Semi-supervised Cycle-GAN for face photo-sketch translation in the wild

Chaofeng Chen\textsuperscript{a}, Wei Liu\textsuperscript{b}, Xiao Tan\textsuperscript{c}, Kwan-Yee K. Wong\textsuperscript{a}

\textsuperscript{a}The Department of Computer Science, The University of Hong Kong, Pokfulam Road, Hong Kong SAR, China
\textsuperscript{b}SenseTime Research, Shenzhen 518000, China
\textsuperscript{c}Baidu Inc, Department of Computer Vision Technology, Beijing 100085, China.

Abstract

The performance of face photo-sketch translation has improved a lot thanks to deep neural networks. GAN based methods trained on paired images can produce high-quality results under laboratory settings. Such paired datasets are, however, often very small and lack diversity. Meanwhile, Cycle-GANs trained with unpaired photo-sketch datasets suffer from the \textit{steganography} phenomenon, which makes them not effective to face photos in the wild. In this paper, we introduce a semi-supervised approach with a noise-injection strategy, named Semi-Cycle-GAN (SCG), to tackle these problems. For the first problem, we propose a \textit{pseudo sketch feature} representation for each input photo composed from a small reference set of photo-sketch pairs, and use the resulting \textit{pseudo pairs} to supervise a photo-to-sketch generator $G_{p2s}$. The outputs of $G_{p2s}$ can in turn help to train a sketch-to-photo generator $G_{s2p}$ in a self-supervised manner. This allows us to train $G_{p2s}$ and $G_{s2p}$ using a small reference set of photo-sketch pairs together with a large face photo dataset (without ground-truth sketches). For the second problem, we show that the simple noise-injection strategy works well to alleviate the \textit{steganography} effect in SCG and helps to produce more reasonable sketch-to-photo results with less overfitting than fully supervised approaches. Experiments show that SCG achieves competitive performance on public benchmarks and superior results on photos in the wild.

1. Introduction

Face photo-sketch translation can be considered as a specific type of image translation between an input face photo and sketch. It has a wide range of applications. For example, police officers often have to identify criminals from sketch images, sketch images are also widely used in social media.

There are lots of works on face photo-sketch translation. Traditional methods are based on patch matching. They usually divide an input photo into small patches and find corresponding sketch patches in a reference dataset composed of well-aligned photo-sketch pairs. In this way, they (Song et al., 2014; Zhou et al., 2012; Zhu et al., 2017b; Wang and Tang, 2009) achieved pleasant results without explicitly modeling the mapping between photos and sketches, which is highly non-linear and difficult. However, sketches generated by these methods are often over-smoothed and lack subtle contents, such as ears in Fig. 1(a)(ii). Moreover, these methods are usually very slow due to the time-consuming patch matching and optimization process. Recent methods based on Convolutional Neural Networks (CNNs) try to directly learn the translation between photos and sketches. However, results produced by simple CNNs are usually blurry (see Fig. 1(a)(iii)), and Generative Adversary Networks (GAN) (Goodfellow et al., 2014) often generate unpleasant artifacts (see Fig. 1(a)(iv)). Finally, due to the lack of large training datasets, these learning-based approaches cannot generalize well to photos in the wild.

Latest works (Yu et al., 2020; Wang et al., 2017a; Fang et al., 2020) utilize Cycle-GAN (Zhu et al., 2017a) to learn the translation between photos and sketches. Cycle-GAN is designed for unpaired translation between different domains. Styles are translated with a discriminator loss and content consistency is guaranteed with a
cycle-consistency loss. However, the cycle-consistency loss used to constrain content is weak, and therefore these methods still require paired data to calculate an MSE (mean squared error) loss between the prediction and ground truth. In experiments, we observed that models directly using unpaired Cycle-GAN fail to preserve facial content (see Fig. 2). This is because Cycle-GAN learns to “hide” information of the input photos in the generated sketches as invisible high-frequency noise, also called steganography (Chu et al., 2017; Bashkirova et al., 2019). It makes it difficult to learn face photo-sketch translation with Cycle-GAN in an unpaired setting. Please refer to Sec. 3.1 for a detailed discussion.

In this paper, we propose a semi-supervised learning framework based on Cycle-GAN, named Semi-Cycle-GAN (SCG), for face photo-sketch translation. To ensure content consistency, we introduce a novel pseudo sketch feature (PSF) to supervise the training of the photo-to-sketch generator $G_{p2s}$. Figure 1(b) shows the pipeline to construct PSF for an input photo without ground truth sketch. Suppose we have a small reference set of photo-sketch pairs and a large face photo dataset without ground-truth sketches. Similar to the exemplar-based approach, we first subdivide an input photo and its VGG-19 (Simonyan and Zisserman, 2014) feature maps into overlapping patches. We then match (in the feature space) these photo patches with the photo patches in the reference set and compose a PSF from the VGG-19 features of the corresponding sketch patches in the reference set. We next supervise the training of $G_{p2s}$ using the MSE between the feature maps of the generated sketch and the PSF of the input photo. The motivation for PSF is that styles of sketches are consistent for facial components with similar shapes. To find corresponding sketch patches for an input photo, we only need to match the facial components with similar shapes in the reference set. Since the shapes of facial components are limited, a small reference set with a few hundreds of photo-sketch pairs is often sufficient for this purpose. However, the same approach cannot be used for training the sketch-to-photo generator $G_{s2p}$ because sketch patches with the same shape may give rise to photo patches of many different styles. Instead, we follow Cycle-GAN and use sketches generated by $G_{p2s}$ to train $G_{s2p}$ in a self-supervised manner. Although the proposed PSF helps to constrain the contents of the output sketches from $G_{p2s}$, we find steganography still exists and is quite harmful to the training of $G_{s2p}$ because it learns to cheat. To solve this problem, we employ a simple noise-injection strategy to disrupt the invisible steganography and force $G_{s2p}$ to learn better translation from sketches to photos. Although the inputs of $G_{s2p}$ are noisy during training, we observed that $G_{s2p}$ can handle clean sketches quite well during testing due to the intrinsic image prior of CNNs (Ulyanov et al., 2017). Experiments demonstrated that the noise-injection strategy can largely benefit the training of $G_{s2p}$.

