Received October 22, 2020, accepted October 24, 2020, date of publication November 3, 2020, date of current version November 13, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3035674

Restoring Latent Vectors From Generative Adversarial Networks Using Genetic Algorithms

YEONGBONG JIN AND BONGGYUN KO
Department of Statistics, Chonnam National University, Gwangju 61186, South Korea
Corresponding author: Bonggyun Ko (bonggyun.ko@jnu.ac.kr)

This work was supported in part by the National Research Foundation of Korea(NRF) Grant Funded by the Korea Government (MSIT) under Grant 2019R1G1A110070412, and in part by the Chonnam National University under Grant 2020-2010.

ABSTRACT Rapid and massive advances in deep learning have made it possible to address with issues with computer vision. In recent years, one type of generative model has emerged, generative adversarial networks (GAN), that enables creating realistic and plausible images. GAN allows for building competition models based on game theory that allows for modeling data probability distributions. Since the introduction of GAN, researchers have conducted many follow–up studies to apply and improve these models. In this article, using a global optimization technique called a genetic algorithm, we suggest the methodology restoring latent vector of pre–trained GAN and measure its performance as hyper–parameters and fitness functions; specifically, we utilize the mating and mutation rate as hyper–parameters of the genetic algorithms and use the mean squared error and the structural similarity as the fitness functions and evaluate their impact. We obtain image reconstruction results through experiment using the MNIST, Fashion–MNIST, Cifar10, and CelebA dataset, and we compare our method with the gradient descent method. We discuss limitations of these experiments and future research topics.

INDEX TERMS Generative adversarial networks, genetic algorithm, neural networks, image processing, latent vector.

I. INTRODUCTION

The generative model is considered to be the core of problems in artificial intelligence because it requires a complete understanding of how data are generated in the real world from a theoretical point of view. The generative model is a method of learning data distribution from a training dataset so as to generate new data points using unsupervised learning, and these models have been studied widely. Traditional generative models such as Hidden Markov Models [1] and Gaussian Mixture Models [2] can be used to approximate actual data distribution, but they are based on rigorous and complex expression, and they must satisfy certain assumptions. Deep learning algorithms have been used to improve the limitations of the traditional methods, and researchers have made impressive progress in recent years. In particular, the generative adversarial networks (GAN), proposed by Goodfellow *et al.* [3] in 2014, has become one of the most widely discussed generative models.

The associate editor coordinating the review of this manuscript and approving it for publication was Junxiu Liu.

The GAN is a network structure that consists of a generator and discriminator that are adversarial to each other [3]. The generator G learns complex real data x to generate realistic fake samples $G(z)$ from latent vector z, and the discriminator D determines whether the data generated $G(z)$ by the generator G are authentic; this competitive learning in turn improves the performance of these distinct networks. After the model is trained, the generator can create plausible samples that the discriminator cannot distinguish.

Since the advent of GAN, investigators have conducted a broad range of related studies. These networks are mainly used in computer vision areas such as creating new images [4]–[7] and recovering low–resolution photos [8], but studies are underway in other fields such as audio synthesis [9], [10], object detection [11], [12] and time series [13]. In order to edit original images, the way to approach the latent vector is also used [14], [15], and some researchers have studied it in applications such as inverting generator of GAN [16] and face–aging [17].

As part of the dimensionality reduction, there have been several studies on the latent vector of GAN, and most of their authors relied on gradient descent [18], [19], but other

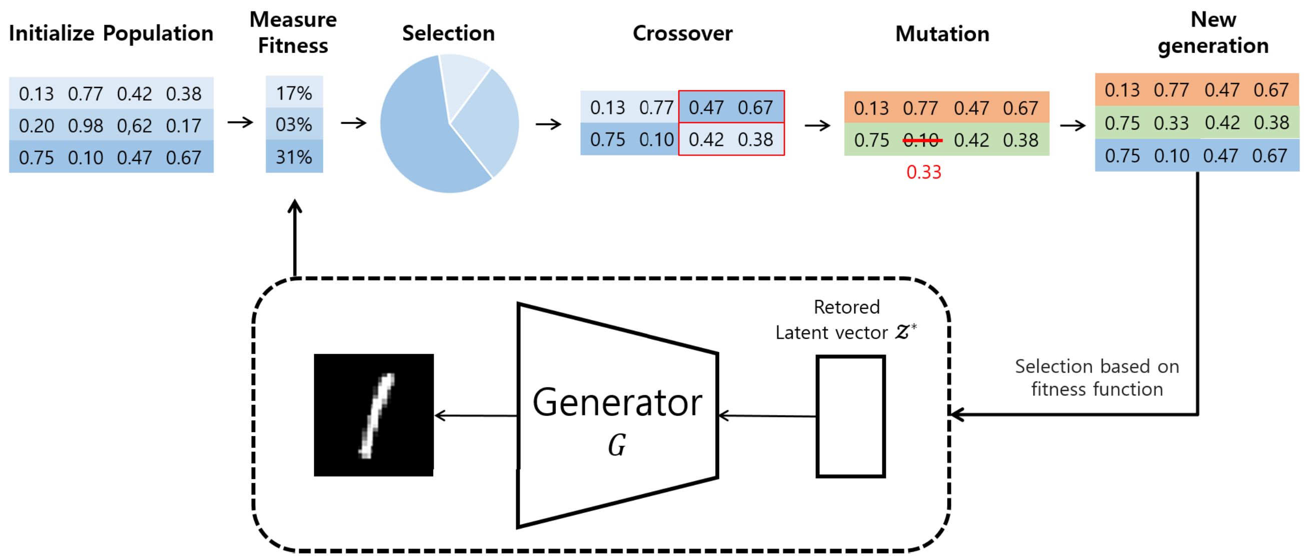


FIGURE 1. The overall framework for a methodology restoring latent vectors of pre-trained GANs using genetic algorithm.

researchers have attempted to overcome the limitations of GAN using genetic algorithms [20], [21]. Their main purpose, however, was not to restore the latent vector but to improve the networks' performance.

Fig. 1 presents our proposed framework, and the contributions of our work are as follows:

1) We propose a method to explore the latent space of the generator in a GAN using genetic algorithms. The genetic algorithm updates a candidate solution as it goes through the process of selection, crossover, and mutation. Through this process. if the restored latent vector z^* is found, then the image produced when it is used as the input of the pre–trained GAN is compared with the actual image.
2) In the genetic algorithms, we compared the results according to the hyper–parameters and the fitness function. In this case, we used the mating and mutation rates as hyper–parameters, and we used mean squared error (MSE) and structural similarity (SSIM) as fitness functions of the genetic algorithms.
3) We evaluated our method on the MNIST, Fashion–MNIST, Cifar10, and CelebA datasets to determine the robustness of the experiment.

