卒 業 研 究 報 告 題 目

GANを用いた二次元キャラクターの表情生成

Generation of 2D character's Expression using GAN

指 導 教 員　　出口 利憲 教授

岐 阜 工 業 高 等 専 門 学 校　電気情報工学科

2015E39 三上　麟太郎

令 和 2 年（ 2 0 2 0 年 ）　　2 月 1 4 日 提 出

Abstract

Recently, acquisition of the creativity of AI is attracting public attention. If AI's creativity is enriched in people's hobbies, it may be social development.

In this research, we use StarGAN, which has high accuracy for image conversion between multiple domains, in the generative adversarial network. In the conventional method, when an image is converted between three or more domains, it was necessary to create a conversion model between each domain. With StarGAN, even when converting images between three or more domains, one model can complete it by implementing classification and reconstructing images.

In this study, we studied whether StarGAN can give various face expressions to 2D characters. As a result, it was possible to convert the parts roughly, but it was not at a level that would enhance the hobbies related to illustration. In order to improve the accuracy of the results, it is necessary to search for more data, more learning times, and hyper-parameters that are optimal for converting the expression of the 2D character.

目 次

Abstract

第 1 章 序論 **1**

第 2 章 ニューラルネットワーク **2**

2.1 ニューロン 2

2.2 ニューロンモデル 2

2.3 ニューラルネットワーク 4

第 3 章 ディープラーニング **7**

3.1 ディープラーニング 7

3.2 畳み込みニューラルネットワーク 7

第 4 章 GAN **10**

4.1 GAN 10

4.2 ACGAN 12

4.3 CycleGAN 12

第 5 章 Stargan **15**

5.1 Stargan 15

5.2 Adversarial Loss 15

5.3 Domain Classification Loss 16

5.4 Reconsutruction Loss 17

5.5 目的関数 17

5.6 Mask Vector 17

5.7 CelebA 18

第 6 章 実験 **24**

6.1 実験準備 24

6.2 実験 24

第 7 章 結論 **31**

謝辞 **32**

参考文献 **33**

第1章　序論

近年、人工知能（AI）が急速な発展を遂げていく中で、AIの創造性の獲得が注目されている。何もないところから存在しない人間やアニメキャラクターの顔を生成するソフトウェアや、動画内の人物に自分の顔を張り付けると、動画内の元の人物の顔の動きに同期して貼り付けた自分の顔が動くアプリはまだ記憶に新しい。

一方、イラストレーションに関しては近年「差分」という手法を用いた作品がみられる。差分とはイラストの全体的な構成はそのままに、表情や衣服、髪型などの一部パーツのみ変更を施したイラストの事である。主にゲームのキャラクターの立ち絵や、バーチャルユーチューバーの顔の動きなどに用いられており、それらは今日の日本において多くの人々の娯楽となっている。

本研究では、GANによる画像の変換技術を用いて、二次元キャラクターイラストの表情の変換を試みる。モデルにはStarGANを使用し、顔のパーツごとに細かい表情の変換が可能になるようにした。これを用いることで、イラストレーションに簡単に様々な表情を与えることができ、製作活動や趣味の充実に貢献することができると考えられる。

第2章　ニューラルネットワーク

2.1　ニューロン

ニューロンとは、神経ネットワークの構成要素である。Figure 2.1 にニューロンの模式図を示す。

ニューロンは細胞体、無数の樹状突起、一本の軸索によって構成されており、軸索の先端は枝分かれして別ニューロンの細胞体や樹状突起と組み合わさることでシナプスと呼ばれる接合部を形成している。人間の脳内には約 140 億個のニューロンが存在し、また、ニューロン一個につき平均千から一万個のシナプスが形成されており、脳全体では 10 兆から 100 兆個のシナプスが存在しているとされている。

ニューロンは、ニューロンの細胞表面に存在するレセプターと呼ばれる化学物質に特異的な受容体によって細胞外の化学物質、機械的刺激、光、電気等の信号を受容した後、これを膜電位という電気的な信号に変換することで細胞体から軸索、シナプスを経て次の細胞体へと電気的情報を伝えている。この長距離を高速で走る電気信号を用いることによって、多細胞体の制御が可能となっている。[1)]

2.2　ニューロンモデル

ニューロンを多入力対 1 出力の情報処理素子と考え、ニューロンの入出力関係モデル化したものをニューロンモデルと呼ぶ。ニューロンモデルは、Figure 2.2 のように表すことができ、動作は以下の式で表すことが出来る。

$$y = \sum_{i=1}^{n} x_i w_i - \theta \tag{2.1}$$

$$z = f(y) \tag{2.2}$$

x_i は 1〜n 番目の入力信号の強さを表しており、非興奮状態で送り出される信号を 0、興奮状態で送り出される信号は 1 である。 w_i はシナプスの結合の強さを表している。 θ はニューロンのしきい値であり、式 (2.1) においてニューロン内部の興奮状態 y がこれを超えるとニューロンは興奮状態となり、出力として 1 が送り出される。この時、興奮性神経からにシナプスに対しては $w_i > 0$、 抑制性神経からのシナプスに対しては $w_i < 0$ となる 。また、 シナプスが結合していない場合は、$w_i = 0$ となる 。

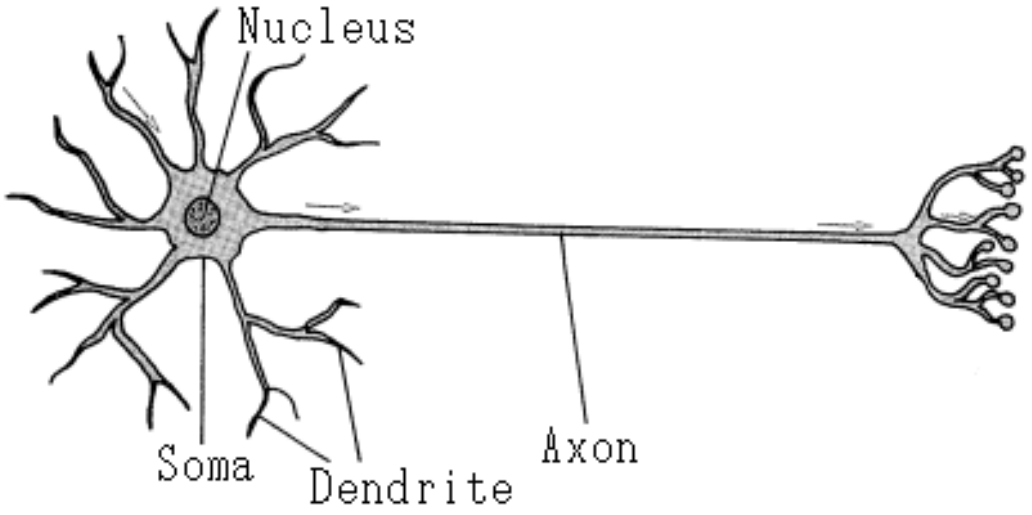


Figure 2.1　Neuron.[1)]

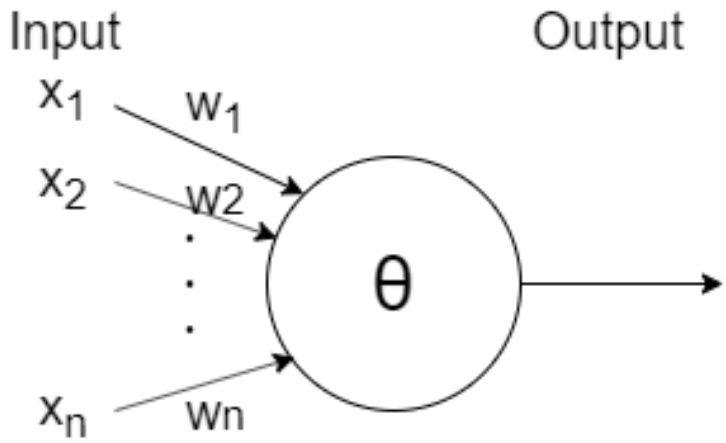


Figure 2.2　Neuron model.

Figure 2.2 のモデルは 1943 年に McCulloch と Pitts によって発表されたモデルであり、形式ニューロンやしきい素子とも呼ばれている。このモデルは出力が 1 と 0 のみのステップ関数とされることが多く、その場合内部状態 y としきい値 θ との大小関係によって出力として 0 か 1 のどちらかの値のみをとるという特徴を持っている。連続的な入力を考える場合は出力関数としてシグモイド関数が用いられることが多い。シグモイド関数は式 (2.3) で表され、Figure 2.3 となる。[2)]

$$f(u) = \frac{1}{1 + \exp(-u)} \tag{2.3}$$

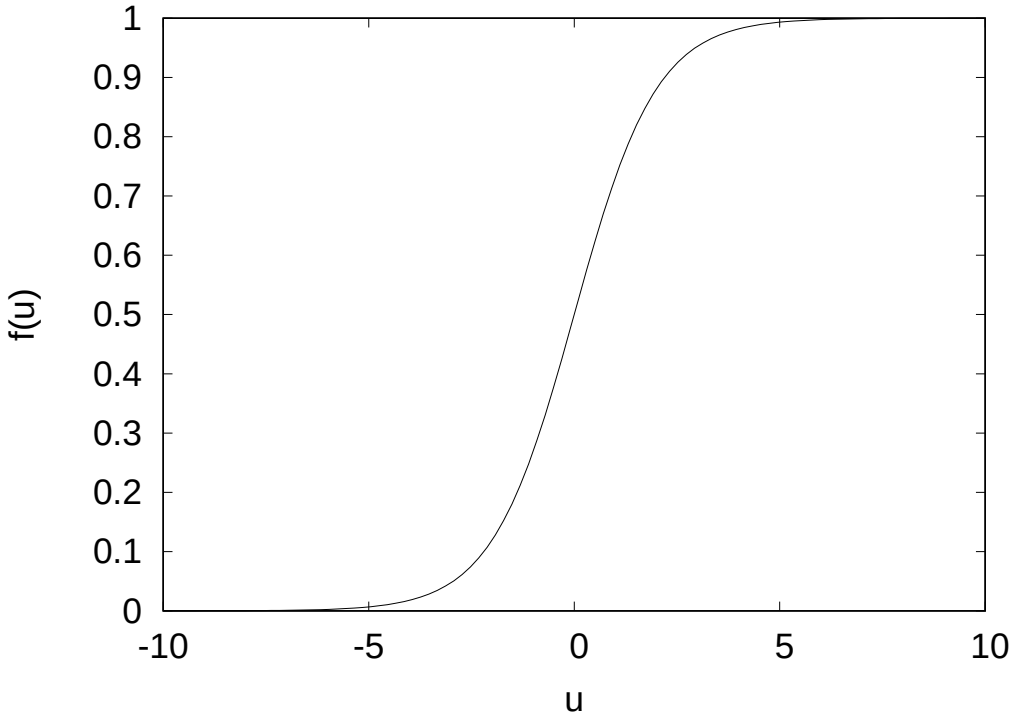


Figure 2.3　Sigmoid function.

