RESEARCH ARTICLE

Cyclic image generation using chaotic dynamics

Takaya Tanaka[1], Yutaka Yamaguti[2]*

1 Graduate school of Engineering, Fukuoka Institute of Technology, Fukuoka, Fukuoka, Japan, **2** Faculty of Information Engineering, Fukuoka Institute of Technology, Fukuoka, Fukuoka, Japan

* y-yamaguchi@fit.ac.jp

Abstract

Successive image generation using cyclic transformations is demonstrated by extending the CycleGAN model to transform images among three different categories. Repeated application of the trained generators produces sequences of images that transition among the different categories. The generated image sequences occupy a more limited region of the image space compared with the original training dataset. Quantitative evaluation using precision and recall metrics indicates that the generated images have high quality but reduced diversity relative to the training dataset. Such successive generation processes are characterized as chaotic dynamics in terms of dynamical system theory. Positive Lyapunov exponents estimated from the generated trajectories confirm the presence of chaotic dynamics, with the Lyapunov dimension of the attractor found to be comparable to the intrinsic dimension of the training data manifold. The results suggest that chaotic dynamics in the image space defined by the deep generative model contribute to the diversity of the generated images, constituting a novel approach for multi-class image generation. This model can be interpreted as an extension of classical associative memory to perform hetero-association among image categories.

Author summary

We have developed a new approach to generate sequences of related images by extending a type of deep learning model called CycleGAN. Our model learns to transform images among three different categories in a cyclic manner. For example, it can turn an image of a T-shirt into a sneaker, then a bag, and back to a T-shirt. Our model generates a series of smoothly changing images by repeatedly applying these transformations. Interestingly, we found that the sequences of generated images exhibit chaotic dynamics. This means that even tiny changes to the starting image can lead to very different sequences. We characterized these chaotic dynamics quantitatively using metrics from chaos theory. Our results demonstrate a novel way to generate diverse images related to multiple categories. The generated images are of high quality but tend to be less diverse than the original training data. Our model may serve as an interesting model of associative memory from a theoretical neuroscience perspective.

Introduction

As deep learning has advanced, there has been extensive research into models that can generate realistic images [1–3], and one such model is the generative adversarial network (GAN) [1]. A GAN consists of two networks—the generator and the discriminator—that are trained adversarially to generate new images that are similar to those in the training dataset. This has led to the development of various extended models, one of which is CycleGAN [4], which is an advanced GAN that performs image translation by learning the relationship between two different image categories.

Elsewhere, associative memory models have been studied as models of biological memory in which data related to a given input pattern are stored and retrieved [5–8]. Such memory can be realized by Hebbian-type synaptic learning. As an extension of associative memory models, dynamic associative memory models that retrieve stored memory patterns by using chaotic dynamics were proposed from the late 1980s to the 1990s [9–11]. It has been shown that these associative memory models based on chaotic dynamics can autonomously generate sequential patterns that resemble memory patterns. From the perspective of dynamical systems, each memory pattern is a pseudo-attractor; that is, the state remains near the pattern for a while and then transitions chaotically to another pattern. Such dynamics is called chaotic itinerancy [12–14]. However, there has been limited research into using modern deep neural networks to generate successive similar images autonomously.

In machine learning, chaotic dynamical systems have been utilized in various ways. For example, chaotic time series are often used as benchmarks for evaluating the performance of time-series prediction models [15]. Moreover, Tanaka and Yamaguti [16] showed that a GAN can be trained to generate chaotic time series, and they evaluated the properties of the generated time series from the perspective of deterministic chaos. In the context of reservoir computing using recurrent neural networks, maintaining a weak chaotic state in the absence of input has been shown to enhance the learning of complex behaviors [17, 18].

However, despite these advancements, there has been limited research into constructing associative memory models that leverage chaotic dynamics within deep learning frameworks. Our study aims to fill this gap by proposing a novel approach that utilizes the power of chaos to explore the high-dimensional space of complex datasets. We construct a model that combines the image-transformation ability of deep learning with the autonomous and rich pattern-generation ability of chaotic dynamics to successively and associatively generate diverse images. Specifically, we develop a model that is an extension of CycleGAN [4], and we generate images iteratively by feeding those produced by the model back into the same model. While the conventional version of CycleGAN transforms images between two different image domains defined by two datasets, we construct a model that transforms images cyclically among three different domains. This successive image generation can be treated as a dynamical system defined in the image space, and so its behavior can be analyzed using concepts from nonlinear dynamical systems, such as attractors and chaos. Specifically, we use Lyapunov exponents and Lyapunov dimensions to quantify the characteristics of this dynamical system, then we relate those characteristics to those of the image generation. Our results show that this model can generate a wide range of images by utilizing chaotic dynamics. Moreover, the quality and diversity of the generated images are evaluated using precision and recall metrics [19].

Background

In this section, we provide some background to this study. Since this research is an interdisciplinary study involving machine learning and chaotic dynamical systems, we explain relevant basic concepts from both disciplines for a broad audience.

Generative adversarial network

A GAN [1] is a generative deep learning model that consists of two networks, namely, a generator and a discriminator. A characteristic of a GAN is that its generator and discriminator are trained adversarially. The generator generates data from given latent variables. The discriminator takes in either training data or data generated by the generator and outputs the probability that the input data are from the training data.

The details of the GAN are as follows. Let $X = \{x_1, \ldots, x_M\}$ be a set of data (dataset) existing in the space of images $J = I^N$ with N pixels, where $I = [-1, 1]$ is the interval from -1 to 1. Here, X is assumed to be a set of data independently sampled as $x \sim p_{\text{data}}(x)$ according to a probability distribution $p_{\text{data}}(x)$ on J. Let z be a latent variable sampled from a latent space J_z according to a known prior distribution p_z on J_z. The generator G is a mapping from $J_z \to J$ and the discriminator D is a mapping from $J \to [0, 1]$, and they have internal adjustable parameters θ_G and θ_D, respectively. The objective function V of the GAN is given by

$$\min_{\theta_G} \max_{\theta_D} V(D, G) = \min_{\theta_G} \max_{\theta_D} \left(\mathbb{E}_{x \sim p_{\text{data}}(x)}[\log D(x)] + \mathbb{E}_{z \sim p_z(z)}[\log\left(1 - D(G(z))\right)] \right), \tag{1}$$

where $\mathbb{E}$ represents the expected value.

When implementing G and D using neural networks and performing minimax training represented by Eq 1, the training phases of G and D are executed alternately. In each training iteration, the generator optimizes its internal parameters to improve its generation ability so that the discriminator will mistakenly recognize the generated data as being real. Conversely, the discriminator optimizes its internal parameters to distinguish between real and fake data. This step is repeated alternately.

CycleGAN

As an extension of a GAN, CycleGAN [4] learns the relationship between two unpaired image datasets and then can translate images between them. For example, it can transform a photograph of an actual landscape into a painting of that landscape in the style of a particular artist. Let $J = I^N$ be the space of images with N pixels, where $I = [-1, 1]$, and suppose that there are two different datasets $X = \{x_1, x_2, \ldots, x_{M_X}\}$ and $Y = \{y_1, y_2, \ldots, y_{M_Y}\}$ on J. It is assumed that each data point in X and Y is independently sampled from the probability distributions P_X and P_Y on J, respectively. CycleGAN has two generators, namely, $G: J \to J$ and $F: J \to J$, and two discriminators, namely, $D_X: J \to [0, 1]$ and $D_Y: J \to [0, 1]$. Generator G learns the transformation from X to Y, while generator F learns the transformation from Y to X. Discriminator D_Y (resp. D_X) distinguishes whether the input data were generated by generator G (resp. F) or came from dataset Y (resp. X).

The adversarial loss used to train the generators and discriminators is the same as in a conventional GAN, except that the generators' input is a sample image. The loss for G and D_Y is expressed as

$$\begin{aligned} \mathcal{L}_{\text{GAN}}(G, D_Y, X, Y) &= \mathbb{E}_{y \sim P_Y}[\log D_Y(y)] \\ &+ \mathbb{E}_{x \sim P_X}[\log\left(1 - D_Y(G(x))\right)]. \end{aligned} \tag{2}$$

However, using only the adversarial loss, the learning of G and F remains independent, and the mutual conversion between domains may not converge well. To address this issue, an additional loss term is introduced to encourage G and F to approach an inverse mapping

relationship, i.e., $F(G(x)) \approx x$. Called the cycle-consistency loss, this term is defined as

$$
\begin{aligned}
\mathcal{L}_{\text{cyc}}(G, F) \quad &= \mathbb{E}_{x \sim p_{\text{data}}(x)}[\| F(G(x)) - x \|_1] \\
&+ \mathbb{E}_{y \sim p_{\text{data}}(y)}[\| G(F(y)) - y \|_1].
\end{aligned}
\tag{3}
$$

The cycle-consistency loss ensures that the learned transformations G and F are consistent with each other, enabling the model to capture the underlying correspondence between the two domains. Here, $\| \|_1$ denotes the L1 norm, also known as the Manhattan distance. The total objective function is defined by combining the adversarial loss and the cycle-consistency loss:

$$
\begin{aligned}
\mathcal{L}(G, F, D_X, D_Y) \quad &= \mathcal{L}_{\text{GAN}}(G, D_Y, X, Y) \\
&+ \mathcal{L}_{\text{GAN}}(F, D_X, Y, X) \\
&+ \lambda \mathcal{L}_{\text{cyc}}(G, F),
\end{aligned}
\tag{4}
$$

where λ is a parameter to control the relative weights of the two losses. Then, learning is performed to find the optimal solution that satisfies

$$
G^*, F^* = \arg \min_{G,F} \max_{D_X, D_Y} \mathcal{L}(G, F, D_X, D_Y),
\tag{5}
$$

where G^* and F^* represent the optimal models with the optimal parameters.

