electronics

MDPI

Article

MemoryGAN: GAN Generator as Heterogeneous Memory for Compositional Image Synthesis

Zongtao Wang [1], Jiajie Peng [2] and Zhiming Liu [1,*]

[1] School of Computer & Information Science, Southwest University, Chongqing 400715, China; wztdream@email.swu.edu.cn
[2] School of Computer Science, Northwestern Polytechnical University, Xi'an 710072, China; jiajiepeng@nwpu.edu.cn
* Correspondence: zhimingliu88@swu.edu.cn

Abstract: The Generative Adversarial Network (GAN) has recently experienced great progress in compositional image synthesis. Unfortunately, the models proposed in the literature usually require a set of pre-defined local generators and use a separate generator to model each part object. This makes the model inflexible and also limits its scalability. Inspired by humans' structured memory system, we propose MemoryGAN to eliminate these disadvantages. MemoryGAN uses a single generator as a shared memory to hold the heterogeneous information of the parts, and it uses a recurrent neural network to model the dependency between the parts and provide the query code for the memory. The shared memory structure and the query and feedback mechanism make MemoryGAN flexible and scalable. Our experiment shows that although MemoryGAN only uses a single generator for all the parts, it achieves comparable performance with the state-of-the-art, which uses multiple generators, in terms of synthesized image quality, compositional ability and disentanglement property. We believe that our result of using the generator of the GAN as a memory model will inspire future work of both bio-friendly models and memory-augmented models.

Keywords: Generative Adversarial Network; compositional image synthesis; memory; disentanglement

Citation: Wang, Z.; Peng, J.; Liu, Z. MemoryGAN: GAN Generator as Heterogeneous Memory for Compositional Image Synthesis. *Electronics* **2023**, *12*, 2927. https://doi.org/10.3390/electronics12132927

Academic Editor: Byung Cheol Song

Received: 24 April 2023
Revised: 26 June 2023
Accepted: 30 June 2023
Published: 3 July 2023

1. Introduction

In recent years, deep learning has made significant progress in image processing [1–3]. Particularly, generative models [4–7] have achieved remarkable success in image generation and manipulation. A more recent trend in image generation is compositional image synthesis. Compositional image synthesis aims to capture the inherent compositional nature of our world, where complex objects are formed by combining simpler components [8]. By incorporating this prior into the generation process, compositional image synthesis allows for the semantic control of its components, enabling the localized manipulation of the image, increasing the interpretability of the generation process, and improving the reusability and efficiency of the model. It has numerous applications, including content creation and design, image editing, virtual reality, and gaming. Despite these important characteristics and potential applications, compositional image synthesis has received relatively little attention from the research community until recently.

Most previous works have primarily focused on generating visually appealing images. One notable approach in this context is the style-based GAN known as StyleGAN [9]. While it is capable of producing high-fidelity images and achieving some extent of disentanglement between coarse and fine features, it lacks the ability to represent the semantic parts of the object and cannot effectively control the attributes of these parts accordingly. Additionally, the achieved disentanglement is limited, since editing one attribute often affects other attributes. These limitations arise from the usage of a single latent vector to govern the generation of the entire image. In this study, we propose the utilization of a set

of latent codes to control the generation process, where each latent code corresponds to a specific semantic part of the object.

Other researchers focus on mapping segmentation masks to realistic images. A notable work in this field is SEAN [10], which encodes the mask as a modulation vector to control the semantic meaning of each region and employs a per-region style to control the appearance of each semantic region. While this approach achieves semantic generation and control, it requires the segmentation mask as input, thus limiting its ability to generate images from scratch. Additionally, since the generation is conditioned on the segmentation mask, it can only manipulate the appearance of the region and necessitates manual effort to change the shape of the region. In contrast, our method can generate images from scratch and automatically control both the shape and appearance of each semantic part.

A recently proposed work called SemanticStyleGAN [11] achieves compositional synthesis. While it demonstrates high-quality image generation and composition, it requires a set of pre-defined generators and uses a separate generator for each component. This structure has several disadvantages. First, the parameter count increases linearly with the semantic part count, rendering it non-scalable. Second, the representation is inefficient because each part is individually modeled without parameter sharing. Third, it lacks flexibility as the generators are designed for specific tasks, making it challenging to transfer the learned representations to new tasks. Manual decisions regarding the choice of generator are necessary for new tasks, introducing subjective bias and potentially leading to suboptimal training. Thus, the question arises: How can we achieve compositional image synthesis without encountering these limitations?

We draw inspiration from the human memory system, which exhibits compositionality as a key feature of human intelligence. Evidence suggests that humans store, retrieve, and manipulate information through a structured memory system. Knowledge is accumulated in long-term memory, which is then retrieved and manipulated in working memory, which is also known as short-term memory [12]. This memory system enables human beings to demonstrate flexible out-of-distribution and systematic generalization abilities, highlighting a significant gap between state-of-the-art machine learning models and human intelligence [8,13]. In terms of compositional learning, this memory system offers efficiency and flexibility. This raises the question: Can we enhance compositional learning in machine learning by introducing a similar memory system? Our work demonstrates that this possibility is indeed achievable.

Interestingly, recent work has shown that the hidden features generated by the generators of GANs posses several important properties, such as disentanglement [14,15] and out-of-domain generalization [16,17]. These results indicate that the generator of GANs could potentially be a suitable memory model. It turns out GANs, especially StyleGANs [9,14,18] can be naturally viewed as a memory system. A random vector $\mathbf{z}$ is sampled from a Gaussian distribution, which is mapped to a latent code $\mathbf{w}$ by Multi-Layer Perceptrons (MLPs). The latent code is used to modulate the generator and produce the final image. We may simply treat $\mathbf{z}$ or $\mathbf{w}$ as the query code and the output of the generator as the corresponding retrieved result. Unfortunately, as a memory model, a basic function is to store heterogeneous information. It is well known that it is hard for GANs to generate heterogeneous objects, and it tends to suffer mode collapse. Most GAN models in the literature are designed to model homogeneous objects sampled from a single distribution, which makes them less useful as a memory model.

In this work, we propose MemoryGAN as a solution to overcome these limitations. MemoryGAN employs a memory system consisting of a long-term memory and a working memory. The long-term memory is a GAN generator, serving as a centralized memory to store information about all the parts. To address the mode collapse issue mentioned earlier, we introduce the working memory, which comprises an auto-regressive model that captures the dependencies between parts and generates the query code for the long-term memory. Consequently, the query vector fed into the long-term memory is conditionally sampled based on previous samples. This enables our model to avoid sampling at non-smooth

boundaries between parts, effectively mitigating mode collapse issue. Despite using a single generator for all parts, our experiments demonstrate that MemoryGAN achieves comparable performance to state-of-the-art models that employ multiple generators, both in terms of image quality and compositional ability.

MemoryGAN offers three key advantages. First, the parameter count required for parts remains constant, ensuring scalability. Second, all parts are modeled within a single generator, facilitating the interaction between parts and enabling parameter sharing. This aspect is crucial for out-of-distribution generalization. Third, as a memory system, MemoryGAN features a flexible query and feedback mechanism, making it convenient to transfer learned representations to new tasks. We believe that our approach of utilizing the GAN generator as a memory model will pave the way for future research on both biologically inspired models and memory-augmented models. Our contributions in this work are summarized as follows:

1. We propose a compositional image synthesis model that explicitly models each component of the object. Our model is more interpretable and enables individual control over the attributes of each semantic part with minimal impact on other components.
2. We propose a method that utilizes a single generator for all semantic parts, making our model scalable. We extensively conduct experiments to evaluate the performance of our model and demonstrate that it achieves comparable results in terms of image quality and compositional ability to state-of-the-art approaches that employ multiple generators.
3. We demonstrate that the generator of a GAN can serve as a heterogeneous memory model and present MemoryGAN that mimics humans' long-term memory and working memory system. Our work has the potential to inspire future research in the development of bio-friendly models and memory-augmented models.

2. Related Work

There are three features of our model: compositional image synthesis; iteratively generating images; and using a memory model to store and retrieve information. In this section, we will review related works from these three aspects.

Compositional Image Synthesis Composition can occur at different levels depending on how the underlying generative factors are disentangled. One approach is to disentangle the generation factors into a background, a foreground shape and a foreground appearance [19,20]. This line of work is usually performed under an unsupervised manner or with the bounding box information to disentangle the foreground object with the background. Different from these models, we are interested in fine-grained disentanglement and composition under supervision. Another line of work tries to decompose scenes or objects into meaningful parts [11,21], with [11] or without [21] supervision. Unfortunately, the work in [21] can only disentangle large parts on the face dataset, such as background, skin and hair. Their disentanglement quality is low and needs to carefully chose the total part count to match the ground truth. Although SemanticStyleGAN [11] achieves high quality part level disentanglement and composition, they use a separate local generator for the generation of each part, while our MemoryGAN uses a single generator for the generation of all the parts, which makes the generator flexible and suitable as a memory model.

The world is intrinsically three-dimensional. Many researchers have recently taken this fact into account for image generation [22–24]. The work in [22], known as NeRF, achieves impressive 3D scene rendering. There, given enough 2D images of one scene taken from various viewpoints, the 3D scene is molded as a neural radiance field, which maps the point location and the camera view direction into a volume density and a color value. Images with novel viewpoints can be generated by volume rendering along a novel camera view direction. GRAF [23] extends NeRF to generate novel objects by conditioning the generator with a set of random codes for the camera view direction and the object shape and appearance. GIRAFFE [24] further extends the model to handle multiple objects and achieves better control of rotation and translation. The work in these

papers is promising, but their disentanglement is at the object level, while we are interested in disentanglement at the semantic part level and emphasize the usefulness of the GAN generator as a memory model.

Iteratively Generate Images Iteratively generating images has a long history in the literature [25–31]. Auto-regressive models [6,32] generate images pixel by pixel, where the sampling for current step is conditioned on the previous sampling. The work of [25] provides a recurrent attention model (RAM) to mimic a human's glimpse process, in which information is accumulated piece by piece through a sequence of attention steps. DRAW [26] and its follow-up convolutional version [27] extend this idea to a generative model. Their work can be viewed as an extension of a variational auto-encoder [5], where the encoder and the decoder are recurrent neural networks. They use a differentiable read and write head to determine where to read, where to write and what to write; as a result, images are drawn on a canvas matrix step by step. Instead of iteratively generating the images, a recently proposed work, PEGANs [33], adopts evolutionary computing techniques to iteratively refine a set of generators. The main difference between these works and ours is that there is no disentanglement in the generation process of these works.

Another line of work considers the fact that a scene usually composes multiple objects. AIR [28] and its follow-up work [29] represent the object with a what code, a where code and a scalar value that represents the presence probability. One object is generated at each step until all the objects are generated. MONet [31] uses an attention network to generate a mask for each object. They use this mask to control what object to generate and where the object is located. IODINE [30] uses a spatial Gaussian mixing model to encode the "where" and "what" information of the object. The parameters of the spatial Gaussian mixing model are updated using iterative amortization [34], which is helpful to disentangle ambiguous objects by leveraging both bottom–up and top–down signals.

These models differ from ours presented in this paper in the following three aspects. First, they are all built upon VAE models, which needs an encoder network to encode the distribution of the statistic layer. Our model is mainly built upon a GAN; thus, there is no encoder network in our model. Second, although these models achieve disentanglement during image generation, their disentanglement is at the object level, while our model is concerned with disentanglement at the semantic part level. Third, these models consider the "where" information of objects: although an important factor to consider and definitely worth exploring, it inevitably complicates the model. Thus, they are only feasible on simple datasets, and there is no evidence that their model can scale to real-world and high-resolution image generation.

Memory Models There are a large number of memory models published in the literature. Early work dates back to the associative memory models, such as the hopfield neural network [35] and sparse distributed memory model [36]. The notion of association there means that given a part of the query information, the whole information can be retrieved from the memory. Those works only focus on memory itself and do not consider how to use the memory to perform compositional learning. With the rise of deep learning, several memory-augmented neural network models are developed [37–40]. NTM [39], and the follow-up work DNC [40] maintains an external memory with differentiable read and write heads. These read and write heads are controlled by a recurrent neural network (RNN) [41,42]. Their work shows that the model can learn to use the memory to solve tasks that need reasoning, such as copy, sorting and shortest path search. Different from these models, our memory model is actually a GAN generator. Compared with these deterministic memory models, the generative memory model has several advantages. First, it does not passively store information but is able to generate novel information that have never been seen. Second, it embeds the entities in a smooth manifold. This property is evidenced by style-based GAN models [9]. Given two images, we can generate smoothly changing images by linearly interpolating the latent codes, which means the latent space representing the images is smooth. This property is beneficial for out-of-distribution generalization and association since, given a latent representation, it is possible to find a similar

but different latent representation near it. Recent work also treats the generator of GAN as a memory model [43,44], but these models focus on lifelong learning [45]. They leverage the generative nature of GAN to perform pseudo-rehearsal to solve a catastrophic forgetting problem [46], which is not our focus in this paper.

3. Methods

Our work is built upon SemanticStyleGAN [11]. In this section, we first overview the main logic of MemoryGAN in Section 3.1. Then, we review SemanticStyleGAN in Section 3.2, and the details of our MemoryGAN are given in Section 3.3.