In summary, our main contributions are:

- We propose a semi-supervised learning framework based on Cycle-GAN, named Semi-Cycle-GAN, for face photo-sketch translation.
- The proposed pseudo sketch feature (PSF) allows us to train $G_{p2s}$ using a small reference set of photo-sketch pairs together with a large face photo dataset without ground-truth sketches. This enables our networks to generalize well to face photos in the wild.
- We introduce a self-supervised approach to train the sketch-to-photo generator $G_{s2p}$ without using real sketches through cycle-consistency. In particular, we find that cycle-consistency loss suffers greatly from invisible steganography, and the simple noise-injection strategy helps a lot to improve it.

A preliminary version of this work appeared in Chen et al. (2018a). We extend it in five aspects: (1) we combine our previously proposed semi-supervised learning framework with cycle-consistency to conduct both photo-to-sketch and sketch-to-photo translations; (2) we find that cycle-consistency loss suffers greatly from invis-
ible steganography, and the simple noise-injection strategy helps a lot to improve it; (3) we add a Gram matrix loss based on PSF which provides second-order style supervision; (4) we provide more comparisons with recently proposed methods such as PS2MAN (Wang et al., 2017a), SCA-GAN (Yu et al., 2020), Knowledge Transfer (Zhu et al., 2019) (denoted as KT), GENRE (Li et al., 2021) and PANet (Nie et al., 2022); (5) we adopt recent perceptual oriented metrics (i.e., LPIPS (Zhang et al., 2018a), DISTS (Ding et al., 2020), and FID (Heusel et al., 2017)) for performance evaluation. In particular, our extended framework shows better performance than Chen et al. (2018a).

2. Related Works

Exemplar-Based Methods Since photos and sketches are in two different modalities, it is not straightforward to learn a direct mapping between them. Tang and Wang (2003) introduced eigentransformation to perform exemplar matching between photos and sketches by assuming a linear transformation between them. Liu et al. (2005) noticed that the linear assumption holds better locally, and proposed the patch-based local linear embedding (LLE). Wang and Tang (2009) introduced a multi-scale markov random fields (MRF) model to resolve inconsistency between adjacent patches. Zhang et al. (2010) extended MRF with shape priors and SIFT features. Zhou et al. (2012) proposed the markov weight fields (MWF) model to synthesize new sketch patches that are not present in the training dataset. Gao et al. (2012) proposed to adaptively determine the number of candidate patches by sparse representation. Wang et al. (2013) proposed a transductive model which optimizes the MRF-based photo-to-sketch and sketch-to-photo models simultaneously. A few works such as Song et al. (2014) and Wang et al. (2017b) tried to improve the efficiency of the sketch generation procedure. Recent methods Zhu et al. (2017b) and Chen et al. (2018b) used features from a pretrained CNN network as the patch feature to replace unrobust traditional features.

Learning-Based Methods In recent years, CNN based methods have become the mainstream. Zhang et al. (2015) proposed to directly translate the input photo to sketch with a fully convolution network (FCN). Zhang et al. (2017) introduced a branched fully convolutional network (BFCN) which is composed of a content branch and a texture branch with different losses. Wang et al. (2017d) improved the vanilla GAN with multi-scale structure for face photo-sketch translation. Wang et al. (2017a) introduced multi-scale discriminators to CycleGAN. Zhang et al. (2018b) proposed multi-domain adversarial learning in the latent feature space of faces and sketches. Fang et al. (2020) introduced VGG-based feature identity loss to better preserve identity information. Yu et al. (2020) extended Cycle-GAN (Zhu et al., 2017a) with facial parsing map and proposed the SCA-GAN. Some recent popular works (Yi et al., 2019, 2020b, a; Huang et al., 2021; Li et al., 2020) consider a different kind of portrait style with simple thick lines and achieve pleasant results. However, it is out of the scope of this paper and hence we do not compare with them in this work.

3. Semi-Cycle-GAN with noise-injection

3.1. Steganography in Cycle-GAN

In this section, we first give a brief review of the unpaired Cycle-GAN for face photo-sketch translation. We then show how Cycle-GAN cheats with invisible steganography. Given a photo set \( P \) and a sketch set \( S \), Cycle-GAN learns two generators: a photo-to-sketch generator \( G_{p2s} \) that maps photo \( p \in P \) to sketch \( s \in S \), and a symmetric sketch-to-photo generator \( G_{s2p} \) that maps sketch \( s \in S \) to photo \( p \in P \). Two discriminators \( D_s \) and \( D_p \) are used to minimize the style differences between the generated and real sketches (i.e., \( \hat{s} \) and \( s \)) and between generated and real photos (i.e., \( \hat{p} \) and \( p \)). Cycle-consistency losses are used to constrain content in-

Figure 2: Illustration of steganography when training Cycle-GAN with unpaired data.
formation in photo-sketch translation and are given by:

\[
\begin{align*}
L_{\text{Cycle}} &= E[\|G_{\hat{s}2p}(G_{p2s}(p)) - p\|], \\
L_{\text{Cycle}} &= E[\|G_{p2s}(G_{\hat{s}2p}(s)) - s\|].
\end{align*}
\] (1)

