

平成 29 年度

卒業研究報告書

敵対的生成ネットワークを利用した  
高解像画像生成の研究

指導教官 藤本 健司  
報告者 入江 一帆

神戸市立工業高等専門学校  
電子工学科

## (論文要旨)

近年、深層学習と呼ばれる手法が、分類、認識の分野で成果を上げている。更には敵対的生成ネットワークを用いた画像生成モデルが考案された。しかし、現在の生成モデルでは、学習コストや出力精度の問題により、高解像度の画像を生成することが難しい。

そこで本研究では、現在盛んに研究されている画像生成モデルである Deep Convolutional Generative Adversarial Networks や、他の敵対的生成ネットワークを用いた高解像度の画像生成モデルを開発することを目的とした。

本研究では、はじめに、画像生成モデルの適切なネットワーク構成の検討や、Hyperparameter の調整を行った。その後、生成モデルを多段階に構成する手法と、生成モデルで生成した画像と超解像ネットワークを組み合わせたモデルを提案し、それぞれの出力結果を比較検討した。

実験の結果として、生成モデルにおいて、適切に調整すると生成画像の出力精度が向上する Hyperparameter と、学習コストと出力精度がトレードオフの関係である Hyperparameter を区別できた。また、画像生成モデルと超解像ネットワークを組み合わせることにより、学習コストを抑えて高解像度の画像を出力できることがわかった。

Recently, a method called deep learning has been successful in the field of classification and recognition. Furthermore, an image generative model using Generative Adversarial Networks has appeared. However, it is difficult for the current model to generate high resolution images due to learning costs and output accuracy problems.

Therefore, in this research, we aimed to develop a high resolution image generative model using Deep Convolutional Generative Adversarial Networks, or other Generative Adversarial Networks.

In this research, first, we examined the network structure, and we adjusted the Hyperparameter. After, we proposed a method structuring the generative model in multiple stages, in addition to a model combining the generative model and the super resolution network. Then, we compared each output.

As a result, in the generative model, the Hyperparameter that improves the accuracy of the generated image when it is adjusted, and the Hyperparameter that are trade-off for the learning cost and the output accuracy, were distinguished. Further, we found that combining the image generative model with the super resolution network is able to output a high resolution image for less learning cost.

# 目 次

第 1 章 序論	1
第 2 章 理論	2
2.1 Neural Network . . . . .	2
2.1.1 Perceptron . . . . .	2
2.1.2 多層 Perceptron . . . . .	2
2.2 活性化関数 . . . . .	3
2.2.1 Sigmoid 関数 . . . . .	3
2.2.2 双曲線正接関数 . . . . .	3
2.2.3 Rectified Linear Unit . . . . .	4
2.2.4 Leaky ReLU . . . . .	4
2.3 Convolutional Neural Network . . . . .	5
2.3.1 畳み込み . . . . .	5
2.3.2 Pooling . . . . .	6
2.3.3 Up Sampling . . . . .	6
2.3.4 Batch Normalization . . . . .	6
2.4 損失関数 . . . . .	7
2.4.1 平均 2 乗誤差 . . . . .	7
2.4.2 交差 Entropy . . . . .	7
2.5 最適化手法 . . . . .	7
2.5.1 Adaptive Moment Estimation . . . . .	7
2.6 敵対的生成ネットワーク . . . . .	7
2.6.1 生成器 . . . . .	8
2.6.2 判別器 . . . . .	8
2.6.3 敵対的生成 . . . . .	8
2.7 Deep Convolutional GAN . . . . .	8
2.8 多段階 GAN . . . . .	9
2.9 Super Resolution CNN . . . . .	9
第 3 章 実験	10
3.1 データセット . . . . .	10
3.2 DCGAN . . . . .	10
3.2.1 Hyperparameter 調整 . . . . .	11
3.2.2 出力解像度による変換 . . . . .	12
3.3 多段階 GAN . . . . .	12
3.4 SRCNN . . . . .	13
3.5 DCGAN+SRCNN . . . . .	14
3.6 画像評価 . . . . .	14
3.6.1 アンケート調査 . . . . .	14