Chaotic dynamics and Lyapunov exponents

Chaos is the irregular and seemingly random behavior in deterministic dynamical systems [20]. In the context of dynamical systems, a "trajectory" refers to the path that the system's state traces over time, starting from a given initial state. A key characteristic of chaos is sensitive dependence on the initial state, meaning that even small differences in the initial state can lead to very different trajectories over time. This sensitivity is quantified by Lyapunov exponents, which measure the average rate of divergence between nearby trajectories.

Lyapunov exponents are important quantities in the characterization of chaos [20]. In an N-dimensional dynamical system, there are generally N Lyapunov exponents that depend on the initial conditions, and the set of these exponents is called the Lyapunov spectrum. In a dynamical system defined by a map $f : \mathbb{R}^N \to \mathbb{R}^N$ on an N-dimensional space as $x_{n+1} = f(x_n)$, $x_n \in \mathbb{R}^N$ ($n = 0, 1, \ldots$), the Lyapunov exponents are calculated as the average of the logarithm of the local expansion rate on the trajectory generated by the map f. The Jacobian matrix M_n for n iterations of f is expressed as

$$
M_n = \prod_{k=0}^{n-1} J_k,
$$

where J_k is the Jacobian matrix $J_k = \frac{\partial}{\partial x} f(x_k)$ at x_k. Under certain conditions, it can be shown that the following matrix exists [21]:

$$
\Lambda = \lim_{n \to \infty} (M_n^T M_n)^{\frac{1}{2n}}.
\tag{6}
$$

The Lyapunov spectrum $\{\lambda_i\}$, $i = 1, \ldots, N$ is obtained as the set of logarithms of the eigenvalues of the matrix Λ.

In the present numerical calculations, the expansion rate was estimated by the commonly used method of Gram–Schmidt orthonormalization [22]. Although the present generator

model has a deep structure, each layer of the model is composed of functions that are differentiable almost everywhere. Consequently, the entire map G is also differentiable almost everywhere, and the Jacobian matrix can be obtained numerically. The Jacobian matrices were calculated numerically by using the "batch_jacobian" function in the TensorFlow [23] library.

Model and methods

Model

We have developed a model that extends CycleGAN [4] to transform input images into different images cyclically, as shown in Fig 1, where X, Y, and Z represent different image categories within the image space J. This model has two generators (G and F), and the transformations that they learn are represented by the solid (G) and dashed (F) arrows in Fig 1. Additionally, the model incorporates three discriminators: D_X, D_Y, and D_Z, each corresponding to one of the image categories. Similar to CycleGAN, G and F are trained so that they are inverse maps of each other.

Below, we provide a brief overview of the network structures of the generator and discriminator used in this experiment, with more details available in S1 and S2 Figs in the supporting information. In the generator, the image is gradually downsized using convolution layers and then upsampled using transposed convolution layers to restore it to its original size. Between the downsampling and upsampling layers, a residual network (ResNet) [24] is inserted. The discriminator extracts features from the input image by downsampling it with convolution layers. The feature map is then averaged using a global average pooling layer and input to the dense layer, and finally a single value is output as the decision result. Dropout layers [25] are incorporated into both the generator and discriminator to prevent overfitting. Note that while this results in stochastic behavior during training, that is the only time; when all the results reported in this study were obtained, the dropouts were disabled and all neurons were activated, resulting in deterministic behavior.

The construction, training, image generation, and evaluation of the model were performed using Python 3 codes with TensorFlow 2.7 [23] library. The model's computations were performed on an NVIDIA GeForce RTX 4090 GPU with 32-bit floating-point precision. The codes used for computations and analyses are available from a public GitHub repository at https://github.com/yymgch/cycle-chaos-gan.
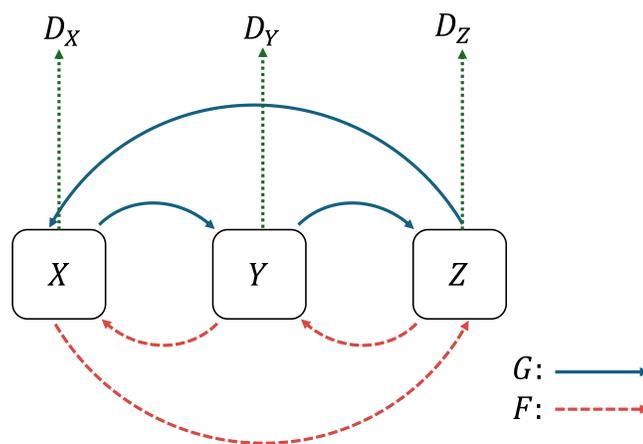
Fig 1. Transformations learned by model. The model consists of two generators, G and F, and three discriminators, D_X, D_Y, and D_Z.

Training datasets

In this study, we performed training using the MNIST [26] and Fashion-MNIST [27] datasets.

MNIST is a widely used dataset consisting of 7×10^4 grayscale images of handwritten digits from 0 to 9. Of these images, 6×10^4 are provided for training and 10^4 are provided for testing. Each image has a resolution of 28×28 pixels, resulting in a total of 784 pixels per image. For our experiments with MNIST, we used subsets of images belonging to specific categories: images of the digit 0 were assigned to dataset X, images of the digit 1 were assigned to dataset Y, and images of the digit 2 were assigned to dataset Z (Fig 1).

Fashion-MNIST [27] is another dataset that serves as a drop-in replacement for MNIST. It consists of 7×10^4 grayscale images from 10 categories of fashion products. The dataset maintains the same image size and split as MNIST, with 6×10^4 training images and 10^4 testing images, each having a resolution of 28×28 pixels. We used subsets of images belonging to specific fashion product categories: images of T-shirts/tops were assigned to dataset X, images of sneakers were assigned to dataset Y, and images of bags were assigned to dataset Z (Fig 1).

Loss function

The loss function for training our model is defined by extending the loss function of Cycle-GAN [4]. For the $X \rightarrow Y$ transformation by generator G, the loss function $\mathcal{L}_{\mathrm{GAN}}(G, D_Y, X, Y)$ defined in Eq 2 for CycleGAN was employed. Similarly, loss functions were prepared for the transformations $Y \rightarrow Z$ and $Z \rightarrow Y$, as well as for the reverse transformation $Y \rightarrow X \rightarrow Z \rightarrow Y$ using F. The overall adversarial loss $\mathcal{L}_{\mathrm{ADV}}$ was the sum of these:

$$
\begin{aligned}
\mathcal{L}_{\mathrm{ADV}}(G, F, D_X, D_Y, D_Z) &= \mathcal{L}_{\mathrm{GAN}}(G, D_Y, X, Y) + \mathcal{L}_{\mathrm{GAN}}(F, D_X, Y, X) \\
&+ \mathcal{L}_{\mathrm{GAN}}(G, D_Z, Y, Z) + \mathcal{L}_{\mathrm{GAN}}(F, D_Y, Z, Y) \\
&+ \mathcal{L}_{\mathrm{GAN}}(G, D_X, Z, X) + \mathcal{L}_{\mathrm{GAN}}(F, D_X, X, Z).
\end{aligned}
\tag{7}
$$

Furthermore, the cycle-consistency loss, which evaluates how close G and F are to being inverse mappings of each other, is formulated as

$$
\begin{aligned}
\mathcal{L}_{\mathrm{CYC}}(G, F) &= \mathbb{E}_{x \sim P_X}[\| F(G(x)) - x \|_1] + \mathbb{E}_{y \sim P_Y}[\| G(F(y)) - y \|_1] \\
&+ \mathbb{E}_{y \sim P_Y}[\| F(G(y)) - y \|_1] + \mathbb{E}_{z \sim P_Z}[\| G(F(z)) - z \|_1] \\
&+ \mathbb{E}_{z \sim P_Z}[\| F(G(z)) - z \|_1] + \mathbb{E}_{x \sim P_X}[\| G(F(x)) - x \|_1].
\end{aligned}
\tag{8}
$$

The total objective function is a combination of the adversarial loss and the cycle-consistency loss:

$$
\mathcal{L}(G, F, D_X, D_Y, D_Z) = \mathcal{L}_{\mathrm{ADV}}(G, F, D_X, D_Y, D_Z) + \lambda \mathcal{L}_{\mathrm{CYC}}(G, F),
\tag{9}
$$

where λ is a coefficient that controls the relative weight of the two losses, and it was set to 10 in our experiments. It is important to note that although the loss function does not explicitly include a term to promote the generation of diverse images, a GAN with the adversarial loss learns to match the distribution of the mapped data (e.g., the distribution of $G(x)$, $x \sim P_X$) with the distribution of the corresponding dataset (e.g., P_Y) [1]. Because of this characteristic, this model is expected to produce a wide variety of images that resemble the distribution of the target dataset.

UMAP

We used uniform manifold approximation and projection (UMAP) [28] to perform dimensionality reduction on the image data and visually evaluate the extent of the generated

data's coverage of the training data distribution. UMAP is a technique that can map high-dimensional data into a low-dimensional space while preserving the original data's local structure. For the UMAP parameters, we used the default values recommended by a previous study [28], i.e., the number of nearest neighbors was set to 15, the minimum distance to 0.1, and the distance metric to Euclidean distance.