2.3　ニューラルネットワーク

ニューロンモデルをネットワーク上に結合したものをニューラルネットワークという。

ニューラルネットワークの構成はさまざまであり、Figure 2.4 のようなフィードフォワードニューラルネットワークや Figure 2.5 のようなリカレントニューラルネットワークなどが存在する。フィードフォワードニューラルネットワークは最も単純なニューラルネットワークであり、入力層から中間層へ、中間層から出力層へと一方向にデータが流れるモデルである。リカレントニューラルネットワークでは、中間層の出力が再帰して中間層の入力となるなど、データのループや双方向にデータが流れるという特徴を持っている。リカレントニューラルネットワークの一つとして、Figure 2.6 のような中間層のみで構成された全結合リカレントニューラルネットワークが存在する。

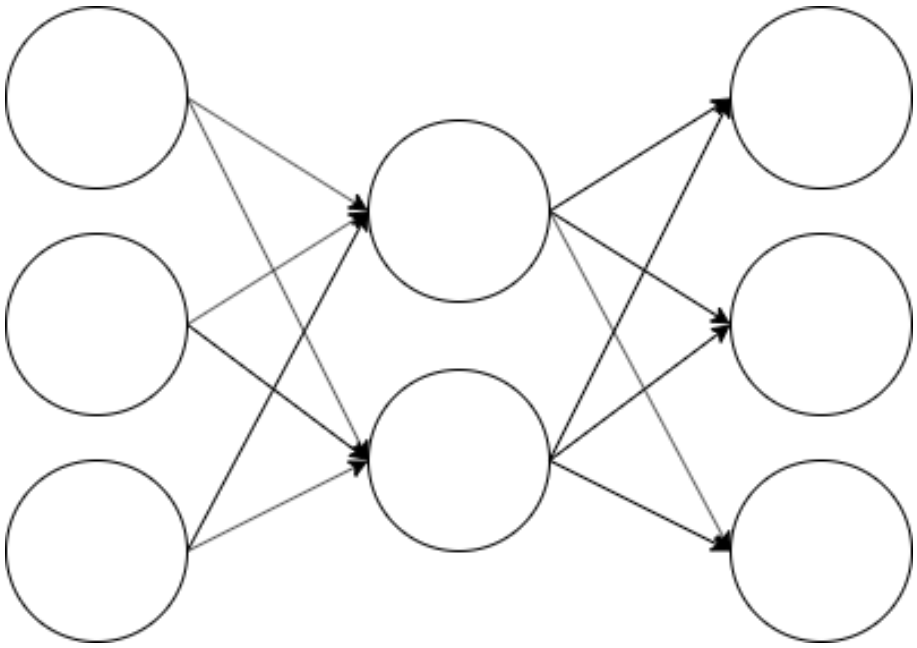


Figure 2.4 Feed forward neural network.

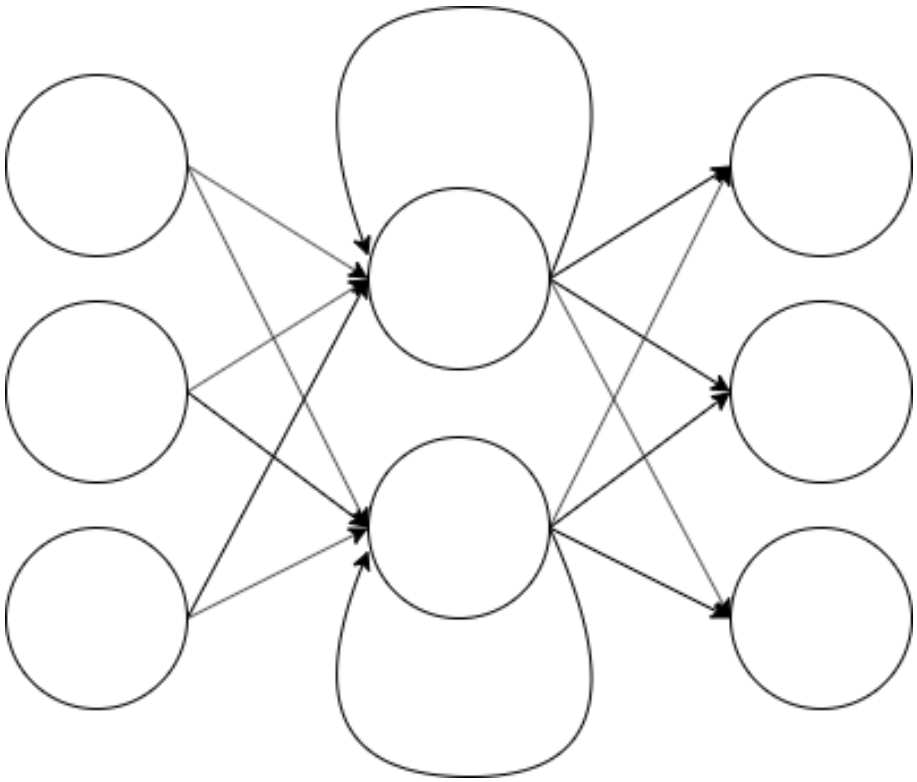


Figure 2.5 Recurrent neural network.

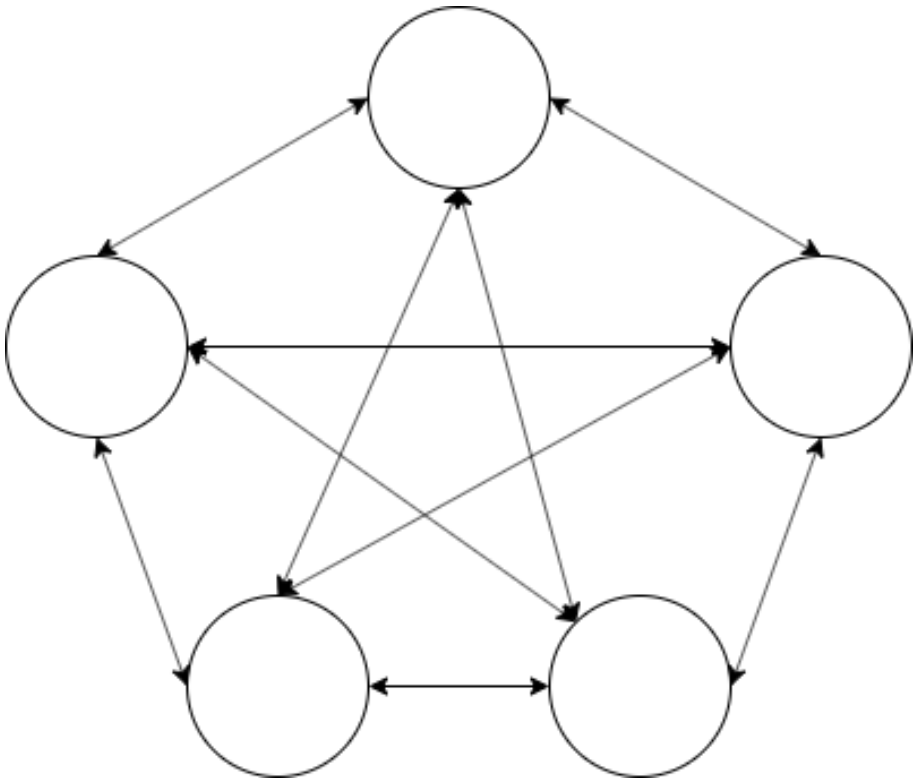


Figure 2.6 Fully recurrent neural network.

第3章　ディープラーニング

3.1　ディープラーニング

中間層を何重にも重ねたニューラルネットワークを使った学習のことをディープラーニングといい、日本語では深層学習と表現する。

ディープラーニングでは、隠れ層に構造を持たせることでより学習が効率的に進むようにされている。ディープラーニングのなかで代表的なものとして、畳み込みニューラルネットワークが挙げられる。

3.2　畳み込みニューラルネットワーク

畳み込みニューラルネットワーク (Convolutional Neural Network) とは、ニューラルネットワークと比較して画像の位置ずれや回転などの変形やゆがみに対して頑強なモデルであり、画像認識に用いられている。畳み込みニューラルネットワークの中間層は、畳み込み層とプーリング層、全結合層によって構成されている。畳み込みニューラルネットワークの全体像を Figure 3.1 に示す。

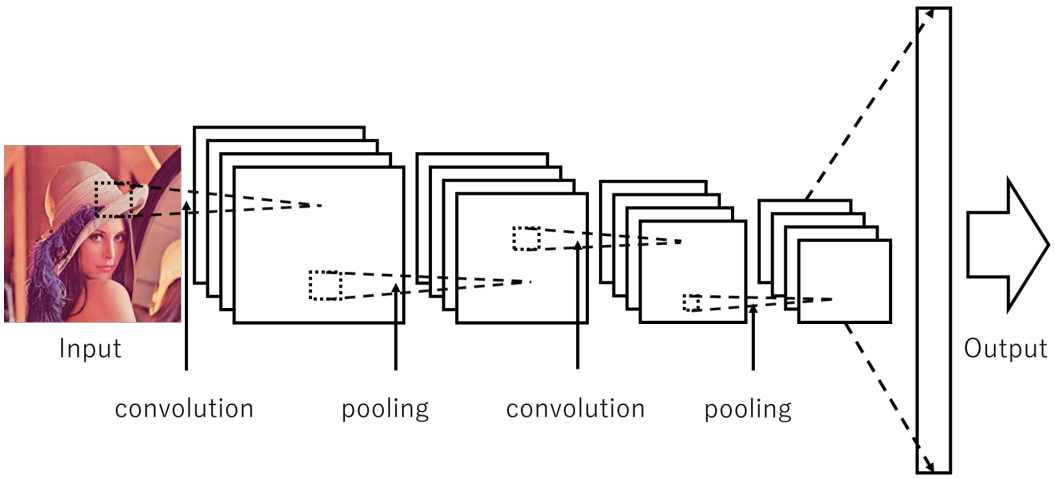


Figure 3.1　Structure of CNN.

