CLIP と微分可能レンダラーを用いたフォントスタイル変換

泉 幸太[1,a)]　柳井 啓司[1,b)]

概要

近年，芸術的な文字はウェブページや広告などで幅広く使われている．しかし，芸術的な文字のデザインは文字ごとに異なり，漢字やひらがな，アルファベットといった様々な種類の文字をデザインするには多大な労力を要する．近年では，深層学習を用いた画像生成技術の発展により，新しい芸術的な文字の生成も容易になった．しかし，従来の文字生成手法は，スタイル画像の入力が必要になる．

そこで本研究では，スタイル画像の入力を必要とせず，入力したテキストに沿ったスタイルを文字画像に転送する．大規模マルチモーダル CLIP と微分可能レンダラーを組み合わせることで，テキストにマッチしたスタイルが文字画像に転送されることを確認した．さらに，文字画像に加え，シンプルなロゴパターンに対しても言語によるスタイル変換が可能なことが確認された．

1. はじめに

近年，芸術的な文字はウェブページや広告などで幅広く使われている．しかし，芸術的な文字のデザインは文字ごとに異なり，漢字やひらがな，アルファベットといったあらゆる文字をデザインするには多大な労力を要する．一方，深層学習を用いた画像生成技術の発展により，新しい芸術的な文字の生成も容易になった．しかし，従来の文字生成手法は，スタイル画像の入力が必要になる，背景にまでスタイルが転送されるといった問題などがある．また，近年，自然言語を用いたスタイル変換できる CLIPstyler [7] も提案されている．この手法は本研究の目的と類似しているが，実画像を入力することを目的としているため，図 1 のようにフォント画像を入力画像としても文字の色が変わるだけで，文字の輪郭やテクスチャに大きな変化はない．

そこで，本研究では，自然言語のみを用いて，文字画像のスタイル変換を行うことを目的とする．微分可能レンダラー [9] を用いてベジェ曲線のパラメータを最適化する手法と，大規模マルチモーダルモデルである CLIP [11] を用いた言葉によるスタイル変換手法を組み合わせた手法を提案する．なお，本稿は既発表ショートペーパーに比較実験を追加して日本語化したものである．

[1] 電気通信大学 大学院情報理工学研究科 情報学専攻
[a)] izumi-k@mm.inf.uec.ac.jp
[b)] yanai@cs.uec.ac.jp

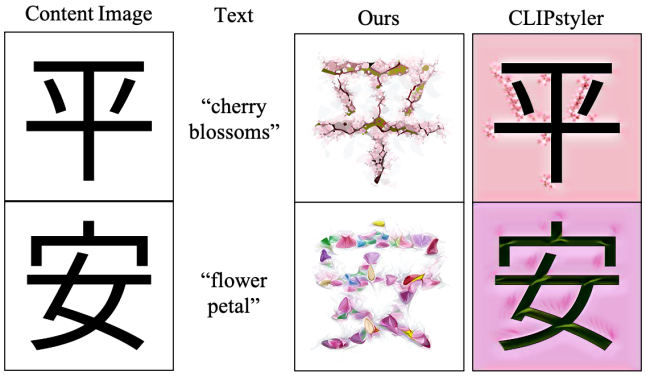

図 1: 本研究と CLIPStyler [7] での文字画像のスタイル変換結果．

2. 関連研究

文字画像のスタイル変換では，通常のスタイル変換とは異なり，文字の形を保ちつつ，スタイル画像のスタイルを転送する必要がある．Atarsaikhan ら [1] は Distrance Transform Loss を neural style transfer [6] に導入することで，文字やロゴパターンの形状を保ったままスタイル変換を行うことに成功した．Typography with Decor [15] では，装飾された文字画像のスタイルを転送するための深層学習ネットワークを提案した．Shape-Matching GAN [16] では，一枚のスタイル画像を学習することで，文字画像のスタイル変換を行うことができる．

近年では，OpenAI が発表した CLIP [11] を用いてテキストで画像を操作する手法が提案されている．StyleCLIP [10] では，StyleGAN [8] と CLIP を組み合わせ，テキストによって生成画像を操作できる．StyleGAN-NADA [5] では，ドメイン画像を追加することなく，テキストに基づいた新たなドメインへ変換することができる．CLIPstyler [7] では，テキストのみを用いて，テキストにマッチしたスタイルを入力画像に転送することを可能にした．