Precision and recall

To quantitatively evaluate the quality and diversity of the generated images, we used the precision and recall metrics proposed by Kynkäänniemi et al. [19] for evaluating generated data. Precision assesses the extent to which the generated images resemble the actual dataset images, while recall measures the extent to which the generated images cover a wide range of features of the real data. These metrics allow us to determine how well the generated image sequences capture the features of the real data and exhibit diversity. In this method, for both the real and generated datasets, we construct explicit and non-parametric representations of the manifolds in which the data lie, independent of the model or parameters. These manifolds are then used to estimate the precision and recall of the generated image set.

Let X_r be the set of real data samples and X_g be the set of generated data samples. The samples in each set are embedded into a high-dimensional feature space using the pre-trained VGG16 model [29]. Pre-trained on the ILSVRC2012 dataset [30], this model allows us to extract high-level features from the images. We used the feature map obtained from the layer before the final layer of the VGG16 model as the feature vector.

Let ϕ_r and ϕ_g represent the feature vectors extracted from a real image and a generated image, respectively, and let Φ_r and Φ_g represent the sets of feature vectors corresponding to X_r and X_g. We took an equal number of samples from each set, i.e., $|\Phi_r| = |\Phi_g|$.

For each set of feature vectors $\Phi \in \{\Phi_r, \Phi_g\}$, we define the corresponding manifold in the feature space. Specifically, we perform the following steps. First, for each feature vector in the set, we consider a hypersphere with a radius equal to the distance to its k-th nearest neighbor. Next, we define the manifold of the dataset Φ as the union of these hyperspheres.

To determine whether a given sample ϕ lies within this manifold, we define the following binary function:

$$f(\phi, \Phi) = \begin{cases} 1 & \text{if } \| \phi - \phi' \|_2 \leq \| \phi' - \text{NN}_k(\phi', \Phi) \|_2 \text{ for at least one } \phi' \in \Phi, \\ 0 & \text{otherwise.} \end{cases} \tag{10}$$

Here, $\text{NN}_k(\phi', \Phi)$ is a function that returns the k-th nearest-neighbor feature vector to ϕ' from the set Φ. Intuitively, $f(\phi, \Phi_r)$ determines whether a given image ϕ looks real, and $f(\phi, \Phi_g)$ determines whether a given image can be reproduced by the generator.

Precision measures how many generated images lie within the manifold of real images and is defined as

$$\text{precision}(\Phi_r, \Phi_g) = \frac{1}{|\Phi_g|} \sum_{\phi_g \in \Phi_g} f(\phi_g, \Phi_r). \tag{11}$$

On the other hand, recall measures how many real images lie within the manifold of generated images and is defined as

$$\text{recall}(\Phi_r, \Phi_g) = \frac{1}{|\Phi_r|} \sum_{\phi_r \in \Phi_r} f(\phi_r, \Phi_g). \tag{12}$$
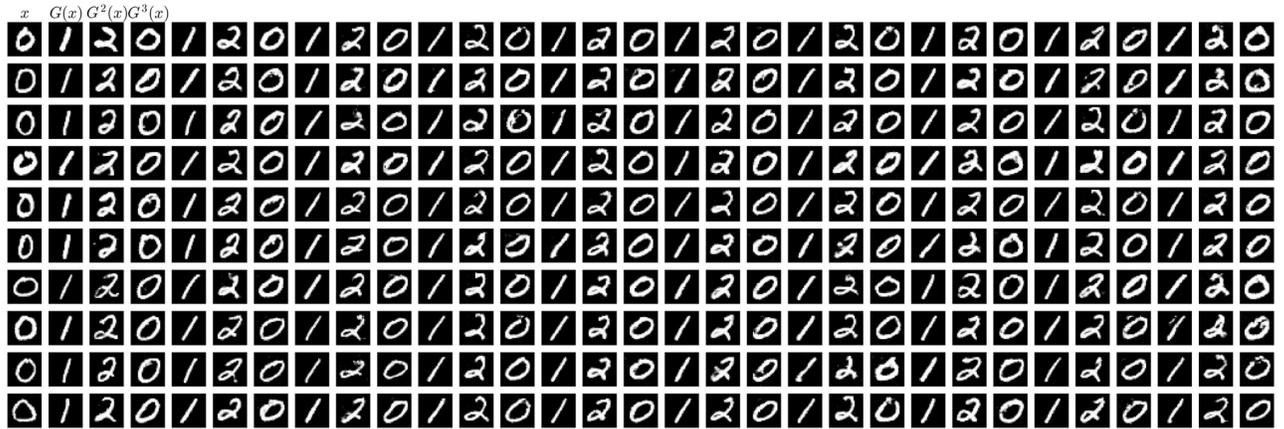
Fig 2. Example of generated image sequences for MNIST. The leftmost image in each row is the initial image, and the subsequent images are generated by iteratively applying the generator *G* to the previous image.

By using these metrics, we can quantitatively evaluate the quality and diversity of the generated image set.

Results

Generation of image sequences

To construct a dynamical system that iteratively generates various images in a cyclic manner, we built a deep generative model that sequentially transforms images from one of three categories into the next category, as described in the section entitled "Model and methods." We trained the model using the loss function defined in Eq 9, and the generators *G* and *F* were trained for 1000 epochs on the MNIST or Fashion-MNIST dataset. In the following sections, we analyze the model from the perspective of dynamical systems and evaluate the image sequences generated using the trained model.

Fig 2 shows the results of the iterative transformation from test image samples of the hand-written digit 0 as the initial values x_0, using the generator *G* by applying $x_{n+1} = G(x_n)$. Similarly, Fig 3 shows the results of the transformation of Fashion-MNIST images, starting from T-shirt
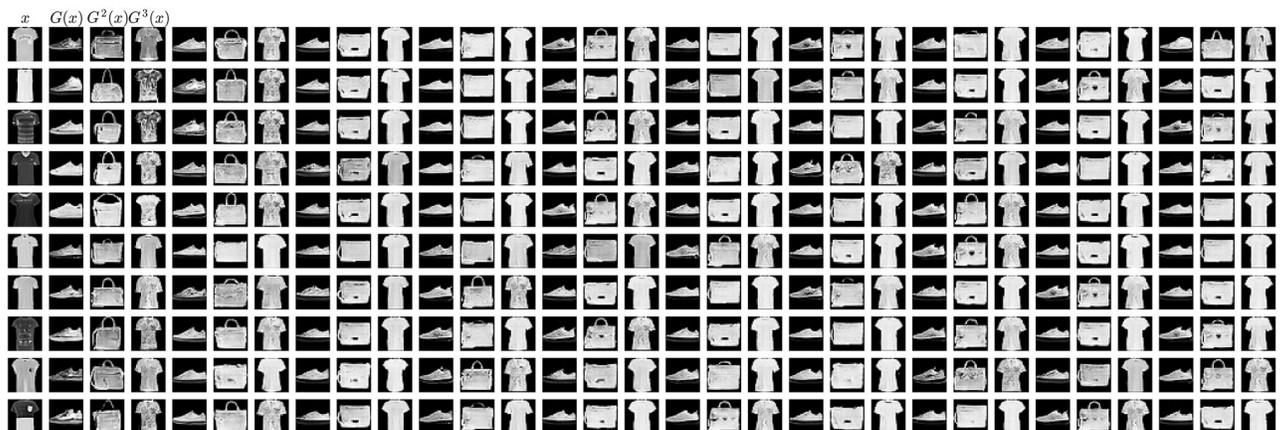
Fig 3. Example of generated image sequences for Fashion-MNIST. The leftmost image in each row is the initial image, and the subsequent images are generated by iteratively applying the generator *G* to the previous image.

images as the initial values. In both figures, multiple rows of images are shown. In each row, the leftmost image represents the initial value, and the subsequent images are generated by iteratively applying the generator to the previous image (left column) and displaying the result in the next column to the right. Each row shows a separate sequence of generated images starting from a different initial image. Both examples show that the images are transformed appropriately into the next digit or fashion product category.

Visualization of generated data distribution using UMAP

To visually assess the distribution of the image sequences generated by the iteration of G, we visualize the distribution using UMAP [28]. We iterated the transformation 5000 times from an initial image and visualized the distribution of the 5000 generated data points on a 2D plane (Fig 4 left and right). To embed, we first trained the UMAP model to embed the training data distribution into a 2D latent space, then we embedded the generated data into the same latent space using the learned model. In the figures, the training data distributions of datasets X, Y, and Z (digits 0, 1, and 2 for MNIST and T-shirt/top, sneaker, and bag for Fashion-MNIST) are represented by cyan, pink, and green points, respectively, while the generated data are represented by purple points. Note that the quantitative analysis in this paper was not performed based on the data in the UMAP-transformed space. The analysis of Lyapunov exponents was performed in the original image space, and the precision/recall metrics were calculated using the feature space obtained by the VGG16 model. We use the UMAP-transformed space only for visualization.