Note that Eq. (1) does not impose a direct constraint over \(G_{p2s}(p)\) and \(G_{\hat{s}2p}(s)\), and this leads to a large solution space. Chu et al. (2017) pointed out that Cycle-GAN tends to hide invisible steganography in the outputs to satisfy the cycle-consistency constraint when two domains have different complexity. Specifically, in face photo-sketch translation, the photo domain \(P\) is much more complex than the sketch domain \(S\), which makes learning of \(G_{\hat{s}2p}\) much more difficult than \(G_{p2s}\). As a consequence, when we train \(G_{\hat{s}2p}\) and \(G_{p2s}\) in an unpaired manner with cycle-consistency, the networks tend to learn a trivial solution by cheating with steganography rather than learning the desired translation networks. Figure 4 provides a theoretical illustration of steganography effect and how noise-injection helps to solve this problem. Given that the high-dimensional photo domain \(P\) contains a more extensive range of information in comparison to the low-dimensional sketch domain \(S\), it poses a considerable challenge for the \(G_{\hat{s}2p}\) network to reconstruct the missing information (e.g., hair color) from grayscale input sketches. The networks tend to learn to conceal the extra information in a low-amplitude signal (i.e., the red curve) to facilitate seamless reconstruction of the high-dimensional signal while retaining the appearance of the sketch signal. Since steganography needs to be low-amplitude signals, it is vulnerable to disruption through the application of random noise. In addition, \(G_{\hat{s}2p}\) with random noise will act as a normal GAN to complement missing information in the low-dimensional sketch domain.

Figure 2 shows some example results when training Cycle-GAN with unpaired dataset. We can observe from the left half of Fig. 2 (photo→sketch→photo) that the lost letter in the generated sketch was recovered in the reconstructed photo, and extra glasses in the sketch were removed. A similar phenomenon also appears in the right half (sketch→photo→sketch). Closely related works including Chu et al. (2017) and Bashkirova et al. (2019) focus on how to avoid adversarial attack that is usually invisible in the images. We, on the other hand, are the first to study the visual effects brought by such steganography in face photo-sketch translation, which have been ignored by previous works based on Cycle-GAN (Yu et al., 2020; Wang et al., 2017a).

To solve this problem, we propose the Semi-Cycle-GAN framework for face photo-sketch translation. As shown in Fig. 3(b), our framework is composed of four networks, namely \(G_{p2s}\), \(G_{\hat{s}2p}\), \(D_s\), and \(D_p\). Unlike Cycle-GAN, we do not use the bidirectional cycle-consistency loss as a content constraint. We use PSF loss (see Sec. 3.2 for details) to supervise the training of \(G_{p2s}\) and cycle-consistency loss with noise-injection to supervise the training of \(G_{\hat{s}2p}\). In this manner, we can train our Semi-Cycle-GAN using a small paired photo-sketch dataset together with a large face dataset.

\[\text{3.2. Pseudo Sketch Feature}\]

Given a test photo \(p\), our target is to construct a pseudo sketch feature \(\Phi^l(p)\) as the supervision using the reference set \(\mathcal{R}(\{p^l_i, s^l_i\})_{i=1}^N\), where \(p^l_i\) and \(s^l_i\) are photos and sketches in \(\mathcal{R}\). We first use a pretrained VGG-19 network to extract a feature map for \(p\) at the \(l\)-th layer, denoted as \(\Phi^l(p)\). Similarly, we can get the feature maps for photos and sketches in the reference dataset, i.e., \(\{\Phi^l(p^l_i)\}_{i=1}^N\) and \(\{\Phi^l(s^l_i)\}_{i=1}^N\). The feature maps are then subdivided into \(k \times k\) patches for the following feature patch matching process. For simplicity, we denote a vectorized representation of a \(k \times k\) patch centered at a point \(j\) of
\( \Phi'(p) \) as \( \Psi_j(\Phi'(p)) \), and the same definition applies to \( \Psi_j(\Phi'(p)_{S_i}) \) and \( \Psi_j(\Phi'(s_{S_i})) \). For each patch \( \Psi_j(\Phi'(p)) \), where \( j = 1, 2, \ldots, m' \) and \( m' = (H'-k+1) \times (W'-k+1) \) with \( H' \) and \( W' \) being the height and width of \( \Phi'(p) \), we find its best match \( \Psi_j(\Phi'(p)_{S_i}) \) in the reference set based on cosine distance, i.e.,

\[
(i', j') = \arg\max_{i'=1-N, j'=1-m'} \frac{\Psi_j(\Phi'(p)) \cdot \Psi_j(\Phi'(p)_{S_i})}{\|\Psi_j(\Phi'(p))\|_2 \|\Psi_j(\Phi'(p)_{S_i})\|_2}. \tag{2}
\]

Since photos and their corresponding sketches in \( \mathcal{R} \) are well aligned, the indices of the best matching result \((i', j')\) can be used directly to find the corresponding sketch feature patch, i.e., \( \Psi_j(\Phi'(s_{S_i})) \) which serves as the pseudo sketch feature patch \( \Psi_j'(\Phi'(p)) \). Finally, we obtain the pseudo sketch feature representation (at layer \( l \)) for \( p \) as \( \{\Psi_j(\Phi'(p))\}_{j=1}^{m'} \). We provide an intuitive visualization of PSF in supplementary material.

### 3.3. Loss Functions

We train generators \((G_{P2s}, G_{S2p})\) and discriminators \((D_s, D_p)\) alternatively with the following loss functions

\[
L^\text{total}_G = \lambda_p L_p + \lambda_{sty} L_{sty} + \lambda_{cyc} L_{cyc} + \lambda_{adv} (L_{G_{S2p}} + L_{G_{S2p}}), \tag{3}
\]

\[
L^\text{total}_D = L_{D_{S2p}} + L_{D_{S2p}}, \tag{4}
\]

where \( \lambda_p, \lambda_{sty}, \lambda_{cyc}, \) and \( \lambda_{adv} \) are trade-off weights for each loss term respectively. We describe details of each term as below.

**Pseudo Sketch Feature Loss** The pseudo sketch feature loss is formulated as

\[
L_p(p, \hat{s}) = \sum_{l=3}^{5} \sum_{j=1}^{m'} \left\| \Psi_j(\Phi'(\hat{s})) - \Psi_j'(\Phi'(p)) \right\|^2_2, \tag{5}
\]

where \( l = 3, 4, 5 \) are relu3\_1, relu4\_1, and relu5\_1 in VGG-19, and \( \hat{s} \) is the predicted sketch from \( G_{p2s} \).

