavier normal distribution, and the convolutional layer used Kaiming normal distribution.

In a standard normal distribution with a mean of 0 and a variance of 1, the latent vector z is sampled and the z size is set to 512. The random seed was set to 0. The mini-batch was set to 8. The image size was set to 256. The number of samples for evaluation was set to 5,000. The number of samples used for each training phase was set to 50,000. The basic step size was set to 6. The step size for evaluation was set to 1,000 and the step size for model save was set to 10,000.

In this paper, the above settings were applied equally to all experiments in order to secure objectivity when drawing conclusions.

3.5 Evaluation metrics

Evaluation indicators such as how to evaluate the synthesized image and whether the trained model can be compared with other models may vary depending on the learning goal [40]. Objective functions of generators and discriminators in GAN are measured by comparing how well they each perform their roles. For example, a particular objective function measures how well a generator deceives a discriminator. Methods of comparing the results of GAN models include the Inception Score (IS), and the Fréchet Inception Distance (FID) [41, 42].

IS represents two performances of GAN. The first is the quality of the synthesized image, and the second is the diversity. A good result is that the conditional probability $p(y|x)$ is easy to predict. That is, when an image is input, it should be possible to easily identify the type of object. IS classifies the synthesized image using the InceptionV3 model, and predicts $p(y|x)$. Here, y is the label, and x is the synthesized image. This reflects the quality of the image. Then, $p(y)$ is the marginal probability calculated as in Eq. (6):

$$\int_z p(y|x = G(z))dz \tag{6}$$

Marginal probability is the probability distribution of X or Y when two random variables X and Y pair, and have a joint probability distribution as (X, Y). Equation (6) eliminates the remaining probabilities through integral or summation. If the images synthesized in Eq. (7) are diverse, the data distribution for y should be uniform. That is, it must have high entropy:

$$IS(G) = exp(E_{X \sim p_{data}} D_{KL}(p(y|x) \parallel p(y))) \tag{7}$$

When synthesizing only one image per class, IS may misrepresent performance. This is because $p(y)$ can still be uniform, even with low diversity.

Introduced in 1957 by the French mathematician Maurice René Fréchet, FID was inspired by the metric. FID also uses Inception V3, and extracts features from the middle layer. FID models the data distribution of the extracted features using multi-variate normal distribution with mean μ and covariance matrix $\sum$ added. The lower the FID value, the better, because the image quality and diversity increase. FID is sensitive to mode collapse. So, the more similar images, the higher the value. FID is resistant to noise, and can detect missing samples within a class.

Because IS improves performance by synthesizing only one kind of class, FID is better than IS. FID calculates the distance between images in pixel space, and is equivalent to Eq. (8):

$$FID(x,g) = ||u_x - u_g||_2^2 + Tr\left(\sum_x + \sum_g - 2\left(\sum_x \sum_g\right)^{\frac{1}{2}}\right) \tag{8}$$

In Eq. (8), μ is the mean, $\sum$ is the covariance matrix, x is the real image, and g is the synthesized image. The Trace of a Matrix (Tr) is a summary of all elements of a diagonal, that is, the sum of the diagonals. In a covariance matrix, the diagonal is an element with the same row and column indices in a square-shaped matrix. The key is to assume that the vector passing through InceptionV3 follows a normal distribution. Without this assumption, the FID value cannot be calculated; and even so, it is not an accurate number. The FID can be transformed into the Fréchet Audio Distance (FAD) in music synthesis, the Fréchet Video Distance (FVD) in video synthesis, and the Fréchet ChemNet Distance (FCD) in molecular synthesis.

Precision, recall, and F1 score are also used as evaluation metrics [43]. The more similar the synthesized image is to the real image, the higher the precision. The more the generator synthesizes the samples from the training dataset without duplication, the higher the recall. Recall is also called hit rate, sensitivity, and true positive rate. The F1 score is the harmonic mean of precision and recall. The harmonic mean is the reciprocal of the arithmetic mean of n positive numbers and their reciprocals.

There are also coverage and density [44]. The author of the paper first forms a manifold with k-Nearest Neighborhood (k NN), before defining fidelity and diversity. In other words, fidelity and diversity are defined after assuming that the Euclidean distance that can contain k pieces of data closest to a specific vector V is formed as the corresponding data manifold. Density is a ratio indicating how much density of synthetic image is included in the manifold synthesized with real image. If the density value is large, the synthetic image is distributed at high density in the manifold and does not deviate significantly from the distribution of the original dataset. The density is given by Eq. (9):

$$Density := \frac{1}{kM} \sum_{j=1}^{M} \sum_{i=1}^{N} 1_{Y_j} \in B(X_i, NND_k(X_i)) \tag{9}$$

Coverage is a ratio indicating how many manifolds contain the synthesized image on multiple manifolds synthesized with the real image. If the model synthesizes images only in a part or a narrow space in the real image manifold, fidelity and diversity have a trade-off relationship; but if the synthesized distribution can cover the real image distribution as a whole, the trade-off problem can be solved. NVIDIA announced improved precision and recall, but claim that they are too sensitive to outliers, and that IS and FID cannot distinguish fidelity and diversity to evaluate, but density and coverage are possible [45]. The Coverage is given by Eq. (10):

$$Coverage := \frac{1}{N} \sum_{i=1}^{N} 1_{\exists j} s.t. Y_j \in B(X_i, NND_k(X_i)) \tag{10}$$

The IS, FID, precision, recall, F1 score, coverage, and density all use Inception V3, so it is a feature extraction-based methodology. If the model is not good at feature extraction, the same value can be calculated, no matter which image is synthesized.

Research is still ongoing regarding end-to-end evaluation metrics that can detect and prevent problems in GAN, such as earlier mode collapse, as well as cost optimization [46]. However, it is difficult to find a satisfactory solution. We use FID, coverage, and density, but we propose to find and apply an evaluation metrics suitable for the model to be used based on a theoretical basis, or through a lot of trial and error.

4 Results

Figure 6 visualizes the synthesized image using the existing weights and fine-tuned weights for the FFHQ dataset. The similar latent code shared the similar meaning after fine-tuning.

Table 3 evaluates the FID values of the existing and proposed methods of models trained with the Stanford Cars and Stanford Dogs datasets. For the Stanford Cars dataset, Freeze D fixed up to discriminator layer 5, and Freeze G fixed it up to generator layer 3. Partial fine-tuning method was fixed up to generator layer 4 and discriminator layer 4. For the Stanford Dogs dataset, Freeze D fixed up to discriminator layer 5, and Freeze G fixed it up to generator layer 2. Partial fine-tuning method was fixed up to generator layer 5 and discriminator layer 7. All tables in Appendix 6.1 shows the rationale for layer freezing. The weights of L2-SP and FD were selected from {0.1, 1, 10}, and in the case of regularization weights, '1' was set for all experiments. FD linearized the fifth activation of the discriminator, and matched the activation of the source and target discriminators. Since the activation size is different for each layer, $L2 - norm$ normalized to the feature dimension was used. The hyperparameter of GLO was set to the value proposed by the author of the paper, and for the minor network, a 2-layer Multi-Layer Perceptron (MLP) and Rectified Linear Unit (ReLU) were used as the activation functions.

In Stanford Cars, the partial fine-tuning method showed the highest performance with FID 10.84. The FID of MineGAN with GLO was 33.65, showing the lowest performance, and there was a performance difference between partial fine-tuning and 22.81. In Stanford Dogs, the Freeze G method showed the highest performance with an FID of

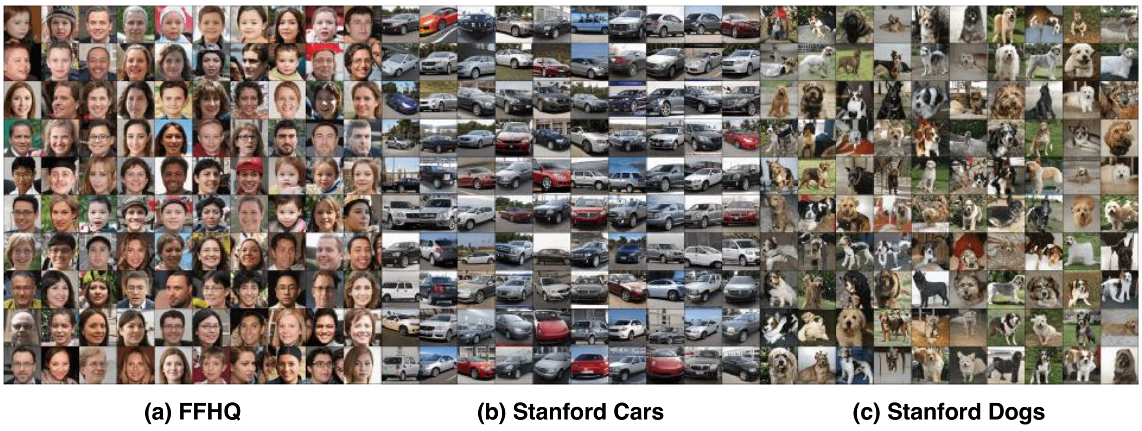


Fig. 6 (a) was synthesized by StyleGAN trained on the FFHQ dataset. (b) shows 100 car images synthesized by the partial fine-tuning method, which had the best performance in the ‘Stanford Cars’ dataset. (c) shows 100 dog images synthesized by the Freeze *G* method, which had the best performance in the ‘Stanford Dogs’ dataset. As a result of the analysis, we found that each item represented a similar latent code and that similar latent codes shared similar meanings even after freezing

29.83. The FID of scale/shift with GLO was 57.30, showing the lowest performance, and there was a performance difference between Freeze *G* and 27.47. While MineGAN could potentially generate high-quality images with specific attributes, such as poses and lighting, it may not be able to capture the full range of diversity present in the training data. There can be several reasons for the performance degradation of FID in scale/shift with GLO. Firstly, updating only the normalization layer while keeping the weights frozen can make it difficult for the model to learn the target dataset and result in unstable training. Secondly, there may be a more appropriate layer than the current normalization layer that can improve performance. Thirdly, hyperparameters such as learning rates may also contribute to performance degradation. Compared to methods other than fine-tuning, the proposed freezing and partial fine-tuning methods were both performance and stability solutions. We chose our method because FD has similar results to the proposed method, but it is twice as slow in speed.

Table 4 evaluates the coverage and density values of the previously published method and the proposed method of the model trained with the Stanford Cars and Stanford Dogs datasets. When evaluating the coverage of Stanford Cars, Freeze *D* fixed up to discriminator layer 5, and when evaluating density, Freeze *D* fixed up to discriminator layer

Table 3 Comparison of various methods under the ‘Stanford Cars’ and ‘Stanford Dogs’ datasets. Values indicate the best FID scores. ‘+’ indicates the styleGAN is trained by GLO loss For each value, the methods are marked with the best performance using gold ●, silver ●, and bronze ● medals

Method	Stanford Cars	Stanford Dogs
Fine-tuning	11.22	30.04
+GLO	25.68	55.65
Scale/shift	11.22	30.50
+GLO	28.45	57.30 ●
MineGAN	33.59	57.24
+GLO	33.65 ●	57.15
L2-SP (*G*)	11.23	30.32
L2-SP (*D*)	11.18	31.06
L2-SP (*G*, *D*)	11.30	30.76
FD	11.04	30.01 ●

Table 4 Comparison of various methods under the ‘Stanford Cars’ and ‘Stanford Dogs’ datasets. Left and right values indicate the best coverage and density scores. ‘+’ indicates the styleGAN is trained by GLO loss

Method	Stanford Cars	Stanford Dogs
Fine-tuning	0.939/7.695	0.246/16.412
+GLO	0.607/**14.845**	0.115/**30.949**
Scale/shift	0.938/7.197	0.241/15.901
+GLO	0.564/9.472	0.108/29.929
MineGAN	0.434/2.668	0.118/27.374
+GLO	0.431/2.714	0.118/27.360
L2-SP (*G*)	0.937/7.702	0.243/16.086
L2-SP (*D*)	0.934/7.443	0.235/15.450
L2-SP (*G*,*D*)	0.936/7.430	0.236/16.055
FD	0.940/7.661	**0.251**/16.161
Freeze*D*	0.941/7.642	0.246/17.396
Freeze*G*	0.943/7.837	0.250/17.833
Partial fine-tuning	**0.945**/7.552	0.246/18.129

Bold items mean the highest performance

6. Freeze G fixed up to generator layer 3 when evaluating coverage and fixed up to generator layer 5 when evaluating density. Partial fine-tuning fixed up to generator layer 7 and discriminator layer 4 when evaluating coverage and fixed up to generator layer 6 and discriminator layer 2 when evaluating density.