3.1. Overview of MemoryGAN

MemoryGAN is inspired by the human memory system, especially the long-term memory and the working memory theory [12,47]. Its structure is illustrated in Figure 1. There are three parts in MemoryGAN: a working memory M_w which is modeled by a *Controller* module and a *Compositional* module; a long-term memory M_l which is the generator of GAN; and as we use adversarial loss to train the whole model, we also need a discriminator D.

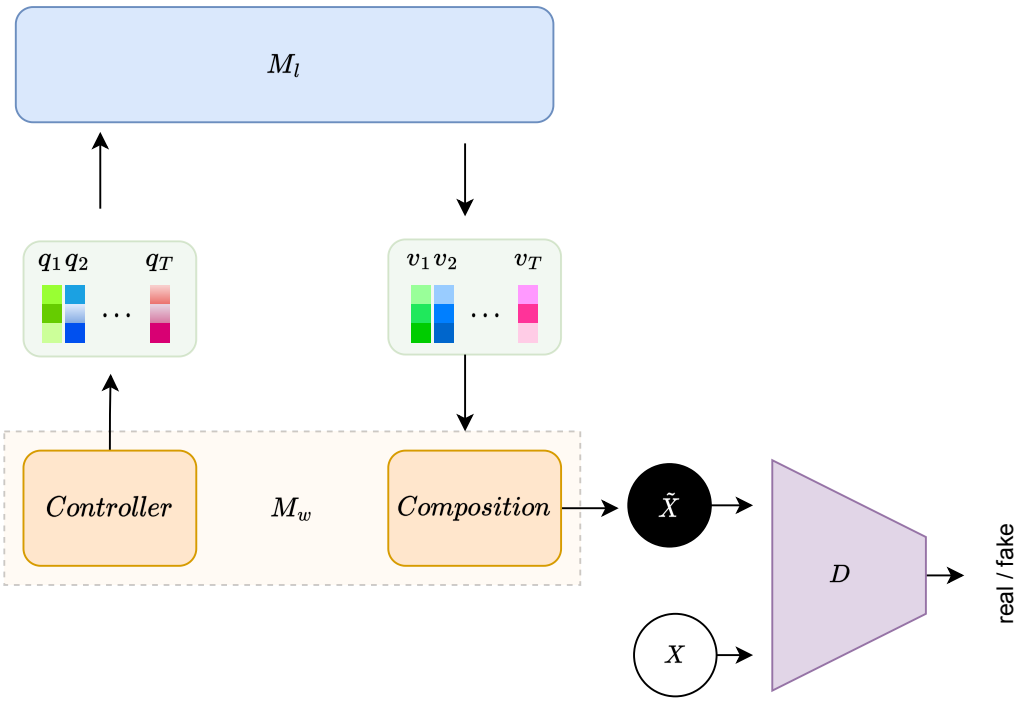


Figure 1. Overview of MemoryGAN. The working memory M_w consists of a *Controller* module and a *Composition* module. The *Controller* provides the query keys $(q_1, q_2, \cdots, q_T)$ to the long-term memory M_l, which returns the values $(v_1, v_2, \cdots, v_T)$. The *Composition* module composites the retrieved values into the final output $\tilde{x}$. The discriminator D tries to distinguish the fake input $\tilde{x}$ with the real input x.

Formally, there are two goals for the working memory M_w. The first goal is to model the joint distribution $P(\mathbf{q}_1, \mathbf{q}_2, \cdots, \mathbf{q}_T)$, where $\mathbf{q}_t$, $t = 1, \cdots, T$, is a random variable, representing the query code at time step t, and T is the maximum step. In practice, we usually set T to semantic part count K. In other words, we query one semantic part in each step. Inspired by the auto-regressive model [6], we factorize the joint distribution as the production of the conditional distributions over query codes:

$$P(\mathbf{q}_1, \mathbf{q}_2, \cdots, \mathbf{q}_T) = \prod_{t=1}^{T} P(\mathbf{q}_t | \mathbf{q}_1, \cdots, \mathbf{q}_{t-1}) \tag{1}$$

The other goal of M_w is to composite the retrieved values $(v_1, v_2, \cdots, v_T)$, which are returned from M_l, into the final output $\tilde{x}$:

$$\tilde{x} = Composition(v_1, v_2, \cdots, v_T) \tag{2}$$

where *Composition* is defined by a neural network.

The role of long-term memory M_l is to respond to the query of working memory M_w:

$$v_t = M_l(q_t),\ t = 1, 2, \cdots, T \tag{3}$$

where $q_t \sim P(\mathbf{q}_t | \mathbf{q}_1, \cdots, \mathbf{q}_{t-1})$.

The training of MemoryGAN follows a normal GAN training procedure with adversarial loss.

3.2. Review of SemanticStyleGAN

In this work, we chose SemanticStyleGAN [11] as our base GAN model. Its local generator and render net design make the memory lightweight and thus computationally efficient. For clarity, we now briefly review SemanticStyleGAN.

SemanticStyleGAN is an extension of StyleGAN [9], which aims to explicitly control semantic part generation and composition under supervision. Suppose we have a dataset $D = \{(x_1, y_1), (x_2, y_2), \cdots, (x_N, y_N)\}$, where $x_i \in \mathbb{R}^{H \times W \times 3}$ is the training image, $y_i \in \{0, 1\}^{H \times W \times K}$ is a pixelwise semantic mask for x_i, H and W are, respectively, the spatial height and width, K is the semantic part count (background is also treated as a part), and N is the total data count. The generator G of SemanticStyleGAN learns a mapping $G : \mathcal{W}^+ \mapsto \mathcal{X} \times \mathcal{Y}$, where $\mathcal{W}^+$, which will be discussed latter, is a latent space that controls the style of each semantic part, and $\mathcal{X}$ and $\mathcal{Y}$ are the image and semantic mask space, respectively. There are mainly three parts in SemanticStyleGAN, which are a set of local generators $\{g_k\}_{k=1}^K$, a render network R and a dual-branch discriminator D. The architecture of SemanticStyleGAN is illustrated in Figure 2.

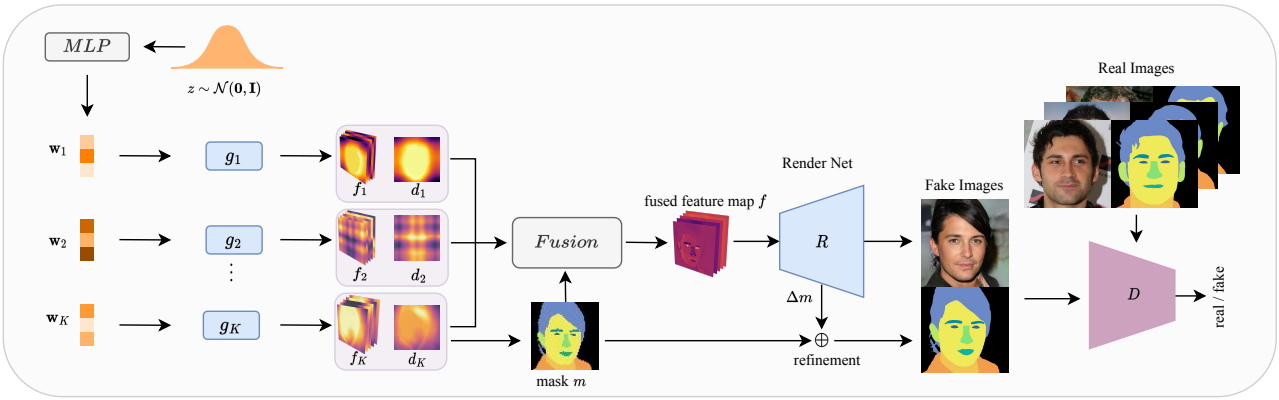


Figure 2. The architecture of SemanticStyleGAN [11]. SemanticStyleGAN employs a set of local generators $\{g_k\}_{k=1}^K$ to generate the features of each semantic part. These part features are then fused together using the *Fusion* module to form a whole feature map f. The render net R utilizes f to generate the image and mask. The discriminator D provides adversarial loss for training.

Noise vector $z \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ is mapped to a latent code $w \sim \mathcal{W}$ by a stack of equal linear layers, where $\mathcal{N}(\mathbf{0}, \mathbf{I})$ is multi-variable Gaussian with zero mean $\mathbf{0}$ and identity covariance matrix $\mathbf{I}$, and $\mathcal{W}$ represents the latent space. $\mathcal{W}$ is further expanded to latent space $\mathcal{W}^+ = \mathcal{W}^{base} \times \mathcal{W}^1 \times \mathcal{W}^2 \times \cdots \times \mathcal{W}^K$, where $\mathcal{W}^{base}$ is the latent space that controls the coarse structure of the object, and $\mathcal{W}^k$ is the latent space that controls the style of the k-th semantic part, which is further factorized into two latent spaces $\mathcal{W}^k = \mathcal{W}_s^k \times \mathcal{W}_t^k$, where $\mathcal{W}_s^k$ controls the shape attribute and $\mathcal{W}_t^k$ controls the texture attribute.

Local generator and fusion The structure of the local generator is illustrated in Figure 3. The local generator g_k, modulated by the latent codes (w^{base}, w_s^k, w_t^k), receives a Fourier feature p and outputs a tuple (f_k, d_k).

$$g_k : (p, w^{base}, w_s^k, w_t^k) \mapsto (f_k, d_k) \tag{4}$$

where $f_k \in \mathbb{R}^{H_f \times W_f \times D_f}$ and $d_k \in \mathbb{R}^{H_f \times W_f \times 1}$ are the feature map and the pseudo-depth map for the k-th semantic part, and H_f, W_f and D_f are the spatial height, spatial width and channel dimension, respectively.

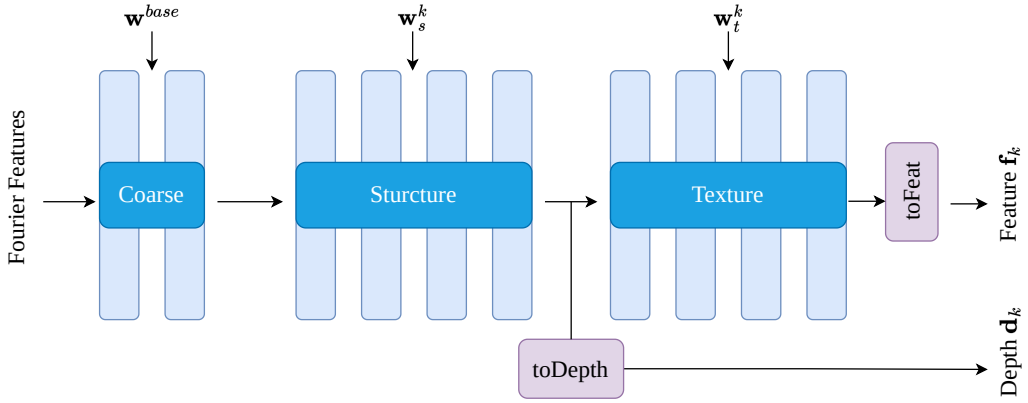


Figure 3. The architecture of the local generator in SemanticStyleGAN [11]. The local generator takes Fourier features as input. The first few layers are modulated by $\mathbf{w}_{base}$, which represents the coarse structure. The middle layers are modulated by $\mathbf{w}_s^k$, which represents the shape of a part. The last few layers are modulated by $\mathbf{w}_t^k$, which represents the texture of a part. *toDepth* and *toFeat* produce the pseudo-depth d_k and the feature f_k, respectively.

The generated feature maps $\{f_k\}_{k=1}^K$ need to be fused into a whole feature before feeding into the render net R. To do this, the depth maps $\{d_k\}_{k=1}^K$ are transformed into a segmentation mask $\{M_k^{coarse} \in [0,1]^{H_m \times W_m \times K}\}_{k=1}^K$ by applying the softmax activation:

$$M_k^{coarse}(i,j) = \frac{exp(d_k(i,j))}{\sum_{k'}^K exp(d_{k'}(i,j))} \tag{5}$$

where (i,j) indexes the spatial dimension. Then, the whole feature f is fused by:

$$f = \sum_{k=1}^{K} M_k^{coarse} \odot f_k \tag{6}$$

where $\odot$ denotes element-wise multiplication.

Render net and discriminator The render net R and the discriminator D are the modified versions of their counterparts in StyleGAN2 [9]. There are two modifications made in R. First, FixedStyledConv is used, where the style of the modulated convolution layer is fixed; thus, the output only depends on the input features. Second, a ToSeg branch is added to generate the high-resolution segmentation mask $M^{fine} \in [0,1]^{H \times W \times K}$. As there is no direct gradient to M^{coarse}, a mask loss L_{mask} is added to feedback the error signal to M^{coarse}, which is defined as

$$\mathcal{L}_{mask} = \|upsample(M^{coarse}) - M^{fine}\|_2 \tag{7}$$

For the discriminator D, a segmentation branch and a separate $R1$ regularization $\mathcal{L}_{R1_{seg}}$ [48] are used.

The final loss is:

$$\mathcal{L}_{all} = \mathcal{L}_{stylegan2} + \mathcal{L}_{mask} + \mathcal{L}_{R1_{seg}} \tag{8}$$

where $\mathcal{L}_{stylegan2}$ is the loss function used in StyleGAN2 [9].