The remainder of this article is summarized as follows: In Section II, we present content related to this study, including GAN and genetic algorithms. The detail of GAN and introduction of fitness function are presented in Section III. In Section IV, we discuss the experimental results, and we conclude the paper in and Section V by discussing the limitations of the study and future research directions.

II. RELATED WORKS

A. GENERATIVE ADVERSARIAL NETWORK

The structure of the GAN consists of two neural networks. The generator approximates the real data distribution in order to generate the fake data sample that would confuse the other neural network and the discriminator classifies whether the sample is generated from the real data or the fake data. If the generator can accurately describe the distribution of the real data, it is impossible to distinguish whether the sample is real or fake.

That is, a GAN [3] is a system of neural networks that have the purpose of strengthening their functions by repeatedly solving a two–player minimax game problem. The generator and discriminator are trained by the loss function:

$$\min_{\mathrm{G}} \max_{\mathrm{D}} V(G, D) = \mathbb{E}_{x \sim p_{data}(x)}[\log D(x)] + \mathbb{E}_{z \sim p_z(z)}[\log(1 - D(G(z)))] \quad (1)$$

where generator G generates a data point $G(z)$ by receiving a latent vector z sampled from a data distribution $p_z(z)$, which is generally a normal or uniform distribution.

From the perspective of minimizing the loss function of G, it can be seen that the first term of the Eq. (1) is not related to G and the second term of the equation is lower the closer $D(G(z))$ gets to 1. In this way, G is trained to deceive the discriminator D, and D meanwhile is trained to maximize Eq. (1). In the first and second terms of Eq. (1), if $D(x)$ is 1 and $D(G(z))$ is 0, the loss function has the maximum value, and this indicates D is trained to distinguish between real and fake.

B. THE LATENT VECTOR OF GAN

The ability of a GAN to generate realistic images by mapping from latent vectors implies that the generator's latent vector can detect the features of real data. To more clearly understand this implication, there have been a number of studies related to the latent vector of GAN.

Some researchers used Riemannian geometry to smoothly change from one fake image to another through geometric analysis and interpolation of latent vectors [22]–[24]. These investigators obtained better images than they did with other metrics, but their main concern was not the latent vector reconstruction or semantic control but the geometric analysis of the latent space.

By applying the arithmetic feature of the latent vector, Radford *et al.* [25] modified images using a simple latent vector operation. Similarly, Upchurch *et al.* [26] showed that image features could be extracted and that the image could be smoothly edited using interpolation.

There has also been work to restore the latent vectors. In particular, to manipulate the existing image, Zhu *et al.* [14] expressed the original image as a low–dimensional latent vector and then reconstructed it using GAN; they assumed that all images could be mapped to an ideal low–dimensional manifold. However, because it is difficult to directly model this manifold, the authors approximated using GAN. In a similar way, Lipton and Tripathi [15] proposed a method of recovering the latent vector using gradient–based optimization that minimized the L_2 norm.

Using the encoder to restore the latent vector, Antipov *et al.* [17] modified face images to fit different age groups. They combined conditional GAN (cGAN) and an encoder. Specifically, through the encoder, the latent vector of the original image is restored, and the conditional vector of cGAN is modified to generate a rejuvenated or aged image. Similarly, Perarnau *et al.* [27] proposed the invertible conditional GAN (IcGAN) that combined two encoders and cGAN for image editing. They extracted the latent vector and the conditional vector using one encoder for each and modified the original image to satisfy the condition through represented conditional vector modification.

Unlike existing methods for restoring latent vector, our novel process attempts to restore latent vector through an approach using genetic algorithms. As a result of restoring the image using this, it was shown that an image similar to the target image was created. Also, we did not adopt the method of introducing encoder for rapid optimization. This is because the use of additional encoders to restore the latent vector increases the parameters that model needs to learn. In addition, images that were not used during GAN learning were restored to confirm whether there was overfitting.

C. GENETIC ALGORITHMS

Genetic algorithm is one of the global optimization techniques: In 1975, Holland [28] proposed a method of modeling the evolutionary process of organisms that adapt to their natural environments through genetic changes. The genetic algorithm expresses a given problem in a genetic form and then derives an optimal solution to the problem through evolutionary calculations such as natural selection, crossover, and mutation. Fig. 2 shows several steps of the genetic algorithm. First, a randomly generated initial population is constructed to search for an optimal solution. The initial population is evaluated based on the fitness function, and among them, the best individuals are naturally selected. Some of the selected individuals then produce the next generation through a crossover that exchanges genetic information and a mutation process that randomly alters the genetic information.

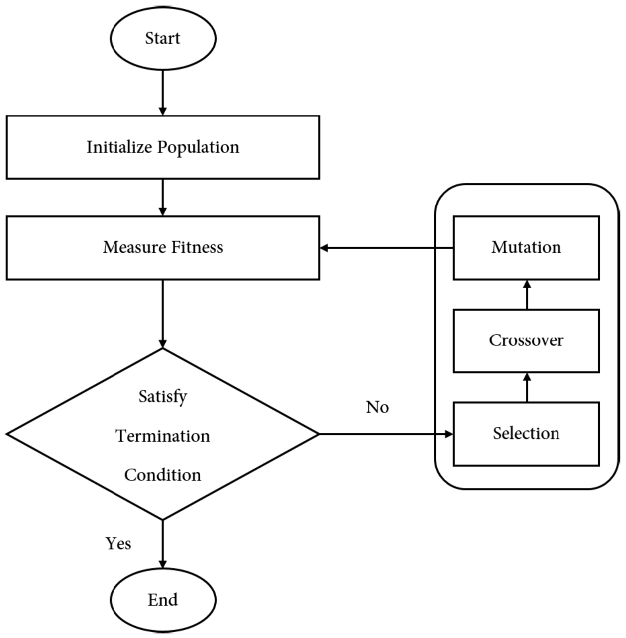


FIGURE 2. The flowchart of genetic algorithms.

Through repetitions of repeating this procedure, the optimal solution can be found.

III. METHODOLOGY

In this work, we propose to restore the latent vector of generator using genetic algorithms as an alternative to gradient descent based on L_p metrics. Similar to our objective, other researchers have attempted to reconstruct images by restoring the latent vector [15], [16]. In restoring the latent vector, genetic algorithms have the advantage of being able to find a global optimal point, while gradient descent methods, have the risk of finding a local optimal point. It is also possible to modify images through restoring the latent vector of pre—trained GAN using genetic algorithms.

