Multi-Domain Image-to-Image Translation using StarGAN with Max-Sliced Wasserstein Distance

Ziyu Zhou
University of Illinois at Urbaba-Champaign
ziyuz2@illinois.edu

Abstract

Recent research has greatly boosted the development of image-to-image translation from a source domain to a target domain. However, translating images to multiple target domains is still difficult because separate generators need to be trained for every source-to-target pair. In this project, we implement a novel model called StarGAN, which is capable of performing multi-domain image-to-image translation using only a single model. It's observed that training the StarGAN can take long because of the instability during the process. To address such issue, we modify the StarGAN objective using the max-sliced Wasserstein distance to form the "Max-Sliced StarGAN" model. We empirically demonstrate that the Max-Sliced StarGAN is able to generate images with high quality, while just requires small sample complexity and projection complexity. It scales easily up to high-dimensional data and is insensitive to the values of the hyper-parameters, making the development process simpler.

1. Introduction

The image-to-image translation task can be considered as translating an input image to an output image with some specific characteristics changed while the others fixed, including converting a person's hair color while keeping the other features of the person intact, transforming an image of a horse to a zebra [18] while retaining the shape of the animal and constructing exactly aligned aerial photos from maps [9].

StarGAN, proposed by Choi *et al*. [2], has demonstrated outstanding abilities in such task, especially in translating an input image to images belonging to multiple target domains. Denote *attribute* as meaningful feature in an image such as hair color or gender, while *attribute value* being the value of the attribute such as the specific color for hair, e.g., black. *Domain* is denoted as a set of images with the same attribute value, such as images of people having black hair. In this setting, the task of multi-domain image-to-image translation is to convert images based on the attributes of multiple domains. Fig. 1 illustrates an example of such task which converts the input image to six domains, namely, 'blond hair', 'brown hair', 'rosy cheeks', 'gender', 'mouth slight open/closed' and 'smiling/not smiling'. Unlike existing models which require to train $k(k-1)$ generators to learn the mappings among k domains, StarGAN is able to learn such conversion using only a pair of discriminator and generator.

Generative Adversarial Networks (GANs), proposed by Goodfellow *et al*. [6], and its variants, have been widely used for image-to-image translation. However, they are known to be tricky and difficult in terms of training and optimization. Typical problems include mode collapse, training instability and vanishing gradients. As a variant of GANs, StarGAN is also faced with such problems. An alternative formulation, the max-sliced Wasserstein distance [3], has been proposed to tackle the issue. It has compelling sample complexity and reduced projection complexity, and scales easily up to high-dimesional data.

In this project, we combine StarGAN and the max-sliced Wasserstein distance together to utilize the advantages from both worlds. The newly formed model is called "Max-Sliced StarGAN". We first introduce the related work to the fields of image-to-image translation and GAN training in Section 2. We then describe our approach in details in Section 3, starting from the original objective of StarGAN and the formulation of max-sliced Wasserstein distance, to the combined Max-Sliced StarGAN and its training algorithm. The actual implementation of the network architectures are illustrated in Section 4. In Section 5, we analyze the behaviors of the Max-Sliced StarGAN from different aspects and compare it to some baseline models to demonstrate their difference. Finally, we conclude the advantages of using Max-Sliced StarGAN in practice.

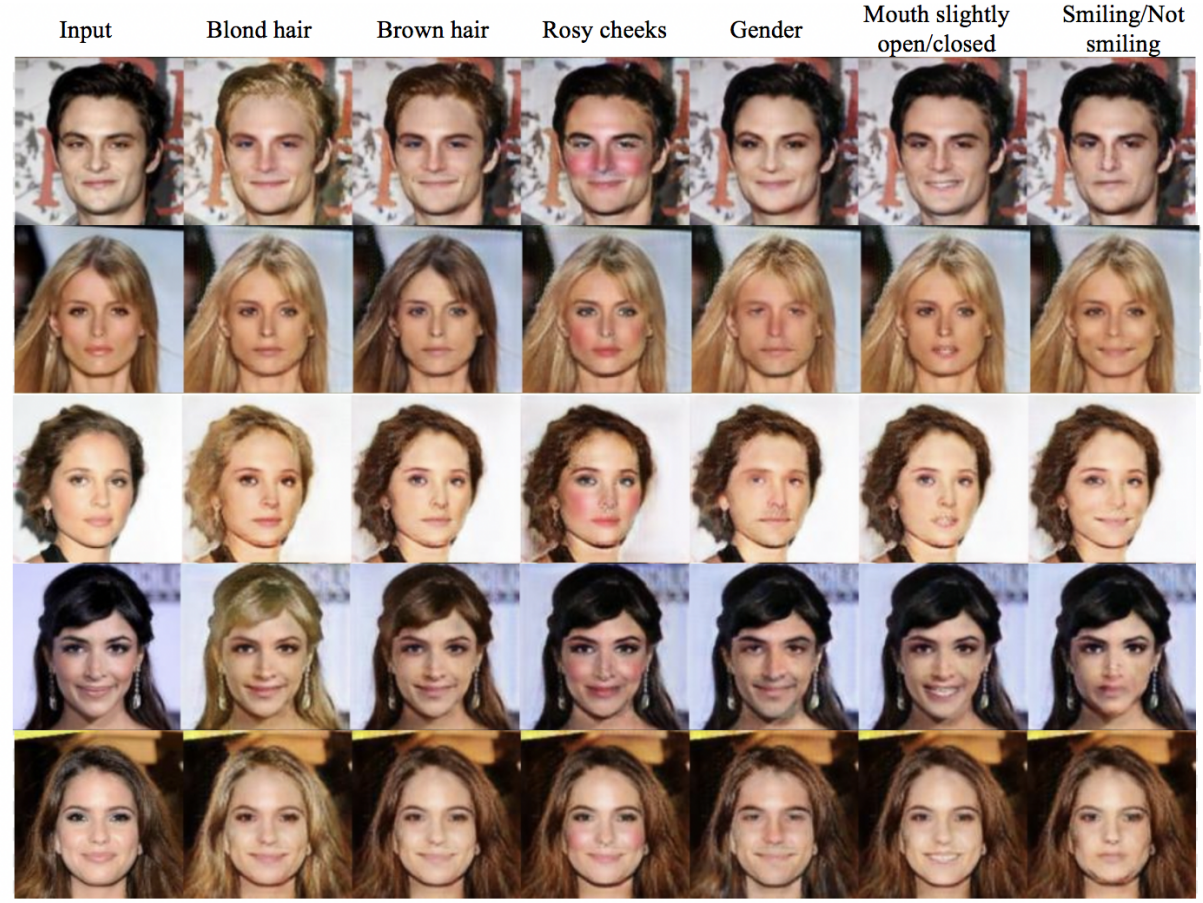

Figure 1. Multi-domain image-to-image translation results on the CelebA dataset for facial attributes using Max-Sliced StarGAN, which combines the recently proposed StarGAN [2] architecture with max-sliced Wasserstein distance [4].

2. Related Work

2.1. Generative Adversarial Networks (GANs)

Generative adversarial networks (GANs) have been applied to a great variety of tasks such as image synthesis [10, 8], image editing [17], representation learning [5, 15] and super-resolution [11]. The basic idea of GANs is to train a generator to generate fake images from random distributions that are similar to and indistinguishable from the real ones, and a discriminator to determine whether an input image is real or fake. The two are trained in an adversarial manner to provide information for each other so that both can improve.