3.3. Detailed Structure of MemoryGAN

As mentioned in Section 3.1, there are two goals for working memory M_w. The first goal is to model the joint distribution over query codes $P(\mathbf{q}_1, \mathbf{q}_2, \cdots, \mathbf{q}_T)$ in Equation (1). This is achieved by the *Controler* module depicted in Figure 4. As the background is a special part which usually has weak dependency with other foreground parts, we treat it separately. We start from a learnable constant $V_{bkg} \in \mathbb{R}^{C_{bkg}}$, which is then mapped to a tuple $(\mu_{bkg}, \log var_{bkg})$ by a two-layer MLP, where C_{bkg} is the dimension of the constant vector. The query code for the background q_{bkg} is then sampled from a diagonal Gaussian distribution defined by $(\mu_{bkg}, \delta_{bkg})$ using a reparameterization trick:

$$q_{bkg} = \epsilon_{bkg} \odot \delta_{bkg} + \mu_{bkg} \tag{9}$$

where $\epsilon_{bkg} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, $\delta_{bkg} = \sqrt{var_{bkg}}$. As there is usually a strong correlation between foreground parts, we use an LSTM [42] to approximate their joint distribution. Similar with SemanticStyleGAN, we divide the query code for a part into a base code and an appearance code. The base code is shared for all the foreground parts which defines the coarse level structures, such as the pose direction. The appearance code is specific for each part and defines the fine level structure of the part, such as shapes and textures. Now, we turn to the base query code generation. A learnable constant $V_{base} \in \mathbb{R}^{C_{base}}$ is mapped to a tuple $(\mu_{base}, \log var_{base})$ by a two-layer MLP, where C_{base} is the dimension of the constant vector; then, the base query code is sampled from a diagonal Gaussian defined by $(\mu_{base}, \delta_{base})$ using a reparameterization trick:

$$q_{base} = \epsilon_{base} \odot \delta_{base} + \mu_{base} \tag{10}$$

where $\epsilon_{base} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ and $\delta_{base} = \sqrt{var_{base}}$. We use q_{base} as the initial input to the LSTM layer. For step t, the LSTM module maintains a hidden state h_t which summarizes the sampled query codes until step t. The query code q_{t-1} sampled at time step $t-1$ and the base query code q_{base} are fed into LSTM at time step t. The role of q_{t-1} is to model the conditional probability defined in Equation (1), while the role of q_{base} is to emphasize the fact that all the part features should follow the same instruction of the base code. The output layer of the LSTM module is a single linear layer that maps the hidden state h_t to a tuple $(\mu_t, \log var_t)$. Then, the query code at step t is sampled from the diagonal Gaussian distribution defined by (μ_t, δ_t) through a reparameterization trick:

$$q_t = \epsilon_t \odot \delta_t + \mu_t \tag{11}$$

where $\epsilon_t \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ and $\delta_t = \sqrt{var_t}$.

The second objective of M_w is to composite the retrieved values $(v_{bkg}, v_1, \cdots, v_T)$ in order to generate the final output $\tilde{x}$. This compositional process is facilitated by the *Composition* module, as illustrated in Figure 5. We use the original setting of SemanticStyleGAN, i.e., using Equations (5) and (6) to composite the part features into the whole feature, and using the render net R to render the whole feature into the final output image $\tilde{x}$.

The goal of the long-term memory M_l is to respond to the query from the working memory M_w, as shown in Figure 6. Here, we use the local generator from SemanticStyleGAN as our long-term memory. Instead of using a separate local generator for each part, we only use a single local generator for all the parts. It should be noted that for both SemanticStyleGAN and our MemoryGAN, we do not need to explicitly factorize the appearance code to a shape code and a texture code, as this factorization is defined by first, their location in the generator as shown in Figure 3, and second, style mixing over these two groups of styles. The final generator is defined as below:

$$g : (p, w^{base}, w^k_{app}) \mapsto (f_k, d_k) \tag{12}$$

where p is the Fourier features, w^{base} is the base latent code and $w^k_{app} = w^k_s = w^k_f$ is the appearance latent code.

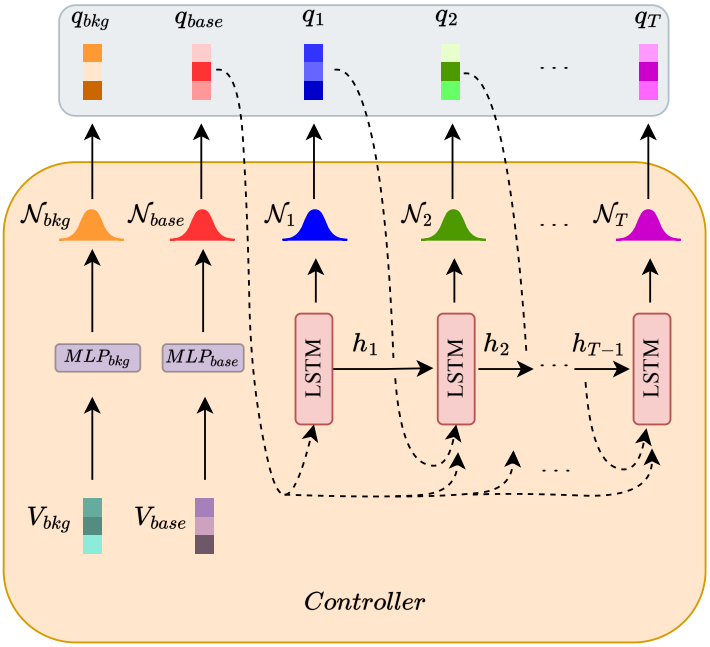


Figure 4. The architecture of the *Controller* module. The LSTM module models dependencies among semantic parts and generates query codes representing them.

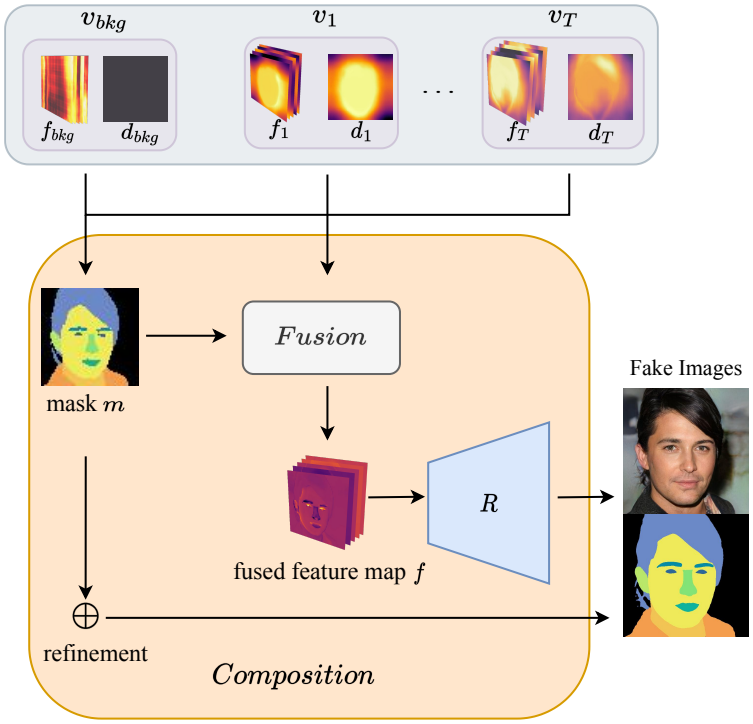


Figure 5. The architecture of the *Composition* module. It consists of the *Fusion* module and the render net R of SemanticStyleGAN. This module takes the retrieved values from the long-term memory as input and composes them together to form the image and segmentation mask.

The training procedures are almost identical with SemanticStyleGAN except we add a Kullback–Leibler divergence term defined as

$$\mathcal{L}_{KL} = \frac{1}{T}\sum_{t=1}^{T} -D_{KL}(P(\mathbf{q}_t|\mathbf{q}_1, \cdots, \mathbf{q}_{t-1})\|\mathcal{N}(\mathbf{0}, \mathbf{I})) \tag{13}$$

Hence, the final loss is

$$\mathcal{L}_{all} = \mathcal{L}_{stylegan2} + \mathcal{L}_{mask} + \mathcal{L}_{R1_{seg}} + \mathcal{L}_{KL} \tag{14}$$

The aim of $\mathcal{L}_{KL}$ is two fold. First, it prevents the value of the query code from becoming too large, which is harmful for training. Second, it encourages feature sharing between the parts.

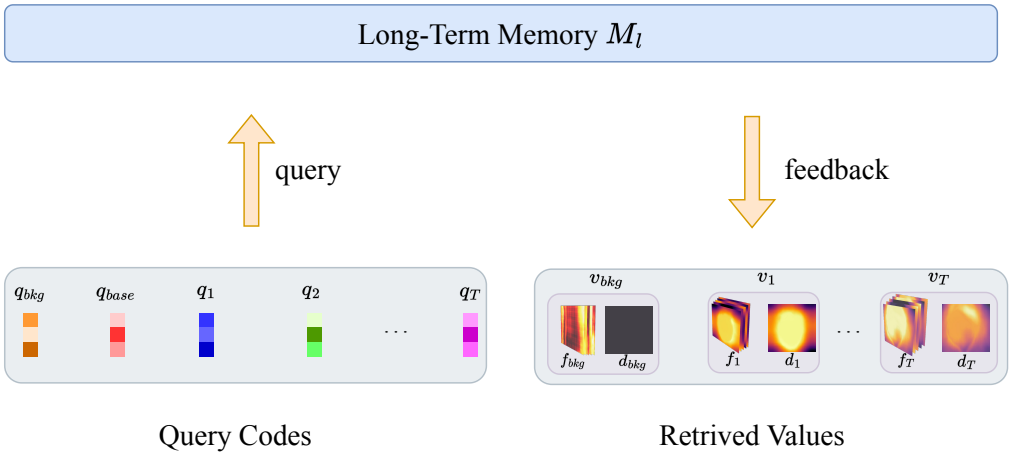


Figure 6. The architecture of the long-term memory module. M_l is a single local generator that takes the query codes as input and provides the corresponding features.

4. Experiments and Results

4.1. Dataset and Training Method

Our experiments are performed on the CelebAMask-HQ dataset [49], which consists of 30,000 facial images with size 512 × 512; all images are annotated with 19 classes of segmentation masks. Following SemanticStyleGAN [11], the first 28,000 images resized to 256 × 256 are used for training. Thirteen segmentation masks are selected for semantic supervision, which are background, skin, eyes, eyebrows, nose, mouth, ears, hair, neck, cloth, eye glass, hat and ear ring. The values of the images and the segmentation masks are normalized to range $[-1, 1]$ before being fed into the network.

As for training, we use the identity setting with SemanticStyleGAN. Specifically, we use an Adam optimizer [50], with $\beta = (0, 0.99)$. The learning rate is 0.002 for all the modules except for the LSTM sub-module, as we found that the learning rate of 0.002 is too large for LSTM and it tends to cause a gradient spike. So, we reduce it to 1×10^{-4}. The path length regularization and $R1$ regularization are performed every 4 and 16 iterations, respectively. The weight factors for $\mathcal{L}_{mask}$ and $\mathcal{L}_{KL}$ are set to 100 and 1×10^{-4}, respectively. The style mixing is performed during training with a probability of 0.3. Our experiments are executed on a single RTX3090 GPU, and we use the largest batch size that can fit our VRAM. We use a batch size of 16 with a gradient accumulation every two steps. We train our model for 150,000 steps, and it took about 7 days.

4.2. Model Complex Comparison

As we use the same discriminator as SemanticStyleGAN, for clarity, we compare the model complexity based on the generator for both models. Algorithm 1 and Algorithm 2 show the inference algorithm for SemanticStyleGAN and MemoryGAN, respectively.

In both algorithms, we use the same symbols as in Sections 3.2 and 3.3, with the exception that we omit the spatial dimension in the calculation of the coarse mask for clarity (line 6 in Algorithm 1 and line 20 in Algorithm 2).

Algorithm 1: Inference algorithm of SemanticStyleGAN

1: **Input** : semantic part count K, Fourier feature p, MLP that maps noise code to latent code MLP_{latent}, a set of local generators $\{g_k\}_{k=1}^{K}$, render network R
2: **Output**: Generated image $\tilde{x}$ and mask M^{fine}
3: $z \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$;
4: $w_{app} = MLP_{latent}(z)$;
5: **for** $k = 1$ **to** K **do**
6: $\quad (f_k, d_k) = g_k(p, w_{app})$;
7: **end**
8: **for** $k = 1$ **to** K **do**
9: $\quad M_k^{coarse} = \frac{exp(d_k)}{\sum_{k'=1}^{K} exp(d_{k'})}$;
10: **end**
11: $f = \sum_{k=1}^{K} M_k^{coarse} \odot f_k$;
12: $(\tilde{x}, M^{fine}) = R(f)$;
13: **return** $\tilde{x}$, M^{fine}

The main difference between our model and SemanticStyleGAN is in line 12 and 18 of Algorithm 2. In line 12, we utilize an LSTM module to model the dependency between the part query codes, which enables us to use a single local generator for generating all the semantic parts, as shown in line 18. As both the LSTM module and the local generator are shared, the complexity with respect to model size is $\mathcal{O}(1)$, which makes our model efficient and scalable. In contrast, SemanticStyleGAN employs a single latent code to modulate a set of local generators, as shown in line 4 of Algorithm 1. It is easy to see that the complexity with respect to model parameters is $\mathcal{O}(n)$ for SemanticStyleGAN. This makes the model impractical when the semantic parts count is large. Another advantage of our algorithm is that we associate each semantic part with a query code (line 4 and line 15), and all part information is stored in a shared memory—the local generator—which can be retrieved by providing the query code, eliminating the need to determine which local generator should be used for storage or retrieval of the part information (line 18). In contrast, as depicted in Algorithm 1, SemanticStyleGAN uses a separate local generator to model each semantic part (line 4), requiring additional effort to determine the appropriate generator for retrieval or storage of the semantic part information, which is inflexible.