A. DATASET

In this article, we used four common public datasets for restoring the latent vector of GAN: MNIST, Fashion–MNIST, Cifar10, and CelebA. MNIST is collection of 70,000 handwritten digit pictures from 0 to 9, consisting of 28×28 grayscale images [29]. Fashion–MNIST is a dataset derived from MNIST that consists of 28×28 grayscale images and has 10 fashion categories [30]. Unlike the previous ones, Cifar10 is a dataset of color images, each 32×32 [31]. CelebA is a dataset of color images of celebrities around the world, each size of 178×218 [32].

For our experiments, we resized CelebA dataset images to 32×32.

B. MODEL STRUCTURE OF GAN

Our goal with the experiments in this study was to restore the latent vector of GAN, so we used a simple structure. The generator consisted of two fully connected layers and two deconvolution layers and used the batch normalization [33] to increase the efficiency of learning to the rest of the layers except the output layer. We used the rectified linear unit (ReLU) function [34] for the rest of the layers except the last output layer as activation function, and in the last output layer, we used the hyperbolic tangent function to match the range of the pixel range with the output value of the actual image.

The discriminator consists of two convolution layers and two fully connected layers and used the batch normalization except the input and output layer. We used leaky ReLU [35] for the rest of the layers except the last output layer as the activation function, and we used the sigmoid function as the activation function to distinguish between real and fake images in the last output layer.

C. FITNESS FUNCTION FOR GENETIC ALGORITHMS

The selection of the loss function did not attract much attention in the computer vision area, but there is study on which loss to choose [36]. The loss function mainly used in the image restoration task is squared L_2 norm, and some studies mentioned in Section II [14], [15], [17] are based on this.

However, the L_2 norm has a limitation that does not reflect the properties of the Human Visual System well [37]. As an alternative to this limitation, SSIM designed to assess visual quality difference and similarity is also used, and we applied these two functions to latent vector optimization through genetic algorithms.

1) L_p DISTANCE

The L_p distance is the most general distance function that describes the distance between two places and defined as:

$$\|\text{x-y}\|_p = \left(\sum_{i=1}^{n} |x_i - y_i|^p\right)^{\frac{1}{p}} \tag{2}$$

It is commonly used when p is 0,2 and ∞ [38]. L_0 counts the number of non-zero elements, that is, the number of data points that have changed. L_2 represents the Euclidean distance between two points x and y, which is the root mean squared error. When p is infinite, it represents the largest change of all data points.

We used MSE, the square of L_2 distance, as a fitness function of the genetic algorithms; in this study, MSE ranged between 0 and 1 because we normalized the images. Therefore, in order to more intuitively understand, we changed the formula so that MSE closer to 0 was different from the original image and MSE closer to 1 was similar to the original image. Eq. 3 presents the actual fitness function that we used, defined as:

$$MSE = \sum_{i=1}^{n} (x_i - y_i)^2 / n$$

$$Fitness = 1 - MSE \tag{3}$$

2) STRUCTURAL SIMILARITY

The SSIM index reflects structural information for measuring the quality of an image [37]. The index is obtained by calculating the luminance, contrast, and structure between two

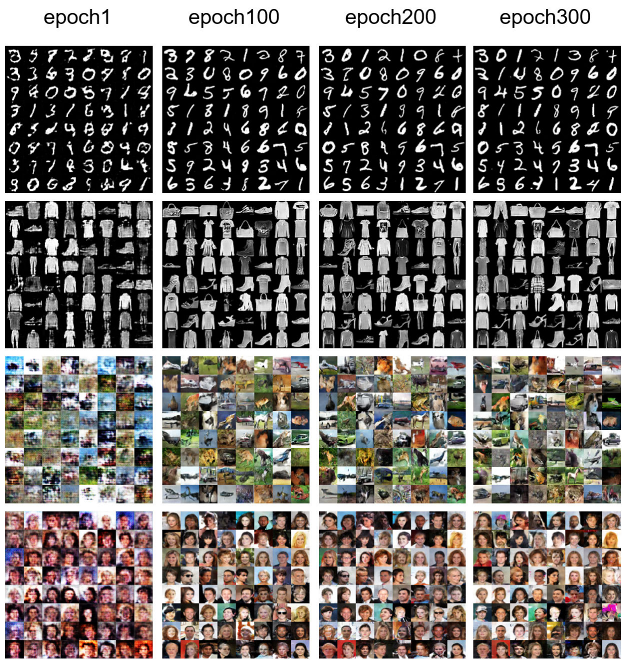


FIGURE 3. The result images of trained GAN.

images and multiplying them. When the image size is $N \times N$, expressed as image $x = \{x_i | i = 1, 2, \cdots, N \times N\}$ and image $y = \{y_i | i = 1, 2, \cdots, N \times N\}$, the SSIM index is defined as:

$$SSIM(x, y) = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)} \tag{4}$$

In this case, μ_x and μ_y are the means within each image window, and σ_x and σ_y are the variances within each window. The SSIM index ranges from 0 to 1 where there is no structural similarity between the two images at 0 they become more structurally similar as the index is closer to 1:

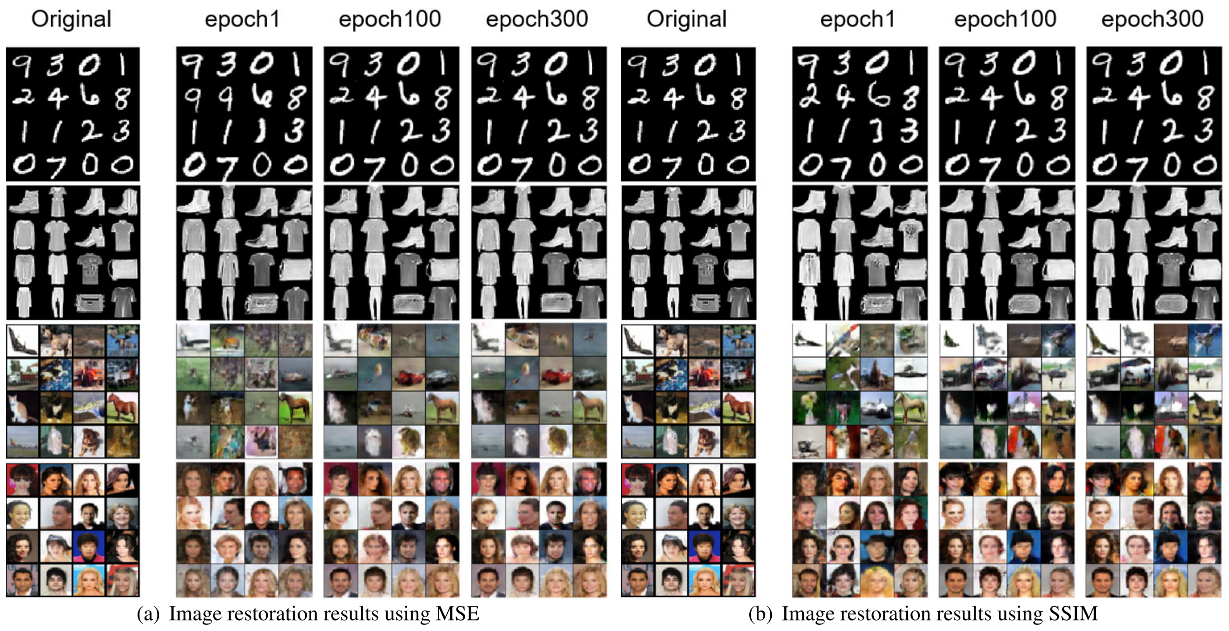
IV. EXPERIMENTAL RESULTS

A. THE RESULT OF TRAINED GAN

As mentioned in III-B, we used a simple structure to train the GAN, and Fig. 3 shows the result; we trained the network on 300 iterations. In the cases of MNIST and Fashion–MNIST, despite the simple structure, the training results looked good, while with Cifar10 and CelebA, the overall outline and color were trained, but not to the image details.

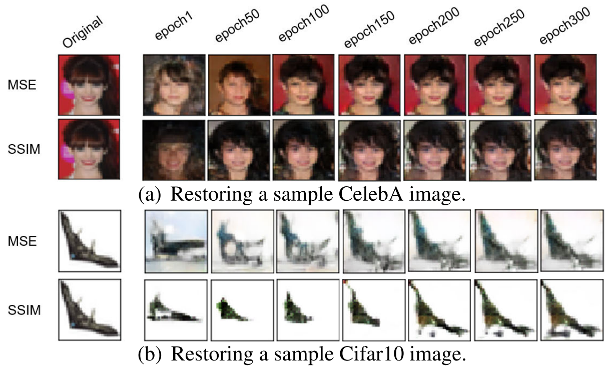
B. FITNESS FUNCTION COMPARISON

We utilized MSE, a converted form of L_2 distance, and SSIM as a fitness function of genetic algorithms, and the overall results were better with MSE than with the SSIM index. However, the difference was not large as shown in Fig. 4. The detailed differences between the two fitness functions can be found in Fig.5. The first and third lines in the Fig. 5 are images with MSE as a function of fitness, and the second and fourth lines are images with SSIM applied. While MSE rather than SSIM as a fitness function showed better overall performance, but tended to be similar for images without background. This is because SSIM does not reflect the color difference [39].



(a) Image restoration results using MSE (b) Image restoration results using SSIM

FIGURE 4. Image restoration according to genetic algorithm iterations after mating rate is fixed.



(a) Restoring a sample CelebA image.
(b) Restoring a sample Cifar10 image.

FIGURE 5. Image restoration according to genetic algorithm iterations, First row: fitness function is MSE; Second row: fitness function is SSIM.

C. HYPER–PARAMETER SETTING

The hyper–parameters that we adjusted were the mating rate and the mutation rate relative to the population of genetic algorithms. We set the population per generation to 200 and experimented with a grid search method, increasing each experimental parameter by 0.1 from 0.1 to 0.9. The experiment showed that both the mating and mutation rate were good around 0.2 and 0.4; with parameters beyond the proper range, the optimal updates are not performed well.

Table. 1 shows the average of all four datasets as scores for each hyper–parameter using MSE as a fitness function. Similarly, Table. 2 shows the result of using SSIM as the fitness function. We can confirm that mutation rate has a greater effect on image restoration than mating rate. In addition, when the mutation rate goes from 0.3 to 0.4, it can be confirmed that the performance decreases rapidly. The range of hyper–parameters that show the best performance for each dataset was 0.2 to 0.3 for mutation rate and 0.2 to 0.5 for mating rate. In fact, the datasets excluding the MNIST show the best performance when the mutation rate was 0.3.

TABLE 1. The result of grid search using MSE as a fitness function for each hyper–parameter value.

Mating rate	Mutation rate								
	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
0.1	0.980	0.984	0.985	0.959	0.961	0.961	0.961	0.960	0.962
0.2	0.981	0.983	**0.987**	0.965	0.963	0.964	0.964	0.966	0.964
0.3	0.983	0.986	0.986	0.968	0.968	0.964	0.965	0.966	0.965
0.4	0.982	0.983	**0.987**	0.965	0.965	0.967	0.965	0.966	0.967
0.5	0.982	0.983	**0.987**	0.968	0.968	0.967	0.968	0.967	0.968
0.6	0.982	0.983	0.984	0.966	0.967	0.966	0.967	0.967	0.967
0.7	0.981	0.983	0.983	0.966	0.966	0.964	0.965	0.966	0.966
0.8	0.980	0.982	0.984	0.964	0.965	0.964	0.965	0.964	0.964
0.9	0.979	0.979	0.981	0.962	0.962	0.963	0.961	0.962	0.961

TABLE 2. The result of grid search using SSIM as a fitness function for each hyper–parameter value.

Mating rate	Mutation rate								
	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
0.1	0.615	0.644	0.649	0.573	0.576	0.570	0.579	0.570	0.584
0.2	0.624	0.647	0.644	0.600	0.591	0.595	0.599	0.596	0.593
0.3	0.624	**0.652**	0.651	0.620	0.625	0.597	0.607	0.614	0.606
0.4	0.615	0.642	0.646	0.606	0.613	0.616	0.610	0.609	0.614
0.5	0.614	0.645	0.641	0.620	0.620	0.632	0.626	0.625	0.627
0.6	0.604	0.636	0.634	0.617	0.617	0.613	0.618	0.623	0.612
0.7	0.603	0.634	0.634	0.592	0.606	0.603	0.607	0.604	0.611
0.8	0.580	0.620	0.619	0.612	0.596	0.592	0.598	0.600	0.592
0.9	0.556	0.593	0.601	0.572	0.578	0.579	0.589	0.587	0.571

To detailed identify parameters that have a greater impact on the restoration of the latent vector, we fixed one element with a value of 0.3, which we thought was appropriate through the experiment, and then conducted the experiment

TABLE 3. Scores of using MSE as a fitness function for each epoch after fixing mating rate.