2.2. Image-to-Image translation

Conditional adversarial networks are widely applied to image-to-image translation task. cGANs [14] are a way to generate samples conditioned on the class, which are then extended by pix2pix [9] that learns a mapping from input to output images. pix2pix requires paired data which is hard to obtain and generalize. Various unpaired image-to-image translation methods have been developed to tackle this problem, such as the work by Liu *et al*. [12] that combines variational auto-encoders and GANs, and CycleGAN by [18] Zhu *et al*. that introduces a cycle consistency loss to ensure that the generated fake images can then be mapped back to the original ones. These methods, although good at transforming images from a source set to a target set, are limited when considering mappings to several target domains as different generators must be trained for each mapping. This paper implements a recently proposed method, StarGAN [2], which can achieve image-to-image translations for multiple domains using a single pair of discriminator and generator by providing conditional domain information.

2.3. Training Generative Adversarial Networks

Consider a dataset $\mathcal{D} = \{x\}$ whose data distribution $\mathbb{P}_d$ is unknown. The original GANs formulation [6] trains a generator $G_{\theta_g}(z)$, parameterized by θ_g, to learn to map prior perturbations z obtained from $\mathbb{P}_z$ to a range whose probability distribution $\mathbb{P}_g$ is close to $\mathbb{P}_d$. A discriminator,

$D_{\theta_d}(x)$, parameterized by θ_d, acts as a binary classifier that learns to distinguish the real sample $x \sim \mathcal{D}$ from the fake sample $G_{\theta_g}(z)$ and outputs a scalar indicating the probability of the input being real or fake. The objective is thus given by

$$\min_{\theta_g} \max_{\theta_d} \quad \mathbb{E}[D_{\theta_d}(x)] - \mathbb{E}[D_{\theta_d}(G_{\theta_g}(z))]. \tag{1}$$

Jensen-Shannon divergence is often used to approximate the expectation:

$$\begin{aligned} \min_{\theta_g} \max_{\theta_d} \quad & \mathbb{E}_{x\sim\mathbb{P}_d}[\log D_{\theta_d}(x)] \\ & + \mathbb{E}_{z\sim\mathbb{P}_z}[\log(1 - D_{\theta_d}(G_{\theta_g}(z)))]. \end{aligned} \tag{2}$$

Arjovsky *et al*. [1] suggests to use the Wasserstein-1 distance to alleviate the problems of instability and mode collapse when training GANs with Jensen-Shannon divergence. To estimate the distance, [1] substitudes it using the Kantorovich-Rubinstein duality and obtains:

$$W(\mathbb{P}_d, \mathbb{P}_g) = \sup_{\|f\|_L \leq 1} \mathbb{E}_{x\sim\mathbb{P}_g}[f(x)] - \mathbb{E}_{x\sim\mathbb{P}_d}[f(x)] \tag{3}$$

where f are 1-Lipschitz functions.

Deshpande *et al*. [4] proposes an alternative formulation called "sliced Wasserstein distance" that estimates the Wasserstein distance directly from samples and enjoys better training stability. Specifically, using the sliced Wasserstein-2 distance, the objective of the generator now becomes

$$\min_{\theta_g} \frac{1}{|\hat{\Omega}|} \sum_{\omega\in\hat{\Omega}} W_2^2(\mathcal{D}^\omega, \mathcal{F}^\omega). \tag{4}$$

where $\mathcal{D}$ and $\mathcal{F}$ represent "real" and "fake" data points sampled from $\mathbb{P}_d$ and $\mathbb{P}_g$ respectively. ω is a projection direction from $\hat{\Omega}$, a set of randomly chosen unit vectors. $W_2^2(\mathcal{D}^\omega, \mathcal{F}^\omega)$ is the Wasserstein-2 distance of samples taken from $\mathcal{D}$, $\mathcal{F}$ projected onto direction ω, which can be estimated directly using samples $\mathcal{D}^\omega$, $\mathcal{F}^\omega$ by sorting them with a permutation σ such that

$$W_2^2(\mathcal{D}^\omega, \mathcal{F}^\omega) = \frac{1}{|\mathcal{D}|} \sum_i \|\mathcal{D}^\omega_{\sigma_\mathcal{D}(i)} - \mathcal{F}^\omega_{\sigma_\mathcal{F}(i)}\|_2^2, \tag{5}$$

$$\mathcal{D}^\omega_{\sigma_\mathcal{D}(i)} \leq \mathcal{D}^\omega_{\sigma_\mathcal{D}(i+1)}, \forall i \in \{1 \leq i < |\mathcal{D}|\}, \tag{6}$$

$$\mathcal{F}^\omega_{\sigma_\mathcal{F}(i)} \leq \mathcal{F}^\omega_{\sigma_\mathcal{F}(i+1)}, \forall i \in \{1 \leq i < |\mathcal{F}|\}. \tag{7}$$

3. Approach

This section first describes the formulation of StarGAN [2], a model tailored for the multi-domain image-to-image translation task, and then introduces a recently proposed method, max-sliced Wasserstein distance [3], to improve the training of StarGAN. The two are combined together to form a "Max-Sliced StarGAN" to utilize the advantages of StarGAN and ease the training difficulties.

3.1. Star Generative Adversarial Networks

3.1.1 StarGAN Objective

Star Generative Adversarial Networks (StarGANs), proposed by Choi *et al*. [2], is able to transform images from one domain to multiple target domains with only a generator and a discriminator. Given an image x sampled from the real dataset $\mathcal{D}$, and a target domain label c, the generator $G_{\theta_g}(x, c)$ learns to map x to an output image y conditioned on c. The discriminator $D_{\theta_d}(x)$ has two roles, one being a critic to determine whether an input image x is real or fake and output a scalar $D_{\theta_d,src}(x) \in \mathbb{R}$ which indicates the probability of x's authenticity, the other being a classifier that outputs a vector $D_{\theta_d,cls}(x) \in \mathbb{R}^{c_{dim}}$ (c_{dim} is the dimension of the mask vector introduced in section 3.1.2), the i^{th} entry of which tells the probability of the i^{th} attribute being classified into domain c_i. To summarize, our goal is to achieve

$$G_{\theta_g} : G_{\theta_g}(x, c) \to y, \tag{8}$$

$$D_{\theta_d} : x \to \{D_{\theta_d,src}(x), D_{\theta_d,cls}(x).\} \tag{9}$$

Adversarial Loss. Based on Eq. (2), the adversarial loss is defined as

$$\begin{aligned} \mathcal{L}_{adv} = \quad & \mathbb{E}_x[\log D_{\theta_d,src}(x)] \\ & + \mathbb{E}_x[\log(1 - D_{\theta_d,src}(G_{\theta_g}(x, c))]. \end{aligned} \tag{10}$$