It should be noted that the computational complexity of both models is $\mathcal{O}(n)$. More effort is needed to further reduce the compositional complexity. One promising direction is to consider taxonomy and use a hierarchy of latent codes to modulate the shared generator. We leave this topic to our future research.

Table 1 and Figure 7 show the real parameter count and Multiply–Accumulate operations (MACs) count with respect to semantic part count. It can be seen in Table 1 that for the parameter count, MemoryGAN has a small overhead for one semantic part (2.3 M) compared to SemanticStyleGAN. However, as the semantic parts count grows, the parameter count for MemoryGAN only increases slightly, whereas for SemanticStyleGAN, it grows dramatically. With 1000 semantic parts, the parameter count for SemanticStyleGAN is over 20 times larger than that for MemoryGAN. This trend is more clearly depicted in Figure 7. The difference in parameter count between SemanticStyleGAN and MemoryGAN becomes obvious when the semantic part count exceeds 10. In accordance with previous analysis, the MACs count reveals that both models exhibit similar trends. The MACs count increases

significantly when the semantic part count exceeds 10, indicating that it remains an open problem that requires further exploration in future work.

Algorithm 2: Inference algorithm of MemoryGAN

Input : semantic part count K, constant vector for background code V_{bkg}, constant vector for base code V_{base}, MLP for background code MLP_{bkg}, MLP for base code MLP_{base}, LSTM module $LSTM$, MLP that maps query code to latent code MLP_{latent}, Fourier feature p, local generator g, render network R

Output: Generated image $\tilde{x}$ and mask M^{fine}

$(\mu_{bkg}, \log var_{bkg}) = MLP_{bkg}(V_{bkg})$;
$\epsilon_{bkg} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$;
$\delta_{bkg} = \sqrt{var_{bkg}}$;
$q_{bkg} = \epsilon_{bkg} \odot \delta_{bkg} + \mu_{bkg}$;
$(\mu_{base}, \log var_{base}) = MLP_{base}(V_{base})$;
$\epsilon_{base} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$;
$\delta_{base} = \sqrt{var_{base}}$;
$q_{base} = \epsilon_{base} \odot \delta_{base} + \mu_{base}$;
$q_1 = q_{bkg}$;
$w^1_{app} = MLP_{latent}(q_1)$;
for $k = 2$ **to** K **do**
    $(\mu_k, \log var_k) = LSTM(q_{base}, q_{k-1})$;
    $\epsilon_k \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$;
    $\delta_k = \sqrt{var_k}$;
    $q_k = \epsilon_k \odot \delta_k + \mu_k$;
    $w^k_{app} = MLP_{latent}(q_k)$;
end
for $k = 1$ **to** K **do**
    $(f_k, d_k) = g(p, w^k_{app})$;
end
for $k = 1$ **to** K **do**
    $M^{coarse}_k = \frac{exp(d_k)}{\sum_{k'=1}^{K} exp(d_{k'})}$;
end
$f = \sum_{k=1}^{K} M^{coarse}_k \odot f_k$;
$(\tilde{x}, M^{fine}) = R(f)$;
return $\tilde{x}$, M^{fine}

Table 1. Model parameter and operation count comparison.

Method	#Semantic Parts	#Parameter (M)	#MACs (G)
SemanticStyleGAN	1	15.4	103.4
	10	19.0	106.5
	100	55.4	136.9
	1000	419.8	440.8
MemoryGAN	1	17.7	103.4
	10	17.7	106.4
	100	17.8	136.3
	1000	19.1	435.4

# denotes the number of the corresponding items.

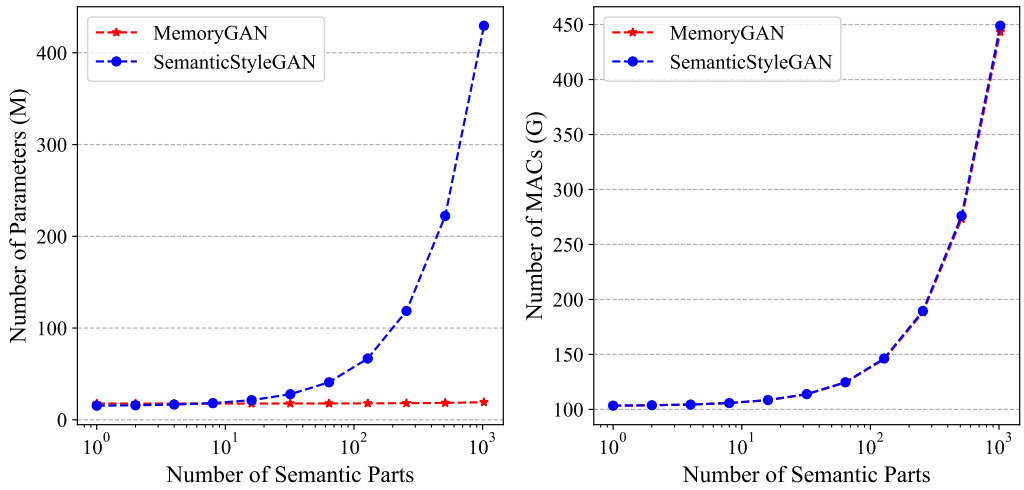


Figure 7. Comparison of parameter count and MACs with respect to semantic part count.

4.3. Image Quality Evaluation

We first evaluate the image quality generated by MemoryGAN. Figure 8 shows the generated images and semantic masks by our model. We can see that MemoryGAN is able to generate perceptually high-quality images and correct segmentation masks. To quantitatively measure the quality of the generated images, we calculate the Fréchet Inception Distance [51] and Inception Score [52]. We compare our results with that reported in SemanticStyleGAN [11]. Table 2 shows that compared with SemanticStyleGAN and SemanticGAN [17], our model achieves comparable FID (the lower the better) and slightly lower IS (the higher the better) values, indicating that our model is able to generate comparable fidelity images.

Figure 8. Examples of the generated images and segmentation masks by MemoryGAN. The model is trained on CelebAMask-HQ with a resolution of 256 × 256. A truncation of 0.7 is used for the generation.

Table 2. Synthesis image quality evaluation.

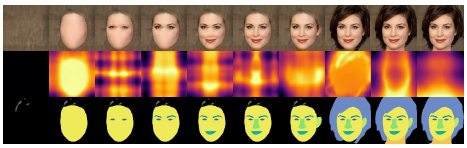
Method	Data	Compositional	FID↓	IS↑
SemanticGAN [1]	img&seg	✗	7.50	3.51
SemanticStyleGAN [1]	img&seg	✓	6.42	3.21
MemoryGAN	img&seg	✓	6.41	3.07

[1] These data are referenced from [11]. ↓ denotes lower values are better, ↑ denotes higher values are better, ✗ denotes the model is not compositional, and ✓ denotes the model is compositional.

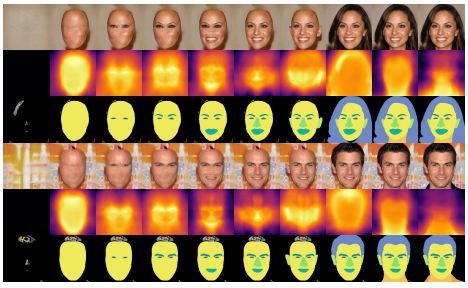
For the CelebAMask-HQ task, SemanticStyleGAN uses 13 local generators, and each controls one semantic part generation. On the contrary, our model only uses a single local generator (the long-term memory in our architecture). One insight of our result is that it is possible to hold all the heterogeneous parts information in a single generator and then recall and composite them together to form high-fidelity images. So, it is unnecessary to use a separate generator for each part object. An interesting consequence of this result is that the single generator can be reasonably treated as a shared memory. Heterogeneous information can be stored in and retrieved from it. We believe this will lead to future work on both bio-friendly models and memory augmented models.

4.4. Sequential Composition

SemanticStyleGAN [11] exhibits an impressive sequential composition property. Starting from the background, each part is sequentially added for composition. This method intuitively demonstrates both the compositional property and the disentanglement property of the model. It is interesting to see to what extent MemoryGAN preserves this desirable property using only a single generator. Our evaluation results are shown in Figure 9. Figure 9a displays the results of SemanticStyleGAN, while Figure 9b shows the results of MemoryGAN. In each sample of the figure, the composed images are shown in the top row, which is followed by the depth maps in the middle row and the segmentation masks in the bottom row. It can be observed that for both models, new parts are sequentially added with minimal disturbance to the previously added parts. The depth maps largely correspond to their respective semantic parts, and the segmentation masks align well with the added semantic parts. These results indicate that MemoryGAN achieves comparable performance to SemanticStyleGAN. It is worth noting that there is no depth map supervision in the training data, yet both models are able to learn the depth maps automatically. We believe this phenomenon provides evidence of the advantages of compositional-aware models. The key to a compositional model is to express the whole object as a composition of its parts. This prior encourages the reuse of learned features by recombining them with different configurations to form new objects. Driven by this reusability prior, the models tend to learn the complete part from occluded parts images, as the complete part is more general and reusable compared to the occluded part. Additionally, guided by the recombination prior, the models tend to learn a configuration strategy to express the occlusions of the parts. The depth maps simply represent the learned configuration strategy.



(**a**) SemanticStyleGAN



(**b**) MemoryGAN

Figure 9. Examples of the sequential part composition of (**a**) SemanticStyleGAN and (**b**) MemoryGAN. Starting from the background feature, each part feature is sequentially added for composition. For each sample, the first row is the synthesized image, the second row is the pseudo-depth map and the third row is the segmentation mask.

Although it is difficult to see the change of the previous part when a new part is added with the human eyes, a closer inspection reveals that there is still interference between them. For example, the jaw is slightly changed when a neck is added in both, as shown in Figure 9a,b. This indicates that there is still an entanglement between parts. To quantify the

entanglement strength, we provide a metric called Sequential Mean Square Error (SMSE), which is defined in Equation (15).

$$\mathrm{SMSE}_i = \frac{\sum_{h=1}^{H}\sum_{w=1}^{W}\left[\left(\sum_{j=1}^{i-1} M_{j,h,w}^{i}\right)\left(I_{h,w}^{i} - I_{h,w}^{i-1}\right)\right]^2}{\sum_{h=1}^{H}\sum_{w=1}^{W}\sum_{j=1}^{i-1} M_{j,h,w}^{i}} \tag{15}$$

where i is the sequence index, h, w are the spatial height and width index, respectively, $I_{h,w}^{i}$ is the image pixel value at location (h, w) at step i, and $M_{j,h,w}^{i} \in \{0, 1\}$ is the mask value at location (h, w) of the j-th part at step i. Intuitively, it describes the varying intensity of the original image when the i-th part is added. A lower value of SMSE indicates a smaller entanglement between current part with the parts previously added.

The SMSE results are shown in Figure 10 and Table 3. It can be seen that both SemanticStyleGAN and MemoryGAN have a relatively larger SMSE value when the skin and hair are added, indicating that there is entanglement in both models, especially for the larger parts. The SMSE value of MemoryGAN is slightly higher than that of SemanticStyleGAN, which is expected. As MemoryGAN uses a single generator for all the parts, there should be stronger interaction between the parts than its multi-generator counterpart.

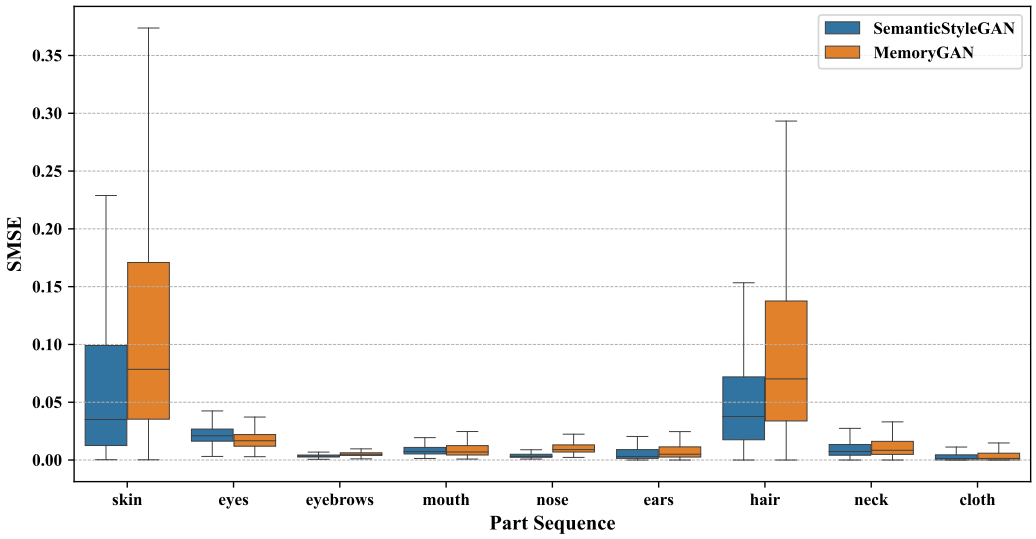


Figure 10. Quartiles of SMSE. The order of the parts corresponds to the order in which they are combined. The statistical calculation is taken over 10,000 samples.