There are three clusters in the distributions of the training data, corresponding to the X, Y, and Z datasets. The generated data points lie within the regions where the training data exist, which suggests that the generated images closely resemble the original images, indicating high quality. However, the generated data distribution does not fully cover the entire distribution of the training data, suggesting that only a portion of the original data was reproduced in the generation sequence. These conjectures about quality and diversity are analyzed quantitatively in a later subsection. S3 Fig in the supporting information shows 64 different image sequences from different initial values. It appears that trajectories from different initial values converge to
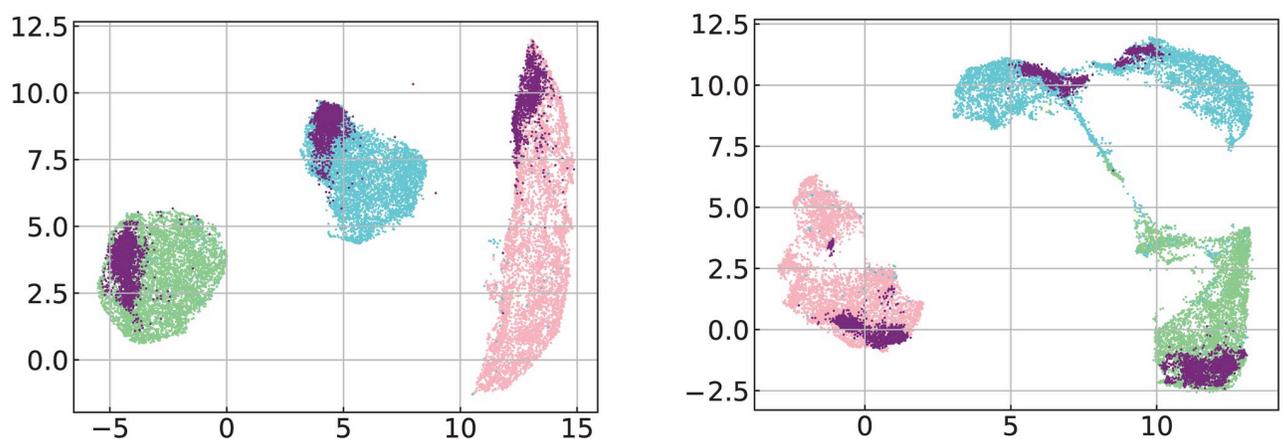
Fig 4. Visualization of distribution of training data and generated data by UMAP. Left: results for MNIST dataset. The green, pink, and cyan points represent 0, 1, and 2 image data in the training dataset, respectively, and the purple points represent the generated images. Right: results for Fashion-MNIST dataset. The green, pink, and cyan points represent T-shirt/top, sneaker, and bag image data in the training dataset, respectively, and the purple points represent the generated images.

very similar distributions. From the perspective of dynamical systems, this suggests that these trajectories converge to the same attractors.

To evaluate whether the UMAP transformation effectively captures the distribution of data and the characteristics of the trajectories in the original space, we performed two analyses. First, we applied k-means clustering to the UMAP-transformed training data to confirm that UMAP projected the three categories into two-dimensional space while preserving their distinctions. The adjusted Rand index [31, 32], a measure of clustering accuracy, was approximately 0.976. This high score indicates that UMAP successfully projected the clusters of the original categories into two dimensions while maintaining their differences. Next, we employed the Trustworthiness and Continuity metrics [33] to verify that the trajectories in the UMAP space effectively capture the high-dimensional dynamics. These metrics quantify the extent to which points in close proximity in high dimensions remain close in low dimensions, and vice versa. The metrics range from 0 to 1, with values closer to 1 indicating better preservation of the data structure. For the trajectories of our dynamical system, the Trustworthiness and Continuity metrics were approximately 0.942 and 0.978, respectively. These high values demonstrate that the UMAP representation effectively preserves the characteristics of the trajectories from the high-dimensional space.

The results in Fig 4 suggest that trajectories starting from an initial condition converge to limited regions within the image space. From the perspective of dynamical systems, this can be interpreted as convergence to attractors. Furthermore, S3 Fig indicates that trajectories from different initial conditions also converge to the same attractors. To gain a more detailed understanding of the convergence process to these attractors, we observe bundles of trajectories starting from a large number of initial conditions. Specifically, we track the transformation of the set of states at each time step and visualize the process of convergence to the attractors. We analyze whether trajectories from different initial conditions are drawn to similar regions and how the region of images that the generator can produce shrinks over time.

We prepared a large number of initial values $\{x_0^{(1)}, \ldots, x_0^{(j)}, \ldots, x_0^{(980)}\}$ from test images of category X and performed 3000 transformations as $x_{n+1}^{(j)} = G(x_n^{(j)})$, $(j = 1, \ldots, 980)$ using the trained generator G. For each time step n, we embedded the set of states $\{x_n^{(1)}, \ldots, x_n^{(980)}\}$ on a 2D plane using UMAP as shown in Fig 5. Similar to Fig 4, the points representing the training data for each dataset are shown in light colors, while the points representing the generated data are shown in purple. The distribution at $n = 0$ represents the distribution of the test images, which almost entirely covers the spread of the training image distribution. In the first few transformations, the region occupied by the generated images shrinks and does not fully cover the training image distribution, although it covers a wider range compared with later time steps, indicating the generation of relatively diverse images. However, as the transformations are repeated, the covered region shrinks further, and the trajectories concentrate on a limited area. After a certain number of iterations, the shape of the region remains relatively stable, and the set of generated images still occupies a finite area even after a long time. This suggests that the attractors are neither fixed points nor periodic points but rather attractors with some spatial extent. To quantitatively support these observations, we perform an analysis using the precision and recall metrics in the next subsection.

Quantitative evaluation of quality and diversity

To quantitatively evaluate the quality and diversity of the generated image sequences, we calculated the precision and recall [19] (P/R) of the generated images from the MNIST-trained model using the method described in the subsection entitled "Precision and recall." Here, quality refers to how similar the generated images are to the real dataset images, whereas
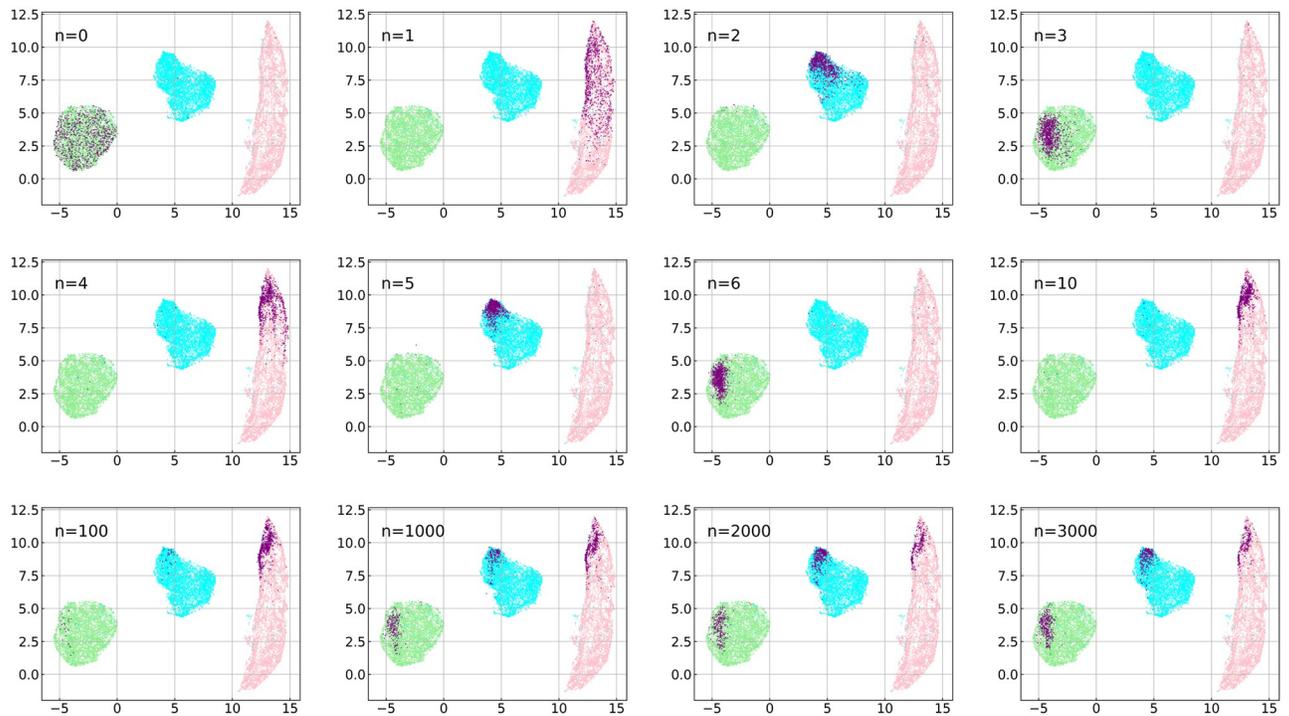
Fig 5. UMAP visualization of transitions of a set of states starting from a category X image for MNIST dataset. The step n indicates the number of transformations applied to the initial image. Green, pink, and cyan points represent image data for digits 0, 1, and 2 in the training dataset, respectively, and purple points represent the generated images. The area where the transitioned points are present shrinks over time relative to the area where the training data exist.

diversity refers to how well the generated images cover a wide range of variations in the real dataset. The important parameters in this method are the number of data samples and the number of nearest neighbors. In this method, an equal number of real and generated data samples should be prepared. In our case, the real data consisted of all the test dataset images from categories X, Y, and Z, amounting to 3178 samples. For the generated images, transformations were repeatedly applied starting from a test image x_0 selected from category X as an initial value, resulting in a trajectory of 3178 points $\{x_1, \ldots, x_{3178}\}$, and these samples included in this trajectory were used for evaluation. Previous research [19] proposed using 2×10^4 samples each for real and generated data as a standard procedure. However, in this case, the number of test images does not reach this number, so direct comparison with other literature results is impossible. To ensure the validity of the evaluation, 3178 samples from the same categories were extracted from the training dataset, and the P/R between the training and test datasets was calculated as a reference value. In the literature [19], the number of nearest neighbors k is typically set to 3. However, because the number of data samples used in this study was different from that in the literature, it is difficult to draw conclusions from results with only a single value of k. Therefore, we varied k from 1 to 10 and observed the resulting changes in P/R values.

