Preference-Based Image Generation

Hadi Kazemi
West Virginia University
hakazemi@mix.wvu.edu

Fariborz Taherkhani
West Virginia University
fariborztaherkhani@gmail.com

Nasser M. Nasrabadi
West Virginia University
nasser.nasrabadi@mail.wvu.edu

Abstract

Deep generative models are a set of promising methods, that are able to model complex data and generate new samples. In principle, they learn to map a random latent code sampled from a prior distribution into a high dimensional data space, such as image space. However, these models have limited utilities as the user has minimal control over what the network produces. Despite the success of some recent work in learning an interpretable latent code, the field still lacks a coherent framework to learn a fully interpretable latent code, without any random part for sample diversity. Consequently, it is generally hard, if not impossible, for a non-expert user to produce a desired image by tuning the random and interpretable parts of the latent code. In this paper, we introduce the Preference-Based Image Generation (PbIG), a new method to retrieve the corresponding latent code of the user's mental image. We propose to adopt preference-based reinforcement learning, which learns from a user's judgment of the generated images by a pre-trained generative model. Since the proposed method is completely decoupled from the training stage of the underlying generative models, it can easily be adopted by any method, such as GANs and VAEs. We evaluate the effectiveness of PbIG framework using a set of experiments on baseline datasets using a pretraind StackGAN++.

1. Introduction

Building generative models, that are able to produce new samples of high-dimensional data distributions, is a fundamental problem in many computer vision applications, such as face generation [17], image-to-image translation [14, 19], image editing [5, 18], domain adaptation [47], and image in-painting [41]. Currently, the most prominent approaches are the Generative Adversarial Networks (GAN) [12], Variational Autoencoders (VAE) [21], and Auto-Regressive Generative Models [28]. These models capture the joint distribution between the data and a set of hidden variables, called latent codes, which represent different variations of the training data. The trained models then generate new samples, given random latent codes, which are sampled from their prior distributions. However, in an unconditional setting, these models lack an inference mechanism to find the corresponding latent representation of a mental image (a desired image which user has in mind). Consequently, prior works conditioned these models on additional information to direct the data generation process. The conditioning could be on another image for image-to-image translation, part of an image for in-painting, some desired data attributes [39], or even class labels [24].

Even though the generative models with their conditional settings sidestep the common problem of random sampling to some extent, they still need a random code to generate diverse samples. To learn meaningful latent representations, MMD-VAE was proposed in [45] which maximizes the mutual information between the input and the latent code. InfoGAN [2], an information-theoretic extension of GAN, allows learning representation which is partially interpretable. Graph-based methods have been also proposed in [33, 32] for semi-supervised learning. The resulting code then consists of a meaningful part corresponding to specific semantic attributes of the data, and a random part which injects diversity among the generated samples. In contrast, two concurrent independent works [8, 6] proposed a full inference of the random code. They have demonstrated that these codes can learn semantic attributes of the data. Several other papers have also investigated supervised representation learning by conditioning the discriminator on certain desired attributes [22, 26]. Transferring attributes between images has also been studied in the literature [5, 13]. Despite all the effort, it remains unclear how non-expert users can exploit the power of generative models to produce their mental images. Moreover, some attributes of the data cannot be encoded explicitly into a user-sensible code. [46] is the colsest work in the literature in which the user is able to genrate or edit an image using a set of computer paiting tools. However, generating the exact mental image is quite hard with this technique. However, this method could be an early step to initialize the latent code or be combined with our method as

978-1-7281-6553-0/20/$31.00 ©2020 IEEE

a condition.

To overcome these limitations, we propose a universal framework that enables users to produce arbitrary images in a sequential fashion. At each step, the users are required to compare a pair of generated images by a pre-trained generative model and select their preferences. The proposed framework then learns the corresponding latent representation of users' mental images. In essence, this framework fits within the Preference Learning (PL) paradigm [10]. Roughly speaking, preference learning is about modeling the preference using a set of instances, which are associated with an order relation. In recent past, extensive research has been conducted to address this problem proposing different techniques to learn from human judgments [1, 11, 36, 38, 30]. After the success of Reinforcement Learning (RL) in many applications [16], they are extended to a new paradigm, namely Preference-based Reinforcement Learning (PbRL), which enables RL algorithms to learn from preferences rather than absolute reward values [37]. Typically, they first learn a preference function using a set of training examples, and subsequently, a policy is learned to make actions which minimize the number of mis-ranked pairs. A series of experiments are conducted in [3, 23], involving actual human feedback, to play video games or perform robotics tasks. In [43], a new technique, called dueling bandit, is proposed which uses preferences in an interactive, but non-sequential setting. One notable difference is that they deal with a sequence of actions in a relatively low dimensional space, while we need to learn a single high dimensional action.

Our framework, which is developed based on PbRL, jointly trains a reward network, which fits a reward function to a user's preferences, and a trainable latent code that generates the user's mental image. More directly, the latent code optimizes the underlying function of the reward network. Compared to prior work, our key contribution is proposing a novel framework, called Preference-Based Image Generation (PbIG), that can find the corresponding random code of a desired image, for a given generative model, through a sequence of preference judgments by the user. We develop a training strategy, and the required considerations for its success, to retrieve a latent code on a relatively small number of judgments. It takes between 5-15 minutes in average for a non-expert user to generate a mental image using PbIG, which is way less than a couple of hours that they need to edit/generate an image in many applications such as sketch-to-photo synthesis in law enforcement or art, interior design (like bedroom/kitchen datasets), and exterior designs (tower dataset).