また，CLIP [11] とベクターグラフィックスを組み合わせた研究も行われている．CLIPDraw [4] では，微分可能レンダラー [9] と CLIP [11] を組み合わせ，ベジェ曲線の

(a) (b) (c)

図 2: (a) 入力画像．(b) ベジェ曲線の初期位置． 制御点はこの点の付近に分布する． (c) 距離変換画像．文字から離れていくほど値が大きくなる．

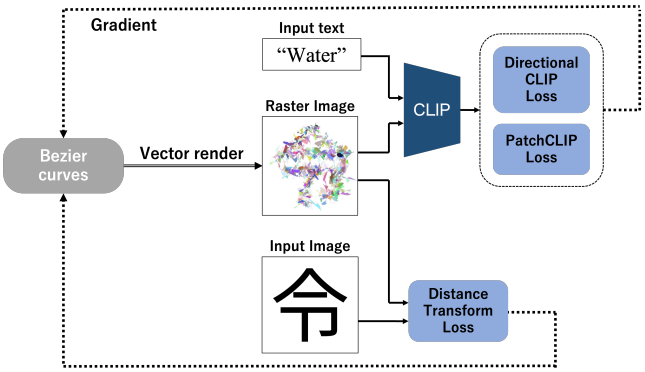

図 3: 本手法の概要図

パラメータを最適化することでテキストのみから描画を生成できる．StyleCLIPDraw [13] では，CLIPDraw で生成される描画にスタイル画像のスタイルを転送できる．

本研究ではこれらの研究を踏まえて，CLIPDraw [4] と CLIPstyler [7] を組み合わせることでテキストによる自然なスタイルの転送を実現し，さらに Atarsaikhan ら [1] が利用した Distrance Transform Loss を導入することで，文字部分のみのスタイル変換を実現する．

3. 手法

本研究は，ベクターグラフィックス用の微分可能なレンダラーを導入した Li ら [9] の研究，Li ら [9] の微分可能レンダラーと CLIP [11] を組み合わせた CLIPDraw [4] を元にしている．本手法の概要図を図 3 に示す．本手法では，描画を，ピクセルで表現されるラスター画像ではなくパラメータとして制御点の位置, 色を持つベジェ曲線で表現する．このベジェ曲線のパラメータを最適化していき，描画を生成する．最適化の際，生成した描画を微分可能なレンダラーを用いてラスター画像に変換する．その後，オーグメンテーションを行った後，CLIP [11] の画像エンコーダー，テキストエンコーダーを用いて損失を計算する．損失の計算は，CLIPstyler [7] で用いられている損失関数をベースとしている．さらに，文字の形を保つために，Atarsaikhan ら [1] がフォントスタイル変換に導入した Distrance Transform Loss も利用する．

3.1 Drawing Representation

描画は，Li ら [9] の方法に基づき，3 次ベジェ曲線の集合で表現する．各曲線は，制御点の位置，色，不透明度で表されいる．閉じたベジェ曲線は，1 つの形状につき 3 から 5 本のセグメントをもち，内部は塗りつぶされる．最適化している間は，曲線と制御点の数は固定し，制御点の位置，色，不透明度のみを勾配降下法によって最適化する．制御点の初期位置は，図 2(b) の様に，文字画像中の黒い領域の座標からストロークの数だけランダムにサンプリングした点の周辺とする．これは，ランダムな位置を初期位置とすると，背景にまでベジェ曲線が描画されてしまうからである．

3.2 Directional CLIP loss

CLIPstyler [7] では，最適化の安定と出力画像の品質向上のために StyleGAN-NADA [5] で提案された Directional CLIP loss を導入している．本手法でもこの損失関数を採用する．Directional CLIP loss は以下のように定義できる．

$$\Delta T = E_T(t_{sty}) - E_T(t_{src})$$

$$\Delta I = E_I(I_{draw}) - E_I(I_c)$$

$$L_{dir} = 1 - \frac{\Delta I \cdot \Delta T}{|\Delta I||\Delta T|} \tag{1}$$

ここで，I_c は入力画像，I_{draw} は微分可能レンダラーを用いてレンダリングした画像．E_I と E_T は，それぞれ CLIP の画像・テキストエンコーダーを示す．また，t_{sty} は入力するテキストである．t_{src} はあらかじめ設定するテキストであり、“a photo of text”, “letters”, “logo” も試したが “a photo” と変化がなかったので，CLIPstyler [7] と同じく “a photo” を採用した．