When evaluating the coverage of Stanford Dogs, Freeze D fixed up to discriminator layer 6, and when evaluating density, Freeze D fixed up to discriminator layer 1. In Freeze G, both coverage and density were fixed up to the generator layer 2 and evaluated. Partial fine-tuning fixed up to generator layer 7 and discriminator layer 1 when evaluating coverage, and fixed up to generator layer 1 and discriminator layer 6 when evaluating density. Table 3 shows the hyperparameter settings of the other methods that were set.

In Stanford Cars, partial fine-tuning showed the highest performance with a coverage of 0.945. The coverage of MineGAN with GLO showed the lowest performance at 0.431, and there was a performance difference between partial fine-tuning and 0.514. For density, fine-tuning with GLO showed the highest performance at 14.845. Density of MineGAN showed the lowest performance at 2.668, and there was a performance difference between fine-tuning with GLO and 12.177. In Stanford Dogs, FD showed the highest performance with coverage 0.251. The coverage of scale/shift with GLO showed the lowest performance at 0.108, and there was a difference in performance between FD and 0.143. For density, fine-tuning with GLO showed the highest performance at 30.949. Density of L2-SP(D) showed the lowest performance at 15.450, and there was a performance difference between fine-tuning with GLO and 15.499. If the size of the target dataset to be learned is smaller than the previously learned dataset, overfitting may occur. As a result, the synthesized images may not be diverse enough to represent the full range of the target dataset, leading to degraded coverage performance. Additionally, updating the normalization layer of GAN only in the freezing state can cause inconsistent normalization of inputs to the generator during training, leading to a degradation in coverage performance. These issues may arise because the normalization layer adapts to a specific distribution of the training data and may not be suitable for samples from the target dataset. If the previously learned dataset and the target dataset are significantly different, density performance may also be degraded. When using L2-SPs to control the distance between the target model and the source model, if the normalization strength is too weak, the target model may deviate too much from the source model during fine-tuning, leading to poor density performance.

Figure 7 shows the (a) FID, (b) Coverage, and (c) Density values for each epoch of the existing method and the proposed method. The early stopping algorithm was not used to observe how each evaluation metric changed until 50,000 epochs. In the first epoch of Fig. 7a, partial fine-tuning

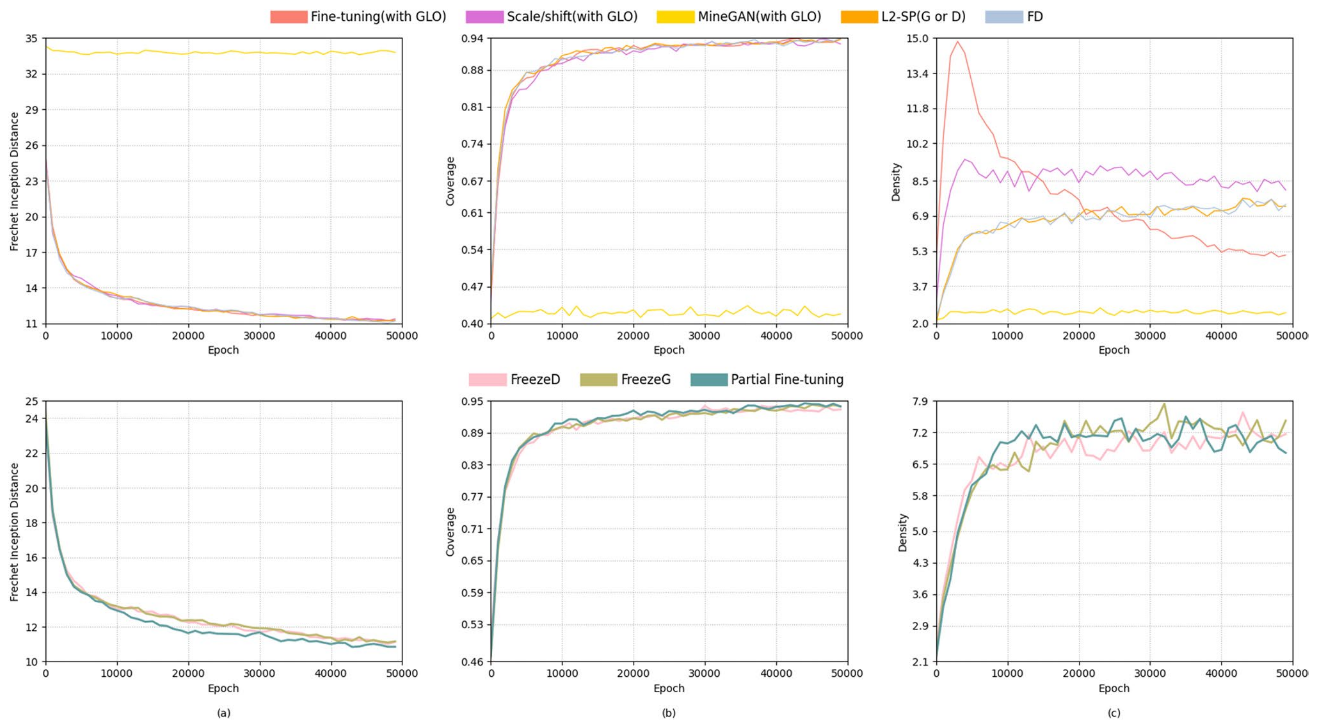


Fig. 7 Trends of FID, coverage, and density scores of the freeze-based methods in the Stanford Cars dataset

showed the highest performance with FID 23.89, and the training speed was faster than Freeze *D* and Freeze *G*. MineGAN showed the lowest performance with FID 34.32, and there was a performance difference between partial fine-tuning and 10.43. Even at 50,000 epochs, partial fine-tuning showed the highest performance at 10.85, while MineGAN showed the lowest performance at 33.79. There was a performance difference of 22.94 between partial fine-tuning and MineGAN. Looking at the graph change trend, the other methods, except MineGAN, showed a smooth upward trend up to 50,000 epochs.

At epoch 1 in Fig. 7b, Freeze *D* showed the highest performance with a coverage of 0.497, while MineGAN showed the lowest performance with a coverage of 0.409. There was a performance difference of 0.088 between Freeze *D* and MineGAN. FD and Freeze *G* started training with the second and fourth highest performance, respectively, but showed higher performance than partial fine-tuning at 50,000 epochs. The increase in value was 0.011 higher in Freeze *G* than in FD. MineGAN still showed the lowest performance with 0.418. There was a performance difference of 0.522 between Freeze *G* and MineGAN. Looking at the graph change trend, except for MineGAN, the other methods showed a smooth upward trend up to 50,000 epochs. The slow initial convergence of the FD-based freezing method may be due to insufficient capacity of the pre-trained generator to synthesize similar images within the target dataset, or limitations in the distillation loss used. In contrast, partial fine-tuning can lead to faster convergence because it only updates a fraction of the pre-trained generator.

At epoch 1 in Fig. 7c, fine-tuning with GLO showed the highest performance with a density of 4.769, while L2-SP(*G*) showed the lowest performance with a density of 2.008. There was a performance difference of 2.761 between fine-tuning with GLO and L2-SP(*G*). Fine-tuning with GLO showed the highest performance during epoch 1, but after a certain section, the performance continued to decline. Scale/shift with GLO started training with the second highest performance, but showed the highest performance of 8.076 at 50,000 epochs. MineGAN with GLO showed the lowest performance at 2.491. Scale/shift with GLO and MineGAN with GLO had a performance difference of 5.585. When using supervised learning loss, the presence of noise in the training data labels or mislabeling can have a negative impact on the performance. Figure 13a-h of Appendix 6.2 shows the image synthesized by the previously announced method.

Figure 8 shows the (a) FID, (b) Coverage, and (c) Density values for each epoch of the existing method and the proposed method. The early stopping algorithm was not used to observe how each evaluation metric changed until 50,000 epochs. In epoch 1 of Fig. 8a, partial fine-tuning showed the highest performance with FID 52.08, while MineGAN with GLO showed the lowest performance with

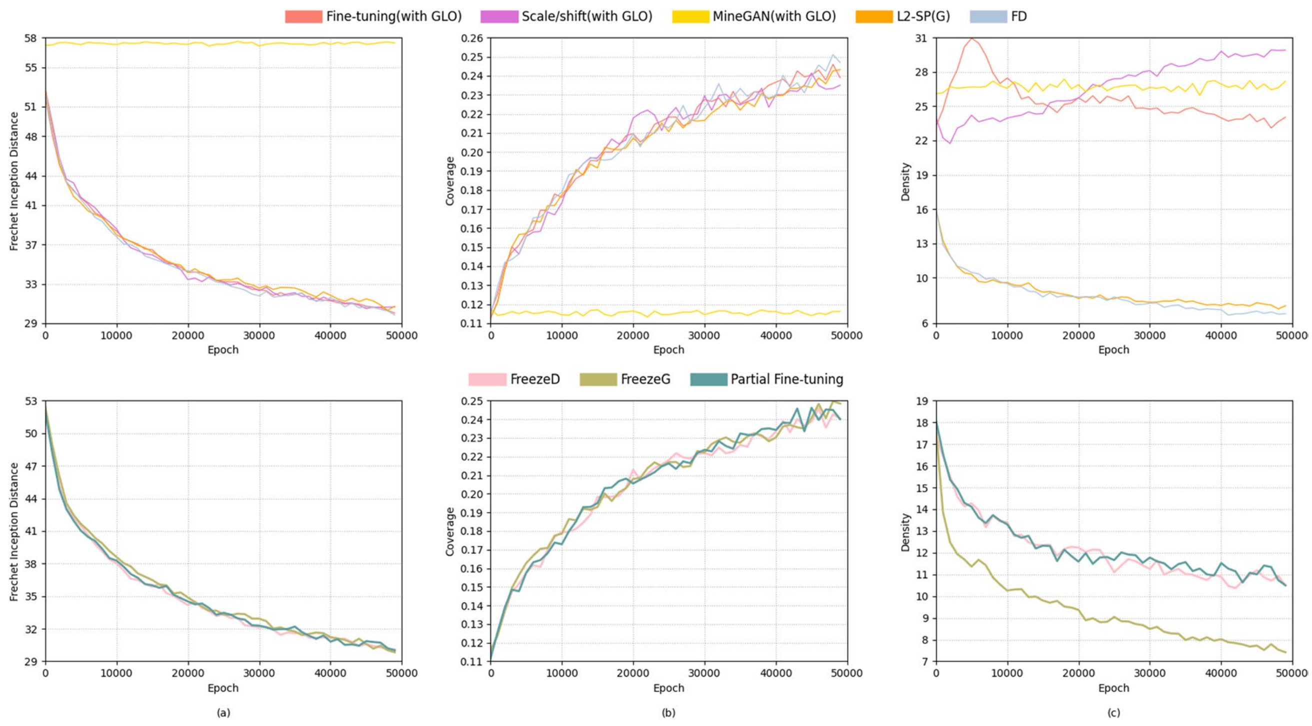


Fig. 8 Trends of FID, coverage, and density scores of the freeze-based methods on the Stanford Dogs dataset

FID 57.23. There was a performance difference of 5.15 between partial fine-tuning and MineGAN with GLO. Freeze *G* started training with the fourth highest performance, but had the highest performance of 29.83 at 50,000 epochs. MineGAN with GLO showed the lowest performance at 57.46; in contrast, the performance decreased by 0.23 compared to the 1st epoch. There was a performance difference of 27.63 between Freeze *G* and MineGAN with GLO. Because the pre-trained dataset may not represent the distribution of the dataset to be trained, the convergence speed of Freeze *G* may be slow enough, and the discriminator may have better learning capabilities than the generator, making it difficult for the generator to synthesize high-quality samples quickly. It is considered that MineGAN failed to train, and more efforts such as increasing the training data or selecting better initialization in the step-by-step hyperparameter tuning process may be necessary.