To the best of our knowledge, this work is the first to leverage the power of preference-based learning in predicting the random codes of generative models to produce user-desired images. Since PbIG is completely decoupled from the training stage of the generative models, it can be adopted by any pre-trained generative models with no more effort. Note that conditioning on some human-sensible attributes can also benefit PbIG, by limiting the latent code search space. In this work, we evaluate the PbIG framework using multiple baseline datasets. We employed StackGAN++ [44] to conduct our experiments, as it can efficiently generate more realistic and diverse images in different domains compared to other generative models. However, the whole framework is identical for any other generative models. Finally, we utilize Model-Agnostic Meta-Learning (MAML) [9] to reduce the number of comparison required by the user.

2. Preliminaries

In this section, we provide some rudiments of GANs and PbRL, necessary to understand the proposed preference-based image generation framework.

2.1. Generative Adversarial Networks (GANs)

GANs [12] are a group of generative models which learn the statistical distribution of the training data, allowing us to synthesize data samples by mapping a random noise z to an output image y: $G(z) : z \longrightarrow y$, where G is the generator network. GAN in its conditional setting (cGAN) is proposed in [15] which learns a mapping from an input x and a random noise z to the output image y: $G(x, z) : \{x, z\} \longrightarrow y$, using an autoencoder network. The generator model $G(x, z)$, is trained to generate an image which is not distinguishable from "real" samples by a discriminator network, D. Simultaneously, the discriminator is learning, adversarially, to discriminate between the "fake" generated images by the generator and the real samples from the train dataset. The objective function of GAN is given by:

$$l_{GAN}(G, D) = \mathbf{E}_{x,y\sim p_{data}}[\log D(x, y)] + \mathbf{E}_{x,z\sim p_z}[\log(1 - D(x, G(x, z)))], \quad (1)$$

where G attempts to minimize it and D tries to maximize it. Since the adversarial loss is not enough to guarantee that the trained network generates the desired output, one may add an extra Euclidean distance term to the objective function to generate images which are near the ground truth. Consequently, the final objective is defined as follows:

$$G^* = \arg\min_G \max_D l_{GAN}(G, D) + \lambda l_{L1}(G), \quad (2)$$

where $l_{L1}(G) = \| \ y - G(x, z) \ \|_1$ and λ is a weighting factor.

2.2. Preference-Based Reinforcement Learning

In standard reinforcement learning setup, an agent interacts with an environment E. At each timestep the agent

receives an observation $o \in O$ from the environment, and takes an action $a \in A$ based on a policy, $\pi : O \rightarrow A$, which maps states to a probability distribution over the actions. Subsequently, the environment assign a scalar reward $r(o, a) \in R$ to the taken action. The goal in reinforcement learning is to learn an optimal policy, which maximizes the discounted sum of rewards through all the steps. However, in PbRL the environment does not supply any rewards to the agent. In contrast, the agent's actions are evaluated by a human user, and a label is provided to the agent in terms of preferences between pairs of actions. In this context, we write $o_1 > o_2$ indicating the observation o_1 is preferred, by the user, to the observation o_2.

In PbRL, the policy and reward estimators are implemented as two neural networks which are updated in the following steps: first, the agent takes an action, a, based on the policy network π, i.e., $a = \pi(o)$. The parameters of π are optimized to maximize the sum of estimated rewards $\hat{r} = N_R(o, a)$ where N_R is the reward network; then, a pair of observations are passed to the user for comparison; finally, the parameters of the reward network are updated to estimate a higher reward for the user preferred observation.

2.3. Model-Agnostic Meta-Learning (MAML)

The goal in meta-learning is to train a model which can rapidly adapt to new tasks with a few training iterations. The Model-Agnostic Meta-Learning (MAML) [9] is a meta learning algorithm which learns a data-driven initialization of the models that accelerate the standard reinforcement learning on new task drawn from a task distribution $p(\tau)$. The meta-training objective of MAML is:

$$\theta_i' = \theta - \alpha \nabla_\theta \mathcal{L}_{\tau_i}(\pi_\theta) \qquad (3)$$
$$\theta = \theta - \beta \nabla_\theta \sum_{\tau_i \sim p(\tau)} \mathcal{L}_{\tau_i}(\pi_{\theta_i'}),$$

where θ is the parameters of the policy network π_θ which maps states to a probability distribution over the actions, and $\mathcal{L}$ represent an arbitrary loss function which is selected based on the application. In effect, the MAML finds model parameters that with small changes will improve the loss function of any task drawn from the $p(\tau)$, when moving in the direction of that loss [9]. Our framework employs MAML to initialize the reward network parameters and accelerate the preference-learning process. Its effectiveness is explored in Sec 5.3.

3. Preference-Based Image Generation (PbIG)

In our proposed Preference-Based Image Generation (PbIG) the state of environment does not change as the user has a fixed mental image at all time, and taking new actions does not change the desired image. In other words, we can look at our problem as a continuous multi-armed bandit (stateless reinforcement learning) [34]. Consequently, the reward is a sole function of the action (see Figure (1)). In our formulation, we do not have a policy network. In contrast, we optimize directly the GAN random code z that produces user's mental image. This code is considered as the action in our RL formulation. Note that for the rest of this paper, for the sake of consistency with the GAN formulation, we use z to refer to the actions.