We want to solve

$$\min_{\theta_g} \max_{\theta_d} \mathcal{L}_{adv} \tag{11}$$

Domain Classication Loss. We want our discriminator to be able to classify a real image x to its original domain c'. Consider $D_{\theta_d,cls}(c'|x)$ as the probability distribution output by the discriminator D_{θ_d}, we have

$$\mathcal{L}^r_{cls} = \mathbb{E}_{x,c'}[-\log D_{\theta_d,cls}(c'|x)], \tag{12}$$

$$\min_{\theta_d} \mathcal{L}^r_{cls}. \tag{13}$$

For the generator, we want it to be able to "fool" the discriminator such that the discriminator will classify the

fake image, generated by $G_{\theta_g}(x, c)$ conditioned on the real input image x and the target domain c, to its target domain. The loss function and the corresponding optimization are thus defined as

$$\mathcal{L}_{cls}^{f} = \mathbb{E}_{x,c}[-\log D_{\theta_d,cls}(c|G_{\theta_g}(x, c))], \tag{14}$$

$$\min_{\theta_g} \mathcal{L}_{cls}^{f}. \tag{15}$$

Reconstruction Loss. To make sure the generated image $G_{\theta_g}(x, c)$ preserve the original contents of its corresponding input image, i.e., the generator must be able to reconstruct the input image based on $G_{\theta_g}(x, c)$ and the original domain label c', a cycle consistency loss [18] is adopted:

$$\mathcal{L}_{rec} = \mathbb{E}_{x,c,c'}[\|x - G_{\theta_g}(G_{\theta_g}(x, c), c')\|_1], \tag{16}$$

$$\min_{\theta_g} \mathcal{L}_{rec}. \tag{17}$$

Putting the above-mentioned loss terms together, the full objective can be defined as

$$\mathcal{L}_D = -\mathcal{L}_{adv} + \lambda_{cls}\mathcal{L}_{cls}^{r}, \tag{18}$$

$$\mathcal{L}_G = \mathcal{L}_{adv} + \lambda_{cls}\mathcal{L}_{cls}^{f} + \lambda_{rec}\mathcal{L}_{rec}, \tag{19}$$

where λ_{cls} and λ_{rec} are hyper-parameters controlling the relative improtance of the domain classification and reconstruction loss. Now we aim to solve:

$$\min_{\theta_d} \mathcal{L}_D, \tag{20}$$

$$\min_{\theta_g} \mathcal{L}_G. \tag{21}$$

3.1.2 Label Representation

This paper uses one-hot vectors to represent the multi-domain labels. For example, consider a set of images containing three attributes, e.g., blond hair, black hair and pale skin. For an input image of a person with blond hair and normal skin, the vector will be defined as $[1, 0, 0]$, meaning that blond hair is true, black hair and pale skin are considered to be false. Formally, for an image x, define a label vector $\tilde{c} = [c_1, c_2, \ldots, c_{c_{dim}}]$, where c_i corresponds to the boolean value of the i^{th} attribute for this image, and $c_{c_{dim}}$ is the total number of the attributes selected to be considered in the training process.

3.2. Max-Sliced Wasserstein Distance

Training StarGAN can be difficult as the training process is unstable and thus takes many steps to converge. To alleviate this problem, this project implements a recently proposed method, Max-Sliced Wasserstein Distance [3], which extends the sliced Wasserstein distance by only projecting the samples onto the max projection direction instead of a number of random directions, and thus reduces the projection complexity while preserving similar sample complexity. The max-sliced Wasserstein-2 distance between two distributions μ and ν is defined as

$$\textit{max}\text{-}\tilde{W}_2(\mu, \nu) = \left[\max_{\omega\in\Omega} W_2^2(\mu^\omega, \nu^\omega)\right]^{\frac{1}{2}}. \tag{22}$$

where Ω is a set of all directions on the unit sphere. When training GANs, we would like the generator to generate a fake dataset $\mathcal{F}$ to be as similar as the real dataset $\mathcal{D}$. Instead of measuring the distance between $\mathcal{D}$ and $\mathcal{F}$ directly, we can compute the distance in a feature space h where it's easy to tell samples from $\mathcal{D}$ and $\mathcal{F}$ apart. We first obtain the distributions of $\mathcal{D}$ and $\mathcal{F}$ in h: $h_{\mathcal{D}}$, $h_{\mathcal{F}}$, and then project them onto random directions $\omega \in \Omega$ to estimate the *max*-$\tilde{W}_2$ distance:

$$\textit{max}\text{-}\tilde{W}_2(h_{\mathcal{D}}, h_{\mathcal{F}}) = \max_{\omega\in\Omega} W_2(h_{\mathcal{D}}^{\omega}, h_{\mathcal{F}}^{\omega}). \tag{23}$$

Let the optimal or max projection direction be ω^*, Eq. (23) is equivalent to

$$\textit{max}\text{-}\tilde{W}_2(h_{\mathcal{D}}, h_{\mathcal{F}}) = W_2(h_{\mathcal{D}}^{\omega^*}, h_{\mathcal{F}}^{\omega^*}). \tag{24}$$

However, it's infeasible to explore all the directions in Ω to find out ω^*. In practice, we can adopt a discriminator to approximate ω^*. Let $\hat{\omega}$ represents for the weights of the last layer of the discriminator which can be used to approximate ω^* when trained for classification as distributions are well differentiated along $\hat{\omega}$. The parameters θ_d of feature space h can also be found using the discriminator's parameters $\hat{\theta}_d$. The penultimate layer of the discriminator now acts as the desired feature space h. Here we have:

$$\begin{aligned}\textit{max}\text{-}\tilde{W}_2(h_{\mathcal{D}}, h_{\mathcal{F}}) &= W_2(h_{\mathcal{D}}^{\hat{\omega}}, h_{\mathcal{F}}^{\hat{\omega}}) \\ &= W_2(\hat{\omega}^T h_{\mathcal{D}}, \hat{\omega}^T h_{\mathcal{F}}).\end{aligned} \tag{25}$$

In the setting of GANs, the generator, parameterized by θ_g, aims to minimize the *max*-$\tilde{W}_2$ distance in Eq. (25):

$$\min_{\theta_g} W_2(\hat{\omega}^T h_{\mathcal{D}}, \hat{\omega}^T h_{\mathcal{F}(\theta_g)}). \tag{26}$$

According to the work of Deshpande *et al.* [4], $W_2(\hat{\omega}^T h_{\mathcal{D}}, \hat{\omega}^T h_{\mathcal{F}(\theta_g)})$ can be approximated by sorting $\hat{\omega}^T h_{\mathcal{D}}$ and $\hat{\omega}^T h_{\mathcal{F}}$ respectively in a monotonically increasing order σ:

$$\begin{aligned}&W_2(\hat{\omega}^T h_{\mathcal{D}}, \hat{\omega}^T h_{\mathcal{F}(\theta_g)}) \\ &= \frac{1}{|\mathcal{D}|}\sum_i \|\hat{\omega}^T h_{\mathcal{D}\sigma_{\mathcal{D}}(i)} - \hat{\omega}^T h_{\mathcal{F}\sigma_{\mathcal{F}}(i)}\|_2^2.\end{aligned} \tag{27}$$

3.3. Max-Sliced StarGAN

To stabilize the training process and generate images with high quality, this project incorporates the max-sliced Wasserstein-2 distance in the StarGAN objective Eq. (19) by using it to replace the $\mathcal{L}_{adv}$ term which adopts the Jensen-Shannon divergence to form the "Max-Sliced StarGAN". The objective of the generator parameterized by θ_g now becomes

$$\begin{aligned}\mathcal{L}_G =& W_2(\hat{\omega}^T h_{\mathcal{D}}, \hat{\omega}^T h_{\mathcal{F}(\theta_g)}) \\ &+ \lambda_{cls}\mathcal{L}^f_{cls} + \lambda_{rec}\mathcal{L}_{rec}. \end{aligned} \tag{28}$$

Instead of passing the scalar outputs $\hat{\omega}^T h_{\mathcal{D}}, \hat{\omega}^T h_{\mathcal{F}(\theta_g)}$ to compute the max-sliced distance, we can also directly pass the real and fake distributions (vectors) in the feature space, i.e. $h_{\mathcal{D}}$ and $h_{\mathcal{F}(\theta_g)}$, to obtain the distance:

$$\begin{aligned}\mathcal{L}_G =& W_2(h_{\mathcal{D}}, h_{\mathcal{F}(\theta_g)}) \\ &+ \lambda_{cls}\mathcal{L}^f_{cls} + \lambda_{rec}\mathcal{L}_{rec}. \end{aligned} \tag{29}$$

Eq. (29) is used in our implementation. Comparison between the two variants can be found in Section 5.