畳み込みの様子を Figure 3.2 に示す。畳み込み層では画像に指定した範囲で様々なカーネルと呼ばれるフィルターをかけている。このフィルターとフィルターをかけられた画像の範囲の類似度の比較を行うことで画像内の特徴量が抽出することができる。このとき、フィルターをかける範囲を 1 ピクセルずつずらすことで、多くの特徴量が抽出でき

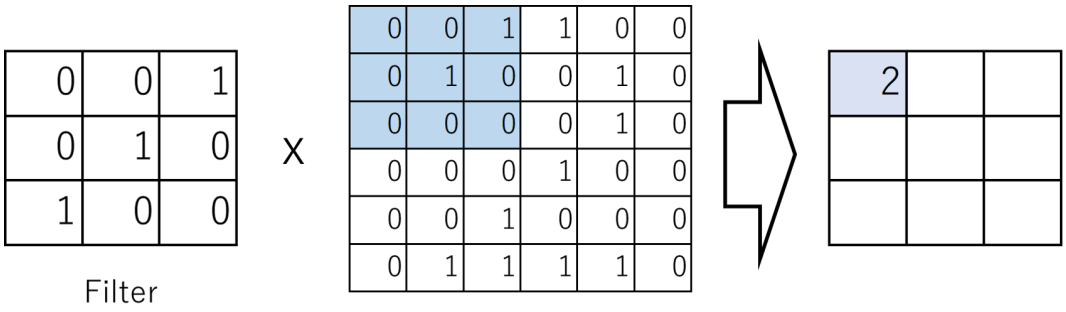


Figure 3.2　Convolution.

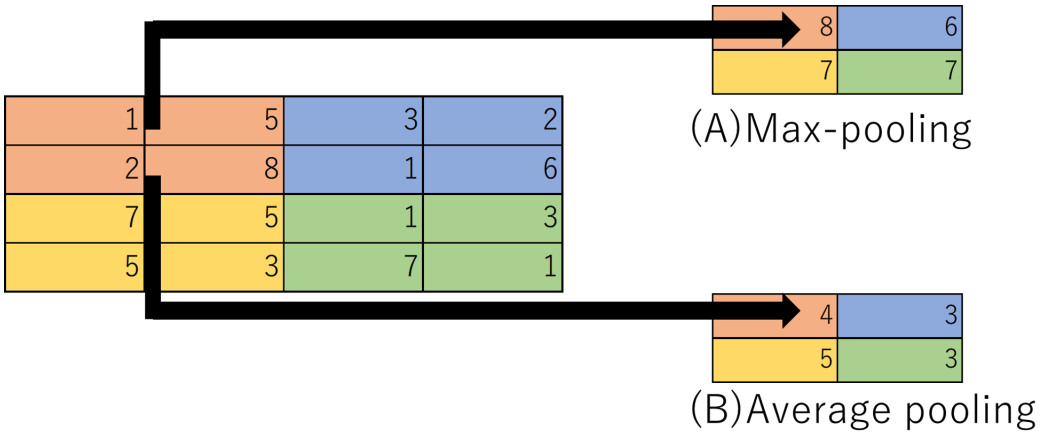


Figure 3.3　Pooling.

る。また、僅かにピクセルがずれたデータを大量に抽出しているため、数ピクセルずれが生じた画像に対しても高い精度で認識を行うことができる。これは、人間の視覚野が持つ局所受容野と同様の働きをしているとみることができる。畳み込みを行い、抽出した特徴量を行列として保持したものを特徴マップと言う。ここで、画像内のどの領域からも正しく特徴を抽出するために、処理する領域に関わらず重みとバイアスはフィルターごとに共通の値を用いている。

プーリング層の目的は特徴マップの空間的特徴を保持したままサイズを圧縮して扱いやすくすることである。そのため、畳み込み層と同様に一定の領域内の値を集約する作業を行う。しかし、畳み込みとは異なり値に対して重みをかけることはしない。

プーリングには 2 種類の方法があり、Max-pooling と Average pooling である。Max-pooling は一定領域内の最大値を抽出するもので、Average pooling は一定領域内全ての値の平均値を抽出するものである。Figure 3.3 にデータに対して $s \times 2$ の領域で各 pooling を実行したときの結果を示す。

全結合層では、何度か畳み込みとプーリングを行った結果の多次元データを 1 次元の

フラットなベクトルに変換し、それを用いることにより画像を認識、分類している。

普通のニューラルネットワークでの画像処理では、画像を直接１次元のフラットなベクトルに変換して処理を行うが、これでは画像の隣り合うピクセル同士の関係、すなわち空間的特徴を学ぶことができない。そのため、情報量の多い複雑な画像に対して画像処理を行う場合畳み込みニューラルネットワークを用いることで精度の高い結果を得ることができる。[3),4)]

第4章　GAN

4.1　GAN

敵対的生成ネットワークとは、Generative Adversarial Network（GAN）とも呼ばれる2014年にGoodfellow氏らが発表したモデルの事である。このモデルでは用意されたデータから特徴を学習し、疑似的なデータを生成することができる。Figure 4.1にGANの構造を示す。

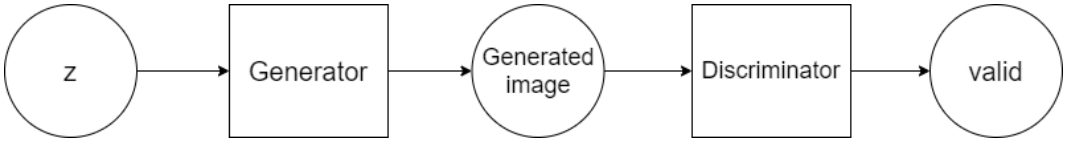


Figure 4.1　Structure of GAN.

GANはGeneratorとDiscriminatorの二つのブロックによって構成されている。GANの動作は、まずノイズzを入力としてGeneratorが画像を生成する。次に生成したデータ、または本物の学習データを入力としてDiscriminatorが入力されたデータが生成データであるか、学習データであるか判別する。そして本物の場合は1、偽物の場合0を出力するというものである。

これらは次の式で表される目的関数を最適化するように学習を行う。

$$\min_G \max_D V(D,G) = E_{x\sim p_{data}}[\log D(x)] + E_{z\sim p_z(z)}[\log(1 - D(G(z)))] \tag{4.1}$$

G :Generator

D :Discriminator

x :訓練データ

z :ノイズ

この時、適切なGが得られなければ最適なDを求めることができず、逆もまた同様である。そのため、GeneratorとDiscriminatorを分けて交互に学習させる。どちらの学習においても画像が生成データか、正しい学習データかを示すラベルとDiscriminatorの出力が等しいとき、目的関数は小さくなる。まず、Discriminatorの学習の際にはGの値を固定しFigure 4.2の形式にすることで式(4.1)における右辺を最大化する。このとき、入力が生成データであるときは0のラベルを、入力が本物の学習データであるときは1のラベルを与える。

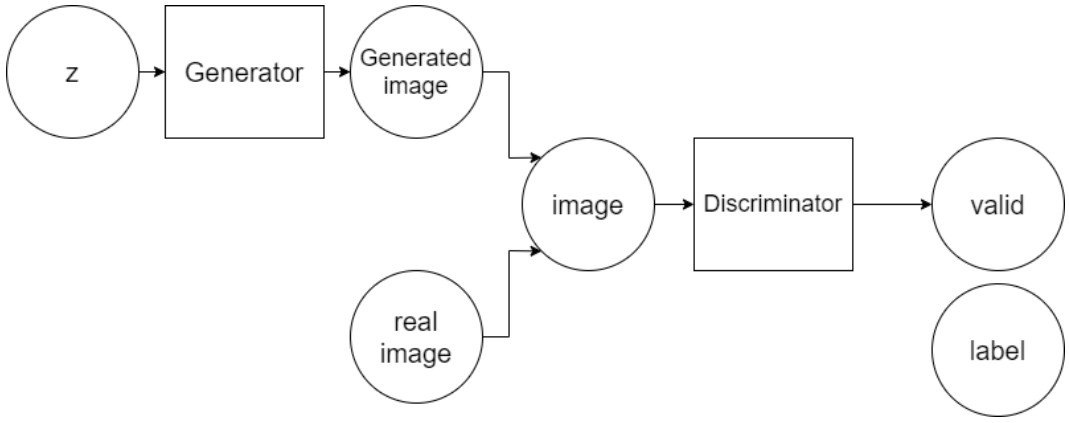


Figure 4.2　Discriminator learning model.

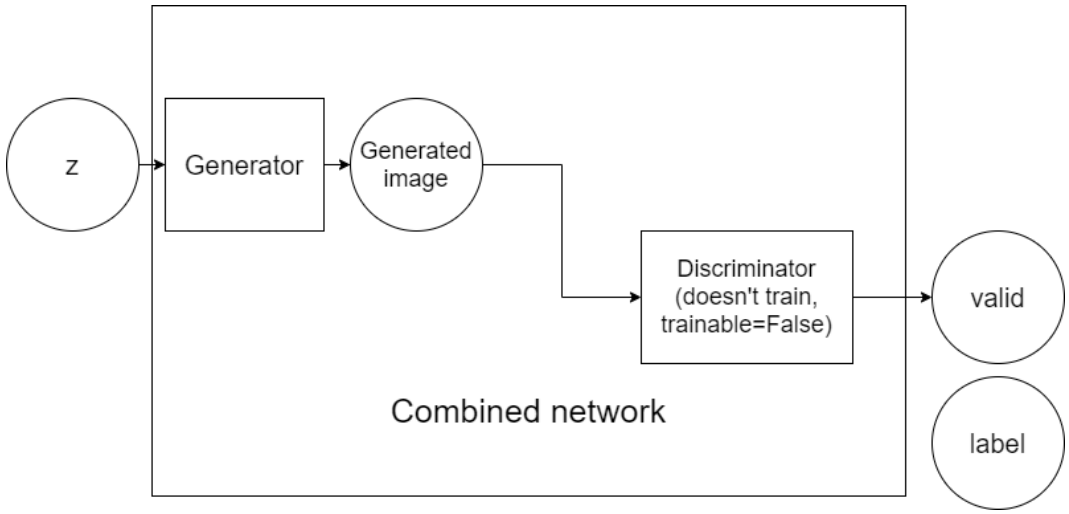


Figure 4.3　Generator learning model.