Following the approach in [3], the reward estimator network, N_R, can be interpreted as a preference-predictor when we consider $\hat{r} = N_R(z)$ as a factor which quantifies the user's judgments. Then, the probability of preferring an action depends exponentially on the value of the estimated reward and can be calculated as follows:

$$P[z_1 > z_2] = \frac{e^{\hat{r}_1}}{e^{\hat{r}_1} + e^{\hat{r}_2}}, \qquad (4)$$

where $r_1 = N_R(z_1)$ and $r_2 = N_R(z_2)$ are the corresponding rewards of the two latent codes z_1 and z_2, respectively.

At each time step, the user is given a pair of images generated from z_1 and z_2 to indicate (a) which image is more similar to the desired image, (b) the two images are equally similar, or (c) none of them is comparable to the desired image. If one of the images is selected as preferable, then its corresponding label l is set to one. Correspondingly, the label of the unfavored image is set to zero. In the case of neutral preference, both the labels are set to 0.5. Finally, when both images are rejected by the user, both labels are set to zero. In other words, the corresponding label of each image, or its corresponding latent code, determines the probability of being preferred by the user.

We can update the parameters of N_R to predict the user's preference labels. To this end, we minimize the cross-entropy loss between the estimated preferred probability and the actual labels provided by the user:

$$L_R(\hat{r}_1, \hat{r}_2) = - \sum_{(z_1, z_2, l_1, l_2) \in B} l_1 \log P[z_1 > z_2] \qquad (5)$$
$$+ l_2 \log P[z_2 > z_1],$$

where B is the training batch, l_1 and l_2 are the corresponding rewards of the two random codes z_1 and z_2, respectively. We update the reward and code with some modifications to the proposed steps in PbRL. Since our formulation is stateless, we cannot generate a pair of codes using a policy network to update the parameters of reward function. In order to sidestep this issue, we select two codes following a modified ϵ-greedy policy. The GAN generator then produces two images from the given codes for user comparison. The parameters of the reward network are updated based on the user-assigned labels and their estimated values to minimize the loss function in (5).

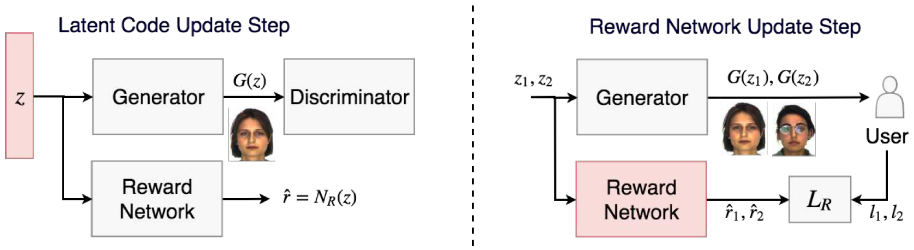


Figure 1: The proposed PbIG framework, is trained in two steps: The reward network update is based on the user's preference labels assigned to two random codes z_1 and z_2; The goal of the latent code update is to find the random code z which maximizes the estimated reward $\hat{r}$. Note that incorporating the generator and discriminator, minimizing the discriminator loss, is optional in this step, but can avoid the codes with non-satisfactory results.

Subsequently, a latent code, z is send to the reward function. The latent code then is optimized to maximize the estimated reward, minimizing the following loss function:

$$L_z = -\sum_{z \in B} N_R(z). \tag{6}$$

As a final step, we find the best initialization of our reward network using the MAML. Note that the MAML in the original paper [9] is used to find the initialization of the policy network. However, in our formulation of a stateless RL, we used the same technique to find the best initialization for the reward network. It usualy being trained based on multiple tasks which could come from a single or multiple datasets. However, in our probelm, since for each experiment we need to train a new reward network from scrach, and it can be considered as a new task, we employ MAML for find the best initial weights for the reward network using a series of synthetic-feedback experiments. The meta-training objective of MAML is defined as:

$$\begin{aligned} \theta_i' &= \theta - \alpha \nabla_\theta \mathcal{L}_{\tau_i}(N_R^\theta) \\ \theta &= \theta - \beta \nabla_\theta \sum_{\tau_i \sim p(\tau)} L_R(N_R^{\theta_i'}), \end{aligned} \tag{7}$$

where L_R is defined in (5).

4. Implementation

We adopt the "memory replay" to speed up the learning process by utilization of earlier samples. However, early experiments showed that the size of memory replay should be relatively small, or the network stops learning due to the very old randomly selected samples with limited information. We also propose to use a weighted memory, which enables us to adopt a bigger memory size. In this setup, we assign a distinct weight to each sample in the memory replay upon its arrival. The assigned weights then decay after each step, which means the older samples in the memory replay have less contributions to the loss function.

We train an ensemble of reward networks on randomly selected pairs from the replay memory. The final estimate of the reward is calculated by normalizing each of the estimators and then averaging the results. Also, similar to [3], we normalize the reward values, estimated by N_R, to have zero mean and constant standard deviation.

Even though GANs can learn to map a random code from a fixed distribution to an image in the training domain, they still fail to generate satisfactory results for some areas of the latent space. Preliminary experiments showed that there is a possibility that the code update stops in any of these areas of the code space. To overcome this issue, we incorporate the discriminator loss when updating the code.