<b>第 4 章 結果</b>	<b>16</b>
4.1 DCGAN . . . . .	16
4.1.1 Hyperparameter 調整 . . . . .	16
4.1.2 出力画像サイズによる変化 . . . . .	18
4.2 多段階 GAN . . . . .	18
4.3 SRCNN . . . . .	18
4.4 DCGAN+SRCNN . . . . .	19
4.5 画像評価 . . . . .	20
4.5.1 反復数共通のとき . . . . .	20
4.5.2 学習時間共通のとき . . . . .	20
<b>第 5 章 結論</b>	<b>21</b>
5.1 考察 . . . . .	21
5.1.1 Hyperparameter 調整 . . . . .	21
5.1.2 解像度による変化 . . . . .	21
5.1.3 DCGAN+SRCNN . . . . .	21
5.1.4 画像評価調査の結果 . . . . .	21
5.2 今後の課題 . . . . .	21

# 第1章 序論

近年，深層学習と呼ばれる手法が，教師あり学習の画像認識や音声認識などの，分類，認識の分野で成果を上げている。更には敵対的生成ネットワーク [1] を用いた画像生成モデルも考案され，教師なし学習での画像生成に成功している [2]。しかし，高解像度の画像を生成したいとき，現在の生成モデルでは，学習コストが増大してしまい，出力精度も低下するため，生成画像をそのまま利用するには有用なレベルであるといえない。そこで本論文では，高解像度の画像を出力するための手法を比較検討し，生成モデルの性能を向上させることで，有用な画像生成を実現し，創造活動の発展に寄与することを目標とした。

まず，画像生成モデルとして，Deep Convolutional Generative Adversarial Networks[3] を用いて画像を生成し，Hyperparameter を調整した。その後，高解像度の画像を生成するために，生成モデルを多段階に構成する手法と，独自に提案する，生成モデルで生成した画像を，Super Resolution Convolutional Neural Network[4] により拡大することにより，高解像度の画像を生成する手法で画像を生成し，それらの学習コスト，出力精度を比較した。

本論文の構成は次のようになっている。第 2 章では，Convolutional Neural Network，及び，各種敵対的生成ネットワークの一般理論について述べる。第 3 章では，作成する各種画像生成モデルの詳細な構造を示し，それらで生成した画像を評価する方法について説明する。第 4 章では，各種画像生成モデルにより出力された生成画像を示し，それらの評価を行う。最後に，第 5 章で本論文の結論を述べ，今後の課題についても述べる。

# 第2章 理論

本研究では、畳み込み Neural Network、及び、各種敵対的生成ネットワークを用いて画像生成を行っている。本章ではそれらの基本的な理論について説明を行う。

## 2.1 Neural Network

生物の脳は多数の神経細胞で構成される。各細胞は複数の細胞から信号を受け取って処理し、複数の細胞に信号を伝達する。これを繰り返して脳は情報を処理する。Neural Network とは、神経細胞の機能を計算機上で実現するたに作られた数式モデルである [5]。

### 2.1.1 Perceptron

図 2.1 のようなモデルを Perceptron という。入力ベクトルを  $\mathbf{x} = (x_1, x_2, \dots, x_n)$ 、重みベクトルを  $\mathbf{w} = (w_1, w_2, \dots, w_n)$ 、バイアスを  $b$  として、活性化関数を  $f$  としたとき、出力  $y(\mathbf{x})$  は以下のように表せる。

$$y(\mathbf{x}) = f(\mathbf{w}^T \mathbf{x} + b) \quad (2.1)$$

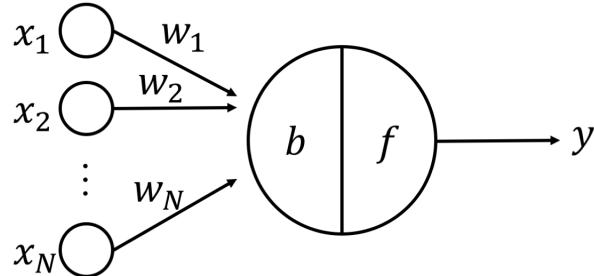


図 2.1 Perceptron

### 2.1.2 多層 Perceptron

Neural Network では Perceptron を複数接続して構成される、多層 Perceptron がよく用いられる。2 層の Perceptron は図 2.2 のように表される。入力  $\mathbf{x} = (x_1, x_2, \dots, x_I)$ 、中間層を  $\mathbf{z} = (z_1, z_2, \dots, z_J)$ 、出力を  $\mathbf{y} = (y_1, y_2, \dots, y_K)$ 、 $n$  層目の重みを  $W^{(n)}$ 、バイアスを  $b^{(n)}$  と表現すると、以下のように表せる。