The objective of the discriminator can just remain the same as Eq. (18). It's also possible to replace its $\mathcal{L}_{adv}$ term with other distance measurement, such as the Wasserstein distance. In our implementation, to further enhance the performance, we replace it with the Wasserstein GAN objective with gradient penalty [1] and have

$$\begin{aligned}\mathcal{L}_D =& \mathbb{E}_x[D_{\theta_d,src}(x)] - \mathbb{E}_{x,c}[D_{\theta_d,src}(G_{\theta_g}(x,c))] \\ &- \lambda_{gp}\mathbb{E}_{\hat{x}}[(\|\nabla_{\hat{x}} D_{\theta_d,src}(\hat{x})\|_2 - 1)^2] \\ &+ \lambda_{cls}\mathcal{L}^r_{cls}, \end{aligned} \tag{30}$$

where $\hat{x}$ is sampled uniformly along a straight line between a pair of a real and a generated images, and λ_{gp} is a hyper-parameter. The overall training process of the Max-Sliced StarGAN is summarized in Alg. 1.

4. Implementation

4.1. Generator Architecture

We implement the generator similar to the one mentioned in [18] which includes three parts: in-netwrok downsampling, residual blocks and in-network upsampling.

The in-netwrok downsampling part consists of two convolutional layers with the stride size of two, each followed by an instance normalization layer [16] and a ReLU activation. The same residual block [7] is applied repeatedly for six times. Each residual block contains a convolution layer followed by an instance normalization layer. A ReLU is added in between the first instance normalization and the second convolution. The output is then added to the original input. The in-netwrok upsampling part consists of two transposed convolutional layers of stride size two following by an instance normalization and a ReLU layer.

4.2. Discriminator Architecture

The discriminator is implemented based on PatchGANs. Unlike the generator, no normalization layer is used here. The first layer is a convolution layer followed by a leaky ReLU activation. Then, a block of two layers, one being convolution and the other being leaky ReLU, are repeated six times, each time with doubled number of output channels.

The discriminator has three outputs. The first is considered as the feature space, obtained from the abovementioned layers. The second is a scalar $D_{\theta_d,src}$ of the probability of the input being real, generated by a 1×1 convolution layer followed. The last one is a vector $D_{\theta_d,cls}$ indicating the probability of the input being classified to the target domains, obtained from a $c_{dim} \times c_{dim}$ convolution layer.

5. Experiments

This section first describes the experimental setups, including baseline models, dataset, hyper-parameter settings and hardware used. Experiments are grouped into two categories based on the models being investigated: the *Max-Sliced StarGAN* model itself and with the *baseline models*. The former examines the behavior of the Max-Sliced StarGAN trained with different sample sizes, different generator update schemes (i.e., number of discriminator updates per generator update) and different estimating methods, i.e., Eq. (28) and (29). The latter investigates the different performance of all these models under same setups.

5.1. Experimental Setups

5.1.1 Baseline models.

We use three baseline models, namely, the *Original StarGAN*, *Sliced StarGAN* and *Sliced StarGAN with feature transformation* to compare the performance of StarGAN under different objective fuctions, and to better understand how max-sliced StarGAN behaves compared to the others. The three models, as well as the *max-sliced StarGAN*, are all trained multiple times using the same network architectures and hyper-parameters.

Original StarGAN. The original StarGAN replaces the $\mathcal{L}_{adv}$ term in both Eq. (18) and Eq. (19) with the Wasserstein GAN objective along with gradient penalty. The objective of the discriminator is the same as Eq. (30), while the generator's objective becomes

Algorithm 1 Max-Sliced StarGAN Algorithm.

Input
  m sample size
  θ_g parameters of generator G_{θ_g}
  θ_d, ω_d parameters of discriminator D_{θ_d}
  α learning rate
  K number of discriminator updates per generator update
  N number of training steps
for step $i \leftarrow 0$ **to** N **do**
  sample images $\{x^{(i)}\}_{i=1}^m \in \mathcal{D}$ and their original domain labels $\{c^{'(i)}\}_{i=1}^m \in \mathcal{C}^{'}$
  generate target domain labels $\{c^{(i)}\}_{i=1}^m \in \mathcal{C}$
  generate fake images $\{G_{\theta_g}(x^{(i)}, c^{(i)})\}_{i=1}^m \in \mathcal{F}$
  update θ_d, ω_d:
    compute $\mathcal{L}_{adv}(D_{\theta_d,src}, \mathcal{D}, \mathcal{F})$, $\mathcal{L}_{cls}^r(D_{\theta_d,cls}, \mathcal{D}, \mathcal{C}^{'})$
    return $\mathcal{L}_D = -\mathcal{L}_{adv} + \lambda_{cls}\mathcal{L}_{cls}^r$
    $(\hat{\theta_d}, \hat{\omega}) \leftarrow (\hat{\theta_d}, \hat{\omega}) - \alpha\nabla_{\theta_d,\omega}\mathcal{L}_D$
  if $(i+1) \mod K = 0$ **then**
    compute max-sliced Wasserstein Distance *max*-$W_2(\hat{\omega}^T h_{\mathcal{D}}, \hat{\omega}^T h_{\mathcal{F}(\theta_g)})$
      obtain $\hat{\omega}^T h_{\mathcal{D}} = D_{\theta_d,src}(\mathcal{D})$
      obtain $\hat{\omega}^T h_{\mathcal{F}} = D_{\theta_d,src}(\mathcal{F})$
      sort $\hat{\omega}^T h_{\mathcal{D}}$ and $\hat{\omega}^T h_{\mathcal{F}}$ to obtain $\sigma_{\mathcal{D}}$, $\sigma_{\mathcal{F}}$
      return $\mathcal{L}_{adv} = \sum_i \|\hat{\omega}^T h_{\mathcal{D}\sigma_{\mathcal{D}}(i)} - \hat{\omega}^T h_{\mathcal{F}\sigma_{\mathcal{F}}(i)}\|_2^2$
    update θ_g:
      compute $\mathcal{L}_{cls}^f(D_{\theta_d,cls}, \mathcal{F}, \mathcal{C})$, $\mathcal{L}_{rec}(\mathcal{D}, \mathcal{F}, \mathcal{C}^{'})$
      return $\mathcal{L}_G = \mathcal{L}_{adv} + \lambda_{cls}\mathcal{L}_{cls}^f + \lambda_{rec}\mathcal{L}_{rec}$
      $\theta_g \leftarrow \theta_g - \alpha\nabla_{\theta_g}\mathcal{L}_G$
  end if
end for

$$\mathcal{L}_G = -\mathbb{E}_{x,c}[D_{\theta_d,src}(G_{\theta_g}(x,c))] + \lambda_{cls}\mathcal{L}_{cls}^f + \lambda_{rec}\mathcal{L}_{rec}. \quad (31)$$