**Style Loss** Inspired by recent style transfer methods, we include Gram Matrix loss (Gatys et al., 2016) as a second-order feature loss to provide better style supervision. We first average pool features in each \( k \times k \) patch for both \( \Psi_j(\Phi'(\hat{s})) \) and \( \Psi_j'(\Phi'(p)) \), resulting in features \( \psi_l \) and \( \psi_l' \) of size \( m' \times c' \), where \( c' \) is the channel number in \( l\)-th layer. We then calculate the Gram Matrix loss as

\[
L_{sty}(p, \hat{s}) = \sum_{l=3}^{5} \frac{1}{(c')^2} \| \psi_l^T \psi_l - \psi_l'^T \psi_l' \|_2^2, \tag{6}
\]

**Cycle-Consistency with Noise-injection** We use the cycle-consistency loss with noise-injection as supervision, which is formulated as

\[
L_{cyc}(p) = \| G_{S2p}(G_{P2s}(p) + \sigma z_1) - p \|_2^2, \tag{7}
\]

where \( z_1 \) is randomly sampled from a normal distribution with the same dimensions as \( G_{P2s}(p) \), and \( \sigma \) is a hyperparameter that controls the noise level.

**GAN Loss** We use the hinge loss to make the training process more stable. The objective functions of hinge loss are given by

\[
L_G = -\mathbb{E}[D(G(x))], \tag{8}
\]

\[
L_D = \mathbb{E}[\max(0, 1 - D(y))] + \mathbb{E}[\max(0, 1 + D(G(x)))], \tag{9}
\]

where \( x, y, D \) refer to \( p, s, D_s \) when \( G \) is \( G_{P2s} \), and \( s, p, D_p \) when \( G \) is \( G_{S2p} \).

### 4. Experiments

#### 4.1. Datasets and Metrics

**Datasets** To compare with previous works, we evaluate our model on two public benchmark datasets, namely the CUFS dataset (combination of CUHK (Tang and Wang, 2003), AR (Martinez and Benavente., 1998) and XM2VTS (Messer et al., 1999)), and the CUFSF dataset (Zhang et al., 2011b). For semi-supervised learning, we use extra face photos from VGG-Face dataset (Parkhi et al., 2015). We randomly select 1,244 photos from VGG-Face to test model performance on natural images. More details are provided in supplementary material.

**Training Details** We set all the trade-off weights \( \lambda_p, \lambda_{sty}, \lambda_{cyc}, \) and \( \lambda_{adv} \) to 1 for simplicity. We use Adam (Kingma and Ba, 2014) with learning rates 0.001 for generators and 0.004 for discriminators, and set \( \beta_1 = 0.9, \beta_2 = 0.999 \). The learning rates are linearly decayed to 0 after the first
10 epochs. The training batch size is 2, and models are trained on Nvidia 1080Ti GPUs.

**Metrics** For test sets with ground truth, we use FSIM (Zhang et al., 2011a), LPIPS (Zhang et al., 2018a) and DISTS Ding et al. (2020) to measure the texture quality, and NLDA score to measure the identity similarity following Wang et al. (2017b). For the evaluation of face-sketch translation in the wild, there are no ground truth sketches to calculate FSIM, LPIPS, and DISTS. We therefore exploit FID (Heusel et al., 2017) to measure the feature statistic distance between the generated sketch datasets and real sketch datasets. We explain details of these metrics in supplementary material.

**4.2. Comparison on Public Benchmarks**

We evaluate our model on both photo-to-sketch and sketch-to-photo translations on CUFS and CUFSF, which were captured under laboratory settings. We compare our results both qualitatively and quantitatively with four exemplar-based methods, namely MWF (Zhou et al., 2012), SSD (Song et al., 2014), RSLCR (Wang et al., 2017b), and DGFL (Zhu et al., 2017b), and five GAN-based methods, namely Pix2Pix-GAN (Isola et al., 2017), PS2MAN (Wang et al., 2017a), MDAL (Zhang et al., 2018b), KT (Zhu et al., 2019) and SCA-GAN (Yu et al., 2020). We obtain the results of MWF, SSD, RSLCR, and DGFL from Wang et al. (2017b), the results of SCA-GAN and KT from the respective authors, and use the public codes of Cycle-GAN and PS2MAN to produce the results. We also compare the photo-to-sketch translation results with our previous work FSW (Chen et al., 2018a). All the models are trained on the CUFS and CUFSF datasets with the same train/test partition.

**4.2.1. Photo-to-Sketch Translation**

Figure 5 shows some photo-to-sketch results on CUFS and CUFSF. Exemplar-based methods (Fig. 5(b,c,d,e)) in general perform worse than learning-based methods (Fig. 5(f,g,h,i,j,k)). Their results are over-smoothed and do not show hair textures. They also fail to preserve contents well, such as hairpins in the first row and glasses in the last row. GAN-based methods can generate better textures, but they usually produce artifacts because of the unstable training. For example, Pix2Pix produces lots of artifacts in the hair and eyes (Fig. 5(f)), and PS2MAN generates lots of artifacts when the facial parts of inputs are not clear or with a strong reflection of light (see the last two rows of Fig. 5(g)). Although the results of SCA-GAN look great, it suffers from incorrect parsing map guidance, such as hairpins in the first row, hairlines in the second row of Fig. 5(h). Referring to Fig. 5(j,k), we have improved our previous results of FSW by introducing $L_{sty}$ and the photo reconstruction branch.

The quantitative results with different metrics in Tab. 1 support our observations. It can be observed that exemplar-based methods perform much worse in terms of all metrics including FSIM, LPIPS, DISTS and NLDA. KT shows the best FSIM score but poor perceptual scores compared with SCG. We can see from Fig. 5(i) that the textures, especially hair textures, generated by KT are much worse than SCG. SCA-GAN generates better textures but the generated images might be different from
Table 1: Quantitative results for photo-to-sketch translation. SCA-GAN* needs a parsing map as guidance.