The P/R calculation results for each k for the set of images included in the generated trajectory are shown in Fig 6. The P/R values between the test and generated data were calculated for 100 different trajectories, and their mean values and standard deviations are shown in the figure. These two values monotonically increase with k because of the nature of the algorithm. The P/R values between the training and test data both rise to around 0.9 at around $k = 6$,
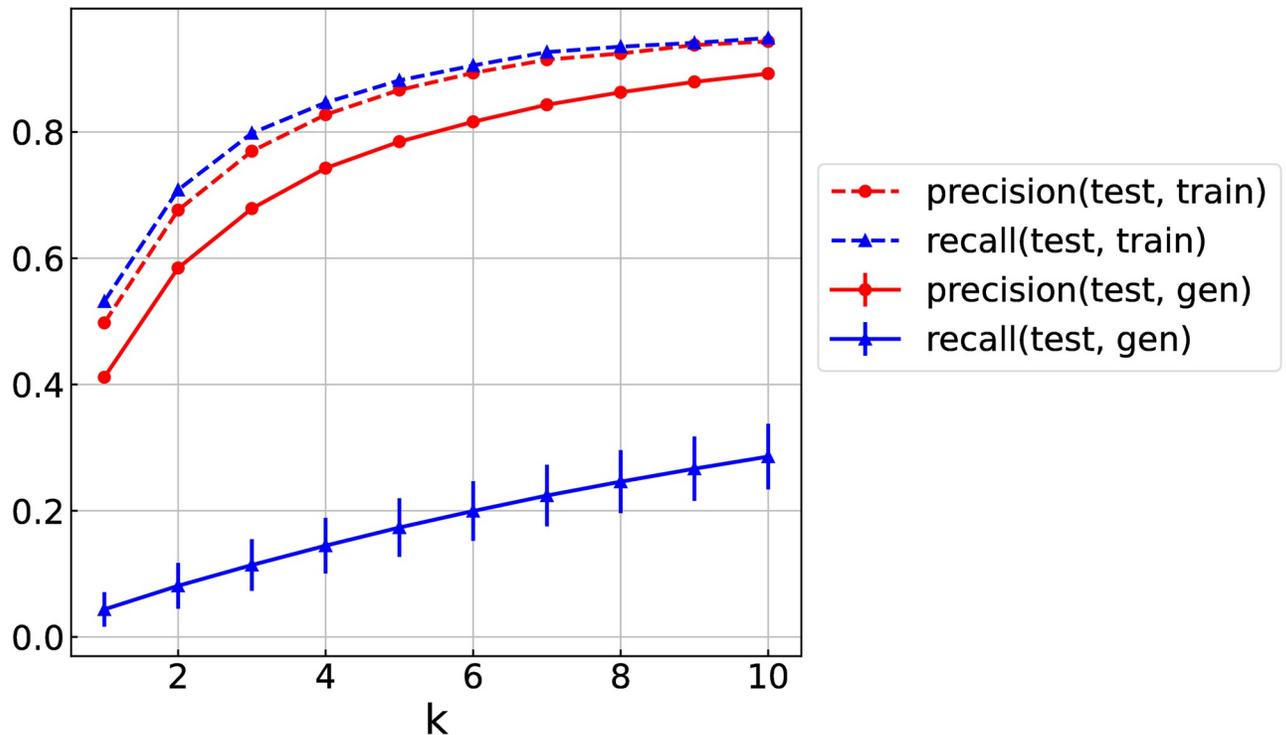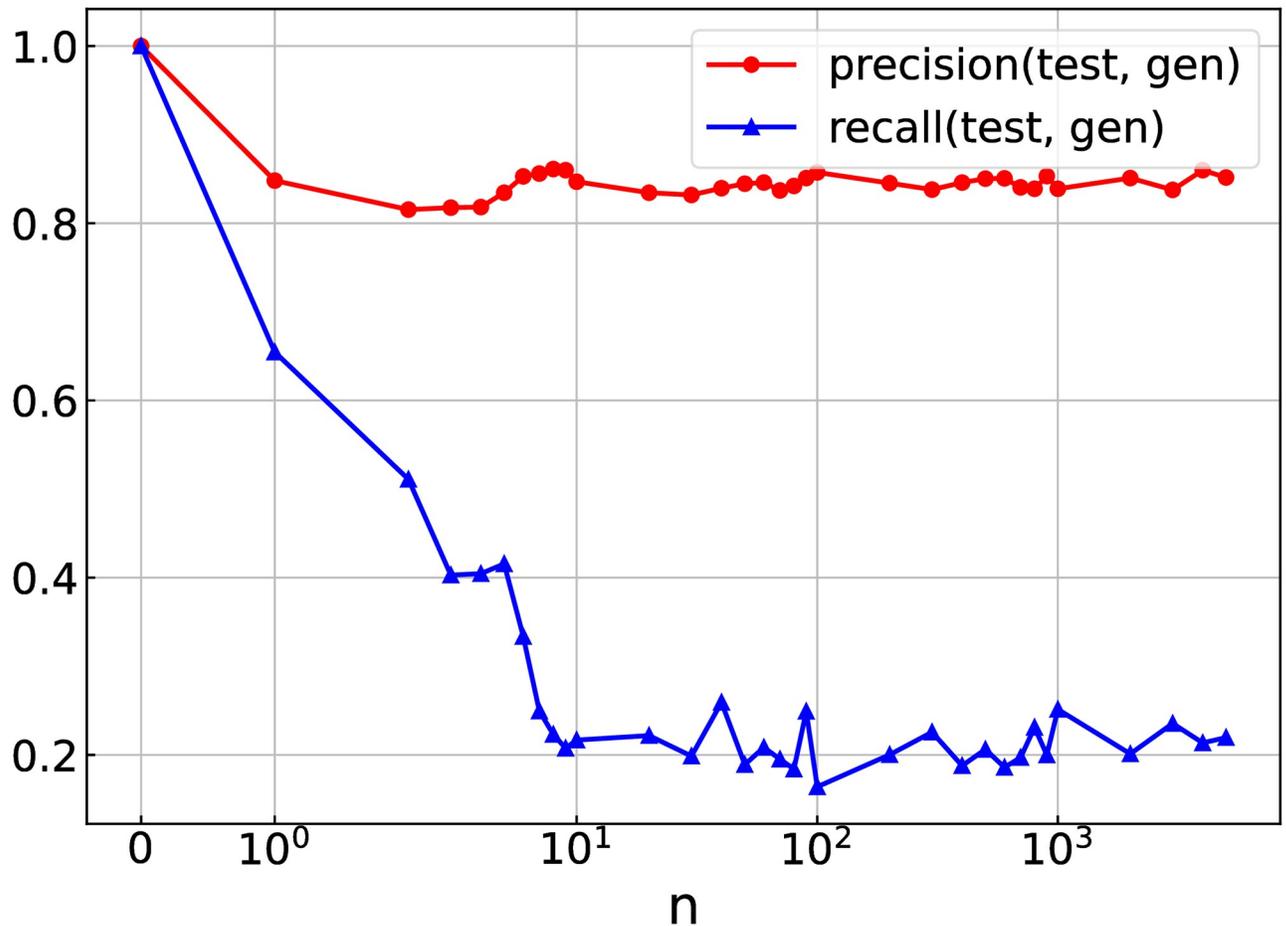
Fig 6. Precision and recall of the generated image sequences. The mean and standard deviation of the precision and recall values between the test and generated data are shown by the solid red and blue lines, respectively, as functions of the number of nearest neighbors k. The P/R values between the training and test data are shown by the dashed lines as references.

indicating high similarity between the training and test datasets. In contrast, while the precision between the generated and test data was slightly lower than that between the training and test data, the recall is considerably lower than that between the training and test data regardless of k. These results indicate quantitatively that while many of the generated images are of high quality and sufficiently close to the real images, the diversity of the generated images is lower than that of the real image dataset, and there are portions of the real images that are not reproduced in the generated image sequences.

Next, we quantitatively evaluate how the images converge to a limited region as transformations are repeatedly applied, as shown in Fig 5. Using the set of images obtained by mapping the entire set of test images for n steps, we calculated the P/R for each step (Fig 7). Referring to the results in Fig 6, the number of nearest neighbors was set to $k = 7$, where the P/R between the training and test data is sufficiently high. The generated dataset at $n = 0$ is the test dataset itself, and thus both the precision and recall are 1. For $n \geq 1$, the precision slightly decreases but remains high at above 0.8 for all steps. On the other hand, the recall decreases with n to less than 0.3 at $n = 10$, and then fluctuates around that value. These results indicate that the decrease in diversity occurs gradually in the early stages of the iterations and then maintains a certain level of diversity for an extended period.

To investigate the dependence of the number of generated sequences on the evaluation of P/R values, we performed an additional analysis using a large number of artificially generated initial images and their trajectories (S4 Fig). While the P/R values changed quantitatively due to the increased number of data points, the overall trends remained consistent. Specifically, we

Fig 7. Precision and recall as functions of time step. Precision and recall of set of generated images as functions of the number of transformations applied to the initial image. The number of nearest neighbors was set to $k = 7$.

observed persistent high precision/low recall patterns and an initial decrease in recall over time.