次に、Generator の学習の際には D の値を固定し Figure 4.3 の形式にすることで式 (4.1) における右辺第二項を最小化するようにする。ここで最小化させるためには、$D(G(z))$ が 1 となる必要がある。そのため、Generator を学習させる際には入力として与える生成データにつけるラベルを本物の学習データを表す 1 とすることで、Generator の目的関数を小さくすることができる。

以上の通りに学習を行うと Generator は Discriminator が判別できない程に精巧なデータの生成ができるように、Discriminator は用意された訓練データと Generator が生成したデータを正確に区別できるように最適化される。

この互いが相反した学習を行うことが GAN の特徴であり、敵対的生成ネットワークという名前の由来となっている。[5),6)]

4.2 ACGAN

ACGANとは、Augustus Odena氏らが2017年に発表したモデルである[7]。従来のGANと異なりDiscriminatorが入力画像に対してrealとfakeだけでなく入力画像の属するクラスも出力することで、それまで以上に高解像度の画像を生成することができる。式(4.2)と式(4.3)にACGANの損失関数を示す。

$$L_s = E[\log P(S = real \mid X_{real})] + E[\log P(S = fake \mid X_{fake})] \tag{4.2}$$

$$L_c = E[\log P(C = c \mid X_{real})] + E[\log P(C = c \mid X_{fake})] \tag{4.3}$$

X_{real} :本物の学習データの画像

X_{fake} :Generatorが生成した画像

$P(S = X \mid Y)$:Yが与えられたときに、Xを返した確率

C :画像が属するクラス

$P(C = c \mid Y)$:与えられた画像Yをクラスcに分類した確率

Discriminatorは$L_s + L_c$を最大化するように、Generatorは$L_s - L_c$を最大化するようにそれぞれ最適化している。

4.3 CycleGAN

CycleGANとは、Jun-Yan Zhu氏らが2017年に発表した、教師なし学習による画像から画像への変換を行う敵対的生成ネットワークモデルのことである。ここにおける教師なし学習とは、以下のFigure 4.4に示すように画像群x_iとそれに対応した画像群y_iというペアの画像を与えるのではなく、画像群Xに対して対応の無い画像群Yというアンペアな画像を与えることである。ペアの画像は用意が非常に困難であり、特に風景写真を歴史上の芸術家の作品風に変換するという命題に対して、その芸術家が描いた風景写真の場所の画像を用意することは不可能であるため、現実的に収集可能である2つの画像群と、その間にある関係性の学習という点がCycleGANの特徴となっている。

このモデルはFigure 4.5に示すようなドメインXからドメインYへの生成器GとドメインYの画像yとドメインXの画像xを変換した$G(x) = Y'$の2つの真偽を判定する識別器D_y、ドメインYからドメインXへの生成器Fと同様の識別器D_xが存在している。実際に2つのドメインXとYの間の変換を学習させるにあたり、目的関数に「敵対性損失」と「サイクル一貫性損失」の2つの項目を与える。

敵対性損失とは、式(4.4)で表される。GANの目的関数と類似した構造をしており、

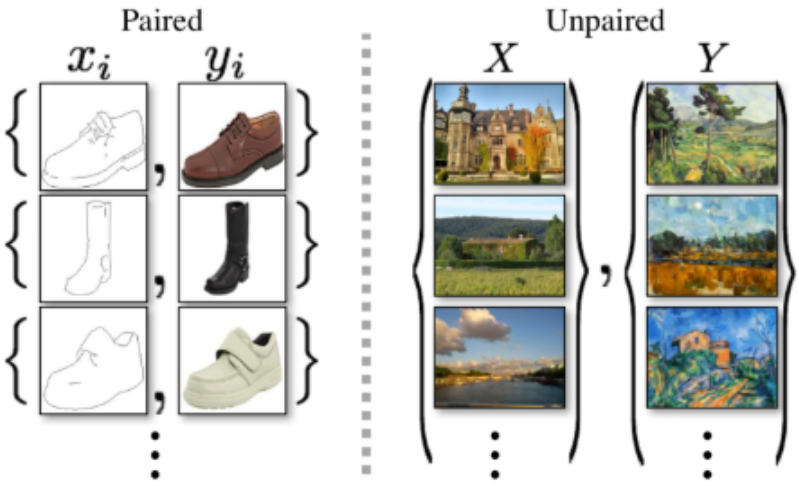


Figure 4.4　Paired data and Unpaired data.[8)]

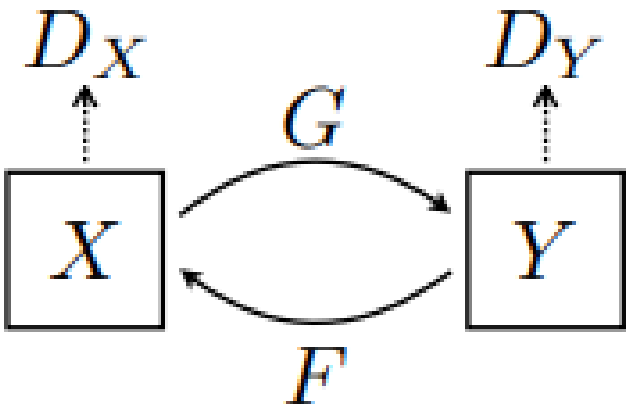


Figure 4.5　Model of CycleGAN.[8)]

GAN と同様に D_y が実際のターゲットドメイン Y のデータを入力として受け取った時に 1 を、G によって生成されたデータを受け取った時に 0 を出力することで目的関数を小さくするものである。しかし、大きな相違点として、敵対性損失にはターゲットドメインを X としたものも存在する。つまり、敵対性損失は以下の式 (4.5) と式 (4.6) の 2 つが存在するということになる。

$$\mathcal{L}_{GAN}(G, D_Y, X, Y) = E_{x\sim p_{data}(y)}[\log D_Y(y)] + E_{x\sim p_{data}(x)}[\log(1 - D_Y(G(x)))] \quad (4.4)$$

G :ターゲットドメイン Y に近いイメージ G(x) を生成する Generator

D_y :生成されたイメージ G(x) と実際のドメイン Y のデータ y を識別する Discriminator

$$\min_G \max_{D_Y} \mathcal{L}_{GAN}(G, D_Y, X, Y) \quad (4.5)$$

$$\min_F \max_{D_X} \mathcal{L}_{GAN}(F, D_X, Y, X) \quad (4.6)$$

サイクル一貫性損失とは、L^1 ノルムを用いて以下の式で表すことができる。

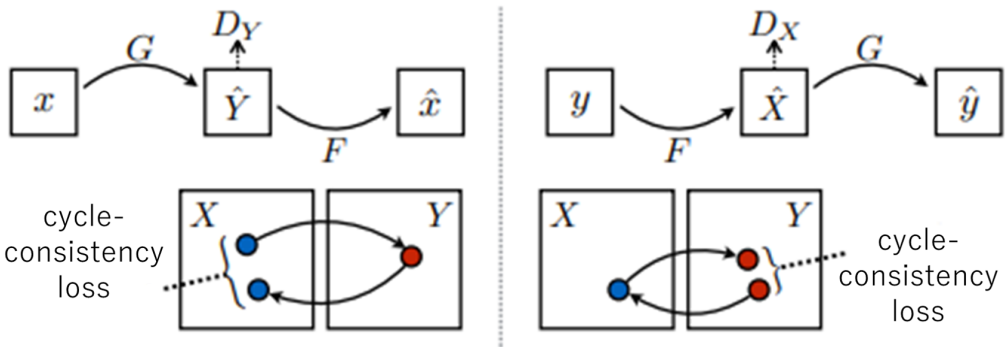


Figure 4.6　Cycle consistency losses.[8)]

$$\mathcal{L}_{cyc}(G,F) = E_{x\sim p_{data}(x)}[||F(G(x)) - x||_1] + E_{y\sim p_data(y)}[||G(F(y)) - y||_1] \quad (4.7)$$

G :ターゲットドメイン Y に近いイメージ G(x) を生成する Generator

F :ターゲットドメイン X に近いイメージ F(y) を生成する Generator

Figure 4.6 に示すようにサイクル一貫性損失とは、ターゲットドメインに対して変換を行ったイメージに対して元のドメインに戻るよう復元したイメージと元の入力データの差を比較することで求めることができる。

これらの敵対性損失とサイクル一貫性損失を組み合わせることで、最終的な目的関数を設定する。この時、これら 2 つの相対的な重みを調整するために、係数 λ を導入する。すると、以下のような式となる。

$$\mathcal{L}(G,F,D_X,D_Y) = \mathcal{L}_{GAN}(G,D_Y,X,Y) + \mathcal{L}_{GAN}(F,D_Y,Y,X) + \lambda\mathcal{L}_{cyc}(G,F) \quad (4.8)$$

最終的に、以下の式を求めることで、CycleGAN は学習を行っている。[8)]

$$G^*, F^* = \arg\min_{G,F}\max_{D_X,D_Y} \mathcal{L}(G,F,D_X,D_Y) \quad (4.9)$$

第5章　Stargan

5.1　Stargan

画像変換という分野において、CycleGAN のように 2 つのドメインの間での画像の変換は顕著な成功が示されていたものの、ドメインが 3 つ以上存在する場合には全ドメインが完全結合となるようにそれぞれモデルを構築しなければならなかったため、拡張性や堅牢性が制限されていた。これの解決策として 2018 年に Yunjey Choi 氏らが発表したモデルが StarGAN である。StarGAN では、単一のモデルで複数のドメイン間で画像の変換を行うことで、前述の問題を解決している。

Figure 5.2 に StarGAN の構造を示す。(a) は Discriminator の学習構造である。Discriminator では、ACGAN のように入力された画像に対してその画像が本物か偽物か、そして本物の画像ならばその画像はどのドメインに属する画像なのかを判別する。(b)、(c)、(d) は Generator の学習構造である。Generator は入力として本物の画像とターゲットドメインを与え、画像を生成する。生成された画像は Discriminator に送られ、本物か偽物か判別されるが、この時、生成された画像を再度 Generator の入力とし、元々の画像のドメインをターゲットドメインとして画像の再構築を行う。再構築した画像と元々の画像を比較することで、サイクル一貫性を導出している。

5.2　Adversarial Loss

式 (5.1) に StarGAN の Adversarial Loss を示す。

$$\mathcal{L}_{adv} = E_x[\log D_{src}(x)] + E_{x,c}[\log(1 - D_{src}(G(x, c)))] \tag{5.1}$$