Data	Mutation rate	1	50	100	150	200	250	300
MNIST	0.2	0.952	0.991	0.993	0.994	0.994	0.994	**0.994**
	0.3	0.960	0.992	0.993	0.993	0.994	0.994	**0.994**
	0.4	0.958	0.970	0.970	0.970	0.970	0.970	0.970
	0.5	0.954	0.969	0.969	0.969	0.969	0.969	0.969
Fashion – MNIST	0.2	0.973	0.990	0.991	0.991	0.991	0.991	0.991
	0.3	0.970	0.991	0.991	0.992	0.992	0.992	**0.992**
	0.4	0.970	0.978	0.978	0.978	0.978	0.978	0.978
	0.5	0.971	0.977	0.977	0.977	0.977	0.977	0.977
Cifar10	0.2	0.953	0.974	0.976	0.977	0.977	0.977	**0.978**
	0.3	0.954	0.973	0.974	0.976	0.976	0.977	0.977
	0.4	0.956	0.960	0.960	0.960	0.960	0.960	0.960
	0.5	0.953	0.959	0.959	0.959	0.959	0.959	0.959
CelebA	0.2	0.956	0.978	0.981	0.982	0.982	0.983	0.983
	0.3	0.958	0.979	0.982	0.983	0.983	0.984	**0.984**
	0.4	0.955	0.964	0.964	0.964	0.964	0.964	0.964
	0.5	0.957	0.965	0.965	0.965	0.965	0.965	0.965

TABLE 4. Scores of using SSIM as a fitness function for each epoch after fixing mating rate.

Data	Mutation rate	1	50	100	150	200	250	300
MNIST	0.2	0.486	0.818	0.849	0.857	0.864	0.868	0.872
	0.3	0.497	0.825	0.855	0.863	0.868	0.872	**0.876**
	0.4	0.478	0.806	0.833	0.845	0.852	0.856	0.860
	0.5	0.524	0.775	0.810	0.826	0.838	0.846	0.850
Fashion – MNIST	0.2	0.474	0.701	0.718	0.728	0.733	0.736	**0.738**
	0.3	0.464	0.702	0.721	0.727	0.732	0.733	0.735
	0.4	0.436	0.682	0.701	0.708	0.713	0.716	0.720
	0.5	0.458	0.677	0.700	0.710	0.716	0.718	0.719
Cifar10	0.2	0.174	0.361	0.392	0.405	0.417	0.423	**0.431**
	0.3	0.176	0.348	0.374	0.390	0.399	0.405	0.409
	0.4	0.165	0.330	0.351	0.357	0.363	0.365	0.367
	0.5	0.169	0.323	0.346	0.355	0.360	0.365	0.369
CelebA	0.2	0.346	0.555	0.576	0.586	0.592	0.597	0.600
	0.3	0.336	0.560	0.586	0.596	0.603	0.608	**0.612**
	0.4	0.338	0.524	0.539	0.547	0.551	0.553	0.556
	0.5	0.325	0.540	0.564	0.576	0.585	0.589	0.592

TABLE 5. Scores of using MSE as a fitness function for each epoch after fixing mutation rate.

Data	Mating rate	1	50	100	150	200	250	300
MNIST	0.2	0.959	0.989	0.993	0.994	0.994	0.994	**0.994**
	0.3	0.960	0.992	0.993	0.993	0.994	0.994	**0.994**
	0.4	0.961	0.991	0.993	0.994	0.994	0.994	**0.994**
	0.5	0.955	0.990	0.993	0.993	0.993	0.994	**0.994**
Fashion – MNIST	0.2	0.971	0.991	0.992	0.992	0.992	0.992	**0.992**
	0.3	0.970	0.991	0.991	0.992	0.992	0.992	**0.992**
	0.4	0.972	0.990	0.991	0.991	0.992	0.992	**0.992**
	0.5	0.970	0.989	0.991	0.991	0.992	0.992	**0.992**
Cifar10	0.2	0.955	0.973	0.975	0.977	0.977	0.978	0.978
	0.3	0.954	0.973	0.974	0.976	0.976	0.977	0.977
	0.4	0.954	0.972	0.975	0.976	0.976	0.977	0.977
	0.5	0.953	0.973	0.975	0.977	0.978	0.978	**0.979**
CelebA	0.2	0.957	0.981	0.982	0.983	0.983	0.984	0.984
	0.3	0.958	0.979	0.982	0.983	0.983	0.984	0.984
	0.4	0.958	0.980	0.982	0.983	0.984	0.984	**0.985**
	0.5	0.955	0.979	0.981	0.982	0.983	0.983	0.984

TABLE 6. Scores of using SSIM as a fitness function for each epoch after fixing mutation rate.

Data	Mating rate	1	50	100	150	200	250	300
MNIST	0.2	0.533	0.831	0.849	0.856	0.860	0.863	0.864
	0.3	0.497	0.825	0.855	0.863	0.868	0.872	**0.876**
	0.4	0.503	0.799	0.842	0.854	0.858	0.861	0.865
	0.5	0.518	0.808	0.836	0.845	0.853	0.856	0.860
Fashion – MNIST	0.2	0.469	0.699	0.709	0.713	0.715	0.716	0.718
	0.3	0.464	0.702	0.721	0.727	0.732	0.733	**0.735**
	0.4	0.488	0.698	0.712	0.718	0.722	0.725	0.727
	0.5	0.464	0.680	0.708	0.717	0.722	0.726	0.728
Cifar10	0.2	0.182	0.358	0.374	0.389	0.397	0.400	0.406
	0.3	0.176	0.348	0.374	0.390	0.399	0.405	0.409
	0.4	0.189	0.356	0.380	0.397	0.405	0.411	**0.417**
	0.5	0.173	0.319	0.347	0.363	0.372	0.376	0.380
CelebA	0.2	0.345	0.562	0.589	0.597	0.604	0.611	0.616
	0.3	0.336	0.560	0.586	0.596	0.603	0.608	0.612
	0.4	0.341	0.546	0.576	0.588	0.598	0.604	0.609
	0.5	0.328	0.560	0.595	0.605	0.617	0.623	**0.627**

by adjusting the values of the other elements. As a result of adjusting the mating and mutation rates, we confirmed that the fitness value was more sensitive to the mutation rate than the mating rate.

1) FIX MATING RATE

Table. 3 and 4 present the fitness function scores using the MSE and SSIM between the regenerated images and the actual images for each mutation rate and the number of iterations after we fixed the mating rate of the genetic algorithms to 0.3. Each value in the Table. 3 and 4 is the average for just one batch, shown in Fig. 4. The figure presents the corresponding batch images from each dataset that used the hyper–parameter with the best MSE and SSIM.