At epoch 1 in Fig. 8b, MineGAN showed the highest performance with a coverage of 0.118, while partial fine-tuning showed the lowest performance with a coverage of 0.111. There was a performance difference of 0.007 between MineGAN and partial fine-tuning. However, MineGAN's 50,000 epoch coverage was 0.116, which was 0.002 lower than the 1 epoch. Fine-tuning and Freeze *G* started training with the second highest performance, but at 50,000 epochs, Freeze *G* performed 0.009 higher. There was a performance difference of 0.132 between Freeze *G* and MineGAN.

At epoch 1 in Fig. 8c, MineGAN showed the highest performance with density 26.086, while L2-SP(*G*) showed the lowest performance with density 16.086. There was a performance difference of 10.0 between MineGAN and L2-SP(*G*). At epoch 1, MineGAN had the highest performance, but at epoch 50,000, scale/shift with GLO allowed inversion. Scale/shift with GLO started training with the second highest performance, but showed the highest performance of 29.907 at 50,000 epochs. FD showed the lowest performance at 6.835, while there was a performance difference of 23.072 with scale/shift with GLO. In the case of scale/shift with GLO, since no issues occurred when updating the normalization layer each time, it can be naturally assumed that it led to performance improvement.

Table 5 evaluates the FID values of vanilla fine-tuning and the proposed method. Vanilla means original, with no added ingredients. For Oxford Flower, Freeze *G* showed the highest performance with FID 14.13. Fine-tuning's FID was 14.51, showing the lowest performance, and there was a performance difference of 0.38 from Freeze *G*. CUB-200–2011 showed the highest performance in partial fine-tuning with FID 14.66. Fine-tuning's FID was 15.14, showing the lowest performance, and there was a performance difference of 0.48 from partial fine-tuning. For Caltech-256, Freeze *G* showed the highest performance with 33.30. Fine-tuning's FID was 33.59, showing the lowest performance, while there was a performance difference of 0.29 from Freeze *G*.

Table 5 FID scores under StyleGAN architecture. Values indicate the best FID scores. For each value, the methods are marked with the best performance using gold 🟡, silver ⚪, and bronze 🟤 medals

Method	Oxford Flower	CUB-200-2011	Caltech-256
Fine-tuning	14.51 🟤	15.14 🟤	33.59 🟤
Freeze*D*	14.23 ⚪	14.71 ⚪	33.39 ⚪
Freeze*G*	14.13 🟡	14.86	33.30 🟡
Partial fine-tuning	14.27	14.66 🟡	33.51

As a result of the overall analysis, the proposed method secures performance and stability, but not in the case of the Caltech-256 dataset. Caltech-256 was more difficult to learn than Oxford Flower and CUB-200–2011, because of the large shift in distribution during training. To train a dataset with large distribution variations, it is necessary to minimize the constrains of the model. The point is, when looking only at the FID values, the proposed method showed better stability than the fine-tuning method for all datasets. Figure 9 shows an Oxford Flower image synthesized by Freeze *D* and Freeze *G*, while Fig. 14q-t of Appendix 6.2 shows the CUB-200–2011 and Caltech-256 images.

Table 6 evaluates the coverage and density values of vanilla fine-tuning and the proposed method. For Oxford Flower, Freeze *D* and partial fine-tuning showed the highest performance with a coverage of 0.783. The coverage of Freeze *G* showed the lowest performance at 0.779, and there was a performance difference of 0.004 between Freeze *D* and partial fine-tuning. For density, fine-tuning showed the highest performance at 12.670. Freeze *D* showed the lowest performance at 11.746, and there was a performance difference between fine-tuning and 0.924. The statistical properties of the Oxford Flower dataset are different from those of the previously learned dataset. Therefore, fine-tuning is better able to adapt to the specific properties of the target dataset than the freezing method. The difference between the Oxford Flower dataset and the CUB-200–2011 and Caltech-256 datasets can be classified into five main categories: number of classes, number of images, image quality, object size and complexity, and annotation quality.

In CUB-200–2011, Freeze *D* showed the highest performance with a coverage of 0.708. The coverage of Freeze *G* was 0.693, showing the lowest performance, while there was a performance difference of 0.015 from Freeze *D*. As for density, Freeze *G* showed the highest performance with 4.133. Partial fine-tuning showed the lowest performance at

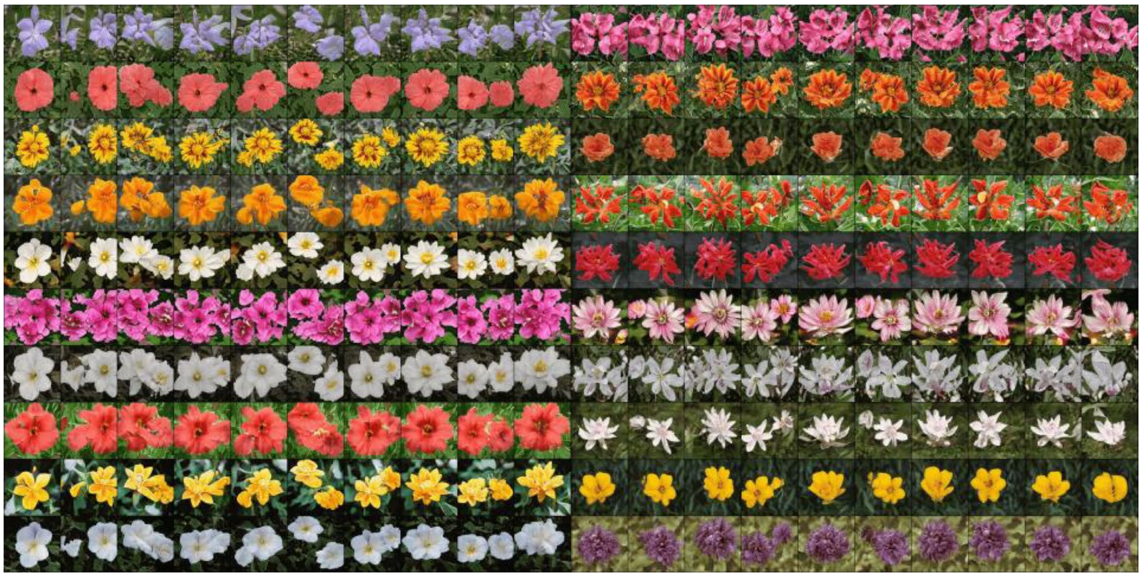


Fig. 9 Samples synthesized by StyleGAN trained by (**a**) Freeze *D*, and (**b**) Freeze *G*, under the Oxford Flower dataset. Each row indicates the same class. Freeze *G* generates more class-consistent samples than does Freeze *D*

3.977, while there was a performance difference of 0.156 with Freeze *G*. When the discriminator was frozen, the coverage performance was high, and when the generator was frozen, the density performance was high.

For Caltech-256, Freeze *G* showed the highest performance with a coverage of 0.361. Freeze *D* showed the lowest performance at 0.348, and there was a performance difference of 0.013 from Freeze *G*. For density, partial fine-tuning showed the highest performance at 59.717. Fine-tuning showed the lowest performance at 58.324, and there was a performance difference between partial fine-tuning and 1.393. The partial fine-tuning method demonstrated excellent density performance on datasets with large distribution shifts during training, such as Caltech-256.

Figure 10 shows the (a) FID, (b) Coverage, and (c) Density values for each epoch of the existing method and the proposed method. The early stopping algorithm was not used to observe how each evaluation metric changed until 50,000 epochs. In Fig. 10a, at epoch 1, the FID of fine-tuning was 22.05, Freeze *D* was 21.91, Freeze *G* was 22.06, and partial fine-tuning was 21.95. In epoch 1, Freeze *D* showed the highest performance, while Freeze *G* showed the lowest performance. There was a performance difference of 0.15 between Freeze *D* and Freeze *G*. At 50,000 epochs, fine-tuning's FID was 14.60, showing an upward trend of 7.45, while Freeze *D*'s was 14.30, showing an upward trend of 7.61. Freeze *G* was 14.18, which was 7.88, while partial fine-tuning was 14.27, showing an upward trend of 7.68. When the expressive ability of the generator is limited, or when pre-trained models are biased toward certain types of data, freezing the generator and training the discriminator can help the generator learn more complex expressions. However, when dealing with complex datasets, fixing only the generator may not be sufficient to capture all the nuances of the data distribution. In addition, there may be cases where freezing the discriminator may not effectively capture the difference between the distribution to be trained and the distribution to be learned.

In Fig. 10b, at epoch 1, the coverage of fine-tuning was 0.628, Freeze *D* was 0.627, Freeze *G* was 0.629, and partial fine-tuning was 0.625. In epoch 1, Freeze *G* showed the highest performance, while partial fine-tuning showed the lowest performance. There was a performance difference of 0.004 between Freeze *G* and partial fine-tuning. At 50,000 epochs, the coverage of fine-tuning was 0.781, showing an upward trend of 0.153, while that of Freeze *D* was 0.775, showing an upward trend of 0.148. Freeze *G* showed an upward trend of 0.148 with 0.777, while partial fine-tuning showed an upward trend of 0.151 with 0.776. Because some layers of the generator are frozen and do not update their

Table 6 Coverage and density scores under StyleGAN architecture. Left and right values indicate the best coverage and density scores

Method	Oxford Flower	CUB-200–2011	Caltech-256
Fine-tuning	0.781/**12.670**	0.697/4.082	0.353/58.324
Freeze*D*	**0.783**/11.746	**0.708**/4.090	0.348/58.398
Freeze*G*	0.779/12.585	0.693/**4.133**	**0.361**/59.272
Partial fine-tuning	**0.783**/12.447	0.700/3.977	0.353/**59.717**

Bold items mean the highest performance

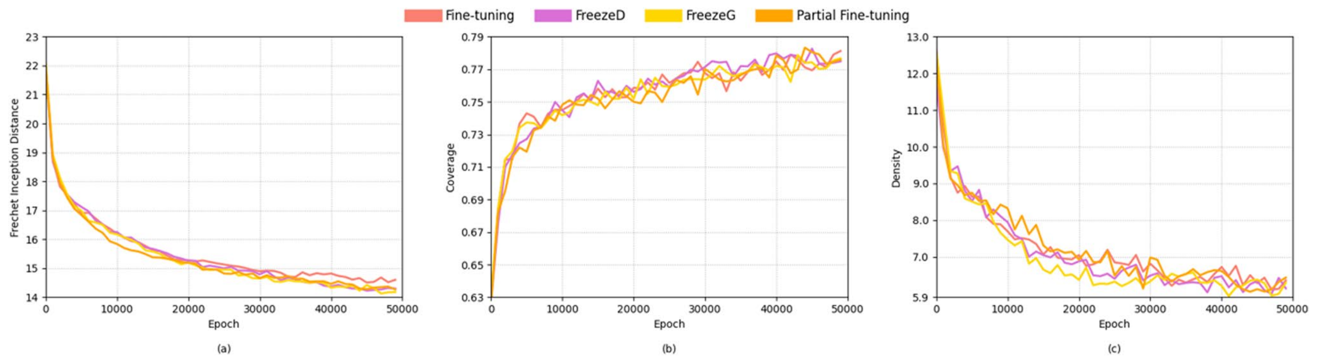


Fig. 10 Trends of FID, coverage, and density scores of freeze-based methods on the Oxford Flower dataset

weights, the existing dataset features remain static and may be limited in their ability to capture new features of the target dataset during training.