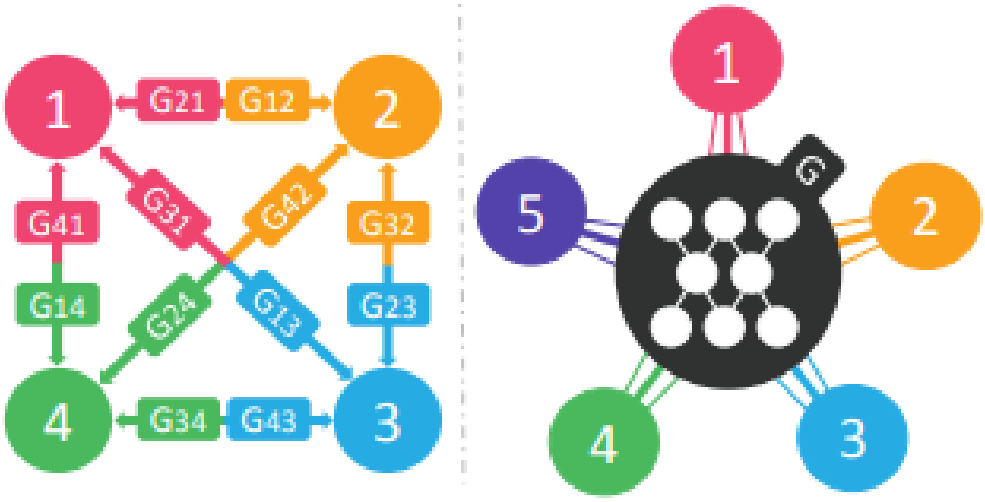
D_{src} :Discriminator からの real,fake の出力

x :元の入力画像

c :ターゲットドメイン

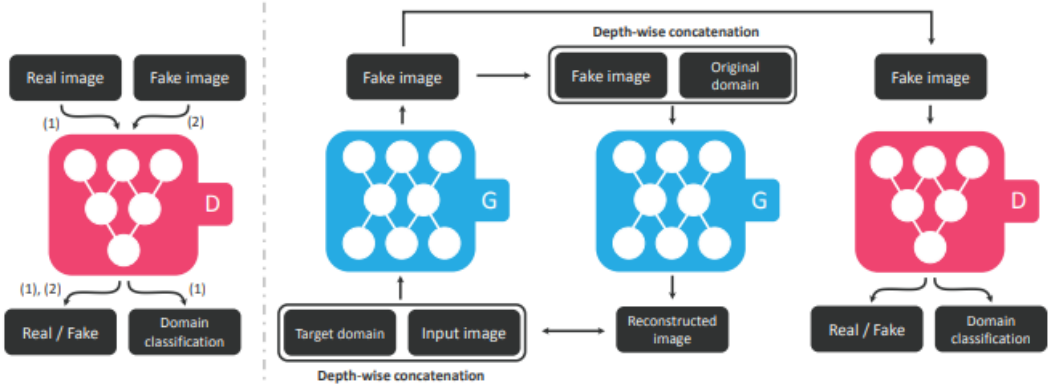
$G(x, c)$:Generater が x に対して c を適用した生成画像

これは他の GAN と同様に生成画像が本物の画像と区別できないようにするための関数であり、Discriminator ではこれを最大化させ、Generator ではこれを最小化させるように学習を行う。実際には学習を安定化させるために、WGAN で用いられている Gradient Penalty を導入し、式 (5.2) を Adversarial Loss としている。



(a) Cross-domain models (b) StarGAN

Figure 5.1 Conversion between multiple domains of each model.[9)]



(a) Training the discriminator (b) Original-to-target domain (c) Target-to-original domain (d) Fooling the discriminator

Figure 5.2 Train of discriminator and generator.[9)]

$$\mathcal{L}_{adv} = E_x[D_{src}(x)] - E_{x,c}[D_{src}(G(x,c))] - \lambda_{gp} E_{\hat{x}}[(||\nabla_{\hat{x}} D_{src}(\hat{x})||_2 - 1)^2] \tag{5.2}$$

5.3 Domain Classification Loss

式 (5.3) に Discriminator の、式 (5.4) に Generator の Domain Classification Loss を示す。

$$\mathcal{L}^r_{cls} = E_{x,c'}[-\log D_{cls}(c'|x)] \tag{5.3}$$

$$\mathcal{L}^f_{cls} = E_{x,c}[-\log D_{cls}(c|G(x,c))] \tag{5.4}$$

$D_{cls}(c, x)$:入力画像 x のドメインを c とした確率

Discriminator は式 (5.3) を、Generator は式 (5.4) を最小化させるように学習を行う。Discriminator の目的は入力画像 x を元々のドメイン c' に分類することであり、Generator の目的は入力画像 x をターゲットドメイン c へと変換させた画像 $y = G(x, c)$ を生成することである。そのため、Discriminator としては入力画像 x が正しく元のラベル c' に分類されることが好ましく、Generator としては生成画像 $G(x, c)$ が本物の画像と認識されたうえでターゲットドメインとした c と分類されることが好ましい。そのため、Domain Classification Loss は Discriminator 用のものと Generator 用のものが存在している。

5.4 Reconsutruction Loss

式 (5.4) に Reconstruction Loss を示す。

$$\mathcal{L}_{rec} = E_{x,c,c'}[||x - G(G(x, c), c')] \tag{5.5}$$

これは CycleGAN における Cycle consitency Loss と同様のものである。入力画像 x が本物のようにターゲットドメイン c に属する画像として変換、生成されたとしても元の入力画像 x の全体的な要素を失ってはならず、画像変換の目的はターゲットドメインとして指定した部分のみの変換であるため、Reconstruction Loss が適用されている。

5.5 目的関数

以上から StarGAN の目的関数は Discriminator は式 (5.6)、Generator は式 (5.7) のように与えられる。

$$\mathcal{L}_D = -\mathcal{L}_{adv} + \lambda_{cls}\mathcal{L}^r_{cls} \tag{5.6}$$

$$\mathcal{L}_G = \mathcal{L}_{adv} + \lambda_{cls}\mathcal{L}^f_{cls} + \lambda_{rec}\mathcal{L}_{rec} \tag{5.7}$$

ここで、λ_{cls} と λ_{rec} はハイパーパラメータである。

5.6 Mask Vector

StarGAN の特徴として、データセットの書式が複数にわたっても制御が可能という点がある。複数の書式によるデータセットを制御するということの問題点として、各デー

タセットから部分的にしかドメインのラベル情報を認識することができず、画像の再構成の際に元のドメインが完全な状態で存在しないために正常に比較することができないという点が挙げられる。この解決法として、StarGAN が特定のラベルを無視することができるように Mask Vector が導入されている。式 (5.8) に Mask Vector を用いた複数種類のデータセットを用いた際の入力値 $\tilde{c}$ を示す。

$$\tilde{c} = [c_1, ..., c_n, m] \tag{5.8}$$

c_i :i 番目のデータセット

n :データセットの種類数

m :Mask Vector

c_i の各データセットはバイナリ属性の場合は、バイナリベクトルとして、カテゴリー化された属性の場合はワンホットベクトルとして表され、未知のラベルはゼロベクトルとして表される。また、Mask Vector は n 次元のワンホットベクトルとして表される。

$\tilde{c}$ を用いて学習を行う時、Generator はゼロベクトルを無視して明示的に示されたラベルに焦点を当てて学習を行う。Discriminator は用いた画像の属するデータセットのラベルに対してのみ Domain classification Loss を最小とするように学習し、それ以外のデータセットのラベルに対しては最小化を行わないようにする。これを各データセット交互に行うことで、全てのデータセットの、全てのラベルの制御が可能となる。

5.7 CelebA

CelebA とは、有名人の顔画像が $178 * 218$ ピクセルで 202599 枚属性ファイル付きで集められたデータセットである。CelebA を用いて StarGAN で顔画像変換を行った結果を Figure 5.3 に示す。なお、λ_{cls}=1,λ_{rec}=10 であり、200000 回繰り返した。結果画像は左から順に元の画像、黒髪に変換した画像、金髪に変換した画像、茶髪に変換した画像、性別を反転させた画像、老若を反転させた画像となっている。

Figure 5.4 から Figure 5.10 に各種損失関数のグラフを示す。グラフの縦軸は損失、横軸は繰り返し回数となっている。[9)]

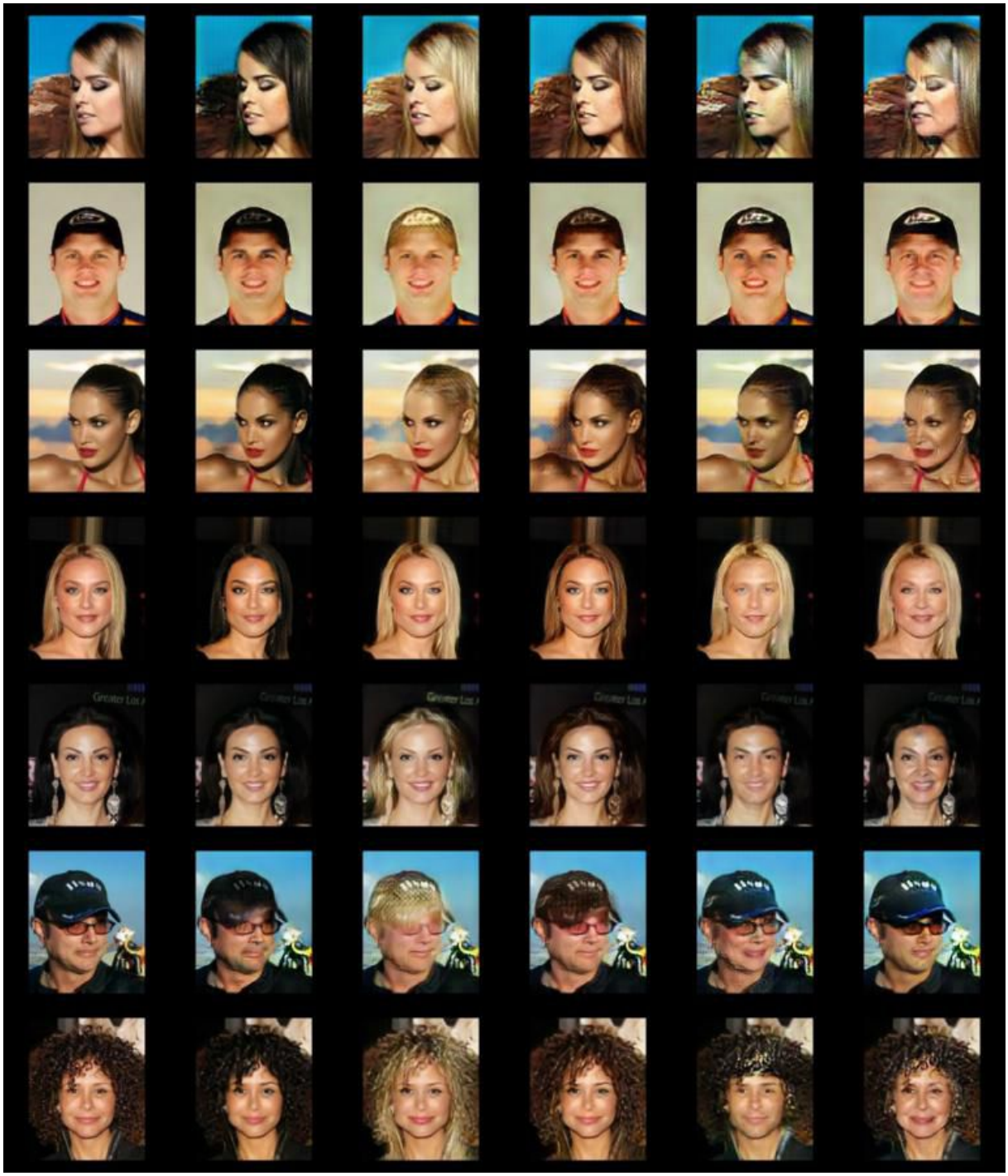


Figure 5.3　Result of CelebA.