$$y_k = f\left(\sum_{j=1}^J W_{kj}^{(2)} z_j + b_k^{(2)}\right) \quad (2.2)$$

$$z_j = f\left(\sum_{i=1}^I W_{ji}^{(1)} x_i + b_j^{(1)}\right) \quad (2.3)$$

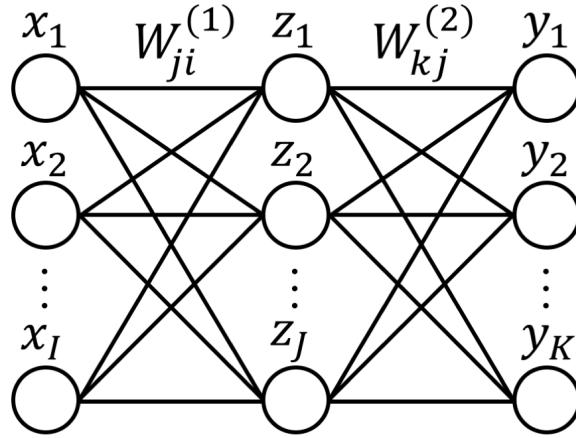


図 2.2 多層 Perceptron

## 2.2 活性化関数

ネットワークを強化または単純化するために Perceptron の出力に活性化関数を適用する。一般的に単調増加する非線形関数が用いられる [6]。

### 2.2.1 Sigmoid 関数

Sigmoid 関数は式 2.4 で表され、ステップ関数に似た性質を持つ連続な関数として、Neural Network で使われてきた [6]。定義域は  $(-\infty \leq x \leq \infty)$  であり、値域は  $(0 \leq f(x) \leq 1)$  である。 $a = 10$  のとき図 2.3 のようになる。

$$f(x) = \frac{1}{1 + \exp(-ax)} \quad (2.4)$$

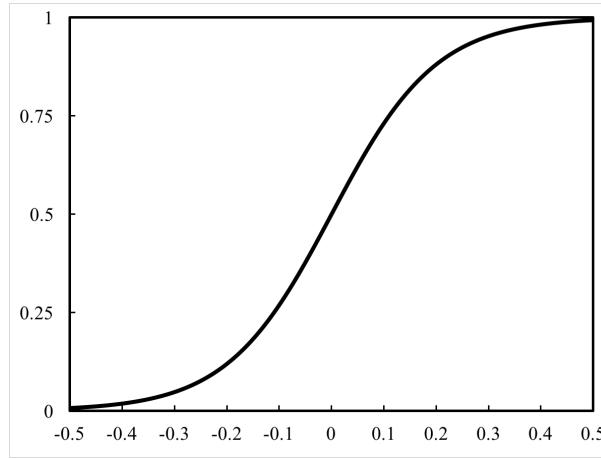


図 2.3 Sigmoid 関数

### 2.2.2 双曲線正接関数

双曲線正接関数は式 2.5 で表され、定義域は  $(-\infty \leq x \leq \infty)$  であり、値域は  $(-1 \leq f(x) \leq 1)$  である。Sigmoid 関数と似た性質を持つが値域に負の値を含む。 $a = 10$  のとき図 2.4 のようになる。

$$f(x) = \tanh(ax) \quad (2.5)$$

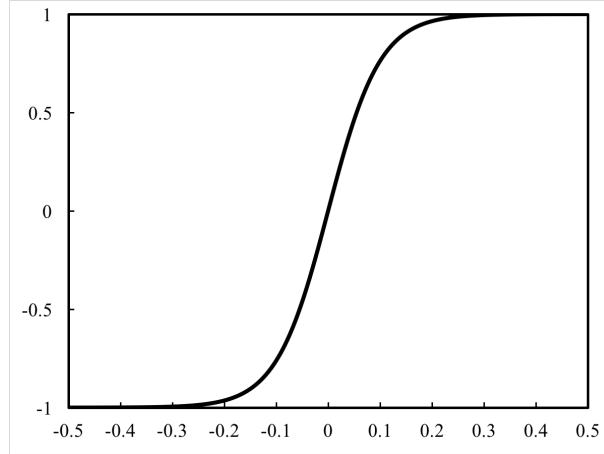


図 2.4 双曲線正接関数

### 2.2.3 Rectified Linear Unit

Rectified Linear Unit (以下, ReLU) は式 2.6 で表され, 定義域は  $(-\infty \leq x \leq \infty)$  であり, 値域は  $(0 \leq f(x) \leq \infty)$  である。単純であるが故に計算が高速である。図 2.5 のようになる。