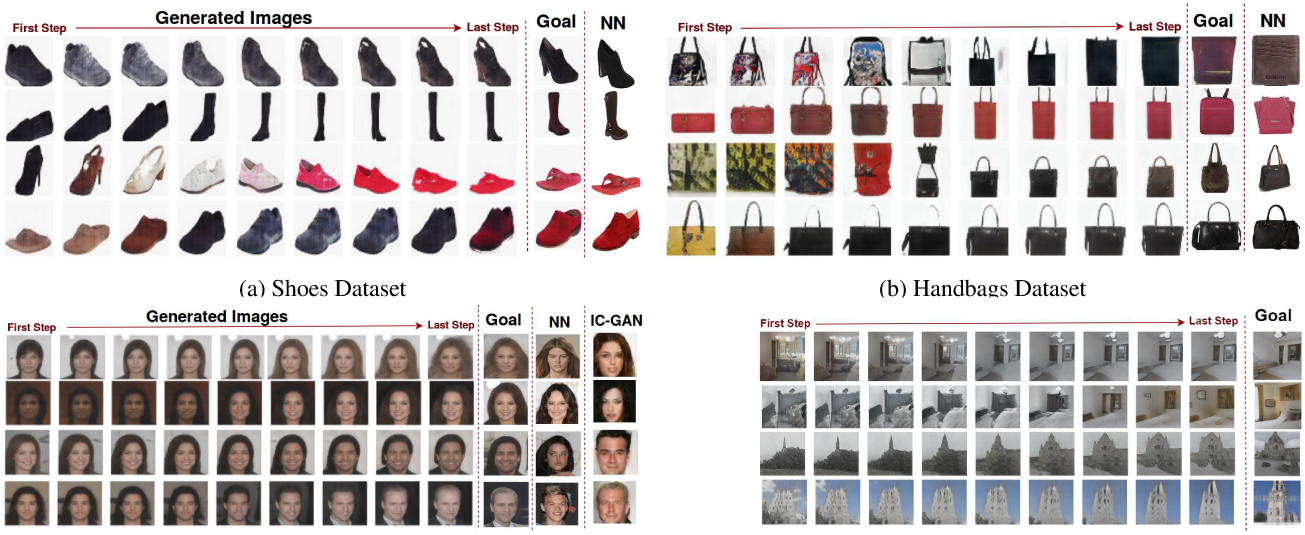
$$L_\pi^{total}(o) = L_z - L_D(G(z)), \tag{8}$$

where $L_D(G(z))$ is the loss of discriminator for the generated image $G(z)$. We use this general notation, as different GAN variants employ different losses.

Finally, with a pre-defined frequency, we generate a query from two randomly selected latent codes among the previous user's preferences. We also let the user select the current preference as the best generated image. If the current preference is not selected as the best generated image, we add the current preference and the best generated image as another pair to the memory. Clearly, the former is labeled as the preferred image.

4.1. Latent Code Sampling Policy

Since the environment is stateless in our formulation, we only have a single action. However, we need a pair of actions to train our reward network. In order to overcome this problem, we select at least one of the codes at random. The random code sampling policy is the key element of the whole framework to succeed. Sampling uniformly from the code distribution tends to destabilize the reward network training. Consequently, at each step we select a pair of



(a) Shoes Dataset

(b) Handbags Dataset

(c) CelebA Dataset

(d) Bedroom/Church Datasets

Figure 2: The evolution of the generated images by the latent code, and the desired image on different databases. Since the latent code is initialized randomly, in each experiment the StackGAN++ generates a random image. In each experiment, the first two rows are generated by human feedback and the last two rows are generated using synthetic feedback. However, for bedroom and church datasets, all samples are generated by human feedback.

codes as follows:

$$z = \begin{cases} z \sim \mathcal{N}(z_{last}, \sigma^2) & \text{w. probability } 1-\epsilon \\ z \sim \mathcal{N}(\mu_c, \sigma_c^2) & \text{w. probability } \epsilon \end{cases}, \quad (9)$$

where ϵ is the probability of taking a completely random action from the code space, z_{last} is the latest learned latent code, $N(z_{last}, \sigma^2)$ is a normal distribution with mean z_{last} and variance σ^2, and $\mathcal{N}(\mu_d, \sigma_d^2)$ is the distribution of GAN latent code, which is assumed to be a normal distribution with mean μ_c and variance σ_c^2 in this paper. Here, σ^2 and ϵ are the two hyper-parameters. To face the need to trade off between exploration and exploitation, ϵ is initialized to one and decays over time. Following this formulation, we generate completely random images and gradually give more chance to explore the neighborhood of the learned latent code. Early experiments showed that simply sampling pairs from the entire code space collapses the model. Similar to [4], we sample multiple pairs at each step, and select the pair with highest reward variance among the ensemble of models, based on the current learned reward networks, as a query to the user.

4.2. Conditional Setting

Even though the PbIG framework can be applied to the generative models, both in their unconditional and conditional settings, conditioning on some human-sensible attributes can benefit PbIG, by limiting the latent code search space. For example, in face generation, conditioning the generated face on some facial attributes, such as the gender, hair and skin colors narrows down the problem into looking exclusively for the geometrical properties of the desired face. This can reduce the average number of comparisons by the user drastically.

In conditional setting, we provide the generator network with an interpretable code c, like facial attributes, and a random code z, which represents the subtle variations of the image. Therefore, our framework only needs to learn the random code z.

5. Experiments

The main goal of our experiments is to investigate if a human subject can generate a desired image in a reasonable time. The framework should be able to find the latent code with the minimum number of user judgments. We evaluate PbIG on CelebA [40], edges $\leftrightarrow$ handbags [46], edges $\leftrightarrow$ shoes [42], LSUN bedroom and church [27], and CUB [35] datasets. To the best of our knowledge, this work is the first attempt to retrieve GAN's random codes based on the human preferences, consequently we use the Nearest Neighbor (NN) sample in training set as the baseline for comparison. Note that NN is not applicable in practice as it needs the ground truth. The image realism and diversity of the output is always the same as the underlying GAN framework and reporting them has no meaning. However, we compared the results of CelebA with IC-GAN [26] which is a conditional GAN based on the facial attributes. We conducted a comprehensive set of experiments in different settings, to evaluate the proposed

PbIG both qualitatively and quantitatively.