In Fig. 10c, at epoch 1, the density of fine-tuning was 12.670, Freeze *D* was 11.746, Freeze *G* was 12.585, and partial fine-tuning was 12.447. In epoch 1, fine-tuning showed the highest performance, while Freeze *D* showed the lowest performance. There was a performance difference of 0.924 between fine-tuning and Freeze *D*. At 50,000 epochs, the density of fine-tuning was 6.358, showing a downward trend of 6.312, while in the case of Freeze *D*, 6.141, showing a downward trend of 5.605. Freeze *G*'s density fell 6.29 percent to 6.295, while partial fine-tuning fell 6.011percent to 6.436. While there may be several reasons for performance degradation, the main problems are considered to be compatibility with the used model, insufficient training time, and inappropriate hyperparameter settings.

Figure 11 shows the (a) FID, (b) Coverage, and (c) Density values for each epoch of the existing method and the proposed method. The early stopping algorithm was not used to observe how each evaluation metric changed until 50,000 epochs. In Fig. 11a, at epoch 1, the FID of fine-tuning was 34.04, Freeze *D* was 33.9, Freeze *G* was 34.11, and partial fine-tuning was 33.63. In epoch 1, partial fine-tuning showed the highest performance, while Freeze *G* showed the lowest performance. There was a performance difference of 0.48 between partial fine-tuning and Freeze *G*. At 50,000 epochs, fine-tuning's FID was 15.14, showing an upward trend of 18.9, while Freeze *D*'s was 14.82, showing an upward trend of 19.08. Freeze *G*'s FID was 14.95, showing an upward trend of 19.16, while partial fine-tuning was 15.01, showing an upward trend of 18.62. Like this, Freeze *D* can specialize in distinguishing between real and synthetic images when not exposed to new information during training.

In Fig. 11b, at epoch 1, the coverage of fine-tuning was 0.246, Freeze *D* was 0.241, Freeze *G* was 0.241, and partial fine-tuning was 0.253. In epoch 1, partial fine-tuning showed the highest performance, while Freeze *D* and Freeze *G* showed the lowest performance. There was a performance difference of 0.012 between partial fine-tuning and Freeze *D* and Freeze *G*. At 50,000 epochs, the coverage of fine-tuning was 0.695, showing an upward trend of 0.449, while that of Freeze *D* was 0.707, showing an upward trend of 0.466. Freeze *G* showed an upward trend of 0.439 with 0.680, while partial fine-tuning showed an upward trend of 0.687 with 0.434. Assuming that the discriminator has learned how to distinguish features and patterns of a wide range of images, using a frozen discriminator can lead to better performance than fine-tuning. In this case, the generator can synthesize more diverse and representative samples without overfitting.

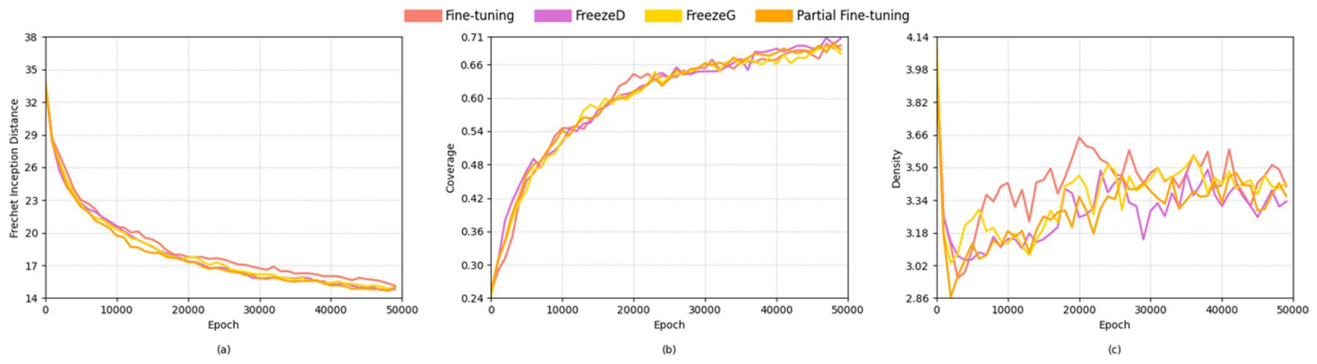


Fig. 11 Trends of FID, coverage, and density scores of freeze-based methods on the CUB-200–2011 dataset

In Fig. 11c, at epoch 1, the density of fine-tuning was 4.082, Freeze *D* was 4.090, Freeze *G* was 4.133, and partial fine-tuning was 3.977. In epoch 1, Freeze *G* showed the highest performance, while partial fine-tuning showed the lowest performance. There was a performance difference of 0.156 between Freeze *G* and partial fine-tuning. At 50,000 epochs, the density of fine-tuning was 3.408, showing a decline of 0.674, and Freeze *D* of 3.333, showing a decline of 0.75. Freeze *G*'s density fell 0.711 to 3.422, and partial fine-tuning fell 0.616 to 3.361. Overfitting may be the cause, such as in the CUB-200–2011 dataset, but the starting point may be different. The biggest difference between the two datasets is complexity. Unlike the Oxford Flower dataset, the CUB-200–2011 dataset contains 200 species of birds with more variability in terms of pose and perspective, making it difficult for the model to learn generalizable features. This is because the Oxford Flower Dataset contains 17 types of flowers with 8,189 images, while the CUB-200–2011 Dataset contains 200 species of birds with 11,000 images.

Figure 12 shows the (a) FID, (b) Coverage, and (c) Density values for each epoch of the existing method and the proposed method. The early stopping algorithm was not used to observe how each evaluation metric changed until 50,000 epochs. In Fig. 12a, at epoch 1, the FID of fine-tuning was 42.82, Freeze *D* was 43.29, Freeze *G* was 42.76, and partial fine-tuning was 43.28. In epoch 1, Freeze *G* showed the highest performance, while Freeze *D* showed the lowest performance. There was a performance difference of 0.53 between Freeze *G* and Freeze *D*. At 50,000 epochs, Fine-tuning's FID was 34.0, showing an upward trend of 8.82, and Freeze *D*'s was 33.65, showing an upward trend of 9.64. Freeze *G*'s FID was 33.51, showing an upward trend of 9.25, and partial fine-tuning was 33.51, showing an upward trend of 9.77. Partial fine-tuning can improve generalization performance by adapting to specific features of the Caltech-256 dataset and utilizing representations learned from a pre-trained model. Additionally, it can reduce the risk of overfitting by reducing the number of parameters that need to be updated during training.

In Fig. 12b, at epoch 1, the coverage of fine-tuning was 0.320, Freeze *D* was 0.315, Freeze *G* was 0.319, and partial fine-tuning was 0.316. In epoch 1, fine-tuning showed the highest performance, while Freeze *D* showed the lowest performance. There was a performance difference of 0.005 between fine-tuning and Freeze *D*. At 50,000 epochs, the coverage of fine-tuning was 0.350, which showed an upward trend of 0.03, while that of Freeze *D* was 0.346, showing an upward trend of 0.031. Freeze *G* showed an upward trend of 0.042 with 0.361, while partial fine-tuning showed an upward trend of 0.037 with 0.353. Freeze *G* may have preserved generator capacity and prevented overfitting by focusing updates on the discriminator. Fine-tuning can result in a significant decrease in the quality of synthesized samples if the situation worsens and the generator forgets the previously learned information.

In Fig. 12c, at epoch 1, the density of fine-tuning was 58.324, Freeze *D* was 58.398, Freeze *G* was 59.272, and partial fine-tuning was 59.717. In epoch 1, partial fine-tuning showed the highest performance, while fine-tuning showed the lowest performance. Partial fine-tuning and fine-tuning had a performance difference of 1.393. At 50,000 epochs, fine-tuning's density fell 33.003 to 25.321, while Freeze *D* fell 33.019 to 25.379. Freeze *G* fell 33.827 to 25.445, while partial fine-tuning fell 35.617 to 24.1. If the dataset that the model has previously learned, such as Caltech-256, has high variability, it may be difficult to generalize as such. Additionally, since there are inherent limitations to the model, it can be difficult to learn a new distribution if the objects or attributes in the training dataset are significantly different from what has been previously learned.

Table 7 evaluates the proposed method FID values of a model trained with specific forest insect class data. The 16 forest insect classes selected and evaluated can be found relatively easily in Jeollanam-do, Korea, and Fig. 14a-p of Appendix 6.2 shows the synthetic images. As a result of the analysis, when looking only at the FID values, Freeze *D* best synthesized the 8 species Cabbage Butterfly, Comostola Subtiliaria, Drepana Curvatula, Dynastid Beetle, Ground Beetle, Short-tailed Blue Butterfly, Stinkbug, and Water

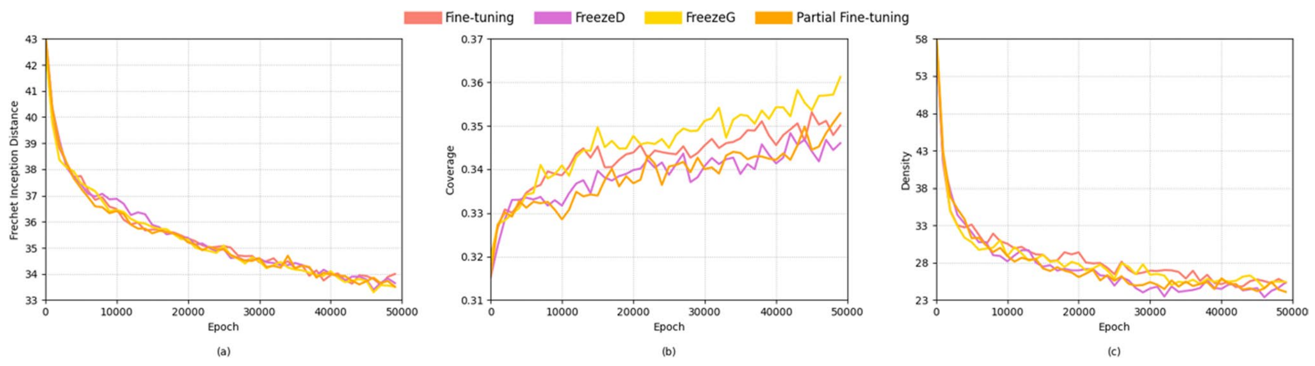


Fig. 12 Trends of FID, coverage, and density scores of freeze-based methods on the Caltech-256 dataset

Table 7 FID scores under the Insect-30 dataset. Values indicate the best FID scores. For each value, the methods are marked with the best performance using gold ● medals

	Actias Gnoma Butler	Apatura Metis	Cabbage Butterfly	Cicada	Comostola Subtiliaria	Drepana Curvatula	Dynastid Beetle	Ground Beetle	Nymphalid	Papilio Bianor	Peach Pyralid	Short-tailed Blue Butterfly	Stag Beetle	Stinkbug	Water Strider	Yellow Swallowtail Butterfly
Freeze*D*	32.93	27.69	29.37 ●	25.19	31.82 ●	28.77 ●	28.73 ●	28.58 ●	29.75	24.96	34.12	26.43 ●	34.70	27.06 ●	34.19 ●	24.74
Freeze*G*	33.21	28.09	29.83	25.63	32.69	29.43	29.66	29.61	30.03	25.54	35.20	27.17	35.49	27.36	34.84	25.34
Partial fine-tuning	32.82 ●	27.68 ●	29.65	25.17 ●	32.13	28.84	29.01	28.80	29.60 ●	24.90 ●	33.98 ●	26.72	34.64 ●	27.29	34.26	24.70 ●

Strider. Eight species of Actias Gnoma Butler, Apatura Metis, Cicada, Nymphalid, Papilio Bianor, Peach Pyralid, Stag Beetle, and Yellow Swallowtail Butterfly were best synthesized by the partial fine-tuning method. Freeze *G* had lower FID values for all experimental settings than Freeze *D* and partial fine-tuning in 16 forest insect species. Both methods include various sizes and colors of species, but Freeze *D* includes larger, more colorful, and more complex patterned species. Freeze *D* includes more aquatic species, while partial fine-tuning includes more terrestrial species. Partial fine-tuning has more species that are active during the day, as well as species that are restricted to certain ranges or specific areas. The method of freezing both generators and discriminators enables better class-specific synthesis, but the method of freezing discriminators focuses on improving overall image quality. In terms of the results alone, there seems to be a class that synthesizes the discriminator well, but it is difficult to affirm when comparing the performance with the method of freezing for both the generator and the discriminator.