Sliced StarGAN. In order to understand the improvement of max-sliced Wasserstein distance over its predecessors, we also build a "Sliced StarGAN" by adopting the sliced Wasserstein distance [4] to replace the $\mathcal{L}_{adv}$ term in (18). The objective of the discriminator is the same as Eq. (30), and the objective of the generator has been changed to

$$\mathcal{L}_G = \frac{1}{|\hat{\Omega}|}\sum_{\omega\in\hat{\Omega}} W_2^2(\mathcal{D}^\omega, \mathcal{F}^\omega) + \lambda_{cls}\mathcal{L}_{cls}^f + \lambda_{rec}\mathcal{L}_{rec}. \quad (32)$$

Sliced StarGAN with feature transformation. Paper [4] also mentions that instead of estimating the distance with datasets $\mathcal{D}$ and $\mathcal{F}$ directly, we can transform them to an adversarially learnt feature space, $h_{\mathcal{D}}$ and $h_{\mathcal{F}}$. The parameters θ_g of h is learnt via a discriminator. In this case, $\mathcal{L}_G$ is

$$\mathcal{L}_G = \frac{1}{|\hat{\Omega}|}\sum_{\omega\in\hat{\Omega}} W_2^2(h_{\mathcal{D}}^\omega, h_{\mathcal{F}}^\omega) + \lambda_{cls}\mathcal{L}_{cls}^f + \lambda_{rec}\mathcal{L}_{rec}, \quad (33)$$

where Ω is a set of randomly chosen projection directions, and $h_{\mathcal{D}}^\omega$ and $h_{\mathcal{F}}^\omega$ are the projections of $h_{\mathcal{D}}$ and $h_{\mathcal{F}}$ onto direction ω.

5.1.2 Dataset, Hyper-parameters and Hardware

Dataset. We train the models on the CelebFaces Attributes (CelebA) dataset [13]. Table 1 shows some insights for this dataset.

Table 1. Insights for the CelebA dataset

Total #samples	#attributes	Image size
$202,599$	40	178×218

The original 178×218 images are cropped around the center to 178×178 then resized to 128×128. The test

set contains 2000 randomly selected images, while the remaining forms the training set. Random horizontal flip is applied to the training images. For the experiments, five domains are constructed using the following attributes: hair color (*blond*), gender (*male/female*) and age (*young/old*).

Hyper-parameters. Some hyper-parameters adopt the same values for all experiments (Table 2), while the others vary according to the experimental setups.

Hardware. Training is conducted on a GeForce GTX 1080Ti GPU.

5.2. Experimental Results on Max-Sliced StarGAN

5.2.1 Experiments on Sample Size

It has been proved by [3] that Max-Sliced GAN has compelling sample complexity, meaning that it can estimate the max-sliced Wasserstein distance using only a small number of samples. This set of experiments is designed to investigate such property. Fig. 2 shows the generated images using sample sizes 16, 32 and 64 respectively. There is no much difference in the quality of the generated images for different samples sizes. The results are visually good even with a sample size as small as 16. Fig. 3 shows the value of the $\mathcal{L}_{adv}$ (i.e., the max-sliced Wasserstein distance) for the generator as the training progresses. We can see that the loss decreases with the number of samples used for its estimation, but the difference is small, which also indicates that a small sample size should be sufficient to approximate the distance.

Figure 3. Training loss $\mathcal{L}_{adv}$ (i.e., the max-sliced Wasserstein distance) for the generator with different sample size.

5.2.2 Experiments on Generator Update Schemes

To show that the Max-Sliced StarGAN is insensitive to hyper-parameters, we experiment on two generator update schemes: (1) performing one generator update after ve discriminator updates, and (2) performing one generator update after one discriminator update. The results are shown in Fig. 4. The images generated by scheme (2) are clearer, but still, the difference is small. The loss trends in Fig. 5 are also similar. In this case, either one of the two schemes can provide reasonable results, thus saving us from the tedious work of hyper-parameter tuning.

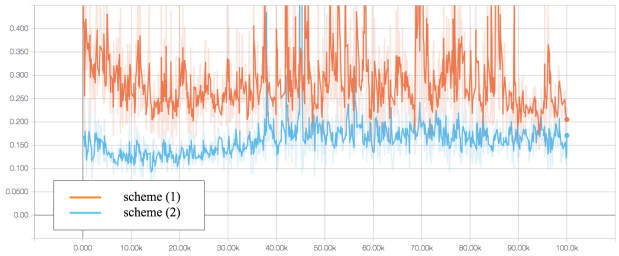

Figure 5. Training loss $\mathcal{L}_{adv}$ (i.e., the max-sliced Wasserstein distance) for generator update schemes (1) and (2).

5.2.3 Experiments on Estimating Methods

Section 3.3 mentions two ways to estimate the max-sliced Wasserstein distance, namely, (1) using *scalar* outputs $\hat{\omega}^T h_{\mathcal{D}}$ and $\hat{\omega}^T h_{\mathcal{F}(\theta_g)}$, Eq. (28); (2) using *vector* outputs $h_{\mathcal{D}}$ and $h_{\mathcal{F}(\theta_g)}$, Eq. (29). We want to see which one performs better. From Fig. 6 we can see that method (1) produces blurred images compared to method (2), and that method (2) works much better in terms of converting to blond hair. Fig. 7 shows that the loss trend of method (2) is more stable. Intuitively, besides the max projection direction, method (2) also provides the information of some other important directions, making the estimation more accurate and thus being able to generate better results.

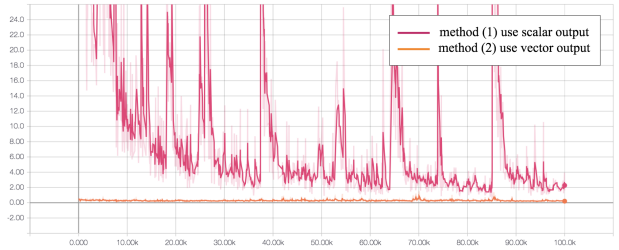

Figure 7. Training loss $\mathcal{L}_{adv}$ (i.e., the max-sliced Wasserstein distance) for different methods to estimate the max-sliced Wasserstein distance.

5.3. Experimental Results on Different Models

5.3.1 Comparing the Original StarGAN with Max-Sliced StarGAN

Fig. 8 shows the images generated by the original StarGAN and the Max-Sliced StarGAN under the same hyper-parameter settings. The original StarGAN works better when translating the *aged* attribute, while the Max-Sliced StarGAN produces better results in terms of *gender*. In general, both of the two can generate images of high quality. In this sense, the advantage of using Max-Sliced StarGAN is that it is insensitive to the values of the hyper-parameters and thus will be easier to train in practice.