Human or synthetic feedback: We trained the models using both human and synthetic feedbacks. In human-based experiments, feedback was provided by a subject who is given a pair of generated images. For synthetic feedback, the agent queries for comparison based on deep image features, extracted from a VGG19 [29], instead of human preferences. Note that, here, we use synthetic feedback in order to conduct more experiments. Any comparison of the human and synthetic feedbacks, which represents how well the synthetic feedback mimics human evaluation is out of scope of this work.

Viewed and Semi-viewed: For human subjects, we set up viewed and semi-viewed scenarios. In former, the desired image was available to the subjects for comparison all the time. In contrast, in the semi-viewed case, the desired image was shown to the subjects for 30 seconds, and they were asked to compare the generated images based on their memories. The latter simulates the forensic facial reconstruction performed by the law enforcement.

Network Structure: We train the PbIG using the Adam optimizer [20], with learning rate of 0.0002, $\beta_1 = 0.5$, $\beta_2 = 0.999$ and mini-batch size of 20. The algorithm is implemented in PyTorch [25]. The reward network has 4 Fully-Connected (FC) layers with 300, 300, 300, 1 nodes. Each FC layer, except the last one, is followed by a LeakyRelu activation function and a dropout layer. For all the three datasets, we train a distinct Stack-GAN++ with a random code of size 100 drawn from a normal distribution with zero mean and unit variance, i.e., $z \sim \mathcal{N}(0, 1)$. The generator is trained to generate images of size 256×256. We set all the hyper-parameters as in the original paper to train our Stack-GAN++. Finally, we use $\sigma = 0.5$ for random code sampling policy.

Synthetic Feedback: We generate the synthetic feedback in a way to simulate the human comparisons. To this end, an L1 loss is computed over deep image features, extracted from a pretrained VGG19 [29], which is sometimes referred to as perceptual loss [7]. To evaluate the similarity using both fine and course features, we calculate an average perceptual loss over conv3_4 and conv5_4 of the VGG19 network. Since the perceptual loss can only represent the content of an image, we use the Gram matrix, which is the inner product between the vectorized feature maps of a layer, over conv2_2 and conv3_4 to consider the style of generated images as well. Finally, we define the total similarity distance between a generated image and the desired image as a weighted average of the content and style losses. Next, the preferred image is defined as the image with the minimum perceptual loss with the desired image

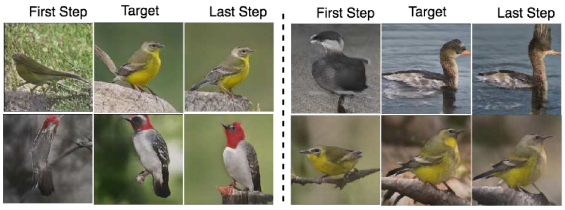


Figure 3: PbIG in conditional setting (CUB dataset).

based on the following formulation:

$$l_i, l_j = \begin{cases} 1, 0 & l_s(img_i) < l_s(img_j) - th_c \\ & l_s(img_i) < th_s \\ 0.5, 0.5 & \parallel l_s(img_i) - l_s(img_j) \parallel < th_c \\ & l_s(img_i) < th_s \text{ or } l_s(img_j) < th_s \\ 0, 0 & \text{else} \end{cases},$$

where $img_i, i = 1, 2$ represent the images to be compared, $l_s(img_i)$ is the total similarity distance between img_i and the desired image, th_c is the minimum difference between the similarity distances of img_1 and img_2 to be comparable, and th_s is the maximum value of similarity distance between img_i and the desired image to select img_i as the preference. More specifically, it simulates the human inability to compare two images when both are too far from the desired image. We also randomly modified the values of l_1 and l_2 with a probability of 5% to simulate user mistakes. We use $th_c = 0.1$ for the synthetic-feedback experiments. The value of th_s selected as 90% of the dataset Average Similarity Distance (ASD), computed between randomly selected images of each dataset (see Table 1).

5.1. Qualitative Analysis

Figures 2a, 2b, and 2c show how the generated images evolve during the comparison steps, following the proposed PbIG framework. For each experiment, the desired image is generated randomly using the StackGAN++ generator. The first two rows of each experiment are generated by the human feedback and the last two rows show the results of synthetic feedback. We stopped each human-feedback experiment as soon as the user is satisfied with the generated image. The user has access to the goal image during the whole experiment. The synthetic-feedback experiments were also executed for 600 comparisons and the resultant generated images are stored to be compared by the user. The stopping step is defined as the earliest step which satisfies the user. The result clearly reveals the success of PbIG to reach the desired image in a limited number of comparisons. For the CelebA dataset, the best achieved results of IC-GAN are also illustrated in Figure 2c. We also conducted more experiments on bedroom and church datasets which has more complexity (see Figure 2d). Note that all the experiments for these datasets are human-feedback generated, as the synthetic-feedback experiments