Table 8 evaluates the coverage and density values of the proposed method of a model trained with specific forest insect class data. As a result of the analysis, when looking only at the coverage value, Freeze *D* synthesized the remaining species relatively well, except for Cabbage Butterfly, Papilio Bianor, and Water Strider types. Freeze *G* synthesized the rest of the species relatively well, except for the 9 species of Actias Gnoma Butler, Cicada, Comostola Subtiliaria, Ground Beetle, Nymphalid, Peach Pyralid, Stinkbug, Water Strider, and Yellow Swallowtail Butterfly. Partial fine-tuning synthesized the remaining 8 species relatively well, except for Apatura Metis, Cicada, Comostola Subtiliaria, Ground Beetle, Nymphalid, Papilio Bianor, Peach Pyralid, and Yellow Swallowtail Butterfly. When using coverage as

Table 8 Coverage and density scores under the Insect-30 dataset. Left and right values indicate the best coverage and density scores

	Actias Gnoma Butler	Apatura Metis	Cabbage Butterfly	Cicada	Comostola Subtiliaria	Drepana Curvatula	Dynastid Beetle	Ground Beetle
Freeze*D*	**0.965**/3.917	**1.000/3.485**	0.954/5.805	**0.993**/5.025	**0.996**/6.636	**1.000**/6.888	**1.000**/6.038	**0.998**/5.250
Freeze*G*	0.962/**3.936**	**1.000**/3.470	**0.960/5.955**	0.992/4.930	0.995/**7.109**	**1.000/7.024**	**1.000/6.074**	0.994/5.343
Partial fine-tuning	**0.965**/3.711	0.999/3.560	**0.960**/5.715	0.992/**5.073**	0.993/7.062	**1.000**/6.905	**1.000**/5.801	0.997/**5.703**
	Nymphalid	**Papilio Bianor**	**Peach Pyralid**	**Short-tailed Blue Butterfly**	**Stag Beetle**	**Stinkbug**	**Water Strider**	**Yellow Swallowtail Butterfly**
Freeze*D*	**1.000/5.413**	0.985/5.996	**0.996/6.336**	**1.000**/8.014	**1.000**/5.355	**1.000**/7.903	0.998/8.984	**0.998**/6.366
Freeze*G*	0.998/5.319	**0.988/6.089**	0.983/5.830	**1.000**/7.710	**1.000/5.918**	0.998/7.829	0.996/**9.039**	0.997/**6.819**
Partial fine-tuning	0.999/5.148	0.979/5.864	0.992/5.920	**1.000/8.153**	**1.000**/5.427	**1.000/8.045**	**0.999**/8.451	0.996/6.384

Bold items mean the highest performance

an evaluation metric, it is difficult to rank well-synthesized classes because many of them are common.

In terms of density values alone, Freeze *D* synthesized the 3 species Apatura Metis, Nymphalid, and Peach Pyralid best. The 9 species of Actias Gnoma Butler, Cabbage Butterfly, Comostola Subtiliaria, Drepana Curvatula, Dynastid Beetle, Papilio Bianor, Stag Beetle, Water Strider, and Yellow Swallowtail Butterfly were best synthesized by Freeze *G*. The 4 species Cicada, Ground Beetle, Short-tailed Blue Butterfly, and Stinkbug were best synthesized by partial fine-tuning. All three methods include insects with diverse appearances, habitats, behaviors, and geographical distributions. However, Freeze *D* includes three species of Lepidopteran insects, partial fine-tuning includes species from different insect orders and families, and Freeze *G* includes species from different insect orders, families, and genera. As with the Insect-30 dataset, pinning only the generator of the GAN can also be effective in improving the density performance if the dataset is relatively small and there is a high risk of overfitting.

5 Conclusion

The deep learning algorithms that led to the development of discriminative modeling have also been applied to generative modeling. Among them, GAN can map from latent space to data space, and can train well on large unlabeled datasets. GAN are useful for image synthesizing tasks, and to improve existing deep learning algorithms. The performance of GAN is very surprising. For tasks that were difficult to perform with traditional methods, GAN showed excellent results. GAN research related to computer vision includes image super-resolution, high-resolution image synthesis, image synthesis using text, painting style simulation, and image-to-image translation. In addition, GAN are already being applied to music synthesis, video games, game design, and the film industry. Recently, GAN have also shown strong performance in image inversion and neural speech synthesis [47–49]. It is anticipated that even fields that are currently underdeveloped will soon make progress [50, 51].

GAN appeared in 2014, and although 7 years have passed, the problem of training instability of GAN still remains. Sometimes, when two neural networks diverge during training, the GAN does not converge at all. Many researchers have tried to stabilize the training of GAN. Initially, solutions such as conditional batch normalization, conditional versions of adaptive instance normalization, mini-batch discrimination, and one-sided label smoothing have been proposed. After that, Feature Statistics Mixing Regularization (FSMR), how to learn various distributions with linear projection applied to synthetic and real distributions with multi-discriminators, and how to provide visual guidelines to generators have been proposed [52–54]. As GAN evolve, we expect that this stabilization will ripen, and we will soon be able to train our models without any problems.

We used coverage and density for various performance evaluations, and found a problem. In all experiments, it is difficult to accurately determine from the coverage and density values whether the model converges or collapses. For example, in Fig. 7c, scale/shift with GLO showed the highest density performance, but in Fig. 13c of Appendix 6.2, a blurry image was inserted. For this reason, to secure the objectivity of the conclusion drawn, coverage and density are excluded from the training method evaluation.

The proposed method divides the generator and discriminator, and then freezing or partially fine-tuning. Based on the FID value alone, the proposed method was superior to the existing method, and based on this, it is judged that the transferability of the generator and discriminator can be universally applied to the image to be synthesized [55]. However, different results may be obtained, depending on the characteristics of the data type used for training. This means that if the synthesis target is significantly different from the present, the optimization configuration can also be changed accordingly. Therefore, the optimal method for the type of task to be solved can so far only be confirmed through experiments.

Although we provide a partial solution to the instability problem of GAN training, we can design a method with better performance than the proposed method. In addition, if there is a higher-spec hardware infrastructure better than ours, images with a resolution higher than 256 × 256 can be generated without restrictions. We believe that the upgraded version of FD will be promising [56–58]. Additionally, ensembles or large and deep models can be difficult to deploy to mobile and Internet of Things (IoT) devices, but this difficulty can be solved by distilling features into small and shallow models. As 3D image synthesis technology is also accelerating, research on how to effectively fine-tune a pre-trained GAN on a 3D dataset is needed in the future [59].

Appendix

Ablation study on freezing layers

Table 9 shows the layer freezing result of the discriminator using the StyleGAN model. Stanford Cars showed the highest performance with FID 11.01 when trained and frozen up to

Table 9 Ablation study on freezing layers of ***D*** on StyleGAN architecture under the ‘Stanford Cars’ and ‘Stanford Dogs’ datasets. Layer ***i*** indicates that the first ***i*** layers of the discriminator are frozen. Values indicate the best FID scores. For each value, the methods are marked with the best performance using gold ●, silver ●, and bronze ● medals

	Layer 1	**Layer 2**	**Layer 3**	**Layer 4**	**Layer 5**	**Layer 6**	**Layer 7**
Stanford Cars	13.71 ●	11.69	11.05 ●	11.07	**11.01** ●	11.22	11.19
Stanford Dogs	41.50 ●	37.01	32.13	30.35 ●	**30.10** ●	30.38	30.65

Table 10 Ablation study on freezing layers of ***D*** on StyleGAN architecture under the 'Stanford Cars' and 'Stanford Dogs' datasets. Layer ***i*** indicates that the first ***i*** layers of the discriminator are frozen. Left and right values indicate the best coverage and density scores

	Layer 1	Layer 2	Layer 3	Layer 4	Layer 5	Layer 6	Layer 7
Stanford Cars	0.880/6.843	0.914/6.245	0.932/6.666	0.937/6.968	**0.941**/7.225	0.935/**7.642**	0.940/7.500
Stanford Dogs	0.160/**17.396**	0.192/17.166	0.227/16.201	0.241/15.522	0.241/15.952	**0.246**/16.072	0.240/15.934

Bold items mean the highest performance

Table 11 Ablation study on freezing layers of ***G*** on StyleGAN architecture under the 'Stanford Cars' and 'Stanford Dogs' datasets. Layer ***i*** indicates that the first ***i*** layers of the generator are frozen. Values indicate the best FID scores. For each value, the methods are marked with the best performance using gold 🟡, silver ⚪, and bronze 🟤 medals

	Layer 1	**Layer 2**	**Layer 3**	**Layer 4**	**Layer 5**	**Layer 6**	**Layer 7**
Stanford Cars	11.15 ⚪	11.28 🟤	**11.10** 🟡	11.17	11.15 ⚪	11.22	11.20
Stanford Dogs	30.16 ⚪	**29.83** 🟡	30.46	30.78	30.86 🟤	30.70	30.33

layer 5. When trained with freezing up to layer 1, FID showed the lowest performance at 13.71, and there was a performance difference of 2.7, compared to when freezing up to layer 5. Stanford Dogs also showed the highest performance with FID 30.10 when trained and frozen up to layer 5. When training with freezing up to layer 1, FID showed the lowest performance at 41.50, while there was a performance difference of 11.4 between freezing up to layer 5 and FID. The choice of updating the weights of all layers was not a good method, and it didn't always follow the law of large numbers.

Table 10 shows the layer freezing result of the discriminator using the StyleGAN model. Stanford Cars showed the highest performance with coverage of 0.941 when frozen up to layer 5, and showed the highest performance with a density of 7.642 when frozen up to layer 6. When only one layer was frozen and trained, the coverage showed the lowest performance at 0.880; and when two layers were frozen and trained, the coverage showed the lowest performance at 6.245. In the case of coverage, the difference between the highest and lowest performance was 0.061; while in the case of density, the difference between the highest and lowest performance was 1.397.

Stanford Dogs showed the highest performance with coverage of 0.246 when 5 layers were frozen, and a density of 17.396 when only 1 layer was frozen. When only one layer was frozen, the coverage showed the lowest performance at 0.160; and when 4 layers were frozen, the density showed the lowest performance at 15.522. In the case of coverage, the difference between the highest and lowest performance was 0.086; while in the case of density, the difference between the highest and lowest performance was 1.874.

Table 11 shows the layer freezing result of the generator using the StyleGAN model. Stanford Cars had the highest performance with FID 11.10 when 3 layers were frozen. When 2 layers were frozen, FID showed the lowest performance at 11.28, and there was a performance difference of 0.18, compared to when 3 layers were frozen. Stanford Dogs had the highest performance with FID 29.83 when 2 layers were frozen. When 5 layers were frozen, FID showed the lowest performance at 30.86, and there was a performance difference of 1.03, compared to when 2 layers were frozen. The decision of whether to freeze a layer or not is not always clear-cut in terms of achieving optimal performance.