First, looking at the instance of optimization using MSE, in the cases of MNIST and Fashion–MNIST, which are grayscale datasets with relatively well–trained GAN learning, the best solution was quickly found, while the color datasets Cifar10 and CelebA, updated at a slower rate. Next, we proceeded with optimization using SSIM. Similar to the case of using MSE, the grayscale image showed good restore results. However, the optimization speed was slow for the color dataset. In addition, when the mutation rate is 0.4 or higher, the optimal update is no longer well performed when the number of iteration exceeds 100.

2) FIX MUTATION RATE

Table. 5 and 6 show the fitness function scores using the MSE and SSIM between the regenerated and actual images for each mating rate and the number of iterations after we fixed the mutation rate of the genetic algorithms to 0.3. Table. 5 and 6 show that the scores by number of repetitions

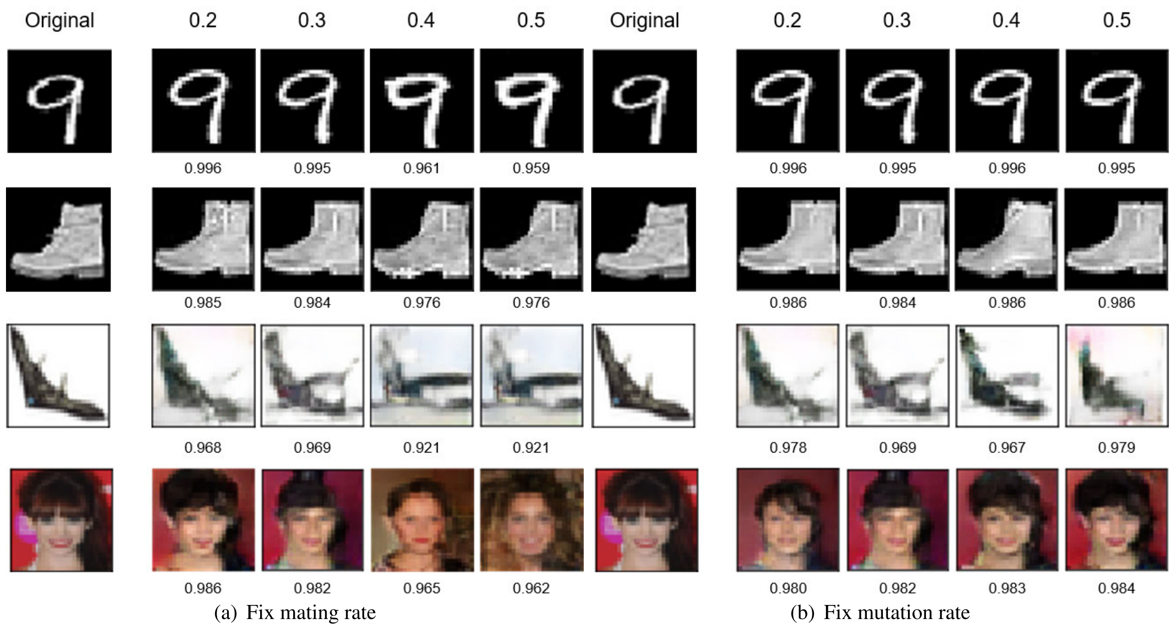


FIGURE 6. Image results by hyper–parameters of genetic algorithms.

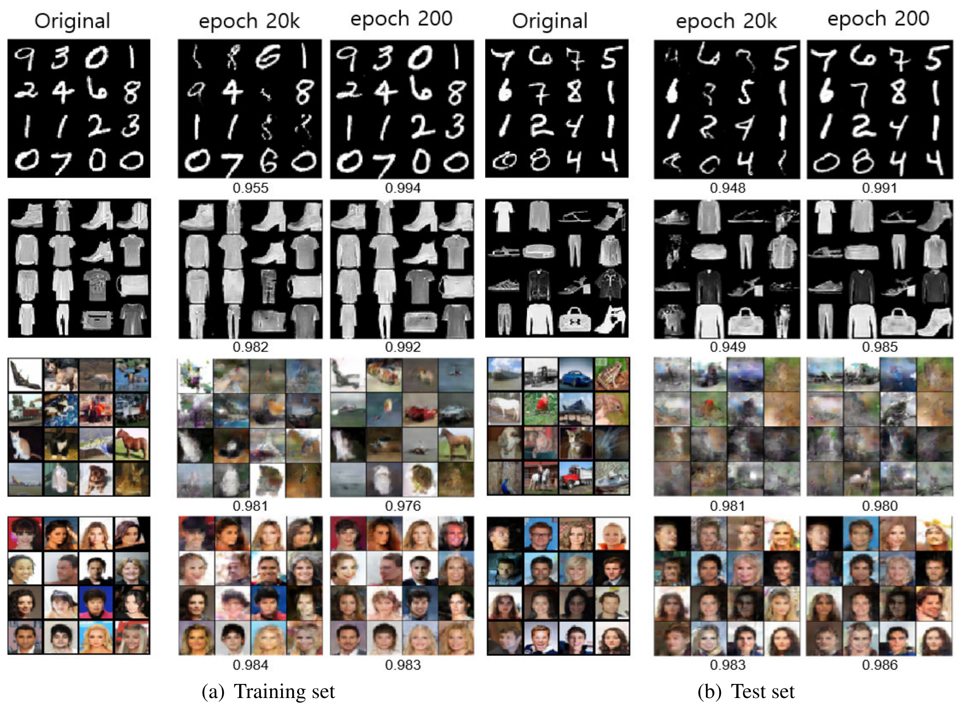


FIGURE 7. Image restoration results and average fitness score for each batch are shown, left column: original image; middle column: gradient descent result; right column: genetic algorithms result.

did not greatly differ. Although the detailed hyper–parameters for each dataset were different, they coalesced to find the optimal within a specific range.

Comparing Tables for each fitness function confirms the impacts of hyper–parameters. The range of values according to the number of repetitions has a larger range when the mutation rate varies for both MSE and SSIM. This suggests that the mutation rate has a greater effect on the optimization of the latent vector. Our experiments demonstrated that the mutation rate has a greater effect on restoring the latent vector, reflected in the results shown in Fig. 6.

The figure presents the results of restoring images with each hyper–parameter using MSE as a fitness function of genetic algorithms. Fig. 6 (a) presents the images and fitness scores after we fixed the mating rate, and (b) shows the images and scores after we fixed mutation rate. If the mutation rate varies a value of 0.4 or higher restores a quite dissimilar image from the original, as shown in Fig. 6 (a).