Figure 5.4 Domain classification loss of discriminator.

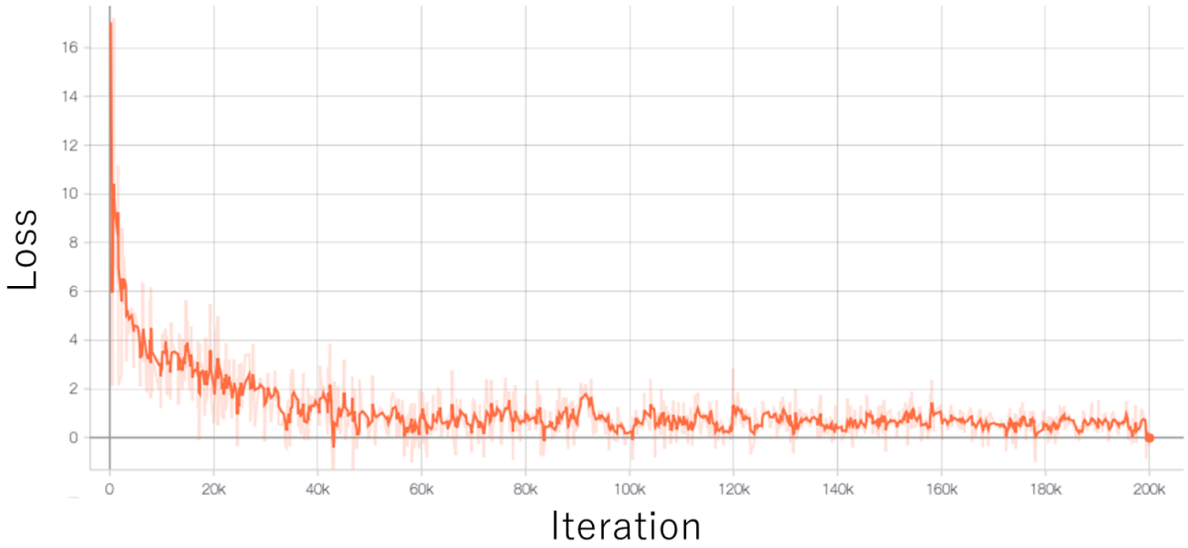


Figure 5.5 Adversarial loss on fake images of discriminator.

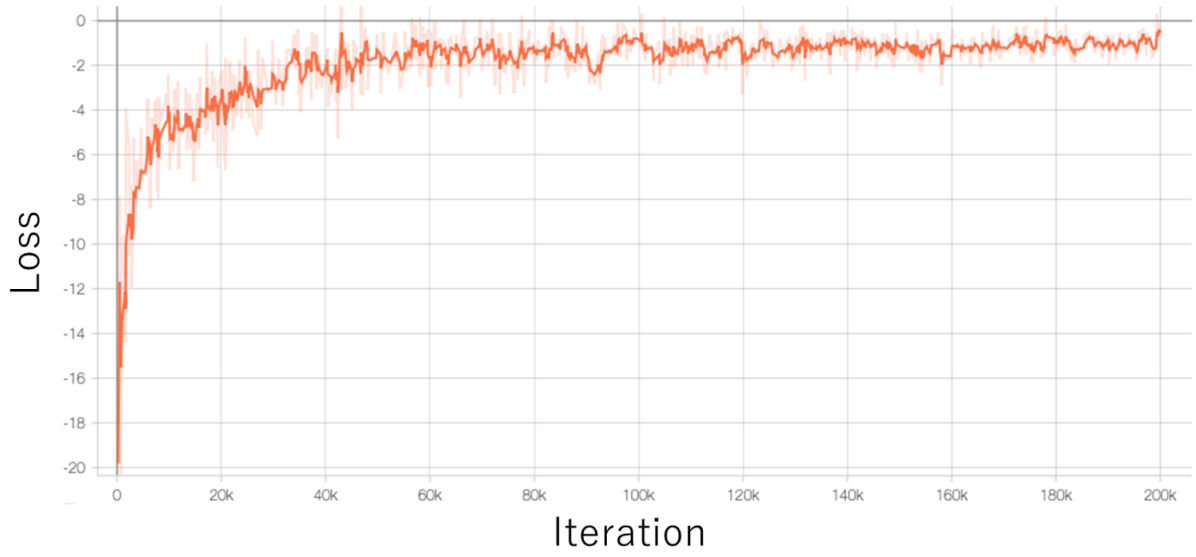


Figure 5.6 Adversarial loss on real images of discriminator.

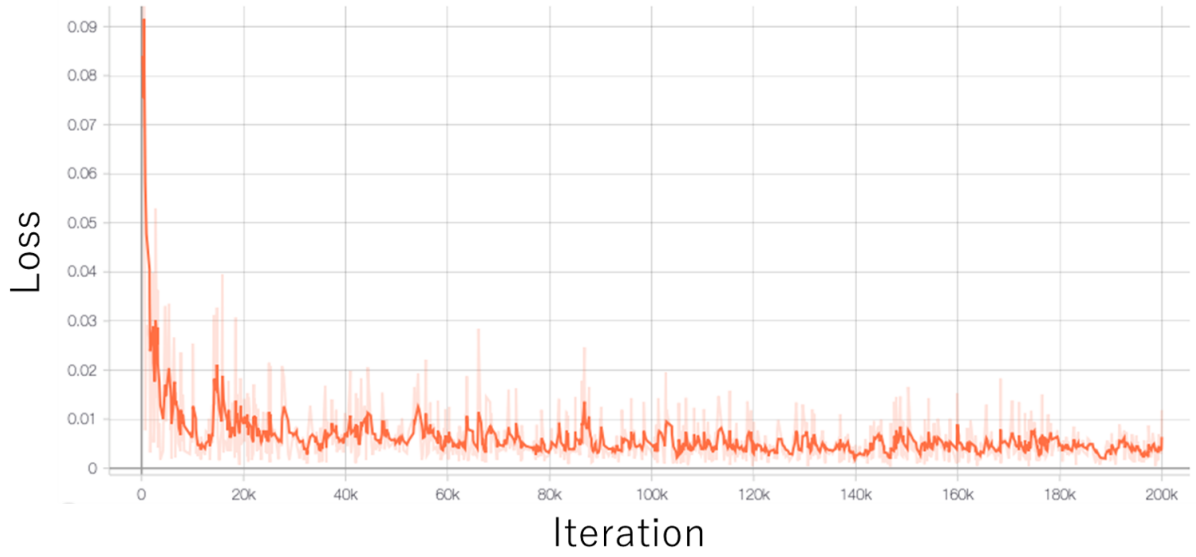


Figure 5.7 Gradient penalty loss.

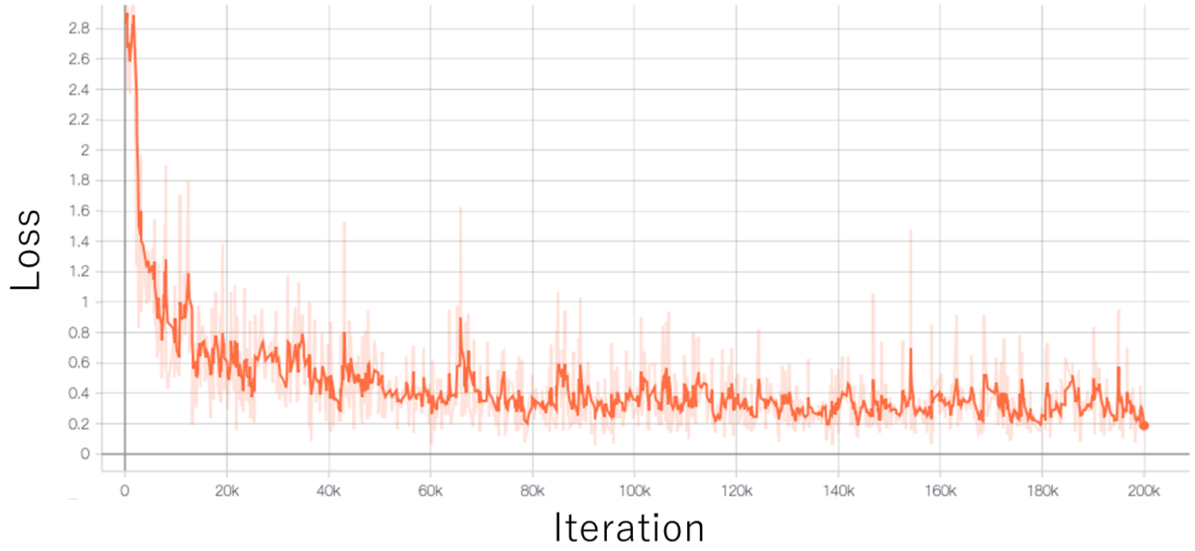


Figure 5.8 Domain classification loss of generator.

Figure 5.9 Adversarial loss of generator.