Table 12 shows the layer freezing results of the discriminator using the StyleGAN model. Stanford Cars showed the highest performance with coverage of 0.943 when frozen up to layer 3, and showed the highest performance with a density of 7.837 when frozen up to layer 5. When 2 layers and 6 layers were frozen, the coverage showed the lowest performance at 0.934; and when 7 layers were frozen, the density showed the lowest performance at 7.173. In the case of coverage, the difference between the highest and lowest performance was 0.009; while in the case of density, the difference between the highest and lowest performance was 0.664.

Stanford Dogs showed the highest performance with coverage of 0.246 and density of 17.396 when frozen up to layer 2. When 4 layers were frozen, the coverage showed the lowest performance at 0.241; and when only 1 layer was frozen, the density showed the lowest performance at 17.163. In the case of coverage, the difference between the highest and lowest performance was 0.005; while in the case of density,

Table 12 Ablation study on freezing layers of ***G*** on StyleGAN architecture under the 'Stanford Cars' and 'Stanford Dogs' datasets. Layer ***i*** indicates that the first ***i*** layers of the generator are frozen. Left and right values indicate the best coverage and density scores

	Layer 1	Layer 2	Layer 3	Layer 4	Layer 5	Layer 6	Layer 7
Stanford Cars	0.939/7.727	0.934/7.726	**0.943**/7.593	0.938/7.462	0.937/**7.837**	0.934/7.242	0.938/7.173
Stanford Dogs	0.244/17.163	**0.250/17.833**	0.244/17.566	0.241/17.331	0.242/17.633	0.243/17.653	0.249/17.531

Bold items mean the highest performance

Table 13 Ablation study on freezing layers of ***G*** and ***D*** on StyleGAN architecture under the 'Stanford Cars' dataset. Layer ***i*** indicates that the first ***i*** layers of the generator and discriminator are frozen. Values indicate the best FID scores. For each value, the methods are marked with the best performance using gold ●, silver ●, and bronze ● medals

Dis. / Gen.	Layer 1	Layer 2	Layer 3	Layer 4	Layer 5	Layer 6	Layer 7
Layer 1	13.58	13.56	13.43	13.46	13.59 ●	13.35	13.43
Layer 2	11.74	11.82	11.86	11.80	11.73	11.68	11.76
Layer 3	11.08	10.98	10.93	10.92	11.33	11.26	11.04
Layer 4	10.95	10.98	10.99	**10.84** ●	10.90	11.00	11.03
Layer 5	11.24	11.09	10.88	11.11	11.11	11.07	11.04
Layer 6	11.20	11.24	11.10	11.09	11.07	11.05	11.07
Layer 7	11.25	10.99	11.08	10.85 ●	11.10	11.05	11.26

Table 15 Ablation study on freezing layers of ***G*** and ***D*** on StyleGAN architecture under the 'Stanford Dogs' dataset. Layer ***i*** indicates that the first ***i*** layers of the generator and discriminator are frozen. Values indicate the best FID scores. For each value, the methods are marked with the best performance using gold ●, silver ●, and bronze ● medals

Dis. / Gen.	Layer 1	Layer 2	Layer 3	Layer 4	Layer 5	Layer 6	Layer 7
Layer 1	42.78 ●	42.51	41.38	42.14	42.28	41.73	42.04
Layer 2	36.52	37.47	36.66	37.16	37.12	36.91	37.41
Layer 3	32.56	32.57	32.46	32.39	32.22	32.44	32.59
Layer 4	30.76	30.52	30.78	30.64	30.58	30.68	31.21
Layer 5	30.31	30.64	30.34	30.56	30.33	30.80	**30.02** ●
Layer 6	30.40	30.48	30.51	30.67	30.23	30.69	30.51
Layer 7	30.21 ●	30.72	30.64	30.23	30.30	30.43	30.57

the difference between the highest and lowest performance was 0.233.

Table 13 shows the layer freezing results of the generator and discriminator using the StyleGAN model. Stanford Cars showed the highest performance with FID 10.84 when frozen and trained up to generator layer 4 and discriminator layer 4. When freezing and training up to generator layer 1 and discriminator layer 5, FID showed the lowest performance at 13.59, and there was a difference between the highest performance and 2.75. In Table 9, the discriminator was optimal when frozen to layer 5, while in Table 11, the generator was optimal when frozen to layer 3, but in Table 13, the generator and discriminator were all optimal when frozen to layer 4. The optimal approach differed when either only the generator or discriminator was frozen compared to when both the generator and discriminator were frozen.

Table 14 shows the layer freezing results of the generator and discriminator using the StyleGAN model. Stanford Cars showed the highest performance with coverage of 0.945 when frozen and trained up to generator layer 7 and discriminator layer 4; and showed the highest performance with a density of 7.552 when frozen and trained up to generator layer 6 and discriminator layer 2. When the generator layer 1 and discriminator layer 5 were frozen and trained, the coverage showed the lowest performance with 0.882; and when frozen and trained until the generator layer 2 and discriminator layer 7, the density showed the lowest performance with 5.779. In the case of coverage, the difference between the highest and lowest performance was 0.063; while in the case of density, the difference between the highest and lowest performance was 1.773.

Table 15 shows the layer freezing results of the generator and discriminator using the StyleGAN model. Stanford Dogs showed the highest performance with FID 30.02 when frozen and trained up to generator layer 5 and discriminator layer 7. When freezing and training up to generator layer 1 and discriminator layer 1, FID showed the lowest performance at 42.78, with a difference of 12.76 from the highest performance. In Table 9, the discriminator was optimal when it was frozen up to layer 5; while in Table 11, the generator was optimal when it was frozen up to layer 2; but in Table 15, when the generator was frozen up to layer 5, the discriminator was optimal when it was frozen up to layer 7. The optimal approach varied when only the generator or discriminator was frozen compared

Table 14 Ablation study on freezing layers of***G***and***D***on StyleGAN architecture under the 'Stanford Cars' dataset. Layer***i***indicates that the first***i*** layers of the generator and discriminator are frozen. Left and right values indicate the best coverage and density scores

Dis / Gen	Layer 1	Layer 2	Layer 3	Layer 4	Layer 5	Layer 6	Layer 7
Layer 1	0.887/6.373	0.883/6.942	0.888/6.463	0.885/6.770	0.882/6.167	0.884/6.448	0.890/6.421
Layer 2	0.914/5.967	0.909/6.264	0.910/6.056	0.916/6.139	0.913/5.874	0.918/6.318	0.905/5.779
Layer 3	0.933/6.545	0.932/6.363	0.937/6.721	0.934/6.483	0.935/6.587	0.934/6.407	0.934/6.660
Layer 4	0.941/6.865	0.940/6.774	0.938/6.957	0.937/6.776	0.942/7.022	0.939/7.026	0.939/7.044
Layer 5	0.938/7.315	0.940/7.543	0.943/7.453	0.938/7.276	0.941/7.089	0.938/7.206	0.938/7.286
Layer 6	0.937/7.229	0.938/**7.552**	0.937/7.194	0.938/6.973	0.934/7.306	0.940/7.433	0.937/7.106
Layer 7	0.941/7.470	0.938/7.259	0.940/7.266	**0.945**/7.453	0.940/7.266	0.940/7.376	0.940/7.192

Bold items mean the highest performance

Table 16 Ablation study on freezing layers of ***G*** and ***D*** on StyleGAN architecture under the 'Stanford Dogs' dataset. Layer ***i*** indicates that the first ***i*** layers of the generator and discriminator are frozen. Left and right values indicate the best coverage and density scores

Dis Gen	Layer 1	Layer 2	Layer 3	Layer 4	Layer 5	Layer 6	Layer 7
Layer 1	0.158/17.757	0.155/17.630	0.161/17.964	0.160/17.785	0.162/17.759	0.158/**18.129**	0.159/18.058
Layer 2	0.192/16.881	0.195/17.036	0.194/17.497	0.194/17.162	0.192/17.029	0.191/17.363	0.190/17.267
Layer 3	0.224/16.011	0.226/15.884	0.224/16.598	0.226/16.351	0.225/16.244	0.226/16.354	0.226/16.619
Layer 4	0.238/15.583	0.239/15.910	0.236/15.753	0.238/15.608	0.243/15.877	0.240/15.815	0.235/15.662
Layer 5	0.245/15.724	0.238/16.102	0.241/15.825	**0.246**/15.935	0.241/15.961	0.237/15.424	0.243/15.447
Layer 6	0.242/16.068	0.241/16.414	0.238/16.022	0.240/16.235	0.245/15.812	0.245/15.983	0.240/15.879
Layer 7	**0.246**/16.097	0.242/16.022	0.244/16.096	0.242/16.290	0.245/16.282	0.240/16.174	0.238/16.371

Bold items mean the highest performance

to when both the generator and discriminator were frozen. In contrast to Table 13, Table 15 showed relatively large differences between the highest and lowest performance.

Table 16 shows the layer freezing results of the generator and discriminator using the StyleGAN model. Stanford Dogs showed the highest performance with coverage of 0.246 when frozen and trained up to generator layer 5, discriminator layer 4, generator layer 7, and discriminator layer 1. Density showed the highest performance at 18.129 when freezing and training up to generator layer 1 and discriminator layer 6. When the generator layer 1 and discriminator layer 2 were frozen and trained, the coverage showed the lowest performance at 0.155; while when the generator layer 5 and discriminator layer 6 were frozen and trained, the density showed the lowest performance at 15.424. In the case of coverage, the difference between the highest and lowest performance was 0.091; while in the case of density, the difference between the highest and lowest performance was 2.705.

Qualitative results for prior and proposed methods

In Fig. 13, scale/shift and L2-SP have some limitations, so they synthesize images that when quantified, are low in diversity, but when qualitatively evaluated, are plausible. GLO synthesized blurry images due to adversarial losses and a lack of knowledge of the source discriminator. MineGAN is completely out of target distribution. MineGAN is not applicable when the source distribution is said to contain the target distribution, but the distribution has split support. As a result, MineGAN did not learn the distribution shift well.