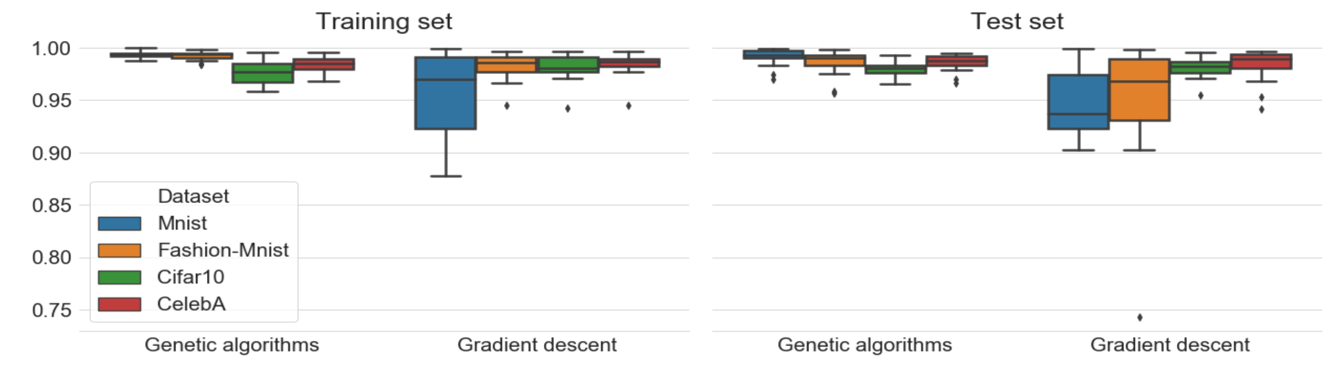


FIGURE 8. **Boxplot according to fitness score of each restoration result.**

In contrast, however when the mating rate varies, the image details differ slightly but the images generated are generally similar, and the fitness score is also similar.

D. COMPARISON OF GRADIENT DESCENT METHOD AND GENETIC ALGORITHMS METHOD

We also compared the result from restoring the latent vector based on genetic algorithms after 200 iterations with the findings for gradient descent after 20,000 iterations for each dataset. We conducted the experiment in the same environment, and the time required was similar. Fig. 7 shows the comparison results, with better overall performance for restoring images using genetic algorithms.

The performance difference between the two methods is clear in the grayscale datasets MNIST and Fashion–MNIST. In the case of the gradient descent method, there were cases where the restoration was not successful to the target, while the genetic algorithms method showed the successful restoration to the same category, and the restoration accuracy was also better. As a result of restoration using Cifar10 and CelebA datasets, both methods were able to capture the overall outline of object, but the restoration accuracy was not high.

In order to examine the generalization ability of each method, restoring latent vectors was performed using test set that unused when GAN learning. The image restoration results are shown in Fig. 7 (b). Both fitness scores and result of images restored are not much different from when using training set.

Fig. 8 is a boxplot using fitness scores of image restoration results depending on dataset and method. Overall, the image restoration results are better when using genetic algorithms, and the results using test set show the similar performance as training set.

Within the same time period, the method using genetic algorithms has a much fewer iterations than the gradient descent method, but showed a better approach to finding the optimal points.

V. CONCLUSION

In this work, we restored the latent vector of GAN using the basic genetic algorithm for convenience with two fitness functions. Based on this experiment, we compared the regenerated images with the original images and confirmed the result by measuring the fitness scores. The score showed higher values when using MSE, but this is due to structural differences in calculations, and the reconstructed result images did not show significant differences between the two functions.

As we expected, our proposed method not only found the latent vector of the images but also allowed for modifying the pre–trained GAN. Comparing with the gradient descent method, our method showed better performance in the restoration results. Since the method to be compared was the gradient-descent method using MSE, we used MSE as the fitness function of genetic algorithms. Genetic algorithms can find global optimal points, so finding hyper–parameters in an appropriate way can result in images similar to the original.

We verified the generalization performance by restoring using test set as well as training set used when learning the GAN. The restored latent vector was able to capture the properties of original images and also showed good performance even for test set that was not used when training GAN.

Despite the light structure of the GAN, it showed good restoration results for grayscale datasets. Although our method succeeded in restoring plausible latent vector, there were several limitations to the study. Primarily, we conducted our experiments using various datasets to test the robustness and effectiveness of our methods. But we did not conduct proper training on the three channel image datasets with the result that the color datasets received low fitness scores. In future work, we plan to use the modified genetic algorithm and various fitness functions, and compare the results with experimental methods based on encoder.

REFERENCES

[1] L. Rabiner and B. Juang, "An introduction to hidden Markov models," *IEEE ASSP Mag.*, vol. 3, no. 1, pp. 4–16, Jan. 1986.

[2] C. E. Rasmussen, "The infinite Gaussian mixture model," in *Proc. Adv. Neural Inf. Process. Syst.*, 2000, pp. 554–560.

[3] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Proc. Adv. Neural Inf. Process. Syst.*, 2014, pp. 2672–2680.

[4] T. Karras, S. Laine, and T. Aila, "A style-based generator architecture for generative adversarial networks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 4401–4410.

[5] T. Karras, T. Aila, S. Laine, and J. Lehtinen, "Progressive growing of GANs for improved quality, stability, and variation," 2017, *arXiv:1710.10196*. [Online]. Available: http://arxiv.org/abs/1710.10196

[6] D. Berthelot, T. Schumm, and L. Metz, "BEGAN: Boundary equilibrium generative adversarial networks," 2017, *arXiv:1703.10717*. [Online]. Available: http://arxiv.org/abs/1703.10717

[7] Y. Li, N. Xiao, and W. Ouyang, "Improved boundary equilibrium generative adversarial networks," *IEEE Access*, vol. 6, pp. 11342–11348, 2018.

[8] C. Ledig, L. Theis, F. Huszar, J. Caballero, A. Cunningham, A. Acosta, A. Aitken, A. Tejani, J. Totz, Z. Wang, and W. Shi, "Photo-realistic single image super-resolution using a generative adversarial network," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 4681–4690.

[9] T. Kaneko, H. Kameoka, N. Hojo, Y. Ijima, K. Hiramatsu, and K. Kashino, "Generative adversarial network-based postfilter for statistical parametric speech synthesis," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, Mar. 2017, pp. 4910–4914.

[10] C. Donahue, J. McAuley, and M. Puckette, "Adversarial audio synthesis," 2018, *arXiv:1802.04208*. [Online]. Available: http://arxiv.org/abs/1802.04208

[11] C. D. Prakash and L. J. Karam, "It GAN DO better: GAN-based detection of objects on images with varying quality," 2019, *arXiv:1912.01707*. [Online]. Available: http://arxiv.org/abs/1912.01707

[12] J. Li, X. Liang, Y. Wei, T. Xu, J. Feng, and S. Yan, "Perceptual generative adversarial networks for small object detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 1222–1230.