<table>
<thead>
<tr>
<th>Method</th>
<th>FSIM ↑</th>
<th>LPIPS ↓</th>
<th>DISTS ↓</th>
<th>NLDA↑</th>
</tr>
</thead>
<tbody>
<tr>
<td>MWF</td>
<td>0.7144</td>
<td>0.7029</td>
<td>0.3671</td>
<td>0.2533</td>
</tr>
<tr>
<td>SSD</td>
<td>0.6957</td>
<td>0.6824</td>
<td>0.4033</td>
<td>0.2336</td>
</tr>
<tr>
<td>RSLCR</td>
<td>0.6965</td>
<td>0.6650</td>
<td>0.4042</td>
<td>0.2556</td>
</tr>
<tr>
<td>DGFIL</td>
<td>0.7078</td>
<td>0.6957</td>
<td>0.3655</td>
<td>0.2410</td>
</tr>
<tr>
<td>Pix2Pix-GAN</td>
<td>0.7153</td>
<td>0.7060</td>
<td>0.3600</td>
<td>0.2151</td>
</tr>
<tr>
<td>PS2MAN</td>
<td>0.7157</td>
<td>0.7219</td>
<td>0.3794</td>
<td>0.2430</td>
</tr>
<tr>
<td>SCA-GAN*</td>
<td>0.7160</td>
<td>0.7268</td>
<td>0.3608</td>
<td>0.2005</td>
</tr>
<tr>
<td>MDAL</td>
<td>0.7275</td>
<td>0.7076</td>
<td>0.3319</td>
<td>0.2037</td>
</tr>
<tr>
<td>KT</td>
<td>0.7369</td>
<td>0.7311</td>
<td>0.3485</td>
<td>0.2116</td>
</tr>
<tr>
<td>FSW</td>
<td>0.7274</td>
<td>0.7103</td>
<td>0.3262</td>
<td>0.2063</td>
</tr>
<tr>
<td>SCG (ours)</td>
<td>0.7343</td>
<td>0.7261</td>
<td>0.3232</td>
<td>0.1967</td>
</tr>
</tbody>
</table>

Table 2: Quantitative results for sketch-to-photo translation. SCA-GAN* needs a parsing map as guidance.

<table>
<thead>
<tr>
<th>Method</th>
<th>FSIM ↑</th>
<th>LPIPS ↓</th>
<th>DISTS ↓</th>
<th>NLDA↑</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pix2Pix-GAN</td>
<td>0.7598</td>
<td>0.7877</td>
<td>0.3977</td>
<td>0.2421</td>
</tr>
<tr>
<td>PS2MAN</td>
<td>0.7645</td>
<td>0.7807</td>
<td>0.3668</td>
<td>0.2254</td>
</tr>
<tr>
<td>SCA-GAN*</td>
<td>0.7633</td>
<td>0.8304</td>
<td>0.3251</td>
<td>0.1794</td>
</tr>
<tr>
<td>KT</td>
<td>0.7794</td>
<td>0.7932</td>
<td>0.3233</td>
<td>0.1821</td>
</tr>
<tr>
<td>SCG (ours)</td>
<td>0.7652</td>
<td>0.7777</td>
<td>0.3374</td>
<td>0.1710</td>
</tr>
</tbody>
</table>

the original images (e.g., missing components) due to incorrect parsing map, which also leads to poor LPIPS and DISTS scores. In contrast, our SCG presents the second best results in terms of FSIM and the best results in terms of LPIPS and DISTS. As for sketch recognition, SCG also demonstrates best NLDA score on CUF and competitive results on CUFSF, which clearly demonstrate its superiority.

4.3. Sketch-to-Photo Translation

Figure 6 shows some example sketch-to-photo results. Same as photo-to-sketch translation, the results of Pix2Pix and PS2MAN contain many undesired artifacts. SCA-GAN produces results with the best visual quality, which is consistent with the quantitative results shown in Tab. 2. However, it still generates results with missing components under incorrect parsing map predictions, such as the missing eyes and glasses in the last row of Fig. 6(d). Without any GAN losses, KT suffers from unrealistic textures. For instance, results in Fig. 6(e) are grainy. Although SCG is trained in a self-supervised manner without seeing any real input sketches, it still shows competitive performance. Referring to Tab. 2, SCG shows the best or second results in 5 out of 8 columns. The biggest problem of SCG is that the synthesized colors are quite different from the ground truth. This is legitimate because the model is not suppose to recover exact color as ground truth unless overfitting.

4.4. Photo-to-Sketch Translation in the Wild

In this section, we will focus on photo-to-sketch translation in the wild. Since there are too many sketch styles in the wild, sketch-to-photo translation in the wild is beyond the scope of this paper, and we will leave it for future work. We compare SCG with other methods which provide codes, including SSD, RSLCR, Pix2Pix-GAN, PS2MAN, Cycle-GAN. Figure 7 shows some photos sampled from our VGG-Face test dataset and the sketches generated by different methods. It can be observed that these photos may show very different lightings and poses etc. Among the results of other methods, exemplar-based methods (see Fig. 7(b,c)) fail to deal with pose changes.
Table 3: Quantitative results and user study for photo-to-sketch translation in the wild.

<table>
<thead>
<tr>
<th>Method</th>
<th>FID</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSD</td>
<td>94.6</td>
</tr>
<tr>
<td>Fast-RSLCR</td>
<td>144.0</td>
</tr>
<tr>
<td>Pix2Pix-GAN</td>
<td>86.7</td>
</tr>
<tr>
<td>PS2MAN</td>
<td>90.8</td>
</tr>
<tr>
<td>Cycle-GAN</td>
<td>87.8</td>
</tr>
<tr>
<td>FSW</td>
<td>81.3</td>
</tr>
<tr>
<td>SCG (ours)</td>
<td>67.9</td>
</tr>
</tbody>
</table>

Table 4: Quantitative comparison on WildSketch dataset.