In Fig. 14, Caltech-256 was difficult to synthesize with high quality using any freezing method, but CUB-200–2011,

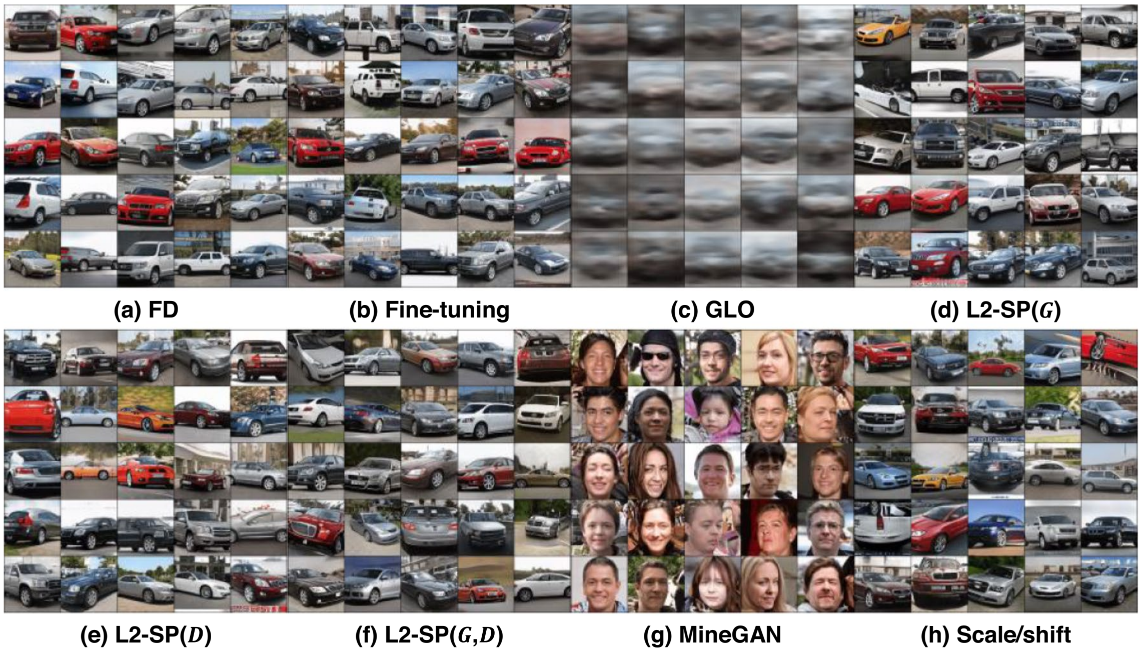


Fig. 13 Samples synthesized by prior methods under the 'Stanford Cars' dataset

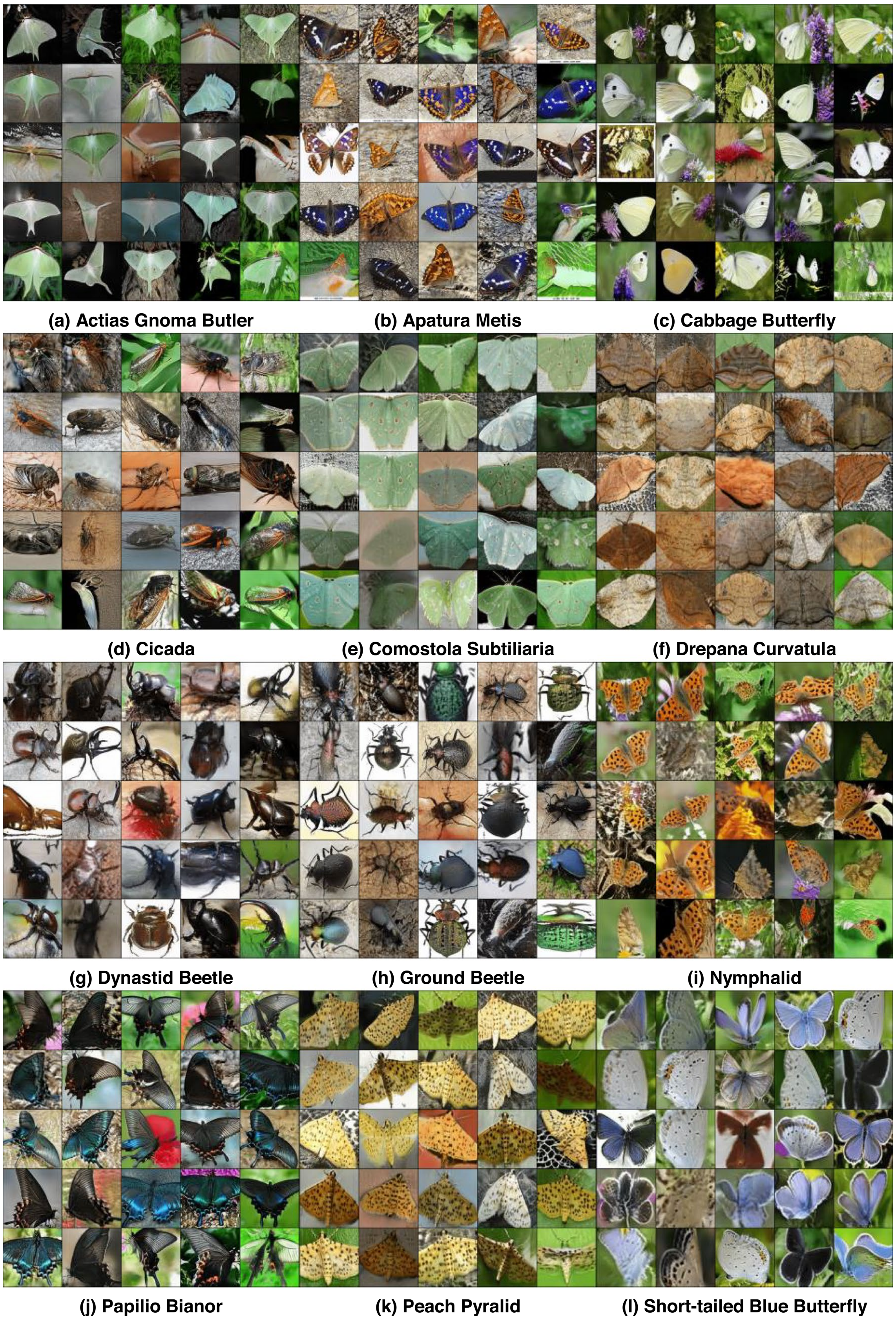
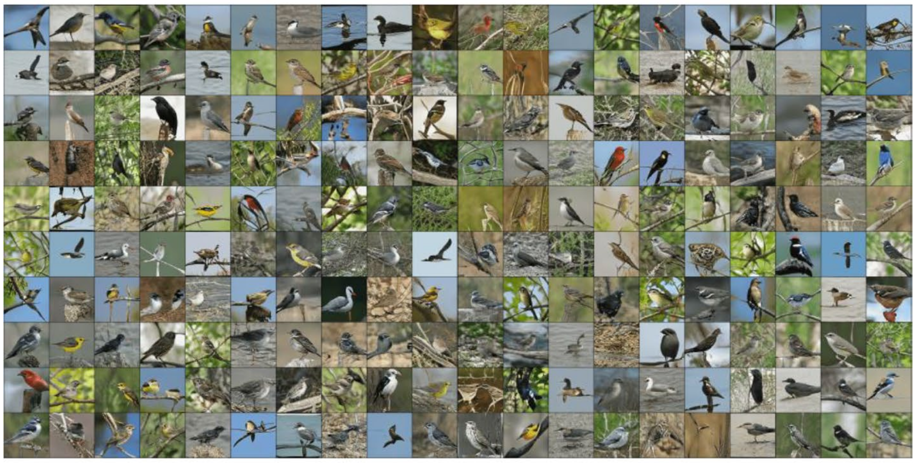


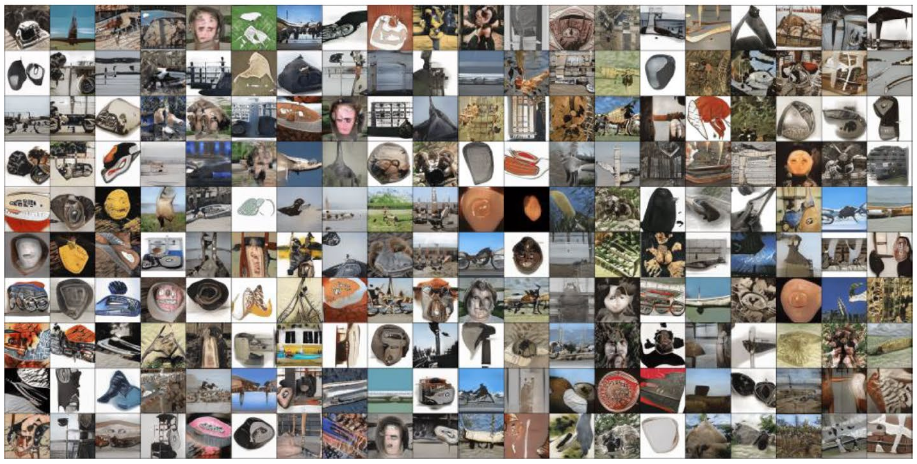
Fig. 14 Samples synthesized by proposed methods

(m) Stag Beetle (n) Stinkbug (o) Water Strider

(p) Yellow Swallowtail Butterfly



(q) CUB-200-2011 (Freeze$D$) (r) CUB-200-2011 (Partial fine-tuning)



(s) Caltech-256 (Freeze$D$) (t) Caltech-256 (Freeze$G$)

Fig. 14 (continued)

a fine-grained dataset, and Insect30, a coarse-grained dataset, synthesized relatively well.

Acknowledgements This work was supported by a Gwangju–Jeonnam local Energy Cluster Manpower training grant of the Korea Institute of Energy Technology Evaluation and Planning (KETEP), funded by the Korea government Ministry of Knowledge Economy (20214000000560); and this research was also supported by the Basic Science Research Program through the National Research Foundation of Korea (NRF), funded by the Ministry of Education (2020R1I1A3054843).

Funding This work was supported by a Gwangju–Jeonnam local Energy Cluster Manpower training grant of the Korea Institute of Energy Technology Evaluation and Planning (KETEP), funded by the Korea government Ministry of Knowledge Economy (20214000000560); and this research was also supported by the Basic Science Research Program through the National Research Foundation of Korea (NRF), funded by the Ministry of Education (2020R1I1A3054843).

Data availability Not applicable.

Declarations

Conflict of interest The Authors declares that there is no conflict of interest.

References

1. Goodfellow I, Pouget-Abadie J, Mirza M, Xu B, Warde-Farley D, Ozair S, Courville A, Bengio Y (2014) Generative adversarial nets. In: Advances in neural information processing systems (NIPS)
2. Brock A, Donahue J, Simonyan K (2019) Large scale GAN training for high fidelity natural image synthesis. In: International Conference on Learning Representations (ICLR)
3. Mo S, Cho M, Shin J (2019) Instagan: Instance-aware image-to-image translation. In: International Conference on Learning Representations (ICLR)
4. Zhou T, Li Q, Lu H, Cheng Q, Zhang X (2023) GAN review: Models and medical image fusion applications. Inf Fusion 91:134–148
5. Park S-W, Huh J-H, Kim J-C (2020) BEGAN v3: avoiding mode collapse in GANs using variational inference. Electronics 9(4):688
6. Park S-W, Ko J-S, Huh J-H, Kim J-C (2021) Review on generative adversarial networks: focusing on computer vision and its applications. Electronics 10(10):1216
7. Kim J-C, Lim S-C, Choi J, Huh J-H (2022) Review for Examining the Oxidation Process of the Moon Using Generative Adversarial Networks: Focusing on Landscape of Moon. Electronics 11(9):1303
8. Chatterjee S, Hazra D, Byun Y-C, Kim Y-W (2022) Enhancement of Image Classification Using Transfer Learning and GAN-Based Synthetic Data Augmentation. Mathematics 10(9):1541
9. Noguchi A, Harada T (2019) Image generation from small datasets via batch statistics adaptation. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV). IEEE, pp. 2750–2758
10. Wang Y, Gonzalez-Garcia A, Berga D, Herranz L, Khan F S, van de Weijer J (2019) Minegan: effective knowledge transfer from gans to target domains with few images. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). IEEE, pp. 9332–9341
11. Liu M-Y, Huang X, Mallya A, Karras T, Aila T, Lehtinen J, Kautz J (2019) Few-shot unsupervised image-to-image translation. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV). IEEE, pp. 10551–10560
12. Zakharov E, Shysheya A, Burkov E, Lempitsky V (2019) Few-shot adversarial learning of realistic neural talking head models. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV). IEEE, pp. 9459–9468
13. Wang T-C, Liu M-Y, Tao A, Liu G, Catanzaro B, Kautz J (2019) Few-shot video-to-video synthesis. In: Advances in neural information processing systems (NIPS). pp. 5014–5025
14. Chen T, Zhai X, Ritter M, Lucic M, Houlsby N (2019) Self-supervised gans via auxiliary rotation loss. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR). IEEE, pp. 12154–12163
15. Lucic M, Tschannen M, Ritter M, Zhai X, Bachem O, Gelly S (2019) High-fidelity image generation with fewer labels. In: International conference on machine learning (ICML). pp. 4183–4192
16. Zhang H, Zhang Z, Odena A, Lee H (2020) Consistency regularization for generative adversarial networks. In: International Conference on Learning Representations (ICLR)
17. Zhao Z, Singh S, Lee H, Zhang Z, Odena A, Zhang H (2021) Improved consistency regularization for gans. Proceedings of the AAAI Conference on Artificial Intelligence 35(12):11033–11041
18. Azadi S, Olsson C, Darrell T, Goodfellow I, Odena A (2018) Discriminator rejection sampling. In: International Conference on Learning Representations (ICLR)
19. Fekri M-N, Ghosh A-M, Grolinger K (2019) Generating energy data for machine learning with recurrent generative adversarial networks. Energies 13(1):130
20. Mo S, Kim C, Kim S, Cho M, Shin J (2019) Mining gold samples for conditional gans. In: Advances in neural information processing systems (NIPS)
21. Tanaka A (2019) Discriminator optimal transport. In: Advances in neural information processing systems (NIPS)
22. Gurumurthy S, Kiran Sarvadevabhatla R, Venkatesh Babu R (2017) Deligan: Generative adversarial networks for diverse and limited data. In: Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR). IEEE, pp. 166–174
23. Sinha S, Zhang H, Goyal A, Bengio Y, Larochelle H, Odena A (2019) Small-gan: Speeding up gan training using coresets. In: International Conference on Machine Learning (ICML). pp. 9005–9015
24. Xu H, Li W, Cai Z (2023) Analysis on methods to effectively improve transfer learning performance. Theor Comput Sci 940:90–107
25. Devlin J, Chang M-W, Lee K, Toutanova K (2019) Bert: Pre-training of deep bidirectional transformers for language understanding. In: 2019 Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)