Chaotic dynamics

Lyapunov spectrum.   As observed above, the trajectories of the generated images do not converge to fixed points or periodic points but instead generate various images. This diverse image generation is not possible with simple dynamics such as fixed-point or periodic attractors, suggesting the presence of chaotic dynamics. To determine whether chaotic dynamics are present, the Lyapunov exponent, which quantifies the chaotic characteristics of the trajectory of the dynamical system produced by the generator G, was estimated numerically. In the numerical calculations, we used the method described in the subsection entitled "Chaotic dynamics and Lyapunov exponents," which involves calculating the Jacobian matrix and utilizing Gram–Schmidt orthonormalization to estimate all the Lyapunov exponents (called the Lyapunov spectrum). The spectrum was estimated by calculating the exponents from 980 trajectories of length 2000 and taking the sample mean to obtain the full set of Lyapunov exponents. To remove the transient period, we first applied the mapping 2000 times to the initial

images, and then used the subsequent 2000-step time series $\{x_{2000}, \ldots, x_{3999}\}$ to estimate the Lyapunov exponents.

S5 Fig shows the histograms of the first five Lyapunov exponents calculated for each of the 980 trajectories. The histograms exhibit unimodal Gaussian-like distributions, suggesting that these trajectories converge to the same attractor without multi-stability. Therefore, it is reasonable to average these values to estimate the Lyapunov exponents.

Fig 8 shows all the spectrum values, with the inset showing an enlarged view of the first 15 Lyapunov exponents. The first seven Lyapunov exponents are definitely positive, and the largest Lyapunov exponent is estimated to be about 0.340. The presence of these positive Lyapunov exponents indicates that the generated trajectories exhibit sensitive dependence on initial conditions and are chaotic, suggesting that the chaotic attractor generates various images. Most Lyapunov exponents are negative except for the seven largest. The phenomenon of most exponents becoming negative is commonly observed in large-scale, high-dimensional dynamical systems where the dimension of the attractor is considerably smaller than the dimension of the system's phase space [13, 34, 35].

Further, S6 Fig demonstrates the convergence of estimated Lyapunov exponents over time, obtained by calculating trajectories from 10 initial values for an extended period (2×10^6 steps). All ten trajectories converge to almost the same value, which matches the average value shown in Fig 8. This result provides an additional evidence for the stability of our Lyapunov exponent calculations in this model.
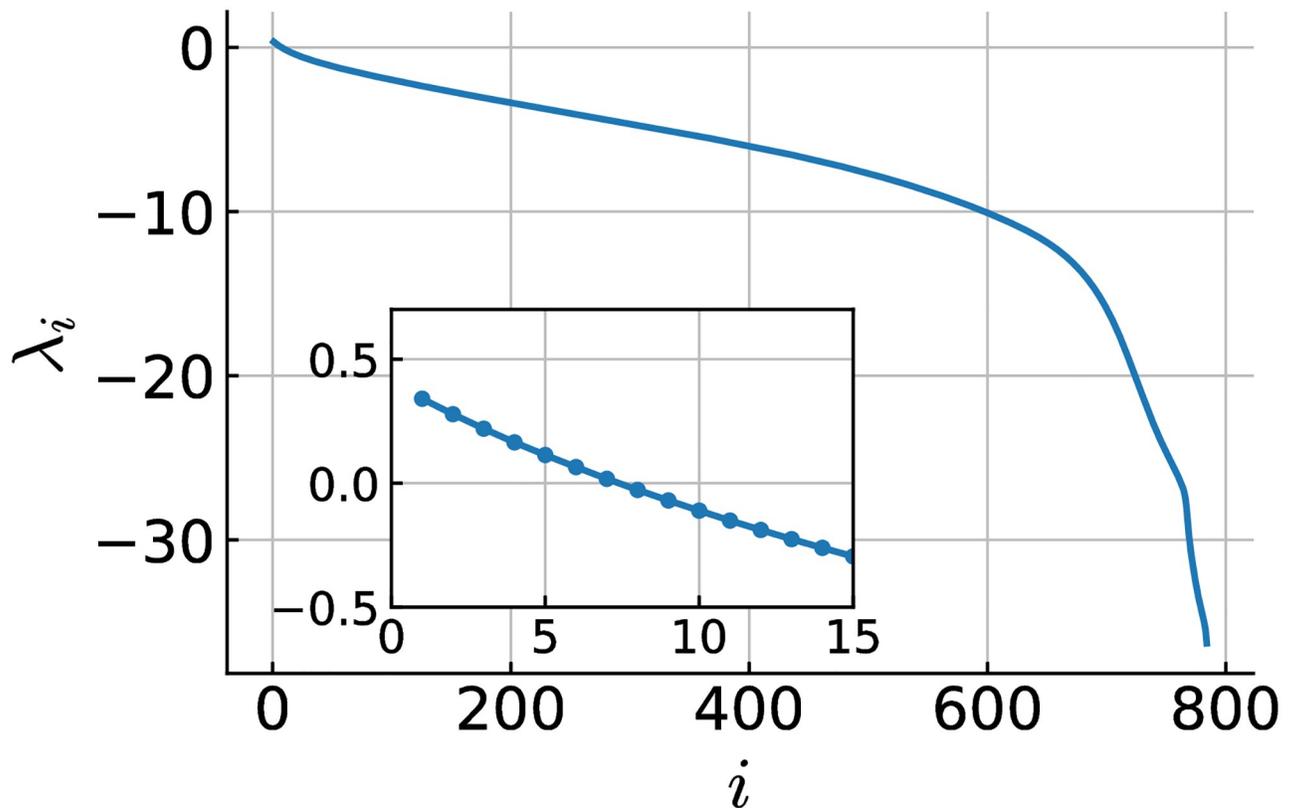
Fig 8. Lyapunov exponents of dynamics defined by generator G. Shown here is the full Lyapunov spectrum, with the inset showing an enlarged view of the first 15 Lyapunov exponents.

https://doi.org/10.1371/journal.pcsy.0000027.g008

Direct observation of trajectory instability. It is well known that numerical computations of certain chaotic dynamical systems can be unstable [13]. Because our deep model requires complex computations with many parameters and these computations are performed using a GPU with finite precision, it is desirable to check the robustness of the numerical results for the Lyapunov exponents estimated in the previous subsection. To do so, we estimate the Lyapunov exponents using a different approach and then check the consistency of the results. For this purpose, we directly observe how trajectories starting from a point within the attractor and trajectories starting from its neighborhood diverge, and we numerically estimate the largest Lyapunov exponent.

Let $\{x_0^{(1)}, \ldots, x_0^{(980)}\}$ be the set of 980 test images of the digit 0 as initial values. To remove the transient period before the dynamics settle into the attractor, we map each point for $T = 2000$ steps using G and denote the resulting set of points as $X_T = \{x_T^{(1)}, \ldots, x_T^{(980)}\}$. We consider the trajectories starting from these points as reference trajectories and observe the difference between these trajectories and those starting from perturbed points. For each point $x_T^{(j)}$, the perturbation is applied by selecting the nearest point $x_T^{(k)}$ from the set X_T (excluding itself) and setting $y_T^{(j)} = x_T^{(j)} + \varepsilon(x_T^{(k)} - x_T^{(j)})/||x_T^{(k)} - x_T^{(j)}||$, where $\varepsilon = 10^{-5}$ is the strength of the perturbation. The intention with this approach is to apply the perturbation in the direction along which the attractor is expanding locally.

We then calculate the difference between these two trajectories as they are transformed by G:

$$d_n^{(j)} = ||G^n(y_T^{(j)}) - G^n(x_T^{(j)})||, \tag{13}$$

and we compute the sample mean of the logarithm of these values for each step, i.e.,

$$\bar{d}_n = \frac{1}{980} \sum_{j=1}^{980} \log d_n^{(j)}. \tag{14}$$

Fig 9 shows the expansion of the differences between the perturbed and reference trajectories and their average. The slope of the green line represents the largest Lyapunov exponent estimated in the previous subsection. As estimated by linear regression, the actual expansion rate of the errors is 0.352, which is in good agreement with this largest Lyapunov exponent. This consistency between the two approaches indicates that estimating the Lyapunov exponents via the Jacobian matrix provides reliable results.

Lyapunov dimension. In the machine learning community, real-world data such as images are assumed to be distributed on a relatively low-dimensional manifold within the high-dimensional space in which they are embedded. The dimension of this manifold is called the intrinsic dimension [36–38]. When generating images using trajectories of a dynamical system, if the attractor's dimension matches the intrinsic dimension of the training data, then this is considered advantageous for generating a set of images with a diversity similar to that of the original set. We characterize the diversity of the generated images by estimating the dimension of the attractor, which can be calculated using the Lyapunov dimension [20].