<table>
<thead>
<tr>
<th>Method</th>
<th>Cycle-GAN</th>
<th>GENRE</th>
<th>CA-GAN</th>
<th>PANet</th>
<th>Ours</th>
</tr>
</thead>
<tbody>
<tr>
<td>FSIM↑</td>
<td>0.6654</td>
<td>0.6902</td>
<td>0.6960</td>
<td>0.6950</td>
<td><strong>0.70</strong></td>
</tr>
</tbody>
</table>

Figure 8: Example comparison with WildSketch dataset.

and different hairstyles. Although GANs can generate some sketch-like textures, none of them can well preserve the contents. The face shapes are distorted and the key facial parts are lost. It can be seen from Fig. 7(g,h) that only FSW and SCG can handle photos in the wild well and generate pleasant results. Compared with FSW, SCG can generate more realistic shadows and textures. The same conclusion can also be drawn from the quantitative results shown in Tab. 3. We also conduct user study to better evaluate their subjective performance, as shown in Tab. 3 right part. We notice that our methods (FSW and SCG) are much preferred over previous methods. By introducing the $G_{	ext{ss}}$ branch and cycle-consistency, SCG further improves the performance of our previous work FSW. Details of user study are in supplementary material.

4.5. Ablation Study

To study the effectiveness of different components of the proposed method, we gradually modify the baseline Semi-Cycle-GAN and compare their results. Table 5 shows the results of all model variations. We discuss the results below.

Table 5: Ablation study of Semi-Cycle-GAN. $\sigma$: noise level, $k$: feature patch size, $L_{sty}$: use second-order style loss or not.

<table>
<thead>
<tr>
<th>Configuration</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\sigma$</td>
<td>0</td>
<td>10</td>
<td>20</td>
<td>30</td>
<td>40</td>
<td>50</td>
<td>60</td>
</tr>
<tr>
<td>$L_{sty}$</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>LPIPS (P2S)</td>
<td>0.3260</td>
<td>0.3273</td>
<td>0.3279</td>
<td>0.3287</td>
<td>0.3273</td>
<td>0.3273</td>
<td><strong>0.3235</strong></td>
</tr>
<tr>
<td>LPIPS (S2P)</td>
<td>0.3454</td>
<td>0.3435</td>
<td>0.3461</td>
<td>0.3433</td>
<td>0.3447</td>
<td>0.3423</td>
<td>0.3423</td>
</tr>
</tbody>
</table>

Figure 9: Effect of noise-injection.

Figure 10: Examples of improvement on $G_{\text{ss}}$ brought by style loss.

Figure 11: Effectiveness of additional training photos.

data dataset, our method achieves SoTA performance. Figure 8 also supports our claim that the inclusion of extra training photos improves generalization abilities of our model. For instance, ours demonstrates greater robustness towards hair color variations in the first row and shows better result with the presence of a hat in the second row. These results underscore the effectiveness of our proposed semi-supervised approach.
Noise injection. We show an example result with and without noise-injection in Fig. 9. It can be observed that Fig. 9(c) with $\sigma = 20$ is much better than Fig. 9(b) with $\sigma = 0$. This demonstrates that noise-injection can greatly improve the performance of $G_{s2p}$. This is because the proposed noise-injection strategy breaks the steganography in the outputs of $G_{p2s}$, and increases the generalization ability of $G_{s2p}$. We explore models with different levels of noise-injection, and the results are shown in columns A, B, C, and D of Tab. 5. We can see that adding more noise is not helpful to the performance of $G_{s2p}$ but degrades the performance of $G_{p2s}$. This is likely because the backward gradients from $G_{s2p}$ are corrupted when noise-injection level is too high. We empirically find $\sigma = 20$ strikes a good balance between the performance of $G_{p2s}$ and $G_{s2p}$.

Patch size. We present the results with patch size 1, 3, and 5 in columns C, E, and F of Tab. 5 respectively. We can observe that $k = 3$ gives the best performance, while $k = 5$ is worse than $k = 3$. This may be caused by the fact that a large patch in the feature space represents a much larger patch in the pixel space and this leads to undesired extra contents in the pseudo sketch feature. We therefore set $k = 3$ in our experiments.

Second-order style loss. Comparing the results in columns E and G of Tab. 5, we can notice that model with $L_{sty}$ shows better performance for both $G_{p2s}$ and $G_{s2p}$. This is because $L_{sty}$ provides better style supervision for $G_{p2s}$, which can in turn benefit the training of $G_{s2p}$. Figure 10 shows some examples of improvement on $G_{p2s}$ brought by style loss.

Extra training photos Introducing more training photos from VGG-Face dataset is the key to improve the generalization ability of our model. As demonstrated in Fig. 11, as we add more photos to the training set, the results improve significantly, see the eyes region.

5. Conclusion

In this paper, we propose a semi-supervised CycleGAN, named Semi-Cycle-GAN (SCG), for face photo-sketch translation. Instead of supervising our network using ground-truth sketches, we construct a novel pseudo sketch feature representation for each input photo based on feature space patch matching with a small reference set of photo-sketch pairs. This allows us to train our model using a large face photo dataset (without ground-truth sketches) with the help of a small reference set of photo-sketch pairs. Since directly training $G_{s2p}$ in a self-supervised manner as Cycle-GAN suffers from steganography, we exploit a noise-injection strategy to improve the robustness. Experiments show that our method can produce sketches comparable to (if not better than) those produced by other state-of-the-art methods on four public benchmarks, and outperforms them on photo-to-sketch translation in the wild.

References


Appendix A. More Methodology Details

Appendix A.1. Visualization of Pseudo Sketch Features

Figure A.13 visualizes examples of the pseudo sketch feature. It can be seen that the pseudo sketch feature provides a good approximation of the real sketch feature (see first two columns of Fig. A.13). We also show naive reconstruction obtained by directly using the matching index to index the pixel values in the reference sketches. We can see such a naive reconstruction does roughly resemble the real sketch, which also justifies the effectiveness of the pseudo sketch feature. Note that we only need alignment between photos and sketches in \( R \). Since we perform a dense patch matching between the input photo and the reference photos, we can also generate reasonable pseudo sketch features for input face photos under different poses (see last row of Fig. A.13).