26. He K, Fan H, Wu Y, Xie S, Girshick R (2020) Momentum contrast for unsupervised visual representation learning. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR). IEEE, pp. 9729–9738
27. Metz L, Poole B, Pfau D, S-D J (2016) Unrolled generative adversarial networks. arXiv preprint, arXiv:1611.02163
28. Arjovsky M, Bottou L (2017) Towards principled methods for training generative adversarial networks. arXiv preprint, arXiv:1701.04862
29. Arjovsky M, Chintala S, Bottou L (2017) Wasserstein generative adversarial networks. In International conference on machine learning. PMLR, pp. 214–223
30. Karras T, Laine S, Aila T (2019) A style-based generator architecture for generative adversarial networks. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR). IEEE, pp. 4401–4410
31. Oyelade O-N, Ezugwu A-E (2023) EOSA-GAN: Feature enriched latent space optimized adversarial networks for synthesization of histopathology images using Ebola optimization search algorithm. Biomed Signal Process Control 84:104734
32. Li Q, Wang X, Ma B, Wang X, Wang C, Gao S, Shi Y (2021) Concealed attack for robust watermarking based on generative model and perceptual loss. IEEE Trans Circuits Syst Video Technol 32(8):5695–5706
33. Li X, Grandvalet Y, Davoine F (2018) Explicit inductive bias for transfer learning with convolutional networks. In: International Conference on Machine Learning (ICML). pp. 2825–2834
34. Hinton G, Vinyals O, Dean J (2014) Distilling the knowledge in a neural network. In: Advances in neural information processing systems (NIPS) Workshop
35. Romero A, Ballas N, Kahou S E, Chassang A, Gatta C, Bengio Y (2015) Fitnets: Hints for thin deep nets. In: International Conference on Learning Representations (ICLR)
36. Yang S, Jiang L, Liu Z, Loy C C (2022) Pastiche Master: Exemplar-Based High-Resolution Portrait Style Transfer. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). IEEE, pp. 7693–7702
37. Zhang Y, Tang F, Dong W, Huang H, Ma C, Lee T-Y, Xu C (2022) Domain enhanced arbitrary image style transfer via contrastive learning. In: ACM SIGGRAPH 2022 Conference Proceedings, pp 1–8
38. Sauer A, Schwarz K, Geiger A (2022) Stylegan-xl: Scaling stylegan to large diverse datasets. In: Special Interest Group on Computer Graphics and Interactive Techniques Conference Proceedings (SIGGRAPH). pp. 1–10
39. Fu J, Li S, Jiang Y, Lin K-Y, Qian C, Loy C-C, Liu Z (2022) Stylegan-human: a data-centric odyssey of human generation. In: Computer Vision–ECCV 2022: 17th European Conference, pp 1–19
40. Theis L, Oord A-V-D, Bethge M (2015) A note on the evaluation of generative models. arXiv preprint, arXiv:1511.01844
41. Barratt S, Sharma R (2018) A note on the inception score. arXiv preprint, arXiv:1801.01973
42. Heusel M, Ramsauer H, Unterthiner T, Nessler B, Hochreiter S (2017) Gans trained by a two time-scale update rule converge to a local nash equilibrium. In: Advances in neural information processing systems (NIPS)
43. Sajjadi M S, Bachem O, Lucic M, Bousquet O, Gelly S (2018) Assessing generative models via precision and recall. In: Advances in Neural Information Processing Systems (NIPS). pp. 5228–5237
44. Naeem M F, Oh S J, Uh Y, Choi Y, Yoo J (2020) Reliable fidelity and diversity metrics for generative models. In: International Conference on Machine Learning (ICML). pp. 7176–7185
45. Kynkäänniemi T, Karras T, Laine S, Lehtinen J, Aila T (2019) Improved precision and recall metric for assessing generative models. In: Advances in Neural Information Processing Systems (NIPS). pp. 32–41
46. Kang M, Shin J, Park J (2022) Studiogan: a taxonomy and benchmark of gans for image synthesis. arXiv preprint, arXiv:2206.09479
47. Yin F, Zhang Y, Cun X, Cao M, Fan Y, Wang X, Yang Y (2022) StyleHEAT: one-shot high-resolution editable talking face generation via pre-trained StyleGAN. In: Computer Vision–ECCV 2022: 17th European Conference, pp 85–101
48. Parmar G, Li Y, Lu J, Zhang R, Zhu J Y, Singh K K (2022) Spatially-Adaptive Multilayer Selection for GAN Inversion and Editing. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). IEEE, pp. 11399–11409
49. Lee S G, Ping W, Ginsburg B, Catanzaro B, Yoon S (2022) BigVGAN: A Universal Neural Vocoder with Large-Scale Training. Accessed https://arxiv.org/abs/2206.04658
50. Tran D-T, Huh J-H (2023) New machine learning model based on the time factor for e-commerce recommendation systems. J Supercomput 79(6):6756–6801
51. Tran D-T, Truong D-H, Le H-S, Huh J-H (2023) Mobile robot: automatic speech recognition application for automation and STEM education. Soft Comput 27:10789–10805
52. Kim J, Choi Y, Uh Y (2022) Feature Statistics Mixing Regularization for Generative Adversarial Networks. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). IEEE, pp. 11294–11303
53. Sauer A, Chitta K, Müller J, Geiger A (2021) Projected gans converge faster. Adv Neural Inf Process Syst (NIPS) 34:17480–17492
54. Wang J, Yang C, Xu Y, Shen Y, Li H, Zhou B (2022) Improving GAN Equilibrium by Raising Spatial Awareness. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). IEEE. pp. 11285–11293
55. Wang S-Y, Wang O, Zhang R, Owens A, Efros AA (2020) Cnn-generated images are surprisingly easy to spot... for now. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR). IEEE, pp. 8695–8704
56. Ahn S, Hu S X, Damianou A, Lawrence N D, Dai Z (2019) Variational information distillation for knowledge transfer. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). IEEE, pp. 9163–9171
57. Jang Y, Lee H, Hwang S J, Shin J (2019) Learning what and where to transfer. In: International Conference on Machine Learning (ICML). pp. 3030–3039
58. Park W, Kim D, Lu Y, Cho M (2019) Relational knowledge distillation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). IEEE, pp. 3967–3976
59. Chan E R, Lin C Z, Chan M A, Nagano K, Pan B, De Mello S, Wetzstein G (2022) Efficient geometry-aware 3D generative adversarial networks. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). IEEE, pp. 16123–16133

Sung-Wook Park received B.E., and M.E. degrees in computer engineering from Sunchon National University, Suncheon, Korea, in 2018, and 2020, respectively. He received the Best Paper Award from Korea Multimedia Society three times (May. 2017, Nov. 2019, Jul. 2021). And he received Best Paper Award from Korea Convergence Software Society (Oct, 2020). Also, he received Best Paper Award from Korea Federation of Information Technology Societies (Nov. 2020). Also, he received Best Paper Award from Korea Electronics and Telecommunications Research Institute (Apr. 2021). Currently, he is a Ph.D. student with the Department of Multimedia Engineering and Interdisciplinary Program in IT-Bio Convergence System, Sunchon National University, Suncheon, Korea. His research interests include the computer vision, deep learning, generative models, explainable AI, and adversarial attacks.

Jun-Yeong Kim received B.E., and M.E. degrees in multimedia engineering from Sunchon National University, Suncheon, Korea, in 2019, and 2021, respectively. He received the Best Paper Award from Korea Multimedia Society three times (Nov. 2018, Apr. 2021, Dec. 2021). And, he received Best Paper Award from Korea Institute of Electronic Communication Sciences (Jun, 2021). Also, he received Best Paper Award from Korean Institute of Smart Media (May, 2020). Currently, he is a Ph.D. student with the Department of Multimedia Engineering and Interdisciplinary Program in IT-Bio Convergence System, Sunchon National University, Suncheon, Korea. His research interests include the computer vision, deep learning, object detection, and anomaly detection.

Jun Park received B.E., and M.E. degrees in multimedia engineering from Sunchon National University, Suncheon, Korea, in 2019, and 2021, respectively. Currently, he is a Ph.D. student with the Department of Multimedia Engineering and Interdisciplinary Program in IT-Bio Convergence System, Sunchon National University, Suncheon, Korea. His research interests include the computer vision, deep learning, anomaly detection, and big data analytics.

Se-Hoon Jung received B.S., M.S., and Ph.D. degrees in multimedia engineering from Sunchon National University, Suncheon, Korea, in 2010, 2012, and 2017, respectively. He was a Assistant Professor (Tenure Track) at Youngsan University, Republic of Korea, from September 2019 to February 2020. And he was a Assistant Professor (Tenure Track) at Andong National University, Republic of Korea, from March 2020 to August 2022. He was the role of Keynote Speaker at the INCITEST held in Bandung, West Java, Indonesia, on 18th July 2019. He was a Center Chair (Director) of AI convergence education center at Andong National University, from February 2020 to August 2022. He was a Center Chair (Director) of SW industry-academic cooperation center, Andong National University, from April 2021 to August 2022. Currently, he is an Assistant Professor (Tenure Track) with the Department of Computer Engineering, Sunchon National University, Suncheon, Korea. His research interests include the software engineering, reinforcement learning, block chain, artificial intelligence, data mining, big data analysis, and big data prediction.

Chun-Bo Sim received B.S., M.S., and Ph.D. degrees in computer engineering from Chonbuk National University, Jeonju, Republic of Korea, in 1996, 1998, and 2003, respectively. From 2004 to 2005, he was Assistant Professor Department of Computer Engineering, Catholic University of Pusan, Republic of Korea. Currently, he is a Full Professor (Tenured) with the Department of Artificial Intelligence Engineering, Sunchon National University, Republic of Korea, from 2005 to Now. Also, he is the head of the Industry-University Collaboration Team at Sunchon National University, Republic of Korea. He was Invitational lecture at the 53rd International Conference of the Korean Society of Japanese Language and Literature, October 12, 2019 (Sat). Also, he was Invitational lecture at the 54th International Conference of the Korean Society of Japanese Language and Literature, October 17, 2020 (Sat). He was the director of the Institute of Information and Computer Science at Sunchon National University, August, 2021. He was an exchange professor at North Carolina Central University, United States of America (February, 2015-July, 2016). His research interests include the big data, block chain, deep learning, generative models, natural language processing, and reinforcement learning.