When the Lyapunov exponents λ_i ($i = 1, \ldots, N$) are arranged in descending order, and j is an integer satisfying $\sum_{i=1}^{j} \lambda_i \geq 0$ and $\sum_{i=1}^{j+1} \lambda_i < 0$, the Lyapunov dimension D_L is defined as

$$D_L = j + \frac{\sum_{i=1}^{j} \lambda_i}{|\lambda_{j+1}|}. \tag{15}$$
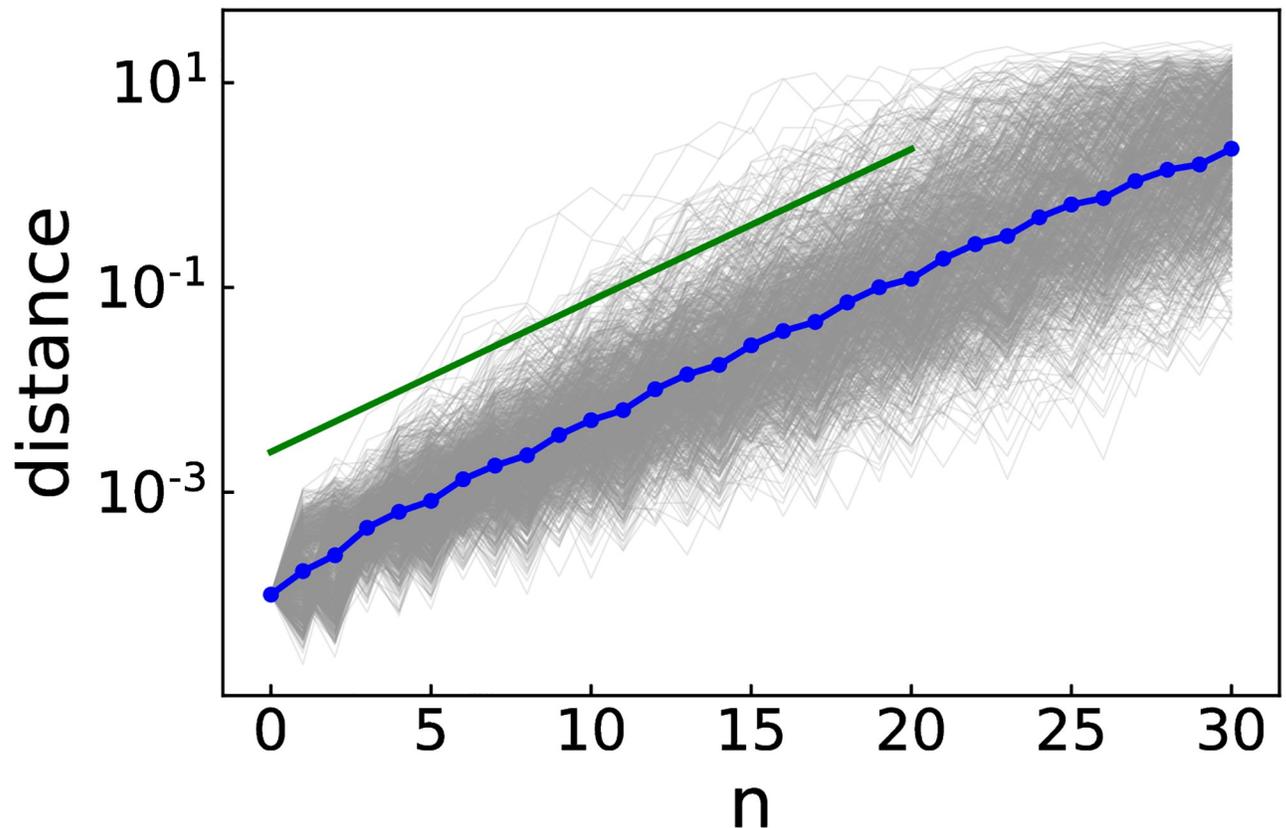
Fig 9. Direct observation of trajectory instability. Shown here is the expansion of the differences between the perturbed and reference trajectories and their average. The slope of the green line represents the largest Lyapunov exponent estimated in the previous subsection. The gray lines represent the individual development of the difference between trajectories. The blue line represents the average of the differences.

Based on the results in Fig 8, the Lyapunov dimension is estimated to be ca. 14.5. According to the literature, the intrinsic dimension of the MNIST dataset is between 10 and 20. Although the specific value depends on the method used to calculate it [37, 38], this range is qualitatively consistent with the result for the Lyapunov dimension. These results suggest that images are generated on an attractor by a chaotic dynamical system, which is thought to contribute to the diversity of the images.

Discussion

In this study, we extended CycleGAN to construct a model that generates images of multiple categories by cyclically transforming images among three different categories. Using the constructed model, we repeatedly generated images and confirmed that they were transformed into images of the following categories. By visualizing the distribution of the generated images using a dimensionality reduction technique, we verified that the images of the generated data were distributed into three clusters corresponding to the same categories as the training dataset.

The process of successive image transformation can be considered as being a dynamical system. A single trajectory of the dynamical system induced by our model can generate a diverse range of images using chaotic dynamics. Attractors with trajectories that transitioned cyclically among the three different categories emerged, producing various images without falling into

fixed points or periodic solutions. This characteristic is considered effective as a method for generating diverse data.

The quality of the generated images was evaluated using the P/R metric, and the precision showed high performance. The high precision suggests that the model can accurately capture and reproduce image features. However, the recall values were relatively low, indicating that the generated images only partially cover the wide distribution of the actual dataset. Evaluating these results using precision and recall allowed for a quantitative assessment of the outcomes and provided a benchmark for future improvements.

We conducted a visual investigation of the generated images to address whether the samples within the attractor possess any specific features that render them distinct from the samples outside the attractor. However, it was difficult to characterize the images within the attractor as either typical or atypical samples of their respective categories.

The estimation of the Lyapunov spectrum suggested that the trajectories of the generated images exhibit sensitive dependence on initial conditions, a characteristic of chaotic systems. This sensitivity was further confirmed by direct observation of trajectories departing from neighboring points and diverging from each other. Furthermore, we estimated the dimension of the attractor by the Lyapunov dimension, which is considered as the dimension of the data manifold on which the generated images lie. The estimated dimension of the attractor was close to the intrinsic dimension of the training dataset. This result suggests that the images generated by the model were spread on attractors with a high-dimensional complexity similar to that of the training dataset, and that chaotic dynamics contribute to the diversity of the generated images.

This emergence of the large chaotic attractor may be related to our dynamical system's design, which learned a cyclic path using only the generator G, without involving the other generator F. For comparison, consider a dynamical system constructed using the original CycleGAN [4], defined as $x_{n+1} = F(G(x_n))$. In this case, the composition $F \circ G$ is trained to approximate the identity mapping. Consequently, the trajectory is expected to converge to a small region, such as a fixed point, where no significant changes in the image are expected. In contrast, we constructed a system that circulates among three categories using only G, without being pulled back by F. We hypothesize that one of the primary reasons for the emergence of the large chaotic attractor is that this structure is not constrained by the requirement to approach the identity map. Further comparison of our model's dynamics with those of Cycle-GAN, and a deeper understanding of the mechanisms underlying the emergence of the chaotic attractor, remain important topics for future research.

Our model can be viewed as an extension of the classical associative memory model that memorizes sequences of patterns using the Hebbian learning rule [10, 39]. Classical associative memory can memorize periodic solutions that cycle through multiple memorized points in the state space of the dynamical system, while the present model cycles among categories instead of points. In other words, we have demonstrated that it is possible to construct a model that achieves hetero-associations among categories. Such a model may offer an interesting tool for the interdisciplinary field between machine learning and neuroscience, and future research may investigate how deep learning models perform transformations among categories and whether properties similar to the classical Hebb's association rule can be found in their model.

To investigate the scope of our method with more challenging real-world examples, we believe it is necessary to extend the approach to handle transformations in latent space, similar to techniques used in StyleGAN [40] or latent diffusion models [41]. To link our method to associative memory processes and achieve more flexible image transformations, several advancements are needed. Potential avenues for future research include: a) Extending the method to allow variation of specific features within a category (e.g., the thickness or

inclination of a character) through external conditional inputs. b) Developing the capability to store and navigate multiple category cycles. c) Adding functionality to dynamically change the association target based on conditional inputs.

As evident from the visualized distribution and quantitative results of P/R evaluation, the generated data did not fully cover the entire distribution of the training data. Understanding the balance between the quality and diversity of the generated images and improving diversity while maintaining quality are challenges for future research. To address this, adjusting the parameters of the dropout layer and improving the network structure of the generator and discriminator are considered to be effective approaches. Furthermore, when calculating the loss function, the discriminator's evaluation of the generated images is currently performed based on only the results of a single mapping from the test images. Considering that the recall value decreased with each successive mapping from the test images, it is expected that applying the discriminator's evaluation to the results of multiple transformations of the test images and incorporating this into the loss function could improve the recall value.

Supporting information

S1 Fig. Model architecture of generator. Each box represents a layer of the generator and shows the name and type of the layer and the input and output sizes.
(EPS)

S2 Fig. Model architecture of discriminator. Each box represents a layer of the discriminator and shows the name and type of the layer and the input and output sizes.
(EPS)

S3 Fig. Trajectories starting from different initial points.
(TIF)

S4 Fig. Precision and recall as functions of time step, calculated from trajectories from perturbed initial images. To investigate the dependence of the number of generated sequences on the evaluation of P/R values, we conducted an analysis using a large number of artificially generated initial images and their trajectories. We perturbed the initial images using the same method as in our maximum Lyapunov exponent analysis (Fig 9) and generated 15735 trajectories (3147×5). The strength of the perturbation ε was set to 0.1. We then evaluated the precision/recall metrics using these trajectories and perturbed test images following the same procedure as in Fig 7.
(TIF)

S5 Fig. Histograms of first five Lyapunov exponents calculated for each of 980 trajectories.
(TIF)

S6 Fig. Convergence of estimated Lyapunov exponents over time.
(TIF)

Acknowledgments

The authors would like to thank Ichiro Tsuda and Shigetoshi Nara for valuable discussions and helpful comments.

Author Contributions

Conceptualization: Yutaka Yamaguti.