Appendix A.2. Hyper-parameters of Pseudo Sketch Feature Loss

For the pseudo sketch feature loss \( L_p \), we set \( l = 3, 4, 5 \) to relu3_1, relu4_1, and relu5_1 in VGG-19. We choose these 3 layers mainly for two reasons: 1) better texture representation; 2) computation efficiency. Li et al. Li and Wand (2016) pointed out that compared with features from shallow layers, deep features after relu3_1 are more robust to appearance changes and geometric transforms. We conduct a simple experiment to verify this, and the results are presented in Fig. A.12. It can be observed that the model cannot synthesize sketch textures using shallow features from relu1_1 and relu2_1, and can generate better textures with high-level features, such as relu5_1. However, with only high-level features, the model also generates artifacts (e.g., eyes of the sketches in Fig. A.12). Besides, shallow feature maps have higher spatial resolutions, and it requires lots of GPU memory to calculate \( L_p \). Based on the above analysis, we set \( l = 3, 4, 5 \) to strike a balance between performance and computation cost.

Appendix B. More Dataset and Implementation Details

Photo-Sketch Pairs. We use four public datasets, namely the CUHK dataset Tang and Wang (2003), the AR dataset Martinez and Benavente (1998), the XM2VTS dataset Messer et al. (1999), and the CUFSF dataset Zhang et al. (2011b), to evaluate our model. In Wang et al. (2017b); Zhu et al. (2017b), the first three datasets were combined to form the CUFS dataset. Note that the CUFS dataset used in Zhu et al. (2017b); Wang et al. (2017a); Yu et al. (2020) contains only grayscale photos. In order to train a universal model for all datasets, we collect a color version of the CUFSF dataset containing 986 photo-sketch pairs. Details are summarized in Table B.6, and Fig. B.14 shows some examples of photo-sketch pairs from them.

Face Photos. VGG-Face dataset Parkhi et al. (2015) is a popular dataset containing face photos in the wild. We use a subset of it in this work. VGG-Face has 2,622 subjects with 1,000 photos for each subject. We randomly select \( N \) photos of 2,000 subjects for training. The resulting dataset are named as VGG-Face\( N \), where \( N = 01, 02, \ldots, 10 \). The VGG-Face\( N \) datasets are used to validate the performance relationship with increasing training photos. For the test dataset, 2 photos are randomly selected for each subject in the test split (no identity overlap with training dataset), which results in a VGG test set of 1,244 photos. Some examples from training and testing datasets are presented in Fig. B.15.

Preprocessing. For the reference datasets, we need the photo-sketch pairs to be well aligned. We perform alignment with similarity transformation based on 68 face landmarks detected using dlib\(^2\). The output faces and sketches are aligned with two eyes located at (75, 125)

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Total Pairs</th>
<th>TrainTest</th>
<th>Align</th>
<th>Var</th>
</tr>
</thead>
<tbody>
<tr>
<td>CUFS</td>
<td>188</td>
<td>88</td>
<td>✓</td>
<td>✗</td>
</tr>
<tr>
<td>AR</td>
<td>123</td>
<td>80</td>
<td>43</td>
<td>✓</td>
</tr>
<tr>
<td>XM2VTS</td>
<td>295</td>
<td>100</td>
<td>195</td>
<td>✗</td>
</tr>
<tr>
<td>CUFSF</td>
<td>1194</td>
<td>250</td>
<td>944</td>
<td>✓</td>
</tr>
</tbody>
</table>

Table B.6: Details of benchmark datasets. (Align: whether the sketches are well aligned with photos. Var: whether the photos have lighting variations.)

---

\(^1\)Downloaded from https://www.nist.gov/itl/iad/image-group/color-feret-database
\(^2\)http://dlib.net/
and (125, 125) respectively. The output size is set to 250×200 in order to perform a fair comparison with previous works. For photos/sketches whose landmarks cannot be detected, we simply discard them.

Appendix B.1. Implementation Details of Patch Matching

Although the patch matching only happens in training stage, it is still time-consuming. We accelerate patch matching in the following three ways. First, feature patches for the photos and sketches in the reference dataset are precomputed and stored in hard disk for fast query. Second, we use a coarse-to-fine strategy to search for the best matching feature patch. We find the best-matched \( n \) reference photos (we set \( n = 3 \) in the whole training process) for the input photo based on the similarity of their relu5_1 feature maps, which can be calculated fast. Fine-scale patch matching is then performed on these \( n \) reference photos. Third, we use the convolution operator to implement Eq. (2) which can be greatly accelerated with GPU.

Appendix C. More Experiment Details

Appendix C.1. Metric analysis

Previous works Wang et al. (2017b); Yu et al. (2020); Zhu et al. (2017b) usually adopted structural similarity (SSIM) Karacan et al. (2013) to evaluate the performance of sketch generation for test datasets with ground-truth sketches (e.g., CUFS and CUFSG). However, many works Ledig et al. (2016); Wang et al. (2017c,a)) pointed out that SSIM is not always consistent with the perceptual quality because SSIM favors slightly blurry images and fails to evaluate images with rich textures. To verify this, we show some sketches generated using different methods together with their SSIM scores in Fig. C.17. We can observe that although the results of Pix2Pix-GAN and our model have better textures, the result of RSLCR still has a better SSIM score because it is smoother. When we smooth all sketches with a bilateral filter, we notice that SSIM score for RSLCR remains almost unchanged, while the SSIM scores for Pix2Pix-GAN and our model improve by more than 1.5%.

Due to the drawbacks of SSIM, we choose FSIM Zhang et al. (2011a) as one of our image quality assessment metrics. FSIM takes local structure into account and gives lower scores to smooth results without textures, see Fig. C.17 for reference. Considering that metrics based on VGG feature space demonstrate better consistency with human perception, we also include two recent VGG-based metrics, namely LPIPS and DISTS. We use the PyTorch codes provided by Chen et al.\(^3\) to calculate these metrics. For the evaluation of face-sketch translation in the wild, there are no ground truth sketches to calculate FSIM, LPIPS, and DISTS. We therefore exploit FID score to measure the feature statistic distance between the generated sketch datasets and real sketch datasets.