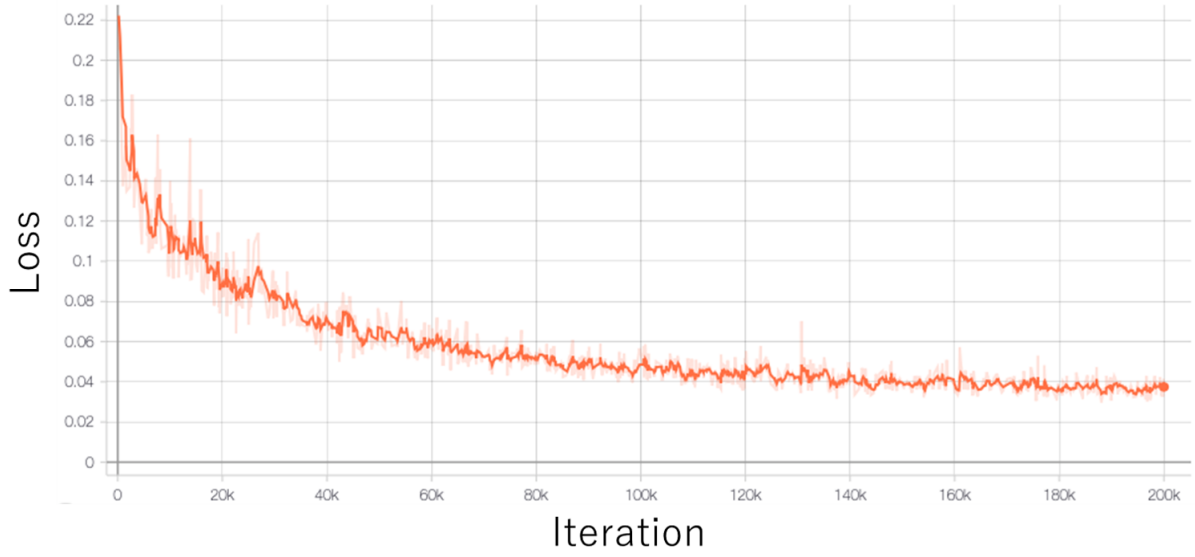


Figure 5.10 Reconstruction loss of generator.

第6章　実験

6.1　実験準備

StarGANの発表者であるYunjey Choi氏らがGit上にStarGANのモデル及びソースコードの公開を行っており[10]、それを用いて実験を行う。ネットワークはTensorflow, Pytorchを用いて作成する。このコードによって実装されるStarGANのGeneratorとDiscriminatorのネットワーク構造をFigure 6.1とFigure 6.2に示す[9]。このネットワークにてGeneratorとDiscriminatorの学習をそれぞれ16000回繰り返して結果を導出する。

学習用のデータとしてゲームアプリのキャラクター画像から顔を抽出した画像を約2500枚用意した。これらの元画像から、ウインクをさせた画像、口を閉じて笑った画像、目を閉じて笑った画像、目を開いて笑った画像、口を開いて笑った画像の5つのドメインに画像を変換させる。なお、元々口を開いている画像に対し口を開いて笑う変換を行っても変更は発生しない。

6.2　実験

Figure 6.3に実験によって得られた変換後の画像を示す。

左から順に元画像、左目を閉じてウインクをさせた変換、口を閉じて口角を上げさせる変換、目を閉じて微笑ませる変換、目を開かせる変換、口を開いて笑う変換となっている。特に変更が顕著に表れた例として、5列目の目を開いた変換と、6列目の口を開いた変換が挙げられる。5列目の1行目、4行目の画像は、元画像では目を閉じているが、変換後の画像では眼球と思われる黒い部分が目の下に形成されているのがわかる。6列目の5行目、6行目の画像では元の画像では口を閉じているものの、返還後の画像では口元が半月状に赤くなり、口を開いて微笑んでいる。特に6行目の画像に関しては、口元の赤くなった部分の周りに黒く輪郭も形成されており、よりターゲットドメインに近い変換ができている。3列目、4列目の口を閉じさせる変換、目を閉じさせる変換に関しては、それぞれ該当の部位の色がキャラクターの肌の色と同化している。口、及び目を開いている状態よりも閉じている状態の方が見える肌の面積は大きくなり、また、4列目の目の変換に関しては、眼球部のみ色が変化しており、閉じて微笑んでいる目と形状が似ているまつ毛の部分は色の変化が発生していない。これらのことより学習はある程度進んでいることが考えられるが、人間が見て表情が形成されていると認識できる段階まで

Part	Input → Output Shape	Layer Information
Down-sampling	$(h, w, 3+n_c) \rightarrow (h, w, 64)$	CONV-(N64, K7x7, S1, P3), IN, ReLU
	$(h, w, 64) \rightarrow (\frac{h}{2}, \frac{w}{2}, 128)$	CONV-(N128, K4x4, S2, P1), IN, ReLU
	$(\frac{h}{2}, \frac{w}{2}, 128) \rightarrow (\frac{h}{4}, \frac{w}{4}, 256)$	CONV-(N256, K4x4, S2, P1), IN, ReLU
Bottleneck	$(\frac{h}{4}, \frac{w}{4}, 256) \rightarrow (\frac{h}{4}, \frac{w}{4}, 256)$	Residual Block: CONV-(N256, K3x3, S1, P1), IN, ReLU
	$(\frac{h}{4}, \frac{w}{4}, 256) \rightarrow (\frac{h}{4}, \frac{w}{4}, 256)$	Residual Block: CONV-(N256, K3x3, S1, P1), IN, ReLU
	$(\frac{h}{4}, \frac{w}{4}, 256) \rightarrow (\frac{h}{4}, \frac{w}{4}, 256)$	Residual Block: CONV-(N256, K3x3, S1, P1), IN, ReLU
	$(\frac{h}{4}, \frac{w}{4}, 256) \rightarrow (\frac{h}{4}, \frac{w}{4}, 256)$	Residual Block: CONV-(N256, K3x3, S1, P1), IN, ReLU
	$(\frac{h}{4}, \frac{w}{4}, 256) \rightarrow (\frac{h}{4}, \frac{w}{4}, 256)$	Residual Block: CONV-(N256, K3x3, S1, P1), IN, ReLU
	$(\frac{h}{4}, \frac{w}{4}, 256) \rightarrow (\frac{h}{4}, \frac{w}{4}, 256)$	Residual Block: CONV-(N256, K3x3, S1, P1), IN, ReLU
Up-sampling	$(\frac{h}{4}, \frac{w}{4}, 256) \rightarrow (\frac{h}{2}, \frac{w}{2}, 128)$	DECONV-(N128, K4x4, S2, P1), IN, ReLU
	$(\frac{h}{2}, \frac{w}{2}, 128) \rightarrow (h, w, 64)$	DECONV-(N64, K4x4, S2, P1), IN, ReLU
	$(h, w, 64) \rightarrow (h, w, 3)$	CONV-(N3, K7x7, S1, P3), Tanh

Figure 6.1 Generator network architecture.[9)]

Layer	Input → Output Shape	Layer Information
Input Layer	$(h, w, 3) \rightarrow (\frac{h}{2}, \frac{w}{2}, 64)$	CONV-(N64, K4x4, S2, P1), Leaky ReLU
Hidden Layer	$(\frac{h}{2}, \frac{w}{2}, 64) \rightarrow (\frac{h}{4}, \frac{w}{4}, 128)$	CONV-(N128, K4x4, S2, P1), Leaky ReLU
Hidden Layer	$(\frac{h}{4}, \frac{w}{4}, 128) \rightarrow (\frac{h}{8}, \frac{w}{8}, 256)$	CONV-(N256, K4x4, S2, P1), Leaky ReLU
Hidden Layer	$(\frac{h}{8}, \frac{w}{8}, 256) \rightarrow (\frac{h}{16}, \frac{w}{16}, 512)$	CONV-(N512, K4x4, S2, P1), Leaky ReLU
Hidden Layer	$(\frac{h}{16}, \frac{w}{16}, 512) \rightarrow (\frac{h}{32}, \frac{w}{32}, 1024)$	CONV-(N1024, K4x4, S2, P1), Leaky ReLU
Hidden Layer	$(\frac{h}{32}, \frac{w}{32}, 1024) \rightarrow (\frac{h}{64}, \frac{w}{64}, 2048)$	CONV-(N2048, K4x4, S2, P1), Leaky ReLU
Output Layer (D_{src})	$(\frac{h}{64}, \frac{w}{64}, 2048) \rightarrow (\frac{h}{64}, \frac{w}{64}, 1)$	CONV-(N1, K3x3, S1, P1)
Output Layer (D_{cls})	$(\frac{h}{64}, \frac{w}{64}, 2048) \rightarrow (1, 1, n_d)$	CONV-(N(n_d), K$\frac{h}{64}$x$\frac{w}{64}$, S1, P0)

Figure 6.2 Discriminator network architecture.[9)]

は到達しなかった。一方で、2 列目のウインクをさせる変換に関してはほとんど目的のドメインへの変化が発生していないように見られる。この原因として、他の 4 つのデータと比較して、ウインクの画像が特に少なかったことが原因として考えられる。全体としてみると、キャラクターの肌の色、髪の毛の色、背景や髪飾りなど、ターゲットドメインとして与えられていない部分にはほとんど変化が発生していないことがわかる。これより、ターゲットドメインのみを変換するという目的は達成していることが分かる。

Figure6.4 から Figure6.10 にこのデータセットを用いた時の各損失関数を示す。データの少なさや繰り返し回数の少なさからグラフは収束に向かってはいるもののぶれが大きく、また、損失関数の正負の反転が CelebA の結果よりも激しく見られた。その一方で、唯一 Generator の Reconstruction loss のみかなり値が安定して収束していた。このこと

Figure 6.3　Result of generation 2D character's expression.

からも、少ないデータ数且つ少ない試行回数ではあったが生成画像が元の画像の全体的な特徴を保持できたことがわかる。

Figure 6.4 Domain classification loss of discriminator.

Figure 6.5 Adversarial loss on fake images of discriminator.

Figure 6.6 Adversarial loss on real images of discriminator.

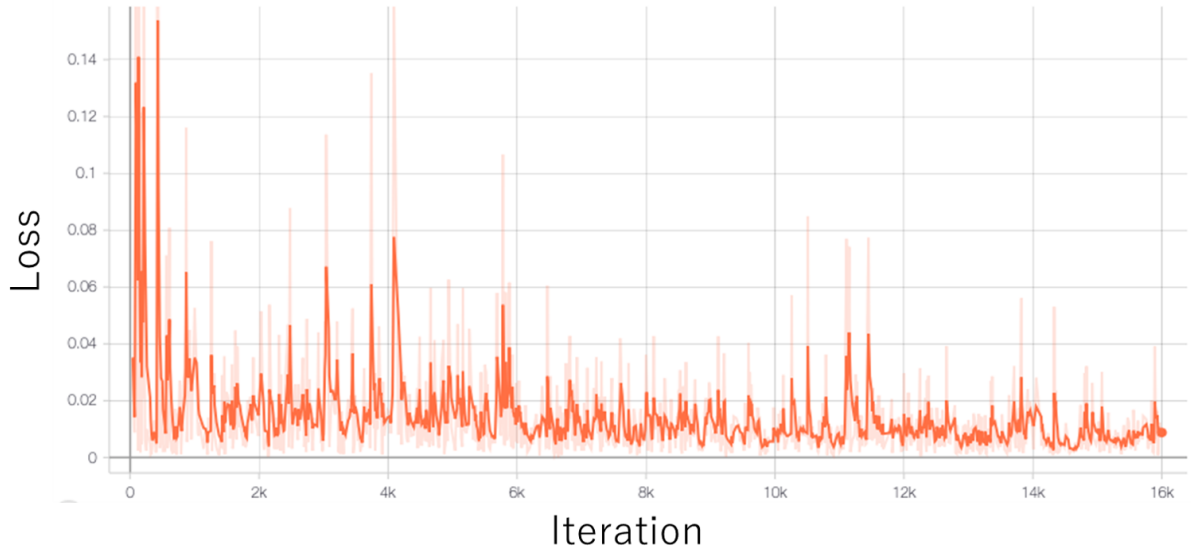


Figure 6.7 Gradient p enalty loss.