Table 3. Mean and standard deviation of SMSE for SemanticStyleGAN and MemoryGAN. The stochastic calculation is taken over 10,000 samples.

Method	**Skin**	**Eyes**	**Eyebrows**	**Mouth**	**Nose**
SemanticStyleGAN	0.079 ± 0.105	0.022 ± 0.009	0.004 ± 0.002	0.009 ± 0.006	0.004 ± 0.003
MemoryGAN	0.123 ± 0.126	0.018 ± 0.008	0.005 ± 0.002	0.010 ± 0.010	0.011 ± 0.006
Method	**Ears**	**Hair**	**Neck**	**Cloth**	
SemanticStyleGAN	0.008 ± 0.012	0.053 ± 0.051	0.011 ± 0.012	0.004 ± 0.007	
MemoryGAN	0.010 ± 0.012	0.107 ± 0.120	0.015 ± 0.021	0.005 ± 0.010	

4.5. Latent Space Property

In this section, we study the latent space property of MemoryGAN through style mixing, interpolation, low-dimensional embedding and correlation.

Style Mixing Style mixing means the information encoded in the latent code of one sample can be transferred to another sample by swapping the latent code on certain layers of the generator. Meaningful style mixing requires the latent codes to be disentangled, so that when swapping the latent code, only certain property is changed. As MemoryGAN is able to control the composition at a semantic part level, we also evaluate its style-mixing property at this level. The results are shown in Figure 11. It can be seen that MemoryGAN is able to mix the styles for most of the parts. However, there are some disadvantages of using MemoryGAN on this task. One disadvantage is that we found it is not enough to swap the latent code for the layers of shape and texture in M_l to transfer the style information for smaller parts, such as the eyebrows, the mouth and the nose. We have empirically found that we have to swap the latent code for the base layer in M_l, too. Here, we additionally swap the latent code for the last base layer for the small parts. This is probably due to the fact that the smaller part occupies fewer spatial regions. Hence, fewer neurons are involved to encode their information. MemoryGAN learns to compensate for this disadvantage by encoding the information of the smaller part in more layers in M_l. Another disadvantage is that there is entanglement for the face parts, especially for the eyes. It can be seen from Figure 11 that the eye color is not affected if we only swap the eye latent code. On the other hand, the eye color is affected by the skin or the hair. This indicates that the eye color is entangled with the colors of the skin and hair. It is actually well known that the colors of the eyes, skin and hair of the human are strongly correlated [53]. Their colors are determined by the same thing: that is, the amount of pigment in the body. We conjecture that both the design of using a single generator and the explicit dependency modeling in the *Controller* module make MemoryGAN sensitive to these dependence and bias in the dataset. MemoryGAN captures this dependency and uses the hair and skin latent code to model the color of the eyes to save memory resources. This conjecture is supported by the results in Section 4.6.1. Considering the result that MemoryGAN is able to disentangle and mix styles for most smaller parts, we believe it is possible to achieve full disentanglement and style mixing with further development. We leave this problem for our further research.

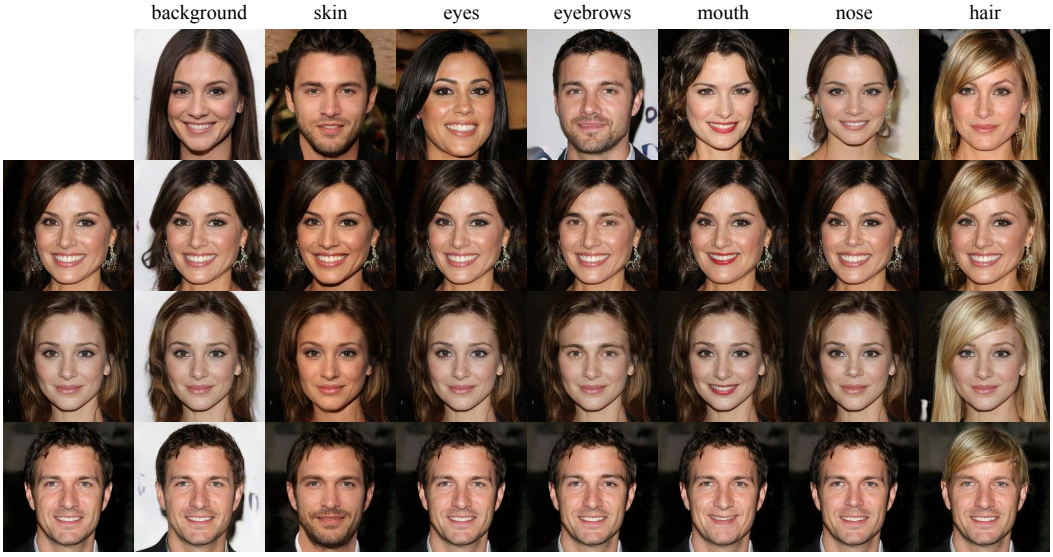


Figure 11. The result of style mixing performed at the semantic part level using MemoryGAN. Here, "skin" means the skin of the face. For the background, we swap the latent code for all the layers in M_l. For the eyes, eyebrows, mouth and nose, we swap the latent code for the last base layer, the shape layers and the texture layers in M_l. For hair and skin, we swap the latent code for the shape and the texture layers in M_l.

Interpolation Latent code interpolation reflects both the smoothness and disentanglement property of the latent space. We interpolated the main parts of the face image, and the results are shown in Figure 12. Similar to the results of style mixing, we have

empirically found that only interpolating the shape and the texture latent code for the small parts usually results in tiny changes to the image. To enhance the impact of the small part's latent code, we additionally interpolate the latent code on the last base layer. It can be seen from the result that for both the whole image and the semantic part, the interpolated image changes smoothly. This indicates that both the whole latent space and the sub-space for each part are smooth in MemoryGAN. Similar to the style-mixing result, we also found that (a) there is an entanglement between parts, such as hair, eyebrows and eyes and (b) the latent code for the eyes seems to mainly control the shape and is unable to control the color. The color is mainly controlled by larger parts such as the skin and the hair.

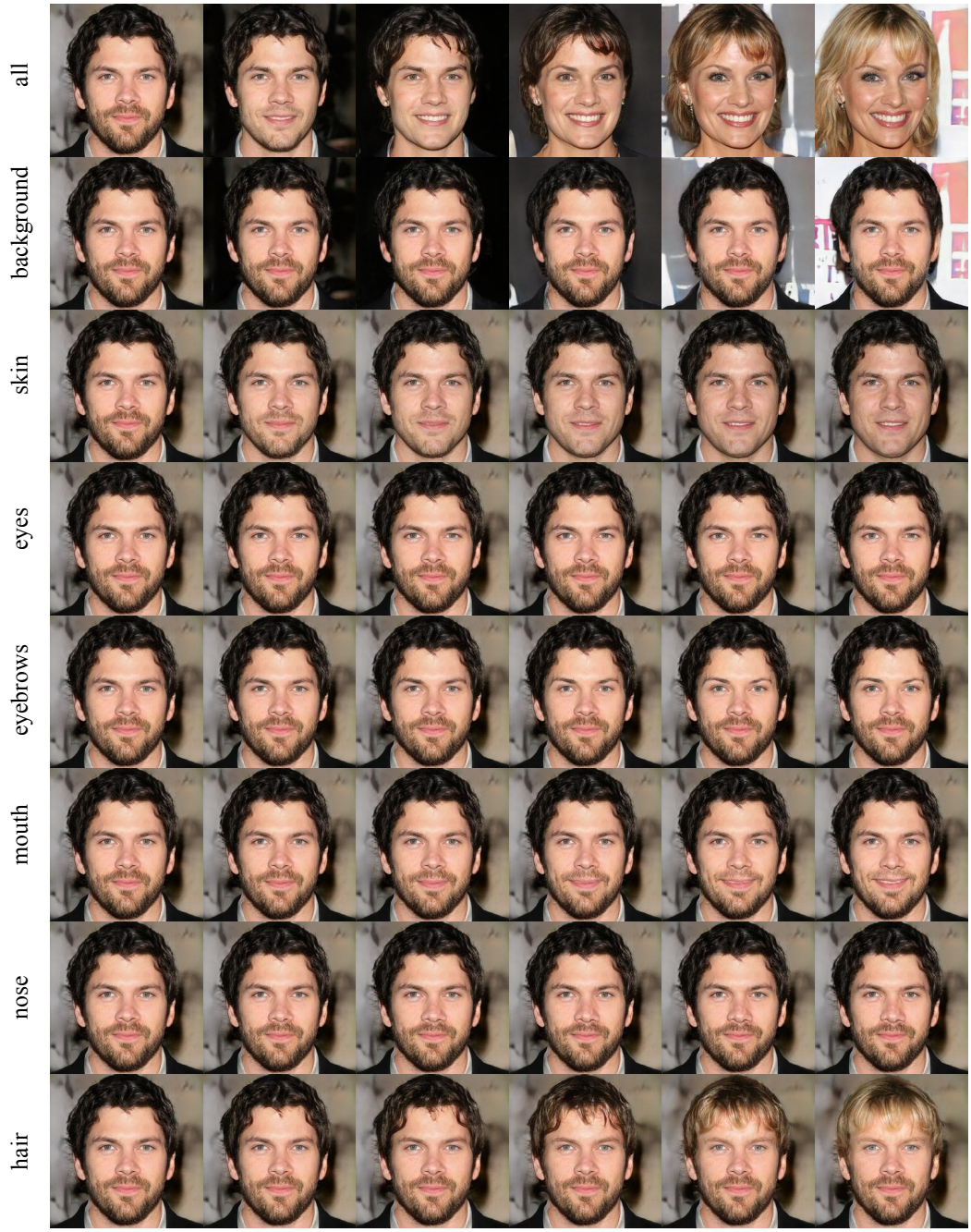


Figure 12. Latent code interpolation for MemoryGAN. Here "skin" means the skin of the face. For the background, we swap the latent code for all the layers in M_l. For skin and hair, we swap the latent code for the shape and the texture layers in M_l. For the eyes, eyebrows, mouth and nose, we swap the latent code for the last base layer, the shape layers and the texture layers in M_l.

Low-Dimensional Embedding As MemoryGAN uses a single generator for all the parts, the latent codes lie in the same semantic space. This allows us to analyze the relationship between the parts directly through the latent codes themselves. On the contrary, we cannot directly analyze the latent codes for SemanticStyleGAN, because each latent code is designed for a specific local generator. Therefore, the latent codes have to be analyzed with the generators together. For example, the same latent code can be applied to different local generators in SemanticStyleGAN, and this produces different semantic parts.

A straightforward analysis is to visualize the latent codes in a two-dimensional space. We use the t-SNE [54] algorithm for dimension reduction. The results are shown in Figure 13. It can be seen that in the query space, parts are already pretty well grouped. We can also observe that the parts generated in succession are more closely grouped, such as the skin and the eyes, the eyebrows, and the mouth. This reflects the fact that the LSTM module in M_w captures the dependence between successive parts. On the contrary, all the parts are well separately grouped in the latent space. This indicates that the latent space is flatter than the query space, and it is also more disentangled than the query space.

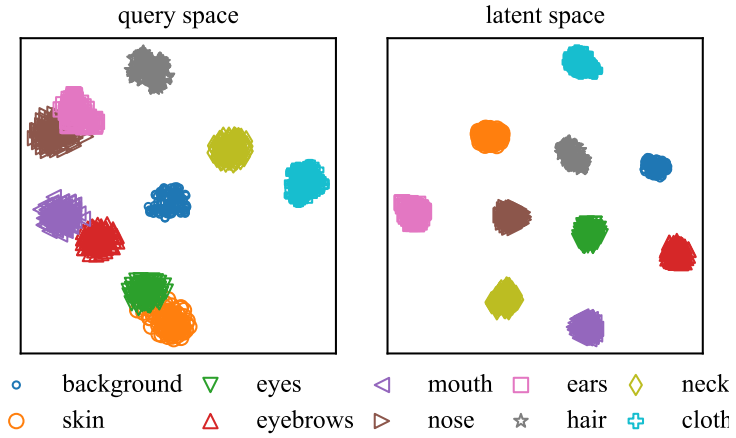


Figure 13. Two-dimensional embedding of the query and latent codes using the t-SNE [54] algorithm.