[13] M. Wiese, R. Knobloch, R. Korn, and P. Kretschmer, "Quant GANs: Deep generation of financial time series," 2019, *arXiv:1907.06673*. [Online]. Available: http://arxiv.org/abs/1907.06673

[14] J.-Y. Zhu, P. Krähenbühl, E. Shechtman, and A. A. Efros, "Generative visual manipulation on the natural image manifold," 2016, *arXiv:1609.03552*. [Online]. Available: http://arxiv.org/abs/1609.03552

[15] Z. C. Lipton and S. Tripathi, "Precise recovery of latent vectors from generative adversarial networks," 2017, *arXiv:1702.04782*. [Online]. Available: http://arxiv.org/abs/1702.04782

[16] A. Creswell and A. A. Bharath, "Inverting the generator of a generative adversarial network," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 30, no. 7, pp. 1967–1974, Jul. 2019.

[17] G. Antipov, M. Baccouche, and J.-L. Dugelay, "Face aging with conditional generative adversarial networks," in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, Sep. 2017, pp. 2089–2093.

[18] S. Santurkar, D. Budden, and N. Shavit, "Generative compression," in *Proc. Picture Coding Symp. (PCS)*, Jun. 2018, pp. 258–262.

[19] A. Bora, A. Jalal, E. Price, and A. G. Dimakis, "Compressed sensing using generative models," 2017, *arXiv:1703.03208*. [Online]. Available: http://arxiv.org/abs/1703.03208

[20] C. Wang, C. Xu, X. Yao, and D. Tao, "Evolutionary generative adversarial networks," 2018, *arXiv:1803.00657*. [Online]. Available: http://arxiv.org/abs/1803.00657

[21] V. Costa, N. Lourenço, and P. Machado, "Coevolution of generative adversarial networks," 2019, *arXiv:1912.06172*. [Online]. Available: http://arxiv.org/abs/1912.06172

[22] G. Arvanitidis, L. K. Hansen, and S. Hauberg, "Latent space oddity: On the curvature of deep generative models," 2017, *arXiv:1710.11379*. [Online]. Available: http://arxiv.org/abs/1710.11379

[23] H. Shao, A. Kumar, and P. T. Fletcher, "The Riemannian geometry of deep generative models," 2017, *arXiv:1711.08014*. [Online]. Available: http://arxiv.org/abs/1711.08014

[24] L. Kuhnel, T. Fletcher, S. Joshi, and S. Sommer, "Latent space non-linear statistics," 2018, *arXiv:1805.07632*. [Online]. Available: http://arxiv.org/abs/1805.07632

[25] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," 2015, *arXiv:1511.06434*. [Online]. Available: http://arxiv.org/abs/1511.06434

[26] P. Upchurch, J. Gardner, G. Pleiss, R. Pless, N. Snavely, K. Bala, and K. Weinberger, "Deep feature interpolation for image content changes," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 6090–6099.

[27] G. Perarnau, J. van de Weijer, B. Raducanu, and J. M. Álvarez, "Invertible conditional GANs for image editing," 2016, *arXiv:1611.06355*. [Online]. Available: http://arxiv.org/abs/1611.06355

[28] J. H. Holland, *Adaptation in Natural and Artificial Systems: An Introductory Analysis With Applicationsto Biology, Control and Artificial Intelligence*. Ann Arbor, MI, USA: Univ. Michigan Press, 1975.

[29] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.

[30] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-MNIST: A novel image dataset for benchmarking machine learning algorithms," 2017, *arXiv:1708.07747*. [Online]. Available: http://arxiv.org/abs/1708.07747

[31] A. Krizhevsky and G. Hinton, *Learning Multiple Layers of Features Fromtiny Images*. Toronto, ON, Canada: Univ. Toronto, 2009.

[32] Z. Liu, P. Luo, X. Wang, and X. Tang, "Deep learning face attributes in the wild," 2014, *arXiv:1411.7766*. [Online]. Available: http://arxiv.org/abs/1411.7766

[33] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," 2015, *arXiv:1502.03167*. [Online]. Available: http://arxiv.org/abs/1502.03167

[34] V. Nair and G. E. Hinton, "Rectified linear units improve restricted Boltzmann machines," in *Proc. 27th Int. Conf. Mach. Learn. (ICML)*, 2010, pp. 807–814.

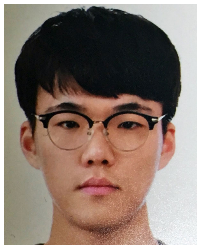
[35] A. L. Maas, A. Y. Hannun, and A. Y. Ng, "Rectifier nonlinearities improve neural network acoustic models," in *Proc. ICML Workshop Deep Learn. Audio, Speech Lang. Process.*, 2013, p. 3.

[36] H. Zhao, O. Gallo, I. Frosio, and J. Kautz, "Loss functions for image restoration with neural networks," *IEEE Trans. Comput. Imag.*, vol. 3, no. 1, pp. 47–57, Mar. 2017.

[37] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: From error visibility to structural similarity," *IEEE Trans. Image Process.*, vol. 13, no. 4, pp. 600–612, Apr. 2004.

[38] X. Yuan, P. He, Q. Zhu, and X. Li, "Adversarial examples: Attacks and defenses for deep learning," 2017, *arXiv:1712.07107*. [Online]. Available: http://arxiv.org/abs/1712.07107

[39] Y. Shi, Y. Ding, R. Zhang, and J. Li, "Structure and hue similarity for color image quality assessment," in *Proc. Int. Conf. Electron. Comput. Technol.*, Feb. 2009, pp. 329–333.



YEONGBONG JIN received the B.S. degree in statistics from Chonnam National University, South Korea, in 2019, where he is currently pursuing the master's degree. His research interests include machine learning algorithm, artificial intelligence methodologies, and financial time series analysis.

BONGGYUN KO received the B.S. and M.S. degrees in mathematical science from the Korea Advanced Institute of Science and Technology, South Korea, in 2013, and the Ph.D. degree in industrial engineering from Seoul National University, South Korea, in 2016.

From 2016 to 2018, he was a Senior Professional with the Big Data Analytics Group of Mobile Communications Business in Samsung Electronics. He was a Senior Data Scientist with the Hana Institute of Technology, Hana TI, South Korea, in 2018. He is currently an Associate Professor with the Department of Statistics, Chonnam National University, South Korea. His research interests include machine learning algorithm, financial time series analysis, and risk management.