3.3 PatchCLIP loss

CLIPstyler [7] では，Gatys ら [6]，Frans ら [4] に触発され，テクスチャ転送のための新しい損失関数，Patch CLIP loss を導入している．本手法でもこの損失関数を導入する．具体的には，微分可能なレンダラーも用いて変換したラスター画像を，十分な数だけ切り出し，切り出したパッチに対して射影変換を行い，Directional CLIP Loss を計算する．また，高スコアのパッチに対する勾配最適化を拒否するために，閾値 τ を設定する．従って，PatchCLIP loss は以下のように定義できる．

$$\Delta T = E_T(t_{sty}) - E_T(t_{src})$$

$$\Delta I = E_I(\mathrm{aug}(\hat{I}^i_{cs}) - E_I(I_c)$$

$$l^i_{\mathrm{patch}} = 1 - \frac{\Delta I \cdot \Delta T}{|\Delta I||\Delta T|}$$

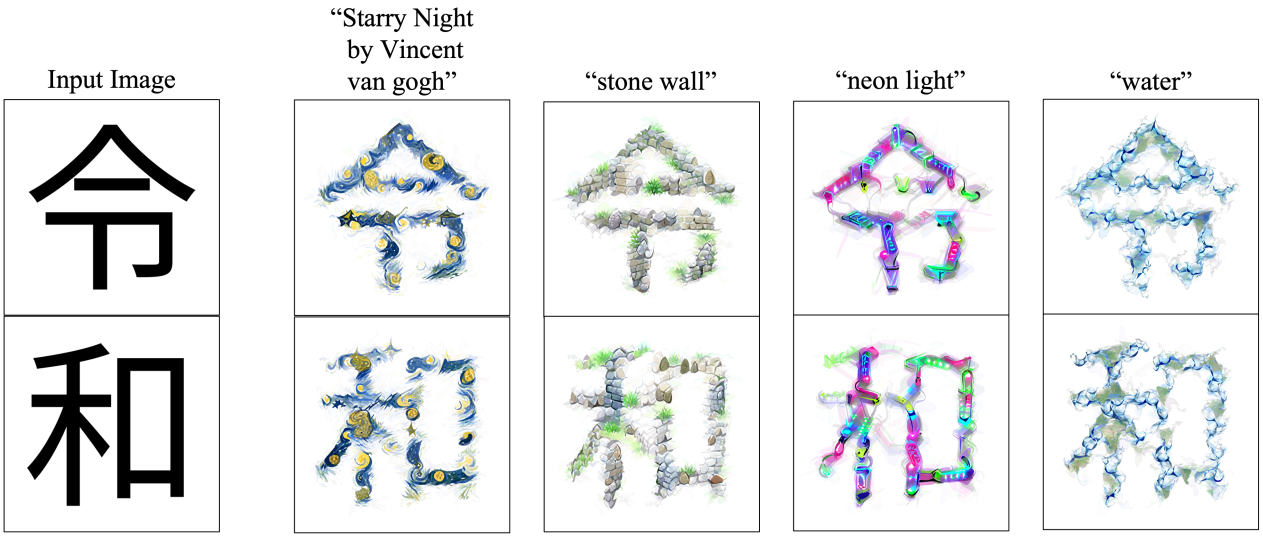

図 4: 本手法を用いて文字をスタイル変換した結果.

$$L_{\text{patch}} = \frac{1}{N}\sum_{i}^{N} R(l^{i}_{\text{patch}}, \tau) \qquad (2)$$

$$\text{where } R(s,\tau) = \begin{cases} 0, & \text{if} s \leq \tau \\ s, & \text{otherwise} \end{cases}$$