Correlation between Parts t-SNE is particularly suitable for encoding the local structure of data points. In other words, if two codes are similar enough, they should be grouped together. It tells less about two codes, which are not similar but may be correlated. So, we also calculate the correlation between parts by pairwise cosine similarity. The results are shown in Figure 14. We can see that the correlation between parts is much weaker in the query space than in the latent space. We hypnosize that this is because the roles of the query code and the latent code are different in MemoryGAN. The role of the query code is to tell the long-term memory what information is needed. So, the codes should be distinguishable, and thus, weaker correlation is helpful. On the other hand, the latent code is used to modulate the generator to produce the features. It is critical for the latent code to be informative. As correlation reflects the structure of the data, it is beneficial for the latent codes to be more strongly correlated. Now, we focus on the details of the correlation matrix. We can see from Figure 14 that in the query space, there is a relatively stronger correlation in the nearby parts. The order of the parts in the correlation matrix matches the order in which they are generated in the working memory. The result shows that our working memory captures the dependency of the consecutive parts successfully. Compared with the query space, there is a much stronger correlation in the latent space between facial parts, such as the correlation between the eyes, eyebrows, mouth and ears. As mentioned above, we believe this result indicates that the latent code encodes the structure of the face much better than the query codes.

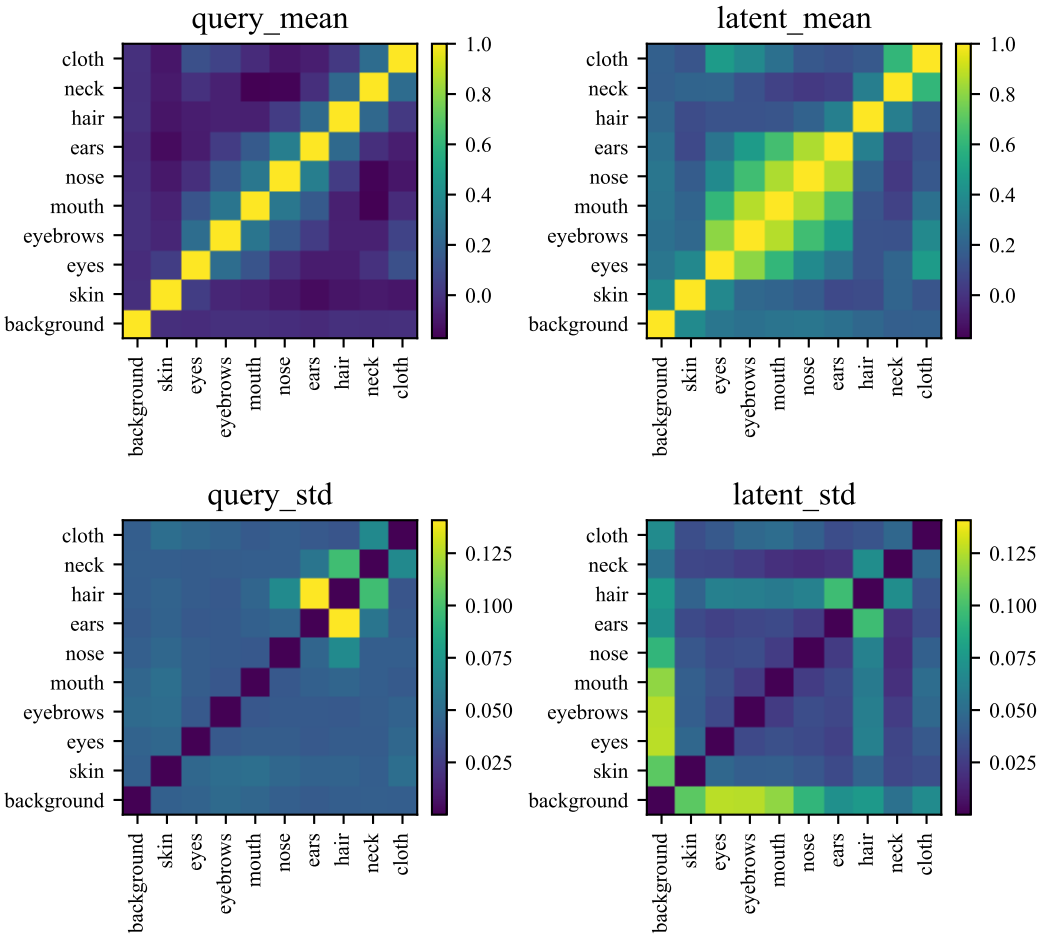


Figure 14. Correlation matrix between the parts. The top row includes the mean value of the correlation matrix; the left is for the query vector, and the right is for the latent code. The bottom row is the standard deviation of the correlation matrix; the left is for the query code, and the right is for the latent code. The value of the correlation matrix is calculated using cosine similarity. A total of 10,000 samples are used for the stochastic calculation.

4.6. Controlled Image Editing

4.6.1. Text Driving Image Editing

One advantage of the semantic part compositional model is that we can edit the attribute of one part individually with no or little affect on the attribute of other parts. Figure 15 shows the results of part attribute editing using the StyleCLIP [55] optimization method. Given a description of the image using natural language, StyleCLIP is able to modify the image to match the description. StyleCLIP uses three losses defined below for optimization:

$$\mathcal{L}_{CLIP} = D_{CLIP}(G(w)_{img}, t) + \lambda_{L2}\|w - w_s\|_2 + \lambda_{ID}\mathcal{L}_{ID}(w) \tag{16}$$

where w is the latent code for optimization, t is the text prompt, w_s is the source latent code, and G is the generator. As MemoryGAN generates both the image and segmentation mask, we use subscript *img* and *seg* to distinguish them. D_{CLIP} is the CLIP loss [56]. It measures the cosine distance between the CLIP embedding of the image and the text prompt. $\mathcal{L}_{ID}$ represents identity loss [57] and is defined below:

$$\mathcal{L}_{ID} = 1 - \langle F(G(w_s)_{img}), F(G(w)_{img}) \rangle \tag{17}$$

where F is a pre-trained ArcFace [58] network and $\langle \cdot, \cdot \rangle$ represents the cosine similarity between its arguments. In MemoryGAN, the parts are usually spatially separated. It is, therefore, possible that the optimized latent code for a part may change too much and thus affect nearby parts. To control this interference, we add a segmentation loss term, so that the final loss is:

$$\mathcal{L} = \mathcal{L}_{CLIP} + \lambda_{seg} \| G(w)_{seg} - G(w_s)_{seg} \|_2 \quad (18)$$

The result of attribute editing using StyleCLIP is shown in Figure 15. For skin, eyebrows, mouth, nose and hair, we only allow optimizing the corresponding part latent code, while the latent codes of other parts are fixed. The results show that StyleCLIP is able to edit the attribute of these parts individually with little effect on other parts. Similar to the results in Section 4.5, we find that we are unable to control the color of the eyes through the latent code of the eyes. To verify the previous conjecture that the color of the eyes may be strongly entangled with the colors of the skin and the hair, we only allow optimizing the latent code of the skin and hair for a text prompt that aims to change the color of eyes, such as "a person with blue eyes". The results show that we can achieve the desired eye color editing, and the colors of the skin and the hair change correspondingly. For "blue eyes", the skin tends to be "white" and the hair color tends to be "red" or "blonde". This results is consistent with our conjecture.

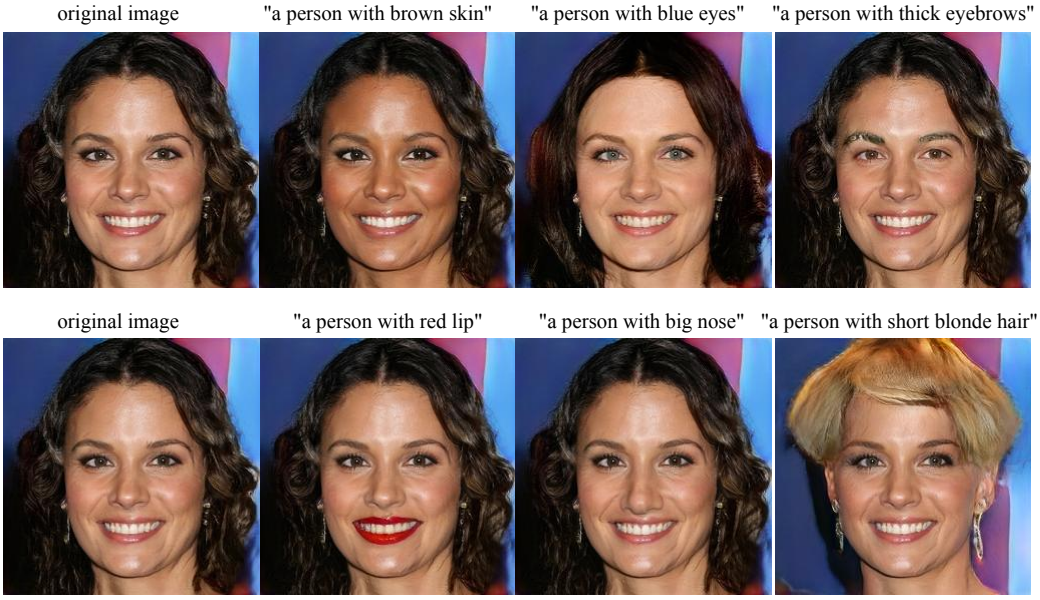


Figure 15. Edit the attribute of parts using StyleCLIP [55] on MemoryGAN. For skin, eyebrows, mouth, nose and hair, we only allow modifying the latent code of the corresponding part. For eyes, we only allow modifying the latent codes of hair and skin (see the text for the reason).

4.6.2. Localized Attribute Editing

One advantage of compositional image synthesis models is their ability to offer a higher level of control and editing over images. In this section, we compare our model with the seminal work of StyleGAN2 [9] in the context of localized attribute editing. We first invert the original image into the latent space of StyleGAN2 and MemoryGAN using e4e [59], and then, we utilize the optimization method of StyleCLIP [55] to edit the attributes. Each attribute uses the same text for editing, and we showcase the results after 300 optimization steps. Specifically, we use "a man with a beard" for beard attribute editing, "a man with gray hair" for hair attribute editing, "a smiling woman" for mouth attribute editing, and "a woman with a big nose" for nose attribute editing. The results are displayed in Figure 16. It is evident that our model achieves significantly better localization for all attributes, particularly for small parts such as the nose and mouth, in comparison to StyleGAN2. Our model demonstrates precise control over attribute editing, wherein

modifying one attribute of a semantic part has a small impact on other parts. These results highlight the fine-grained control that our model provides.

We have observed that StyleGAN2 tends to generate darker images when used in conjunction with StyleCLIP. We conjecture that this behavior can be attributed to the fact that StyleGAN2 employs a single latent code to generate the entire image, which makes it challenging for StyleCLIP to accurately identify the corresponding semantic meaning of a word (e.g., hair) in the latent space. The blind search in the latent space may lead to an erroneous penalization of image appearance, resulting in darker outputs. In contrast, our model utilizes a set of latent codes to modulate the generator, with each latent code corresponding to a specific semantic meaning. Consequently, it becomes much easier for StyleCLIP to identify the corresponding latent code for a given word, leading to improved results.

One way to quantitatively assess the localization property is by calculating the percentage of preserved pixels required to achieve a certain amount of reduction in CLIP loss [56]. In this evaluation, we utilize L_1 loss between the inverted image and the edited image to measure the difference between the images. Additionally, we use the complement of the L_1 loss to determine the percentage of preserved pixels after editing. The results are displayed in Figure 17, where the term "score gain" refers to the reduction in CLIP loss. Notably, our model consistently achieves a significantly higher percentage of preserved pixels, particularly for editing small semantic parts, which aligns with previous findings. Although StyleGAN2 tends to yield lower CLIP loss, it does so at the expense of affecting unrelated areas in the images, as demonstrated in Figure 16.

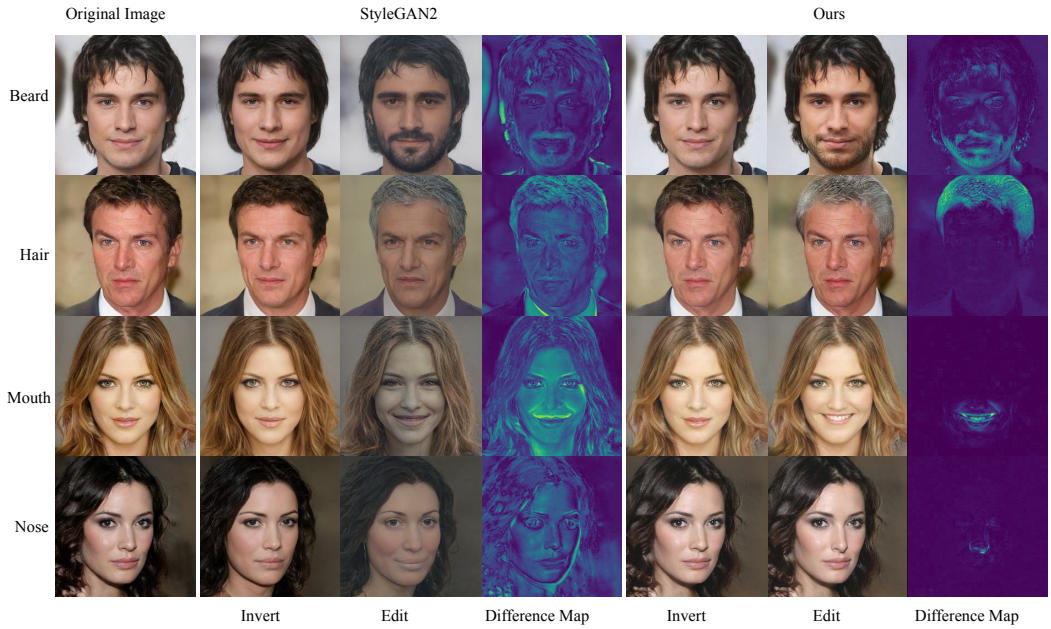


Figure 16. Results of localized attribute editing for StyleGAN2 [9] and our model. We first invert the input image into latent space using e4e [59]; then, we edit the attribute using StyleCLIP [55]. For each attribute, we show the original image, the inverted image, the edited image and the difference map.