Table 2. Hyper-parameter setting

Optimizer	Learning rate	λ_{cls}	λ_{rec}	λ_{gp}	#Steps
Adam ($\beta_1 = 0.5, \beta_2 = 0.999$)	0.0001	1	10	10	100,000

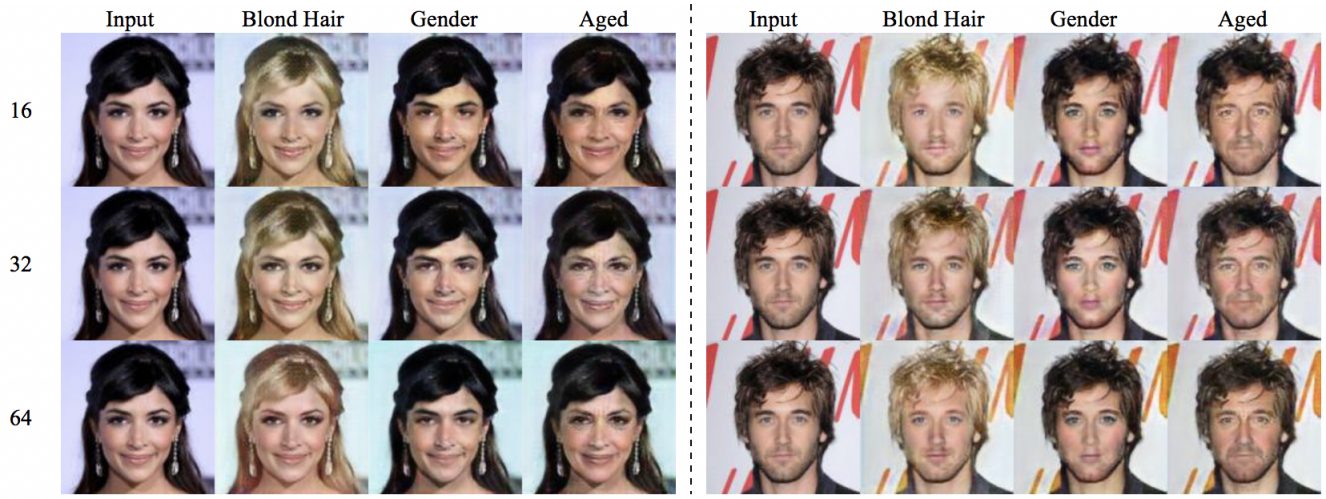

Figure 2. Facial attribute transfer results for sample sizes of 16, 32 and 64. The left and right parts show two sets of generated images. For each part, the first column shows the input image, and the next three columns show the single attribute transfer results.

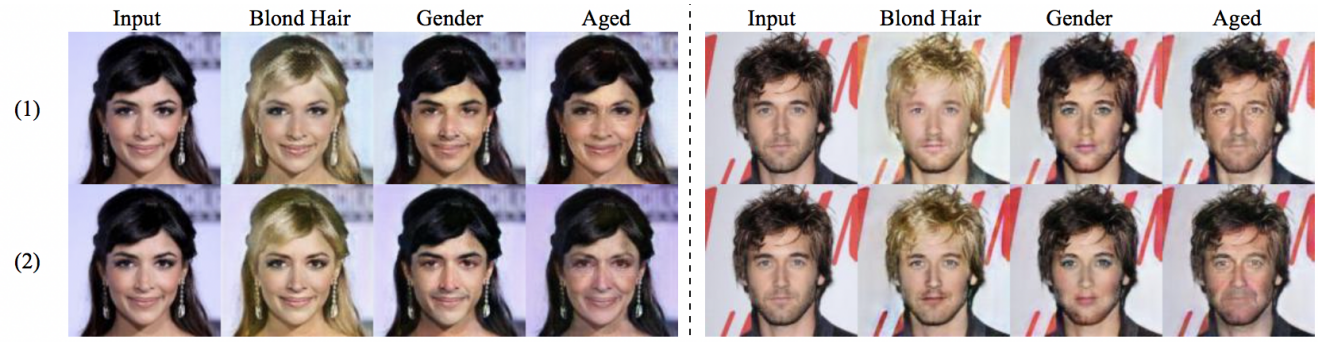

Figure 4. Facial attribute transfer results for two generator update schemes: (1) performing one generator update after ve discriminator updates, and (2) performing one generator update after one discriminator update.

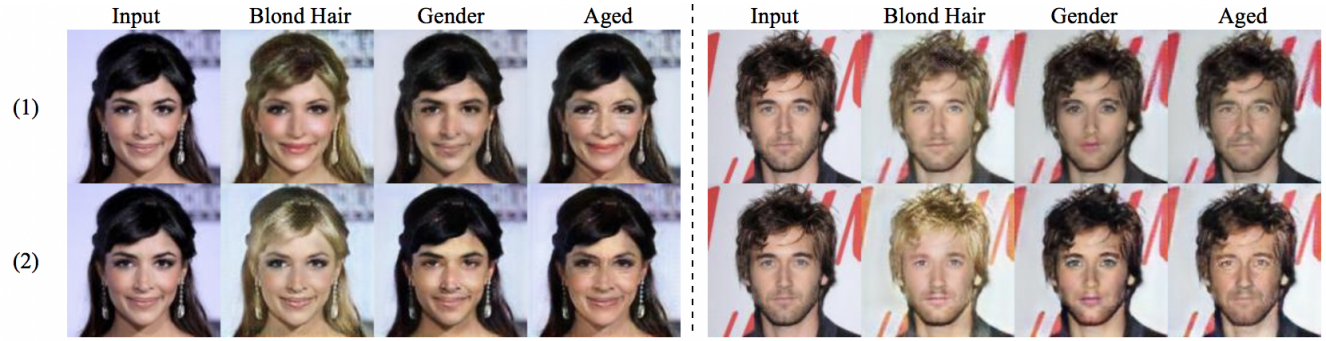

Figure 6. Facial attribute transfer results for two estimating methods: (1) using scalar outputs $\hat{\omega}^T h_{\mathcal{D}}$ and $\hat{\omega}^T h_{\mathcal{F}(\theta_g)}$, Eq. (28); and (2) using vector outputs $h_{\mathcal{D}}$ and $h_{\mathcal{F}(\theta_g)}$, Eq. (29).

5.3.2 Comparing Sliced StarGAN with Max-Sliced StarGAN

Images generated by the Sliced StarGAN model with different number of projection directions and Max-Sliced Star-