Dataset	Dataset ASD	NN ASD	Feedback	Final ASD		ANS	
				w/ D_{GAN}	w/o D_{GAN}	w/ D_{GAN}	w/o D_{GAN}
CelebA	1.135	0.483	Human	0.46 ± 0.17	0.47 ± 0.20	310 ± 33	322 ± 40
			Synthetic	0.39 ± 0.14	0.40 ± 0.13	302 ± 29	276 ± 39
Handbags	1.782	0.532	Human	0.54 ± 0.16	0.56 ± 0.14	360 ± 52	381 ± 59
			Synthetic	0.49 ± 0.12	0.49 ± 0.15	332 ± 50	297 ± 62
Shoes	1.637	0.419	Human	0.48 ± 0.19	0.51 ± 0.18	347 ± 45	369 ± 53
			Synthetic	0.46 ± 0.13	0.47 ± 0.15	311 ± 42	289 ± 64
CUB (conditional)	1.72	0.387	Human	0.43 ± 0.22	0.49 ± 0.25	98 ± 39	124 ± 51
			Synthetic	0.40 ± 0.18	0.42 ± 0.20	102 ± 36	123 ± 40

Table 1: The average perceptual distance between the final generated image and the goal image, and the average number of steps to stop training the latent code on different datasets for human and synthetic feedbacks.

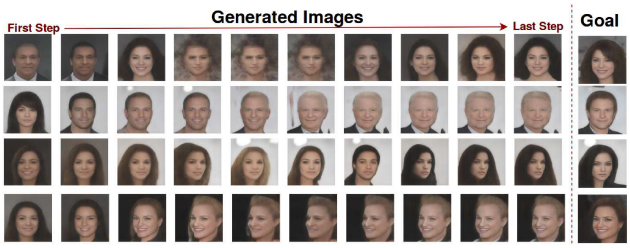


Figure 4: The evolution of the generated image by the latent code, and the goal image on CelebA database in semi-viewed setting. The users looked at goal images for 30 seconds and then asked to generate it.

fail to converge to an user-acceptable image. The final results on these datasets show the lack of our framework's ability on capturing small image details. A possible reason could be the poor performance of the StackGAN++ in these datasets, having too much of artifacts. Note that the performance of our framework is bounded by the performance of the underlying GAN framework in generating high quality images. Figure 3 also shows the PbIG results in conditional setting (CUB). The images are generated using conditional StackGAN++, conditioned on text description, which narrowed down the search space.

We also conducted a series of semi-viewed human-feedback experiments. Figure 4 illustrates the results of our semi-viewed experiments on CelebA dataset.

5.2. Quantitative Analysis

Table 1 presents the quantitative analysis of the generated images by the proposed framework. We evaluate the PbIG using two metrics, namely average similarity distance (ASD), and average number of steps (ANS). The ASD is calculated by averaging the similarity distance between the generated image after the experiment termination, based on the explained stopping criteria, and the goal image (over 100 experiments in the synthetic-feedback setting, and 20 experiments in the human-feedback setting). The effect of incorporating the discriminator of StackGAN++, D_{GAN}, in training the latent code is also investigated. Based on our human-feedback experiments each query takes 3 seconds in average to receive a user feedback. That means, based on the values of ANS in Table 1, it takes roughly 15 (5) minutes for the users to generate their goal images using unconditional (conditional) PbIG. We also tried to redo the human-feedback CelebA experiments for IC-GAN by tuning its conditions. The ASD was calculated as 0.694, compared to 0.458 of PbIG, which confirms the visual difference in Figure 2c. Table 2 is also lists the results of more experiments on LSUN bedroom and church datasets. Note that we conducted it only in human-feedback setting, as the synthetic-feedback did not generate acceptable results. The reason is that the poor performance of the StackGAN++ in these datasets, having too much of artifacts.

Since the ASD might not be a perfect alternative to the human comparison ability, its value for the human-feedback experiments is greater than the synthetic-feedback. A possible reason could be that the network, when using synthetic feedback, learns to apply subtle modifications to the generated images that are almost undetectable to the human eyes (see adversarial samples [31]). For the sake of clarity, the average similarity distance between 1000 randomly generated images is also reported for each dataset. We also reported ASD for NN training sample.

Employing the discriminator of StackGAN++, D_{GAN}, unexpectedly increases the ANS of the synthetic-feedback experiments while the ASD remains almost unaltered. This increase is a direct consequence of changing the direction of policy search by the discriminator to satisfy the realism of the generated image. However, the final result could be in average more plausible to the human user as some unrealistic images might have a low similarity distance to the goal image. In contrast, for human-feedback experiments, using D_{GAN} reduces the ANS notably. Here, removing the D_{GAN}, results in the latent code to generate unrealistic images in some experiments, while learning,

which increases the number of selecting "not comparable" by the user (see the generated images in the first row of Figure 4). The ASD of the human-feedback experiments is rather similar with and without D_{GAN}, while its larger variation, in comparison with synthetic-feedback experiments, is associated with its relatively smaller number of experiments over which the ASD is calculated.

We also study how sensitive is the proposed framework for the complexity of the reward network. To this end, we calculated the ASD on the synthetic-feedback experiments for multiple reward networks with different complexity, namely $2 \times$ fc200, $3 \times$ fc200, $3 \times$ fc300, $4 \times$ fc300, $3 \times$ fc400, and $4 \times$ fc400. Here, "n x fc$m$" represents n subsequent fully-connected layers with m nodes. Figure 5a shows how the reward network complexity affects the ASD (models sorted as reported above). The proposed learning scheme is quite robust to the complexity of the reward network. However, for very simple or complex networks, the ASD increases drastically due to the underfitting and overfitting, respectively. More specifically, in both cases, the learned latent code diverges significantly from its mean (zero in StackGAN++). Figure 5b illustrates how the ASD decreases as more feedbacks are provided to the framework. The ASD eventually starts settling down which means the PbIG is able to generate an acceptable representation of the goal image in less than 350 comparisons. Note that the results of this figure, as well as the results on Table 1, are reported for the reward network #3.