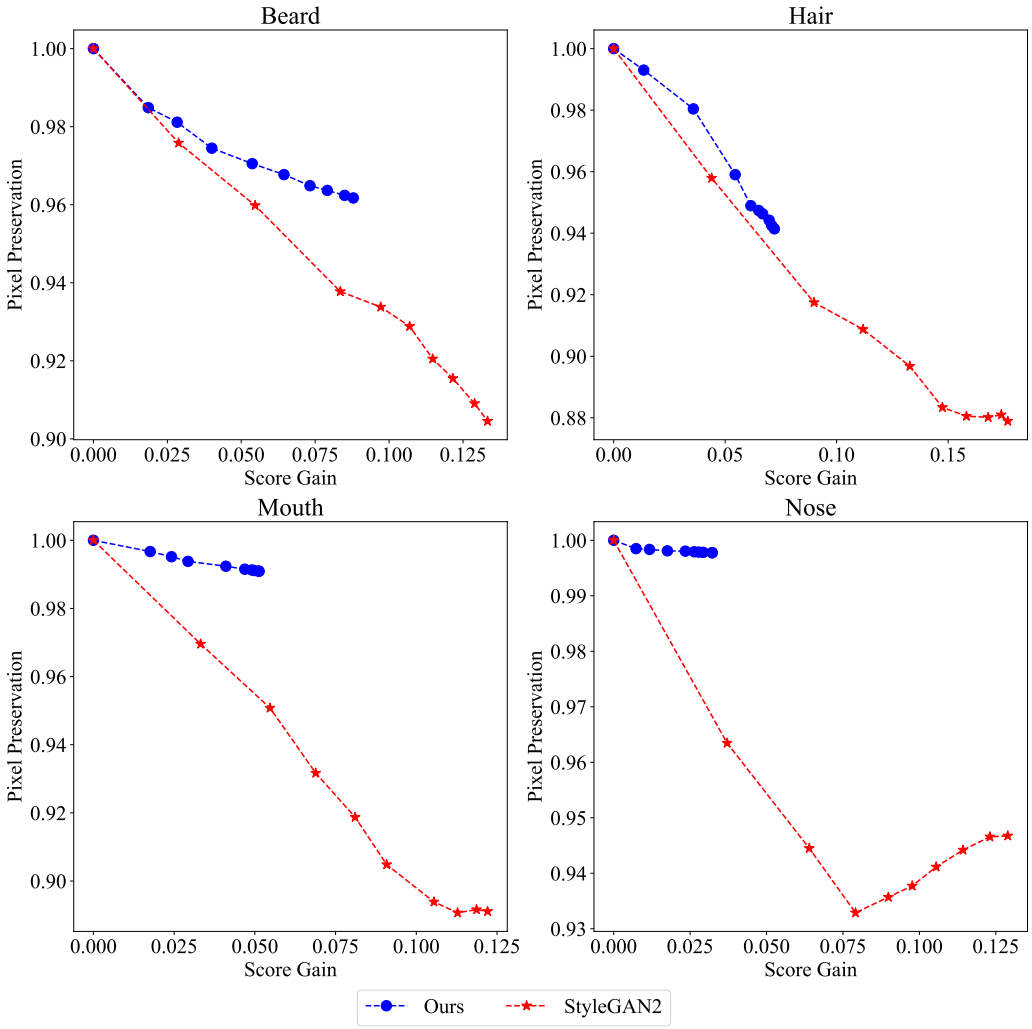


Figure 17. Quantitative comparison of localized attribute editing between StyleGAN2 and our model.

4.6.3. Comparison with Layout-to-Image Model

Another research direction that explores semantic control is layout-to-image generative models, which map layouts to corresponding images. In this section, we have chosen the representative model SEAN [10] for comparison. As discussed in Section 1, the primary limitation of these models is their reliance on input layouts and inability to generate images from scratch. Another consequence of this limitation is that these models can only automatically transfer the texture of the image and require manual intervention to modify the shape. To validate this observation, we conducted style-mixing experiments for both SEAN and our model, and the results are presented in Figure 18. We selected six images with diverse hair styles as source images and aimed to transfer these styles to a target image. With SEAN, style mixing can be constrained to a specific region (such as the hair region), allowing the transfer of hair texture to the target image. However, due to the shape being controlled by the input layout, SEAN cannot transfer the hair shape of the source image to the target image. In contrast, our model offers the flexibility to transfer hair styles either for the texture only or incorporate both shape and texture. By enabling the swapping of either just the texture code or both the shape and texture codes, we can achieve texture-level or shape-and-texture-level hair style transfers. These results clearly highlight the advantages of compositional-aware models.

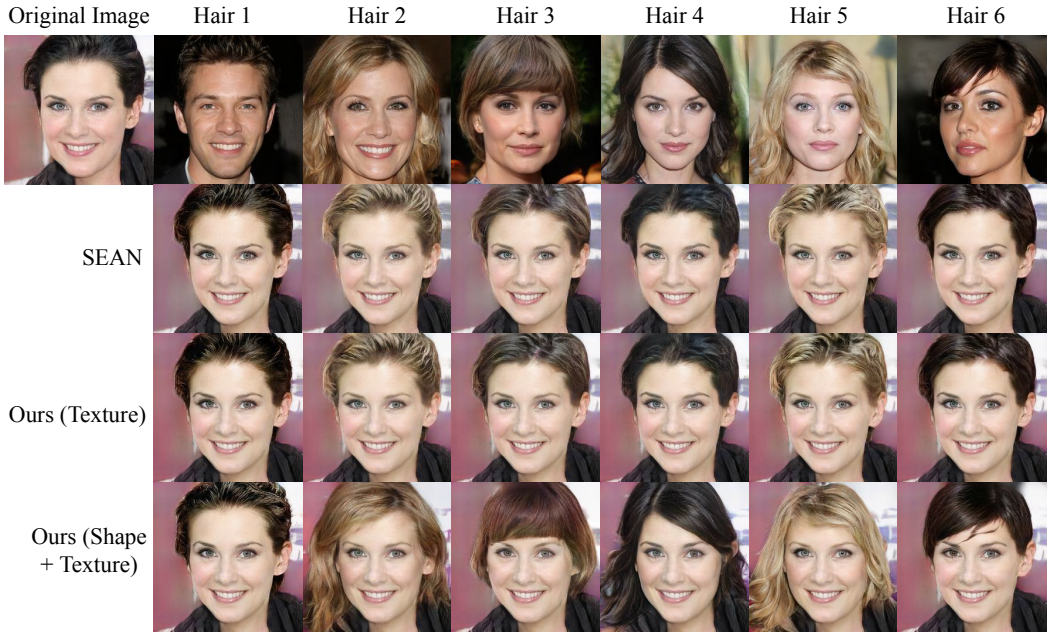


Figure 18. Comparison between SEAN and our model regarding controllability of attribute editing.

5. Conclusions

Inspired by humans' structured memory system, we present MemoryGAN, which is a compositional generative model for image synthesis. MemoryGAN consists of a long-term memory and a working memory. The long-term memory is a shared generator responsible for storing heterogeneous information about the parts. The working memory utilizes LSTM to model the dependency between the parts and generate query codes for the long-term memory. The retrieved part features are then combined in the working memory to produce the complete image. Despite employing a single generator for all parts, MemoryGAN achieves comparable image synthesis performance and compositional ability to state-of-the-art models that utilize multiple generators. The advantages of MemoryGAN are as follows: First, as a compositional model, it provides a higher level of control over the generation and image editing processes. Second, MemoryGAN utilizes a shared generator for all semantic parts, making the representation efficient and the model scalable. Third, as a memory system, its information retrieval mechanism is general, making it a suitable base framework for knowledge transformation. We believe that our approach, utilizing the generator of GAN as a memory model, will lead to future advancements in bio-friendly models and memory-augmented models. Furthermore, we anticipate that this memory framework will pave the way for addressing challenges in large-scale compositional image generation.

While our model achieves promising results in compositional image synthesis, there are still several challenges that need to be addressed in future work. First, MemoryGAN only reduces the parameter requirements for the semantic parts, while the computational complexity remains the same as SemanticStyleGAN. A promising solution to this issue is to consider taxonomy and use a hierarchy of latent codes to modulate the shared generator, thereby significantly reducing both the parameters and computational complexity. Second, MemoryGAN does not take into account the scale of the semantic parts, resulting in the need to process with the entire image resolution to generate a small part feature, which leads to unnecessary computation. This issue could potentially be resolved by introducing a hard attention mechanism. Last, MemoryGAN requires full supervision for the semantic parts. A valuable direction for future research would be to explore semi-supervised approaches.

Author Contributions: Conceptualization, Z.W., J.P. and Z.L.; methodology, Z.W.; software, Z.W.; validation, Z.W., J.P. and Z.L.; formal analysis, Z.W.; investigation, Z.W. and J.P.; resources, Z.L.; data curation, Z.W.; writing—original draft preparation, Z.W.; writing—review and editing, Z.W., J.P. and Z.L.; visualization, Z.W.; supervision, J.P. and Z.L.; project administration, Z.L.; funding acquisition, Z.L. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded in part by the National Natural Science Foundation of China grant number 62032019 and 61732019.

Data Availability Statement: The code is available at https://gitee.com/swu-wzt/memgan-pytorch (accessed on 29 June 2023).

Acknowledgments: We sincerely thank the anonymous reviewers for their insightful comments and suggestions, which greatly enhanced the quality of this research. We would also like to acknowledge the GPU support from the Center for Research and Innovation in Software Engineering (RISE).

Conflicts of Interest: The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript; or in the decision to publish the results.

Abbreviations

The following abbreviations are used in this manuscript:

IS	Inception Score
FID	Fréchet Inception Distance
MLP	Multi-Layer Perceptron
SMSE	Sequential Mean Square Error
LSTM	Long Short-Term Memory
GAN	Generative Adversarial Network
MACs	Multiply–Accumulate operations

References

1. Tian, C.; Zheng, M.; Zuo, W.; Zhang, B.; Zhang, Y.; Zhang, D. Multi-Stage Image Denoising with the Wavelet Transform. *Pattern Recognit.* **2023**, *134*, 109050. [CrossRef]
2. Tian, C.; Zhang, Y.; Zuo, W.; Lin, C.W.; Zhang, D.; Yuan, Y. A Heterogeneous Group CNN for Image Super-Resolution. *IEEE Trans. Neural Networks Learn. Syst.* **2022**, 1–13. [CrossRef] [PubMed]
3. Zhang, Q.; Xiao, J.; Tian, C.; Lin, J.C.-W.; Zhang, S. A Robust Deformed Convolutional Neural Network (CNN) for Image Denoising. *CAAI Trans. Intell. Technol.* **2022**, *8*, 331–342. [CrossRef]
4. Goodfellow, I.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.; Bengio, Y. Generative Adversarial Nets. In *NIPS*; Curran Associates, Inc.: Red Hook, NY, USA, 2014; Volume 27.
5. Kingma, D.P.; Welling, M. Auto-Encoding Variational Bayes. *arXiv* **2022**, arXiv:1312.6114.
6. van den Oord, A.; Kalchbrenner, N.; Kavukcuoglu, K. Pixel Recurrent Neural Networks. In Proceedings of the International Conference on Machine Learning, New York, NY, USA, 19–24 June 2016; Volume 48, pp. 1747–1756.
7. Tian, C.; Zhang, X.; Lin, J.C.W.; Zuo, W.; Zhang, Y.; Lin, C.W. Generative Adversarial Networks for Image Super-Resolution: A Survey. *arXiv* **2022**, arXiv:2204.13620.
8. Goyal, A.; Bengio, Y. Inductive biases for deep learning of higher-level cognition. *Proc. R. Soc. A Math. Phys. Eng. Sci.* **2022**, *478*, 20210068. [CrossRef]
9. Karras, T.; Laine, S.; Aittala, M.; Hellsten, J.; Lehtinen, J.; Aila, T. Analyzing and Improving the Image Quality of StyleGAN. In Proceedings of the Conference on Computer Vision and Pattern Recognition, Seattle, WA, USA, 13–19 June 2020; pp. 8110–8119.
10. Zhu, P.; Abdal, R.; Qin, Y.; Wonka, P. SEAN: Image Synthesis With Semantic Region-Adaptive Normalization. In Proceedings of the Conference on Computer Vision and Pattern Recognition, Seattle, WA, USA, 13–19 June 2020; pp. 5104–5113.
11. Shi, Y.; Yang, X.; Wan, Y.; Shen, X. SemanticStyleGAN: Learning Compositional Generative Priors for Controllable Image Synthesis and Editing. In Proceedings of the Conference on Computer Vision and Pattern Recognition, New Orleans, LA, USA, 18–24 June 2022; pp. 11254–11264.
12. Atkinson, R.C.; Shiffrin, R.M. Human memory: A proposed system and its control processes. In *Psychology of Learning and Motivation*; Elsevier: Amsterdam, The Netherlands, 1968; Volume 2, pp. 89–195.
13. Lake, B.M.; Ullman, T.D.; Tenenbaum, J.B.; Gershman, S.J. Building Machines That Learn and Think Like People. *Behav. Brain Sci.* **2017**, *40*, e253. [CrossRef]
14. Karras, T.; Laine, S.; Aila, T. A Style-Based Generator Architecture for Generative Adversarial Networks. In Proceedings of the Conference on Computer Vision and Pattern Recognition, Long Beach, CA, USA, 15–20 June 2019; pp. 4401–4410.