ここで，$\hat{I}^{i}_{cs}$ は，ラスター画像から切り出した i 番目のパッチを表す．また，aug は射影変換を意味する．

3.4 Distance Transform loss

文字画像を描画する際，背景は入力画像と同じく白いままである必要がある．また，文字の形が崩れすぎると識字しづらい文字となってしまう．この問題を解決するために，Atarsaikhan らは Distance Transform Loss を追加した [1]．本手法でもこの Distrance Transform Loss を導入する．Distance Transform Loss の計算には，まず入力画像に対して距離変換を行い距離変換画像を作成する．その後，入力画像と出力画像それぞれに距離変換画像を掛け合わせ，その平均二乗誤差を計算する．距離変換は図 2(c) の様に，白黒画像シルエットとなる画素の値を 0 とし，シルエットから離れた画素ほど値が大きくなるようにする．Distance Transform Loss は以下の式で定義できる．

$$L_{\text{distance}} = \frac{1}{2}\left(I_c \circ I_d - I_{draw} \circ I_d\right)^2 \qquad (3)$$

ここで，I_d は距離変換画像を表す．演算子 $\circ$ は，同じ大きさの行列の要素同士の積を取ることを意味する．

3.5 Total loss

以上の 3 つの損失関数に加え,CLIPstyler [7] と同様に，生成画像をなめらかにする効果がある全変動正規化損失 L_{tv} を導入する．従って，最適化の際の損失関数は以下の様に定義できる．

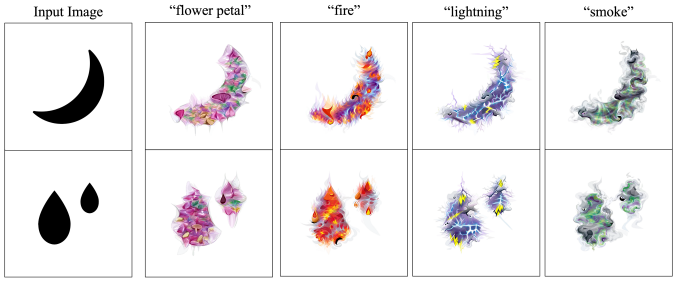

図 5: シンプルなロゴに適用した結果.

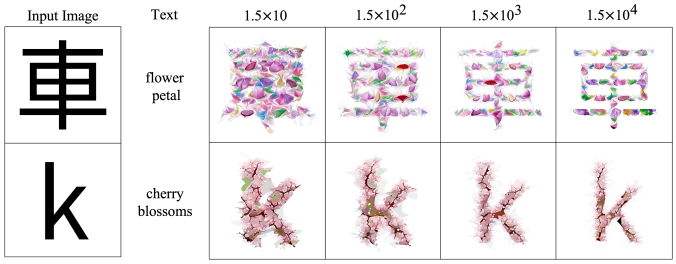

図 6: Distance Transform Loss の重み $\lambda_{\textbf{distance}}$ を変えた結果.(a) $\lambda_{\textbf{distance}} = 1.5 \times 10$. (b) $\lambda_{\textbf{distance}} = 1.5 \times 10^2$. (c) $\lambda_{\textbf{distance}} = 1.5 \times 10^3$. (d) $\lambda_{\textbf{distance}} = 1.5 \times 10^4$.

$$L = \lambda_d L_{\text{dir}} + \lambda_p L_{\text{patch}} + \lambda_{distance} L_{\text{distance}} + \lambda_{tv} L_{\text{tv}} \qquad (4)$$

4. 実験

4.1 実験設定

入力画像の解像度は 512×512 を用いた．距離変換画像は，OpenCV ライブラリ内の関数を用いて作成した．重みについては，λ_d, λ_p, $\lambda_{distance}$, λ_{tv} をそれぞれ，5×10^2, 9×10^3, 1.5×10^3, 2×10^{-3} とした．Directional CLIP loss，PatchCLIP loss，全変動正規化損失の重みは，CLIP-

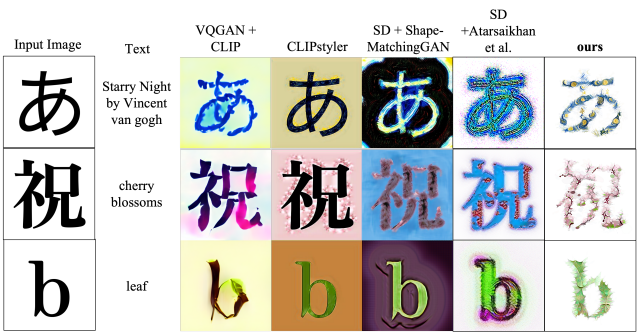

図 7: 他の手法との比較.