Appendix C.2. Face Recognition Details

Following the practice of Wang et al. (2017b), we employed the null-space linear discriminant analysis (NLDA) Chen et al. (2000) to perform the recognition experiments. For CUFS, we randomly selected 150 synthesized sketches and their ground-truth sketches from the test set (338 test photos) to train a classifier and used the rest 188 for testing. For CUFSG(gray, crop), we randomly selected 300 synthesized sketches and their ground-truth sketches from the test set (944 test photos) to train a classifier and used the rest 644 for testing. Each experiment was repeated 20 times. We do not include SCA-GAN in

---

\(^3\)https://github.com/chaofengc/IQA-PyTorch
GT

Photo

PSF

Pixel

Projection

Figure A.13: Examples of PSF in the relu3 layer and the pixel level projection of the patch matching result. First row: ground truth feature and sketch with the photo in second row as input. Second row: results of laboratory images. Third row: results of natural images. (Note that the pixel level results are for visualization only, and they are not actually being computed or used in training.)

this comparison because it uses an extra face parsing map as guidance, and adopts a slightly different training/test split.

Figure C.16 shows the recognition accuracy of different methods on these two datasets. The results for photo-to-sketch translation are shown in Fig. C.16(a,b). We can see that the proposed SCG achieves the best NLDA scores with different feature dimensions on CUFS and competitive results on CUFSF. We notice that the recognition rate on CUFSF is similar to FSW and slightly worse than KT. This is mainly because the ground truth sketches in CUFSF are deformed too much compared with the input photos. We believe the NLDA scores on CUFSF cannot represent the quality of generated sketches. In fact, we observe that the result of SCG is clearer and without extra shadows or artifacts.

The results for sketch-to-photo translation are given in

Appendix C.3. User Study Details

We also conduct a user study for sketch synthesis in the wild. The methods considered include RSLCR, Pix2Pix-GAN, PS2MAN, Cycle-GAN, FSW (our previous method), and SCG (proposed). Different from our previous project which asked the subjects to rank results of different methods, we adopt a more comprehensive strategy to do the subjective study on an online human crowdsourcing platform. To be specific, we employed a two-alternative forced choice (2AFC) method. The crowd workers were shown two generated sketches at one time, and were asked to choose the sketch with better quality. An example photo-sketch pair was shown as a reference. We randomly select 30 images from VGG-Test dataset, generate sketches with the above methods and create 6 different surveys. Each survey contains results for 5 different images and \( \binom{6}{5} \times 5 = 75 \) questions in total for the 6 compared methods. Each worker was asked to do one
of these surveys. We collected 60 survey results from different crowd workers in total. We used the well-known Bradley–Terry model Bradley and Terry (1952) to convert the paired comparison results to global ranking. Give method $i$ and $j$, the probability that $i$ is better than $j$ is defined as

$$P(i > j) = \frac{e^{\beta_i}}{e^{\beta_i} + e^{\beta_j}}$$  \hspace{1cm} (C.1)

where $e^{\beta_i}$ indicates the ranking score of the method $i$. We estimate $\beta = \beta_1, \ldots, \beta_n$ by minimizing the following negative log-likelihood using gradient descent

$$L(\beta) = -\sum_{i=1}^{n} \sum_{j=1, j\neq i}^{n} \log w_{i,j} P(i > j)$$  \hspace{1cm} (C.2)

where $w_{i,j}$ indicates the total numbers that $i$ is better than $j$. The “prefer score” in Tab. 3 of main text refers to $e^{\beta_i}$.

### Appendix D. More Results

In this part, we provide more photo-to-sketch results for CUFS in Fig. F.18, CUFSF in Fig. F.19 and natural images of VGGFace in Fig. F.20.

### Appendix E. Limitations

Although the proposed model shows good generalization ability to images in the wild, it cannot generate unknown structures which are not included in the small reference dataset. For example, SCG fails to generate the teeth in the last row of Fig. F.20. Note that existing methods cannot even produce pleasant results for these natural images. The proposed SCG also cannot generalize to sketches with different styles, such as sketches drawn in thick lines Yi et al. (2019, 2020b,a). It is quite challenging to synthesize satisfactory sketches with different styles using the same model. In a word, the above two problems are difficult to be solved with current small face sketch datasets, and we will leave them to future work.

### Appendix F. Links to public codes

We also provide links to the public codes used in our experiments below:

- Pix2Pix-GAN: [https://github.com/phillipi/pix2pix](https://github.com/phillipi/pix2pix)
- PS2-MAN: [https://github.com/andidan1/PhotoSketchMAN](https://github.com/andidan1/PhotoSketchMAN)
Figure F.18: More qualitative results for photo-to-sketch translation on CUFS test dataset.

<table>
<thead>
<tr>
<th>Sketch (ours)</th>
<th>GAN (ours)</th>
<th>PS2MAN (ours)</th>
<th>CycleGAN (ours)</th>
<th>SCGAN (ours)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(a) Photo</td>
<td>(b) MWF</td>
<td>(c) SSD</td>
<td>(d) RSLCR</td>
<td>(e) DGFL</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>(f) Pix2Pix</th>
<th>(g) Cycle-</th>
<th>(h) PS2MAN</th>
<th>(i) SCGAN</th>
<th>(j) KT</th>
</tr>
</thead>
<tbody>
<tr>
<td>(k) FSW</td>
<td>(l) SCG</td>
<td>(m) GT</td>
<td>(n) Photo</td>
<td>(o) Photo</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>(p) MWF</th>
<th>(q) Photo</th>
</tr>
</thead>
</table>
Figure F.19: More qualitative results for photo-to-sketch translation on CUFSF test dataset.
Figure F.20: More qualitative results for photo-to-sketch translation in the wild on VGG test dataset.