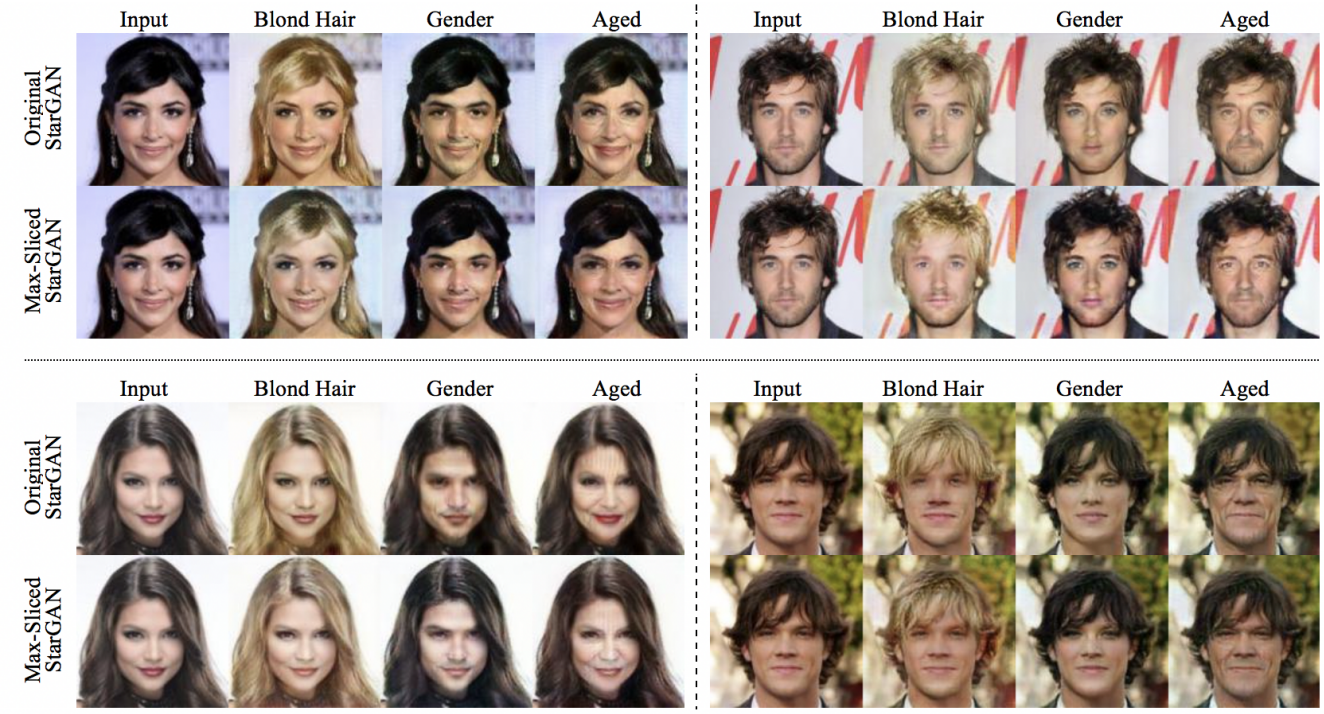

Figure 8. Facial attribute transfer results for the original StarGAN and the Max-Sliced StarGAN. Four sets of generated images are shown from the top left to the bottom right sections. For each section, the first column shows the input image, and the next three columns show the single attribute transfer results.

GAN are shown in Fig. 9. The Sliced StarGAN fails to generate reasonable images, indicating that it's hard for it to scale up to high-dimensional distributions. This is intuitive since paper [4] also suggests using a discriminator to transform the data samples before passing them to compute the sliced Wasserstein distance.

5.3.3 Comparing Sliced StarGAN using Feature Transformation with Max-Sliced StarGAN

As mentioned in Section 5.3.2, we may need to use a discriminator to help the Sliced StarGAN scale up to high-dimensional data. Fig. 10 shows the results for Sliced StarGAN with Feature Transformation (Sliced StarGAN-F) and Max-Sliced StarGAN. Compared to the results in Fig. 9, Sliced StarGAN-F works much better than Sliced StarGAN itself. Sliced StarGAN-F even generates images as good as the Max-Sliced StarGAN. It's possible that the role of the discriminator is more important for our multi-domain image-to-image translation task than for other tasks, and therefore whether to use a discriminator to transform the samples or not makes such a huge difference.

Although the Sliced StarGAN-F model can produce high-quality results, it takes much longer to train than the Max-Sliced StarGAN. The average time needed for each generator update is shown in Table 3. Max-Sliced StarGAN is the fastest one because it needs orders of magnitude fewer projection directions (i.e., smaller projection complexity) and thus fewer sorting operations.

Table 3. Comparison of time required for each generator update.

Method	Time (s)
Sliced StarGAN-F, 100	1.05
Sliced StarGAN-F, 1000	2.16
Sliced StarGAN-F, 10,000	5.25
Max-Sliced StarGAN	**1.00**

5.4. Other Experiments

To test the scalability of the Max-Sliced StarGAN, more attributes are selected to train the model, including *blond hair*, *brown hair*, *rosy cheeks*, *gender*, *mouth slightly open/closed*, *smiling/not smiling*. The results are shown in Fig. 1. We can see the model is able to generate images with high quality for all the selected attributes.

6. Conclusions

In this project, we propose and implement "Max-Sliced StarGAN" which combines the StarGAN objective with max-sliced Wasserstein distance to improve the training process of the original StarGAN. We summarize the the-

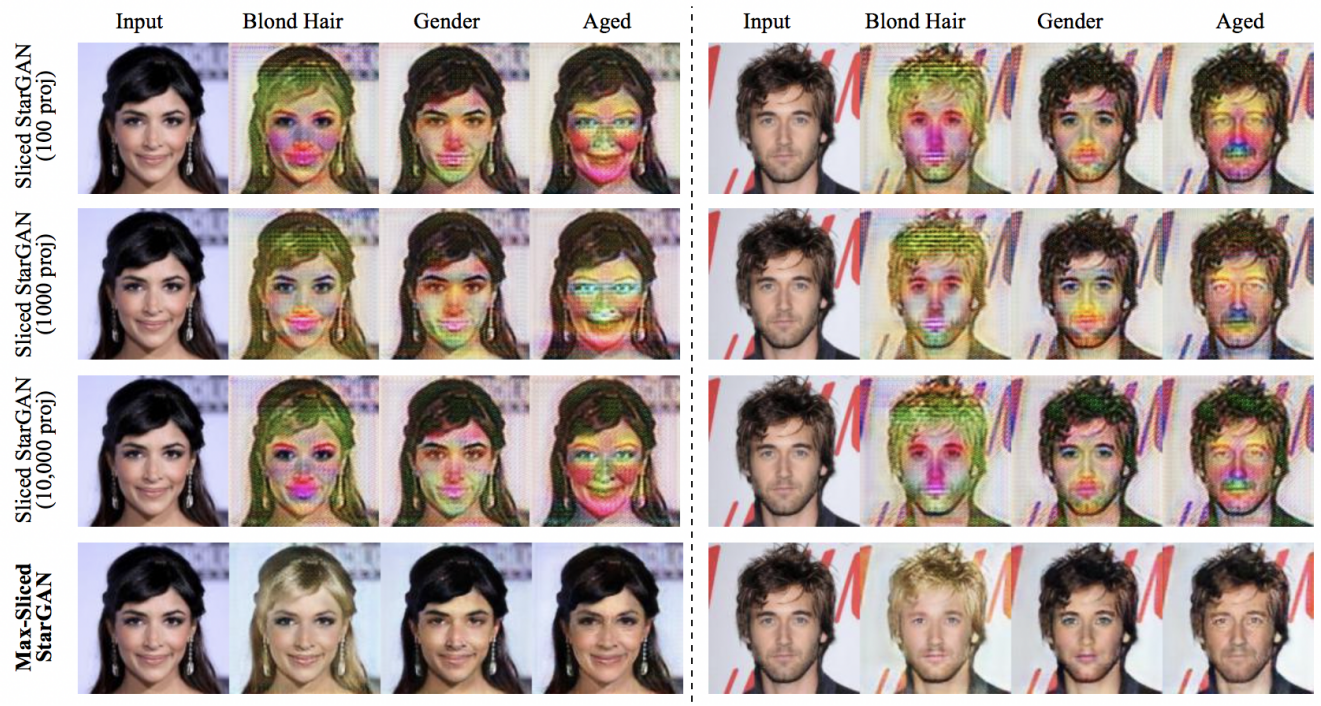

Figure 9. Facial attribute transfer results for the Sliced StarGAN and the Max-Sliced StarGAN. Images from the first to the last row are generated by the four models respectively: Sliced StarGAN with 100 projections, Sliced StarGAN with 1000 projections, Sliced StarGAN with 10,000 projections and Max-Sliced StarGAN.