5.3. Ablation Study

To further gain deep insights of the improvements obtained by each part of the proposed method, we conduct more additional synthetic-feedback experiments for ablation studies:

MAML initialization: Adding MAML initialization to the reward network results in a roughly %25 improvement in ANS. Table 3 shows the ANS of synthetic feedback experiments with and without MAML initialization. However, the final ASDs changed for less than $\pm\%2$.

Random queries: Replacing the proposed random code sampling policy, in which we gradually give more chance to explore the neighborhood of the learned latent code, with complete random selection, results in the learned latent code converging to a point with a high similarity distance.

Best sample tracking: Removing the best sample tracking, significantly increases the ANS. Table 3 shows the ANS for the same experiments as in Table 1, without best sample tracking. The improvement in ANS is a result of incorporating automatically generated pairs, to train the reward network, which provides a global information about the most significant direction of reward maximization.

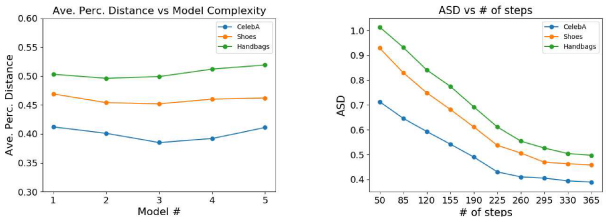
Weighted replay memory: We conducted the same experiments with un-weighted replay memory. The ASD at step 500 increased by 18%, 27%, and 20% for CelebA, Shoes, and Handbags datasets, respectively. Note that, as was mentioned, using a weighted replay memory will allow the network to learn faster, specifically in the early steps as most of the samples are drawn randomly from the latent code space. However, in late steps, there is more chance of sampling from the neighborhood of the learned latent code.

	Feedback	ASD	ANS
Bedroom	Human	0.67	320
Church	Human	0.71	291

Table 2: Human feedback results on LSUN bedroom and church datasets.

w/o	CelebA	Handbags	Shoes
BST	365 (21% ↑)	378 (14% ↑)	367 (18% ↑)
MAML	359 (19% ↑)	417 (25% ↑)	387 (24% ↑)

Table 3: Removing the best sample tracking (BST) or MAML increases the average number of steps significantly.



(a) ASD vs Complexity (b) ASD vs Steps

Figure 5: ASD vs. model Complexity and number of steps.

Comparison of previous preferences: Asking the users to compare their preferred images from the previous steps decreases the ANS by 9%, 12%, and 14% for CelebA, Shoes, and Handbags datasets, respectively. This trend of comparison, provides relative scores between previous preferred samples, which results in training of a better reward estimator.

6. Conclusion

In this paper, we present PbIG which enables the user to control the generation process of a generative model. To the best of our knowledge, it is the first universal framework which retrieves the desired latent code of a generative model which produce the user's desired image. The proposed method leverages the power of preference-based reinforcement learning to find the desired latent code from a set of user's preferences. The proposed framework can be easily be adopted by any generative model in its conditional or unconditional setting. Our future work will explore a more systematic sampling policy to minimize the number of comparisons by the human user.

References

[1] R. Akrour, M. Schoenauer, and M. Sebag. Preference-based policy learning. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 12–27. Springer, 2011. 2

[2] X. Chen, Y. Duan, R. Houthooft, J. Schulman, I. Sutskever, and P. Abbeel. InfoGAN: Interpretable representation learning by information maximizing generative adversarial nets. In *Advances in neural information processing systems*, pages 2172–2180, 2016. 1

[3] P. F. Christiano, J. Leike, T. Brown, M. Martic, S. Legg, and D. Amodei. Deep reinforcement learning from human preferences. In *Advances in Neural Information Processing Systems*, pages 4299–4307, 2017. 2, 3, 4

[4] C. Daniel, M. Viering, J. Metz, O. Kroemer, and J. Peters. Active reward learning. 2014. 5

[5] H. Ding, K. Sricharan, and R. Chellappa. Exprgan: Facial expression editing with controllable expression intensity. *AAAI*, 2018. 1

[6] J. Donahue, P. Krähenbühl, and T. Darrell. Adversarial feature learning. *arXiv preprint arXiv:1605.09782*, 2016. 1

[7] A. Dosovitskiy and T. Brox. Generating images with perceptual similarity metrics based on deep networks. In *Advances in Neural Information Processing Systems*, pages 658–666, 2016. 6

[8] V. Dumoulin, I. Belghazi, B. Poole, O. Mastropietro, A. Lamb, M. Arjovsky, and A. Courville. Adversarially learned inference. *arXiv preprint arXiv:1606.00704*, 2016. 1

[9] C. Finn, P. Abbeel, and S. Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International Conference on Machine Learning*, pages 1126–1135, 2017. 2, 3, 4

[10] J. Fürnkranz and E. Hüllermeier. Preference learning. In *Encyclopedia of Machine Learning*, pages 789–795. Springer, 2011. 2

[11] J. Fürnkranz, E. Hüllermeier, W. Cheng, and S.-H. Park. Preference-based reinforcement learning: a formal framework and a policy iteration algorithm. *Machine learning*, 89(1-2):123–156, 2012. 2

[12] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014. 1, 2

[13] X. Huang, M.-Y. Liu, S. Belongie, and J. Kautz. Multimodal unsupervised image-to-image translation. *arXiv preprint arXiv:1804.04732*, 2018. 1

[14] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros. Image-to-image translation with conditional adversarial networks. *CVPR*, 2017. 1