15. Shen, Y.; Gu, J.; Tang, X.; Zhou, B. Interpreting the Latent Space of GANs for Semantic Face Editing. In Proceedings of the Conference on Computer Vision and Pattern Recognition, Seattle, WA, USA, 13–19 June 2020; pp. 9243–9252.
16. Tritrong, N.; Rewatbowornwong, P.; Suwajanakorn, S. Repurposing GANs for One-Shot Semantic Part Segmentation. In Proceedings of the Conference on Computer Vision and Pattern Recognition, Nashville, TN, USA, 20–25 June 2021; pp. 4475–4485.
17. Li, D.; Yang, J.; Kreis, K.; Torralba, A.; Fidler, S. Semantic Segmentation With Generative Models: Semi-Supervised Learning and Strong Out-of-Domain Generalization. In Proceedings of the Conference on Computer Vision and Pattern Recognition, Nashville, TN, USA, 20–25 June 2021; pp. 8300–8311.
18. Karras, T.; Aittala, M.; Laine, S.; Härkönen, E.; Hellsten, J.; Lehtinen, J.; Aila, T. Alias-Free Generative Adversarial Networks. *Adv. Neural Inf. Process. Syst.* **2021**, *34*, 852–863.
19. Singh, K.K.; Ojha, U.; Lee, Y.J. FineGAN: Unsupervised Hierarchical Disentanglement for Fine-Grained Object Generation and Discovery. In Proceedings of the Conference on Computer Vision and Pattern Recognition, Long Beach, CA, USA, 15–20 June 2019; pp. 6490–6499.
20. Ojha, U.; Singh, K.K.; Lee, Y.J. Generating Furry Cars: Disentangling Object Shape and Appearance across Multiple Domains. In Proceedings of the International Conference on Learning Representations, Virtual Event, Austria, 3–7 May 2021.
21. Kwak, H.; Zhang, B.T. Generating Images Part by Part with Composite Generative Adversarial Networks. *arXiv* **2016**, arXiv:1607.05387.
22. Mildenhall, B.; Srinivasan, P.P.; Tancik, M.; Barron, J.T.; Ramamoorthi, R.; Ng, R. NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis. In Proceedings of the European Conference on Computer Vision, Glasgow, UK, 23–28 August 2020; Vedaldi, A., Bischof, H., Brox, T., Frahm, J.M., Eds.; Lecture Notes in Computer Science; Springer: Cham, Switzerland, 2020; pp. 405–421. [CrossRef]
23. Schwarz, K.; Liao, Y.; Niemeyer, M.; Geiger, A. GRAF: Generative Radiance Fields for 3D-Aware Image Synthesis. In Proceedings of the Thirty-Fourth Annual Conference on Neural Information Processing Systems, Vancouver, BC, Canada, 6–12 December 2020; Volume 33, pp. 20154–20166.
24. Niemeyer, M.; Geiger, A. GIRAFFE: Representing Scenes As Compositional Generative Neural Feature Fields. In Proceedings of the Conference on Computer Vision and Pattern Recognition, Nashville, TN, USA, 20–25 June 2021; pp. 11453–11464.
25. Mnih, V.; Heess, N.; Graves, A.; Kavukcuoglu, K. Recurrent models of visual attention. In Proceedings of the Twenty-Eighth Conference on Neural Information Processing Systems, Montreal, QC, Canada, 8–13 December 2014; Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N., Weinberger, K., Eds.; Curran Associates, Inc.: Red Hook, NY, USA, 2014; Volume 27.
26. Gregor, K.; Danihelka, I.; Graves, A.; Rezende, D.; Wierstra, D. DRAW: A Recurrent Neural Network For Image Generation. In Proceedings of the 32nd International Conference on Machine Learning, Lille, France, 6–11 July 2015; Volume 37, pp. 1462–1471.
27. Gregor, K.; Besse, F.; Jimenez Rezende, D.; Danihelka, I.; Wierstra, D. Towards Conceptual Compression. In Proceedings of the Thirtieth Annual Conference on Neural Information Processing Systems, Barcelona, Spain, 5–10 December 2016; Curran Associates, Inc.: Red Hook, NY, USA, 2016; Volume 29.
28. Eslami, S.M.A.; Heess, N.; Weber, T.; Tassa, Y.; Szepesvari, D.; Kavukcuoglu, K.; Hinton, G.E. Attend, Infer, Repeat: Fast Scene Understanding with Generative Models. In Proceedings of the Thirtieth Annual Conference on Neural Information Processing Systems, Barcelona, Spain, 5–10 December 2016; Curran Associates, Inc.: Red Hook, NY, USA, 2016; Volume 29.
29. Kosiorek, A.; Kim, H.; Teh, Y.W.; Posner, I. Sequential Attend, Infer, Repeat: Generative Modelling of Moving Objects. In Proceedings of the Thirty-Second Conference on Neural Information Processing Systems, Montréal, QC, Canada, 3–8 December 2018; Curran Associates, Inc.: Red Hook, NY, USA, 2018; Volume 31.
30. Greff, K.; Kaufman, R.L.; Kabra, R.; Watters, N.; Burgess, C.; Zoran, D.; Matthey, L.; Botvinick, M.; Lerchner, A. Multi-Object Representation Learning with Iterative Variational Inference. In Proceedings of the International Conference on Machine Learning, Long Beach, CA, USA, 9–15 June 2019; pp. 2424–2433.
31. Burgess, C.P.; Matthey, L.; Watters, N.; Kabra, R.; Higgins, I.; Botvinick, M.; Lerchner, A. MONet: Unsupervised Scene Decomposition and Representation. *arXiv* **2019**, arXiv:1901.11390.
32. Salimans, T.; Karpathy, A.; Chen, X.; Kingma, D.P. PixelCNN++: Improving the PixelCNN with Discretized Logistic Mixture Likelihood and Other Modifications. *arXiv* **2017**, arXiv:1701.05517.
33. Xue, Y.; Tong, W.; Neri, F.; Zhang, Y. PEGANs: Phased Evolutionary Generative Adversarial Networks with Self-Attention Module. *Mathematics* **2022**, *10*, 2792. [CrossRef]
34. Marino, J.; Yue, Y.; Mandt, S. Iterative Amortized Inference. In Proceedings of the 35th International Conference on Machine Learning, Stockholm, Sweden, 10–15 July 2018; pp. 3403–3412.
35. Hopfield, J.J. Neural Networks and Physical Systems With Emergent Collective Computational Abilities. *Proc. Natl. Acad. Sci. USA* **1982**, *79*, 2554–2558. [CrossRef]
36. Kanerva, P. *Sparse Distributed Memory*; MIT Press: Cambridge, MA, USA, 1988.
37. Weston, J.; Chopra, S.; Bordes, A. Memory Networks. *arXiv* **2015**, arXiv:1410.3916.
38. Sukhbaatar, S.; szlam, a.; Weston, J.; Fergus, R. End-To-End Memory Networks. In Proceedings of the Conference on Neural Information Processing Systems, Montreal, QC, Canada, 7–12 December 2015; Cortes, C., Lawrence, N.D., Lee, D.D., Sugiyama, M., Garnett, R., Eds.; Curran Associates, Inc.: Red Hook, NY, USA, 2015; pp. 2440–2448.
39. Graves, A.; Wayne, G.; Danihelka, I. Neural Turing Machines. *arXiv* **2014**, arXiv:1410.5401.

40. Graves, A.; Wayne, G.; Reynolds, M.; Harley, T.; Danihelka, I.; Grabska-Barwińska, A.; Gómez, S.; Grefenstette, E.; Ramalho, T.; Agapiou, J.; et al. Hybrid Computing Using a Neural Network With Dynamic External Memory. *Nature* **2016**, *538*, 471–476. [CrossRef]
41. Goodfellow, I.; Bengio, Y.; Courville, A. *Deep Learning*; MIT Press: Cambridge, MA, USA, 2016; pp. 378–393.
42. Hochreiter, S.; Schmidhuber, J. Long Short-Term Memory. *Neural Comput.* **1997**, *9*, 1735–1780. [CrossRef] [PubMed]
43. Wu, C.; Herranz, L.; Liu, X.; Wang, Y.; van de Weijer, J.; Raducanu, B. Memory Replay GANs: Learning to Generate New Categories without Forgetting. In Proceedings of the Conference on Neural Information Processing Systems, Montreal, QC, Canada, 2–8 December 2018; Curran Associates, Inc.: Red Hook, NY, USA, 2018; Volume 31.
44. Cong, Y.; Zhao, M.; Li, J.; Wang, S.; Carin, L. GAN Memory with No Forgetting. In Proceedings of the Conference on Neural Information Processing Systems, Virtual, 6–12 December 2018; Curran Associates, Inc.: Red Hook, NY, USA, 2020; Volume 33, pp. 16481–16494.
45. Robins, A. Catastrophic Forgetting, Rehearsal and Pseudorehearsal. *Connect. Sci.* **1995**, *7*, 123–146. [CrossRef]
46. McCloskey, M.; Cohen, N.J. Catastrophic Interference in Connectionist Networks: The Sequential Learning Problem. In *Psychology of Learning and Motivation*; Bower, G.H., Ed.; Academic Press: Amsterdam, The Netherlands, 1989; Volume 24, pp. 109–165. [CrossRef]
47. Baddeley, A. Working Memory. *Science* **1992**, *255*, 556–559. [CrossRef]
48. Mescheder, L.; Geiger, A.; Nowozin, S. Which Training Methods for GANs Do Actually Converge? *arXiv* **2018**, arXiv:1801.04406.
49. Lee, C.H.; Liu, Z.; Wu, L.; Luo, P. MaskGAN: Towards Diverse and Interactive Facial Image Manipulation. *arXiv* **2020**, arXiv:1907.11922.
50. Kingma, D.P.; Ba, J. Adam: A Method for Stochastic Optimization. *arXiv* **2017**, arXiv:1412.6980.
51. Heusel, M.; Ramsauer, H.; Unterthiner, T.; Nessler, B.; Hochreiter, S. GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium. In Proceedings of the Conference on Neural Information Processing Systems, Long Beach, CA, USA, 4–9 December 2017; Curran Associates, Inc.: Red Hook, NY, USA, 2017; Volume 30.
52. Salimans, T.; Goodfellow, I.; Zaremba, W.; Cheung, V.; Radford, A.; Chen, X.; Chen, X. Improved Techniques for Training GANs. In Proceedings of the Conference on Neural Information Processing Systems, Barcelona, Spain, 5–10 December 2016; Curran Associates, Inc.: Red Hook, NY, USA, 2016; Volume 29.
53. Sulem, P.; Gudbjartsson, D.F.; Stacey, S.N.; Helgason, A.; Rafnar, T.; Magnusson, K.P.; Manolescu, A.; Karason, A.; Palsson, A.; Thorleifsson, G. Genetic determinants of hair, eye and skin pigmentation in Europeans. *Nat. Genet.* **2007**, *39*, 1443–1452. [CrossRef] [PubMed]
54. Van der Maaten, L.; Hinton, G. Visualizing data using t-SNE. *JMLR* **2008**, *9*, 2579–2605.
55. Patashnik, O.; Wu, Z.; Shechtman, E.; Cohen-Or, D.; Lischinski, D. StyleCLIP: Text-Driven Manipulation of StyleGAN Imagery. In Proceedings of the International Conference on Computer Vision, Montreal, QC, Canada, 10–17 October 2021; pp. 2085–2094.
56. Radford, A.; Kim, J.W.; Hallacy, C.; Ramesh, A.; Goh, G.; Agarwal, S.; Sastry, G.; Askell, A.; Mishkin, P.; Clark, J.; et al. Learning Transferable Visual Models From Natural Language Supervision. In Proceedings of the International Conference on Machine Learning, Virtual Event, 18–24 July 2021; pp. 8748–8763, ISSN: 2640-3498.
57. Richardson, E.; Alaluf, Y.; Patashnik, O.; Nitzan, Y.; Azar, Y.; Shapiro, S.; Cohen-Or, D. Encoding in Style: A StyleGAN Encoder for Image-to-Image Translation. In Proceedings of the Conference on Computer Vision and Pattern Recognition, Nashville, TN, USA, 20–25 June 2021; pp. 2287–2296.
58. Deng, J.; Guo, J.; Yang, J.; Xue, N.; Kotsia, I.; Zafeiriou, S. ArcFace: Additive Angular Margin Loss for Deep Face Recognition. *IEEE Trans. Pattern Anal. Mach. Intell.* **2022**, *44*, 5962–5979. [CrossRef] [PubMed]
59. Tov, O.; Alaluf, Y.; Nitzan, Y.; Patashnik, O.; Cohen-Or, D. Designing an encoder for StyleGAN image manipulation. *ACM Trans. Graph.* **2021**, *40*, 133:1–133:14. [CrossRef]