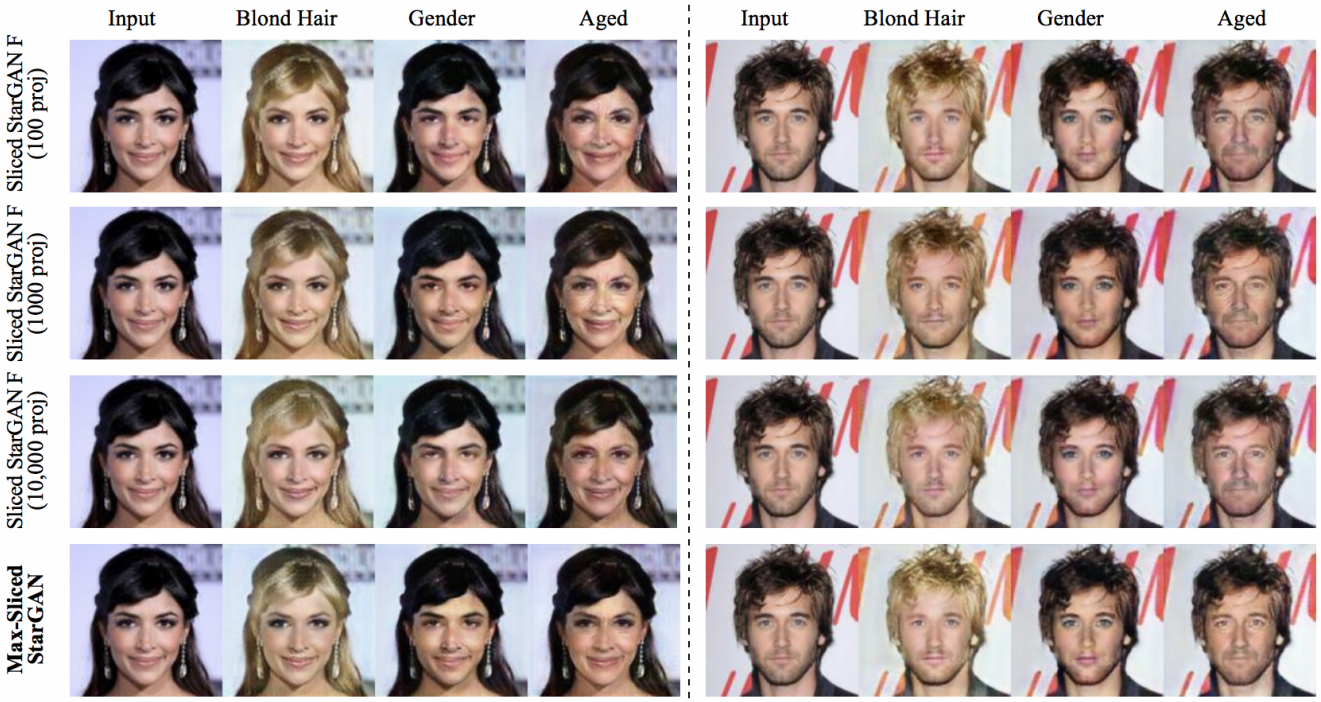

Figure 10. Facial attribute transfer results for the Sliced StarGAN with Feature Transformation (Sliced StarGAN-F) and Max-Sliced StarGAN. Images from the first to the last row are generated by the four models respectively: Sliced StarGAN-F with 100 projections, Sliced StarGAN-F with 1000 projections, Sliced StarGAN-F with 10,000 projections and Max-Sliced StarGAN.

oretical details of the formulations of the original StarGAN, sliced Wasserstein distance, max-sliced Wasserstein distance, as well as the architectures of the generator and the discriminator. We also introduce the full objective of the Max-Sliced StarGAN, along with the overall training algorithm. Empirical results show that Max-Sliced StarGAN achieves similar results as the original model, and outperforms the Sliced StarGAN in terms of image quality, training time and projection complexity. The Max-Sliced StarGAN also enjoys compelling sample complexity and only requires slight hyper-parameter tuning, which may be more useful in practice.

Acknowledgement

This work is an independent study project under the guidance of Professor Ruoyu Sun at University of Illinois at Urbana-Champaign. I would like to express my sincere gratitude to his kindness, patience and encouragement. During the project, Professor Sun has offered valuable suggestions and inputs which helps me a lot. I also want to thank him for providing the GPU server which highly accelerates the entire project.

References

[1] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein gan. *arXiv preprint arXiv:1701.07875*, 2017.

[2] Yunjey Choi, Minje Choi, Munyoung Kim, Jung-Woo Ha, Sunghun Kim, and Jaegul Choo. Stargan: Unified generative adversarial networks for multi-domain image-to-image translation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8789–8797, 2018.

[3] Ishan Deshpande, Yuan-Ting Hu, Ruoyu Sun, Ayis Pyrros, Nasir Siddiqui, Sanmi Koyejo, Zhizhen Zhao, David Forsyth, and Alexander Schwing. Max-sliced wasserstein distance and its use for gans. *arXiv preprint arXiv:1904.05877*, 2019.

[4] Ishan Deshpande, Ziyu Zhang, and Alexander G Schwing. Generative modeling using the sliced wasserstein distance. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3483–3491, 2018.

[5] Vincent Dumoulin, Ishmael Belghazi, Ben Poole, Olivier Mastropietro, Alex Lamb, Martin Arjovsky, and Aaron Courville. Adversarially learned inference. *arXiv preprint arXiv:1606.00704*, 2016.

[6] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[7] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[8] Xun Huang, Yixuan Li, Omid Poursaeed, John Hopcroft, and Serge Belongie. Stacked generative adversarial networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5077–5086, 2017.

[9] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei 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.

[10] Taeksoo Kim, Byoungjip Kim, Moonsu Cha, and Jiwon Kim. Unsupervised visual attribute transfer with reconfigurable generative adversarial networks. *arXiv preprint arXiv:1707.09798*, 2017.

[11] Christian Ledig, Lucas Theis, Ferenc Huszár, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, et al. Photo-realistic single image super-resolution using a generative adversarial network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4681–4690, 2017.

[12] Ming-Yu Liu, Thomas Breuel, and Jan Kautz. Unsupervised image-to-image translation networks. In *Advances in Neural Information Processing Systems*, pages 700–708, 2017.

[13] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *Proceedings of the IEEE international conference on computer vision*, pages 3730–3738, 2015.

[14] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014.

[15] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

[16] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. Instance normalization: The missing ingredient for fast stylization. *arXiv preprint arXiv:1607.08022*, 2016.

[17] Jun-Yan Zhu, Philipp Krähenbühl, Eli Shechtman, and Alexei A Efros. Generative visual manipulation on the natural image manifold. In *European Conference on Computer Vision*, pages 597–613. Springer, 2016.

[18] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, pages 2223–2232, 2017.