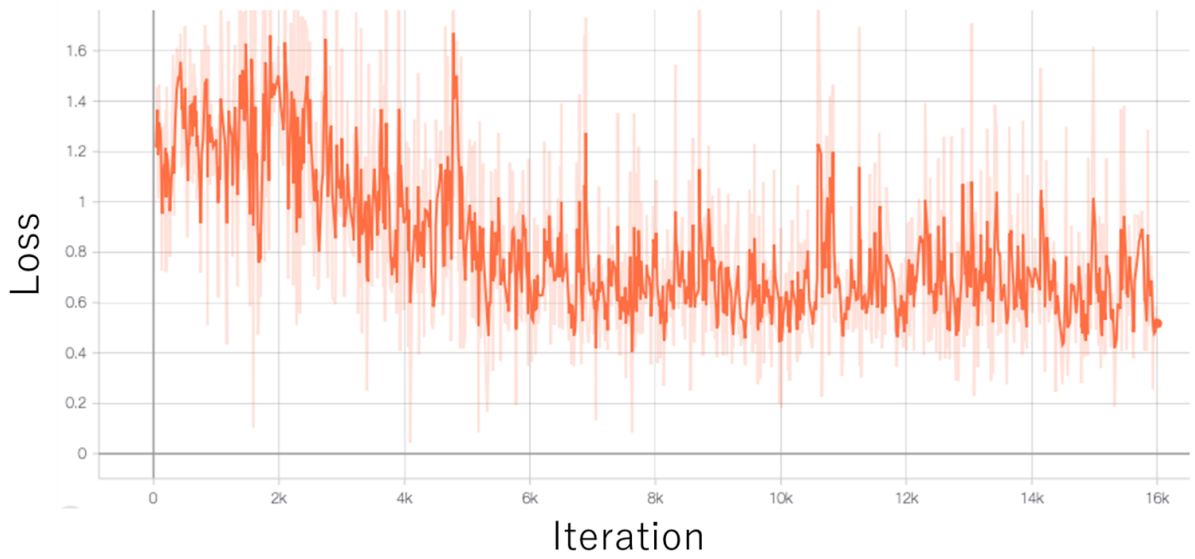


Figure 6.8 Domain classification loss of generator.

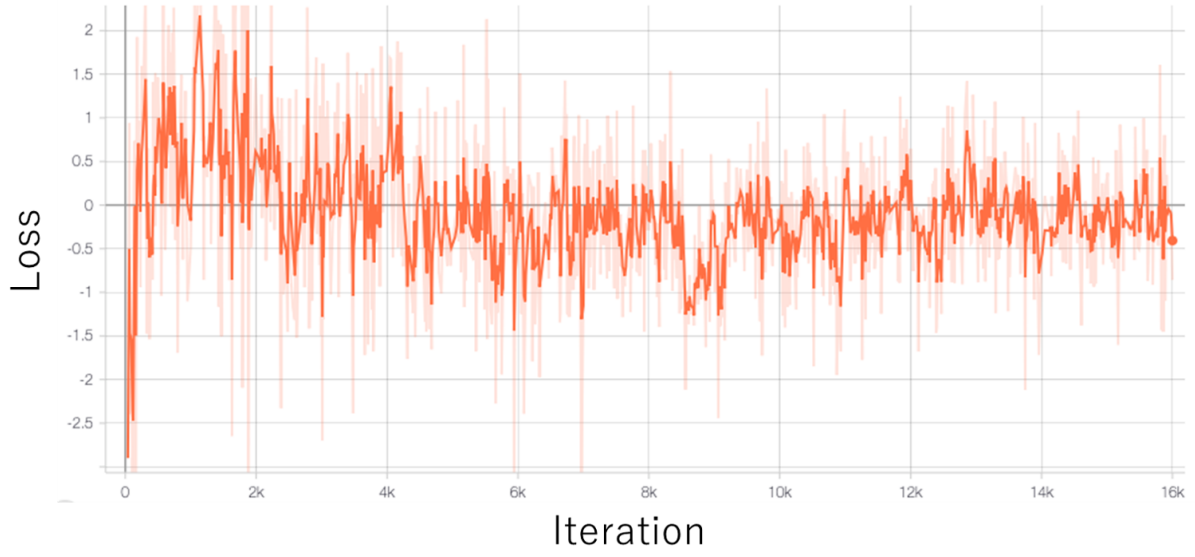


Figure 6.9 Adversarial loss of generator.

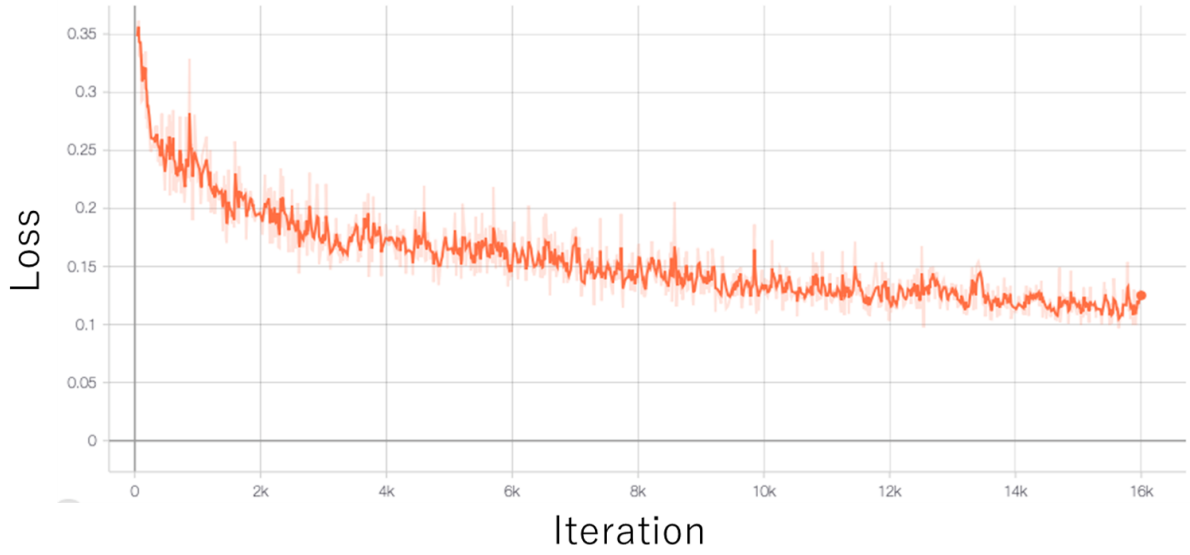


Figure 6.10 Reconstruction loss of generator.

第7章　結論

本研究では、StarGAN を用いた複数ドメイン間における画像の表情変換を行った。

CelebA の画像は約 200000 枚集まっている点に対し、自作データセットには約 2500 枚しか画像を集めることができず、また、時間の関係により事前実験では設定された学習の繰り返し回数 200000 回を終了できたのに対し、自作データセットによる実験では 16000 回しか繰り返しを行うことができなかった。そのため眼球と思わしき部位や口元の色合いなど、概念的な変換をするに留まった。

また、今回の実験では λ_{cls} や λ_{rec} のようなハイパーパラメータを変化させたときの各損失関数の変化の確認をするに至らなかった。StarGAN で用いられたハイパーパラメータは現実の人間の顔の構成要素の変換に最適化されたものであるため、各パーツの大きさ、色合いなどの違いから 2D キャラクターに適用するにはより最適なパラメータが存在することが考えられる。

今回の実験の結果では、理想とする趣味の充実という点を達成する点にはまだ遠いが、今後このシステムを発展させるために進むべき道が示された。今後はより高精度な表情の生成や、より多岐にわたる画風の 2D キャラクターに対して表情が生成できるように研究を続けていきたい。

謝辞

本研究を進めるにあたり、御多忙中にも関わらず多大なご指導を賜りました出口利憲先生に深く感謝すると共に、同研究室において共に勉学に励んだ浅野蒼汰氏、日下部完氏に厚く御礼を申し上げます。

参考文献

1) 木島博正, 木島祥子:ニューロンとシナプスにおける情報の変換と伝達, 日本物理学会誌, 47 巻 4 号, p.277-284 （1992）
2) 久保宏一郎:Studies on learning of layered neural networks and its applications, 九州芸術工科大学博士論文（1998）
3) 涌井良幸, 涌井貞美:ディープラーニングがわかる数学入門 （2017）
4) Antonio Gulli, Sujit Pal 著, 大串正矢, 久保隆宏, 中山光樹 訳:直感 DeepLearning（2018）
5) Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley,Sherjil Ozair, Aaron Courville, Yoshua Bengio:Generative Adversarial Nets, Advances in Neural Information Processing Systems 27（2014）
6) 今さら聞けないGAN（1） 基本構造の理解:https://qiita.com/triwave33/items/1890ccc71fab6cb 2020 年 1 月 31 日閲覧
7) Augustus Odena, Christopher Olah, Jonathon Shlens:Conditional Image Synthesis with Auxiliary Classifier GANs(2017)
8) Jun-Yan Zhu, Taesung Park, Phillip Isola, Alexei A. Efros:Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks,Berkeley AI Research (BAIR) laboratory, UC Berkeley（2017）
9) Yunjey Choi, Minje Choi, Munyoung Kim, Jung-Woo Ha, Sunghun Kim, Jaegul Choo :StarGAN: Unified Generative Adversarial Networks for Multi-Domain Image-to-Image Translation
10) Yunjey Choi, Minje Choi, Munyoung Kim, Jung-Woo Ha, Sunghun Kim, Jaegul Choo :StarGAN - Official PyTorch Implementation :https://github.com/yunjey/stargan 2020 年 2 月 13 日閲覧