styler [7] を参考に値を設定した．Distance Transform Loss の重みは，スタイルを反映しつつ，最も文字が認識しやすい重みに設定した．最適化関数は Adam を使用し，学習率は，制御点の位置，色（不透明度も含む）に対してそれぞれ異なる値 $1, 1 \times 10^{-2}$ とした．イテレーション回数は 200 回とした．切り出す画像のサイズは，最も良い品質を示す 160×160 を選択した．切り出す枚数は 64 枚とした．閾値 τ は 0.7 とした．また，Radford ら [11] が提案したプロンプトエンジニアリングを用いる．プロンプトエンジニアリングは，CLIP に “a photo of {テキスト}” のようなテンプレートを用いて，単語から文章を作成し，CLIP に入力する手法である．512 個の閉じたベジェ曲線で 1 文字を表現する．生成時間は，NVIDIA GeForce RTX2080Ti で，画像 1 枚あたり 60 秒から 80 秒程度である．

4.2 定性評価

本手法での結果を図 4 に示す．文字画像がテキストに合致したスタイルに変換されていることがわかる．例えば，テキストを “Starry Night by Vincent van gogh” とした場合，元の有名な絵画のような月が現れていることがわかる．また，背景には何も描画されることなく，画像が生成できていることもわかる．さらに，テキストを “stone wall” とした場合，色が変わっているだけでなく，石垣のようなパターンが生成されていることがわかる．さらに，石垣にあるような雑草のようなものが現れていることもわかる．

図 5 には．簡単なロゴを入力画像とした場合の結果を示す．文字画像のみではなく，シンプルなシルエット画像にもスタイルが転送できることがわかる．

4.3 アブレーション研究

Distance Transform Loss に異なる重みをかけた際の結果を図 6 に示す．重みを大きくするほど，入力した文字画像の文字の太さに近づいていき，反対に小さくすると，文字の輪郭からベジェ曲線が逸脱していく．従って，この重みを調整することにより，文字の装飾の度合いを調整することが可能であると考えられる．

4.4 他の手法との比較

また，本研究の目的と類似した，自然言語を用いたスタイル変換手法と比較を行う．比較する手法として，CLIPstyler [7]，CLIP [11] を用いて事前学習済み VQGAN [3] の潜在コードを最適化する手法である VQGAN+CLIP[*1] [2] を用いた．フォントのスタイル変換を目的とした Shape-MatchingGAN [16] や Atarsaikhan ら [1] の手法とも比較を行う必要があるが，これらの手法はスタイル画像を必要とするため，本手法と直接比較するのは困難である．そこで，現在自然言語から画像を生成する手法として注目を浴びている Stable Diffusion[*2]を用いてスタイル画像を生成し，フォントスタイル変換手法の入力とした結果と比較した．Stable Diffusion は，Latent Diffusion [12] を大規模データセット LAION-5B [14] で学習させた学習済みモデルである．

スタイル画像は，フォント画像にスタイルが転送しやすいように，なるべく背景と前景がはっきりしているものを生成画像から選択した．

他の手法との比較結果を図 7 に示す．左の 2 列は入力した文字画像，テキストを表す．右側の 5 列はそれぞれの手法で生成した結果を示している．VQGAN+CLIP では，文字の形が崩れており，読みにくくなってしまっている．CLIPstyler は，文字の形は保っているが，テキストの意味が文字にあまり反映されていない．Stable Diffusion+Shape-Matching GAN では，テキストの意味を反映はしているものの，質感が表現できていない．Stable Diffusion+Atarsaikhan et al. では，テキストの意味が文字の一部分にしか反映されておらず，品質も悪い画像となっている．一方で，本手法は，他の手法と比べ，背景にスタイルが転送されることなく，よりテキストに沿った文字画像になっていることがわかる．

5. 結論

本論文では，微分可能レンダラー [9] と CLIPStyler [7] で提案された損失関数と文字の形を保つための Distance Transform Loss [1] を組み合わせ，スタイル画像なしで，文字画像にテキストにマッチしたスタイルを転送する手法を提案した．本手法は，文字の形を保ったまま，スタイルを文字画像に転送することを実現した．また，シンプルなロゴに対しても本手法は有効であることが確認された．さらに，アブレーション研究の結果から，損失関数の重みをを調整することで，文字の装飾の度合いを変化させられることが確認できた．今後の課題として，最適化にかかる時間の短縮が挙げられる．

[*1] https://github.com/nerdyrodent/VQGAN-CLIP
[*2] https://github.com/CompVis/stable-diffusion

参考文献

[1] Atarsaikhan, G., Iwana, B. K. and Uchida, S.: Contained Neural Style Transfer for Decorated Logo Generation, *In 13th IAPR International Workshop on Document Analysis Systems* (2018).