Formal analysis: Takaya Tanaka, Yutaka Yamaguti.

Funding acquisition: Yutaka Yamaguti.

Investigation: Takaya Tanaka, Yutaka Yamaguti.

Methodology: Takaya Tanaka, Yutaka Yamaguti.

Project administration: Yutaka Yamaguti.

Software: Takaya Tanaka, Yutaka Yamaguti.

Supervision: Yutaka Yamaguti.

Visualization: Takaya Tanaka, Yutaka Yamaguti.

Writing – original draft: Takaya Tanaka, Yutaka Yamaguti.

Writing – review & editing: Yutaka Yamaguti.

References

1. Goodfellow I, Pouget-Abadie J, Mirza M, Xu B, Warde-Farley D, Ozair S, et al. Generative Adversarial Nets. In: Ghahramani Z, Welling M, Cortes C, Lawrence N, Weinberger KQ, editors. Advances in Neural Information Processing Systems. vol. 27. Curran Associates, Inc.; 2014.

2. Kingma DP, Welling M. Auto-encoding variational Bayes. arXiv:1312.6114 [Preprint]; 2013. Available from: https://arxiv.org/abs/1312.6114.

3. Ho J, Jain A, Abbeel P. Denoising diffusion probabilistic models. In: Larochelle H, Ranzato M, Hadsell R, Balcan MF, Lin H, editors. Advances in Neural Information Processing Systems. vol. 33. Curran Associates, Inc.; 2020. p. 6840–6851.

4. Zhu JY, Park T, Isola P, Efros AA. Unpaired image-to-image translation using cycle-consistent adversarial networks. In: Proceedings of the IEEE international conference on computer vision (ICCV); 2017. p. 2242–2251.

5. Anderson JA. A simple neural network generating an interactive memory. Math Biosci. 1972; 14(3-4):197–220. https://doi.org/10.1016/0025-5564(72)90075-2

6. Nakano K. Associatron—a model of associative memory. IEEE Trans Syst Man Cybern Syst. 1972;(3): 380–388. https://doi.org/10.1109/TSMC.1972.4309133

7. Kohonen T. Correlation matrix memories. IEEE Trans Comput. 1972; 100(4):353–359. https://doi.org/10.1109/TC.1972.5008975

8. Hopfield JJ. Neural networks and physical systems with emergent collective computational abilities. Proc Natl Acad Sci U S A. 1982; 79(8):2554–2558. https://doi.org/10.1073/pnas.79.8.2554 PMID: 6953413

9. Tsuda I, Koerner E, Shimizu H. Memory dynamics in asynchronous neural networks. Prog Theor Phys. 1987; 78(1):51–71. https://doi.org/10.1143/PTP.78.51

10. Nara S, Davis P. Chaotic wandering and search in a cycle-memory neural network. Prog Theor Phys. 1992; 88(5):845–855. https://doi.org/10.1143/ptp/88.5.845

11. Adachi M, Aihara K. Associative dynamics in a chaotic neural network. Neural Netw. 1997; 10(1): 83–98. https://doi.org/10.1016/S0893-6080(96)00061-5 PMID: 12662889

12. Tsuda I. Dynamic link of memory: chaotic memory map in nonequilibrium neural networks. Neural Netw. 1992; 5(2):313–326. https://doi.org/10.1016/S0893-6080(05)80029-2

13. Kaneko K, Tsuda I. Complex systems: chaos and beyond: a constructive approach with applications in life sciences. New York: Springer Verlag; 2001.

14. Tsuda I. Chaotic itinerancy and its roles in cognitive neurodynamics. Curr Opin Neurobiol. 2015; 31: 67–71. https://doi.org/10.1016/j.conb.2014.08.011 PMID: 25217808

15. Gilpin W. Chaos as an interpretable benchmark for forecasting and data-driven modelling. arXiv:2110.05266 [Preprint]; 2021. Available from: https://arxiv.org/abs/2110.05266.

16. Tanaka Y, Yamaguti Y. Evaluating generation of chaotic time series by convolutional generative adversarial networks. JSIAM Lett. 2023; 15:117–120. https://doi.org/10.14495/jsiaml.15.117

17. Sussillo D, Abbott LF. Generating coherent patterns of activity from chaotic neural networks. Neuron. 2009; 63(4):544–557. https://doi.org/10.1016/j.neuron.2009.07.018 PMID: 19709635

18. Laje R, Buonomano DV. Robust timing and motor patterns by taming chaos in recurrent neural networks. Nat Neurosci. 2013; 16(7):925–933. https://doi.org/10.1038/nn.3405 PMID: 23708144

19. Kynkäänniemi T, Karras T, Laine S, Lehtinen J, Aila T. Improved precision and recall metric for assessing generative models. Advances in Neural Information Processing Systems (NeurIPS). 2019; 32.

20. Alligood KT, Sauer TD, Yorke JA. Chaos: An Introduction to Dynamical Systems. Springer New York; 2000.

21. Eckmann JP, Ruelle D. Ergodic theory of chaos and strange attractors. Rev Mod Phys. 1985; 57(3): 617. https://doi.org/10.1103/RevModPhys.57.617

22. Shimada I, Nagashima T. A numerical approach to ergodic problem of dissipative dynamical systems. Prog Theor Phys. 1979; 61(6):1605–1616. https://doi.org/10.1143/PTP.61.1605

23. Abadi M, Agarwal A, Barham P, Brevdo E, Chen Z, Citro C, et al.. TensorFlow: large-scale machine learning on heterogeneous systems; 2015. Available from: https://www.tensorflow.org/.

24. He K, Zhang X, Ren S, Sun J. Identity Mappings in Deep Residual Networks. In: Leibe B, Matas J, Sebe N, Welling M, editors, Computer Vision—ECCV 2016. Cham: Springer International Publishing; 2016. p. 630–645.

25. Srivastava N, Hinton G, Krizhevsky A, Sutskever I, Salakhutdinov R. Dropout: a simple way to prevent neural networks from overfitting. J Mach Learn Res. 2014; 15(1):1929–1958.

26. LeCun Y, Bottou L, Bengio Y, Haffner P. Gradient-based learning applied to document recognition. Proc IEEE. 1998; 86(11):2278–2324. https://doi.org/10.1109/5.726791

27. Xiao H, Rasul K, Vollgraf R. Fashion-MNIST: a novel image dataset for benchmarking machine learning algorithms. arXiv:1708.07747 [Preprint]; 2017. Available from: https://arxiv.org/abs/1708.07747.

28. McInnes L, Healy J, Melville J. Umap: Uniform manifold approximation and projection for dimension reduction. arXiv:180203426 [Preprint]; 2018. Available from: https://arxiv.org/abs/180203426.

29. Simonyan K, Zisserman A. Very deep convolutional networks for large-scale image recognition. In: 3rd International Conference on Learning Representations (ICLR 2015). Computational and Biological Learning Society; 2015. arXiv:1409.1556.

30. Russakovsky O, Deng J, Su H, Krause J, Satheesh S, Ma S, et al. ImageNet large scale cisual recognition challenge. International Journal of Computer Vision (IJCV). 2015; 115(3):211–252. https://doi.org/10.1007/s11263-015-0816-y

31. Hubert L, Arabie P. Comparing partitions. Journal of classification. 1985; 2:193–218. https://doi.org/10.1007/BF01908075

32. Steinley D. Properties of the hubert-arable adjusted rand index. Psychological methods. 2004; 9(3): 386. https://doi.org/10.1037/1082-989X.9.3.386 PMID: 15355155

33. Venna J, Kaski S. Neighborhood Preservation in Nonlinear Projection Methods: An Experimental Study. In: Dorffner G, Bischof H, Hornik K, editors. Artificial Neural Networks — ICANN 2001. Berlin, Heidelberg: Springer Berlin Heidelberg; 2001. p. 485–491.

34. Engelken R, Wolf F, Abbott LF. Lyapunov spectra of chaotic recurrent neural networks. Physical Review Research. 2023; 5(4):043044. https://doi.org/10.1103/PhysRevResearch.5.043044

35. Kobayashi M, Nakai K, SAIKI Y. Lyapunov analysis of data-driven models of high dimensional dynamics using reservoir computing: Lorenz-96 system and fluid flow. Journal of Physics: Complexity. 2024;.

36. Camastra F, Staiano A. Intrinsic dimension estimation: advances and open problems. Inf Sci. 2016; 328:26–41. https://doi.org/10.1016/j.ins.2015.08.029

37. Pope P, Zhu C, Abdelkader A, Goldblum M, Goldstein T. The intrinsic dimension of images and its impact on learning. In: International Conference on Learning Representations (ICLR); 2021. Available from: https://openreview.net/forum?id=XJk19XzGq2J.

38. Facco E, d'Errico M, Rodriguez A, Laio A. Estimating the intrinsic dimension of datasets by a minimal neighborhood information. Sci Rep. 2017; 7:12140. https://doi.org/10.1038/s41598-017-11873-y PMID: 28939866

39. Amari SI. Learning patterns and pattern sequences by self-organizing nets of threshold elements. IEEE Trans Comput. 1972; C-21(11):1197–1206. https://doi.org/10.1109/T-C.1972.223477

40. Karras T, Laine S, Aila T. A style-based generator architecture for generative adversarial networks. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition; 2019. p. 4401–4410.

41. Rombach R, Blattmann A, Lorenz D, Esser P, Ommer B. High-resolution image synthesis with latent diffusion models. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition; 2022. p. 10684–10695.