[15] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros. Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1125–1134, 2017. 2

[16] L. P. Kaelbling, M. L. Littman, and A. W. Moore. Reinforcement learning: A survey. *Journal of artificial intelligence research*, 4:237–285, 1996. 2

[17] T. Karras, T. Aila, S. Laine, and J. Lehtinen. Progressive growing of GANs for improved quality, stability, and variation. *ICLR*, 2018. 1

[18] H. Kazemi, S. M. Iranmanesh, and N. Nasrabadi. Style and content disentanglement in generative adversarial networks. In *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 848–856. IEEE, 2019. 1

[19] H. Kazemi, S. Soleymani, F. Taherkhani, S. Iranmanesh, and N. Nasrabadi. Unsupervised image-to-image translation using domain-specific variational information bound. In *Advances in Neural Information Processing Systems*, pages 10348–10358, 2018. 1

[20] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 6

[21] D. P. Kingma and M. Welling. Auto-encoding variational bayes. *NIPS*, 2014. 1

[22] S. Liu, Y. Sun, D. Zhu, R. Bao, W. Wang, X. Shu, and S. Yan. Face aging with contextual generative adversarial nets. In *Proceedings of the 2017 ACM on Multimedia Conference*, pages 82–90. ACM, 2017. 1

[23] J. MacGlashan, M. K. Ho, R. Loftin, B. Peng, D. Roberts, M. E. Taylor, and M. L. Littman. Interactive learning from policy-dependent human feedback. *arXiv preprint arXiv:1701.06049*, 2017. 2

[24] M. Mirza and S. Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014. 1

[25] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer. Automatic differentiation in pytorch. In *NIPS-W*, 2017. 6

[26] G. Perarnau, J. van de Weijer, B. Raducanu, and J. M. Álvarez. Invertible conditional GANs for image editing. *arXiv preprint arXiv:1611.06355*, 2016. 1, 5

[27] A. Radford, L. Metz, and S. Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015. 5

[28] T. Salimans, A. Karpathy, X. Chen, and D. P. Kingma. Pixelcnn++: Improving the pixelcnn with discretized logistic mixture likelihood and other modifications. *arXiv preprint arXiv:1701.05517*, 2017. 1

[29] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 6

[30] P. D. Sørensen, J. M. Olsen, and S. Risi. Breeding a diversity of super mario behaviors through interactive evolution. In *Computational Intelligence and Games (CIG), 2016 IEEE Conference on*, pages 1–7. IEEE, 2016. 2

[31] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013. 7

[32] F. Taherkhani, H. Kazemi, A. Dabouei, J. Dawson, and N. M. Nasrabadi. A weakly supervised fine label classifier enhanced by coarse supervision. In *The IEEE International Conference on Computer Vision (ICCV)*, October 2019. 1

[33] F. Taherkhani, H. Kazemi, and N. M. Nasrabadi. Matrix completion for graph-based deep semi-supervised learning. In *Thirty-Third AAAI Conference on Artificial Intelligence*, 2019. 1

[34] J. Vermorel and M. Mohri. Multi-armed bandit algorithms and empirical evaluation. In *European conference on machine learning*, pages 437–448. Springer, 2005. 3

[35] C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie. The caltech-ucsd birds-200-2011 dataset. 2011. 5

[36] S. I. Wang, P. Liang, and C. D. Manning. Learning language games through interaction. *arXiv preprint arXiv:1606.02447*, 2016. 2

[37] C. Wirth, R. Akrour, G. Neumann, and J. Fürnkranz. A survey of preference-based reinforcement learning methods. *The Journal of Machine Learning Research*, 18(1):4945–4990, 2017. 2

[38] C. Wirth, J. Furnkranz, G. Neumann, et al. Model-free preference-based reinforcement learning. In *30th AAAI Conference on Artificial Intelligence, AAAI 2016*, pages 2222–2228, 2016. 2

[39] X. Yan, J. Yang, K. Sohn, and H. Lee. Attribute2image: Conditional image generation from visual attributes. In *European Conference on Computer Vision*, pages 776–791. Springer, 2016. 1

[40] S. Yang, P. Luo, C.-C. Loy, and X. Tang. From facial parts responses to face detection: A deep learning approach. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3676–3684, 2015. 5

[41] R. A. Yeh, C. Chen, T. Y. Lim, A. G. Schwing, M. Hasegawa-Johnson, and M. N. Do. Semantic image inpainting with deep generative models. 2017. 1

[42] A. Yu and K. Grauman. Fine-grained visual comparisons with local learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 192–199, 2014. 5

[43] Y. Yue, J. Broder, R. Kleinberg, and T. Joachims. The k-armed dueling bandits problem. *Journal of Computer and System Sciences*, 78(5):1538–1556, 2012. 2

[44] H. Zhang, T. Xu, H. Li, S. Zhang, X. Wang, X. Huang, and D. Metaxas. Stackgan++: Realistic image synthesis with stacked generative adversarial networks. *arXiv preprint arXiv:1710.10916*, 2017. 2

[45] S. Zhao, J. Song, and S. Ermon. Infovae: Information maximizing variational autoencoders. *arXiv preprint arXiv:1706.02262*, 2017. 1

[46] J.-Y. Zhu, P. Krähenbühl, E. Shechtman, and A. A. Efros. Generative visual manipulation on the natural image manifold. In *European Conference on Computer Vision*, pages 597–613. Springer, 2016. 1, 5

[47] F. Zohrizadeh, M. Kheirandishfard, and F. Kamangar. Class subset selection for partial domain adaptation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2019. 1