[2] Crowson, K., Biderman, S., Kornis, D., Stander, D., Hallahan, E., Castricato, L. and Raf, E.: VQGAN-CLIP: Open Domain Image Generation and Editing with Natural Language Guidance, *European Conference on Computer Vision* (2022).

[3] Esser, P., Rombach, R. and Ommer, B.: Taming transformers for high-resolution image synthesis, *Proc.of IEEE Computer Vision and Pattern Recognition*, pp. 12873–12883 (2021).

[4] Frans, K., Soros, L. and Witkowski, O.: Clipdraw: Exploring text-to-drawing synthesis through language-image encoders, *arXiv preprint arXiv:2106.14843* (2021).

[5] Gal, R., Patashnik, O., Maron, H., BERMANO, A. H., Chechik, G. and Cohen-Or, D.: Stylegan-NADA: Clip-guided domain adaptation of image generators, *ACM Transactions on Graphics*, Vol. 41, No. 141, pp. 1–13.

[6] Gatys, L. A., Ecker, A. S. and Bethge, M.: Image style transfer using convolutional neural networks, *Proc.of IEEE Computer Vision and Pattern Recognition*, pp. 2414–2423 (2016).

[7] Gihyun, K. and Chul, Y. J.: Clipstyler: Image style transfer with a single text condition, *Proc.of IEEE Computer Vision and Pattern Recognition*, pp. 18062–18071 (2022).

[8] Karras, T., Laine, S., and Aila, T.: A style-based generator architecture for generative adversarial networks, *Proc.of IEEE Computer Vision and Pattern Recognition*, pp. 4401–4410 (2019).

[9] Li, T.-M., Lukáč, M., Gharbi, M. and Ragan-Kelley, J.: Differentiable Vector Graphics Rasterization for Editing and Learning, *ACM Transactions on Graphics (Proc. SIGGRAPH Asia)*, Vol. 39, No. 6, pp. 193:1–193:15 (2020).

[10] Patashnik, O., Wu, Z., Shechtman, E., Cohen-Or, D. and Lischinski, D.: StyleCLIP: Text-driven manipulation of stylegan imagery, *arXiv preprint arXiv:2103.17249* (2021).

[11] Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Pamela Mishkin, J. C., Krueger, G. and Sutskever, I.: Learning transferable visual models from natural language supervision, *Proc.of the 38th International Conference on Machine Learning*, Vol. 139, pp. 8748–8763 (2021).

[12] Rombach, R., Blattmann, A., Lorenz, D., Esser, P. and Ommer, B.: High-Resolution Image Synthesis with Latent Diffusion Models, *Proc.of IEEE Computer Vision and Pattern Recognition* (2022).

[13] Schaldenbrand, P., Liu, Z. and Oh, J.: StyleCLIPDraw: Coupling Content and Style in Text-to-Drawing Translation, *arXiv preprint arXiv:2202.12362* (2021).

[14] Schuhmann, C., Beaumont, R., Vencu, R., Cade Gordon, R. W., Mehdi Cherti, T. C., Katta, A., Mullis, C., Wortsman, M., Schramowski, P., Kundurthy, S., Crowson, K., Schmidt, L., Kaczmarczyk, R. and Jitsev, J.: LAION-5B: An open large-scale dataset for training next generation image-text models, *36th Conference on Neural Information Processing Systems* (2022).

[15] Wang, W., Liu, J., Yang, S. and Guo, Z.: Typography with decor: Intelligent text style transfer, *Proc.of IEEE Computer Vision and Pattern Recognition*, pp. 5889–5897 (2019).

[16] Yang, S., Wang, Z., Wang, Z., Xu, N., Liu, J. and Guo, Z.: Controllable artistic text style transfer via shape-matching GAN, *Proc.of IEEE International Conference on Computer Vision*, pp. 4442–4451 (2019).