$$f(x) = \begin{cases} x & (x > 0) \\ 0 & (x \leq 0) \end{cases} \quad (2.6)$$

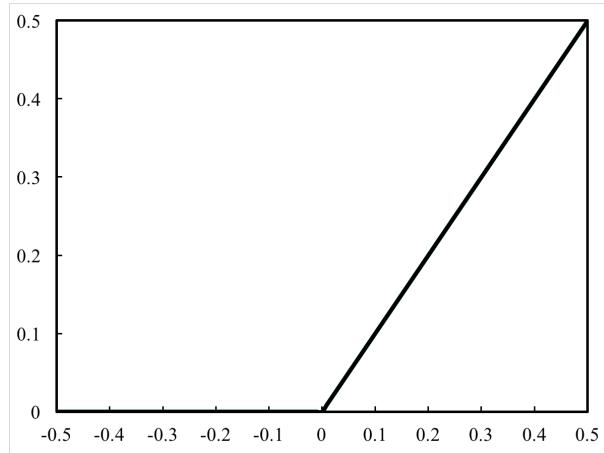


図 2.5 ReLU

### 2.2.4 Leaky ReLU

Leaky ReLU は式 2.7 で表され, 定義域は  $(-\infty \leq x \leq \infty)$  であり, 値域は  $(-\infty \leq f(x) \leq \infty)$  である。ReLU の長所はそのまま, 負側の値にも対応させている。 $a = 0.2$  のとき図 2.6 のようになる。

$$f(x) = \begin{cases} x & (x > 0) \\ ax & (x \leq 0) \end{cases} \quad (2.7)$$

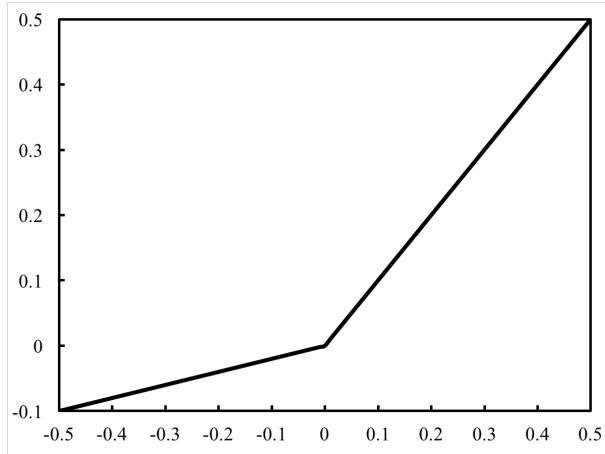


図 2.6 Leaky ReLU

## 2.3 Convolutional Neural Network

全結合層だけのネットワークに畳み込み層と Pooling 層を追加して、畳み込みフィルタのパラメータも学習するネットワークのことを Convolutional Neural Network(以下、CNN) という [7]。

### 2.3.1 畳み込み

2 階のテンソルに対する畳み込みについて考える。図 2.7 は入力  $X$  を  $I \times J$  の 2 階のテンソル、フィルタ  $F$  を  $P \times Q$  の 2 階のテンソル、出力  $Y$  を  $\hat{I} \times \hat{J}$  の 2 階のテンソルとしたときの畳み込みの様子である。入力のある点を  $X_{i,j}$ 、フィルタのある点を  $F_{p,q}$  とすると、出力のある点  $Y_{i,j}$  は次式で表せる。

$$Y_{i,j} = \sum_{p=1}^P \sum_{q=1}^Q X_{(i+p),(j+q)} F_{p,q} \quad (2.8)$$

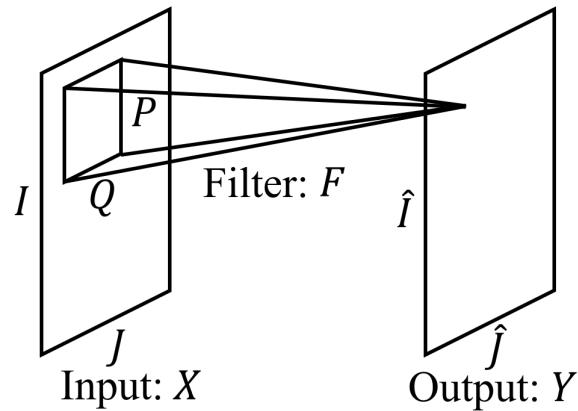


図 2.7 畳み込み

また、図 2.8 のように入力チャンネル数  $N$  と出力チャンネル数  $M$  を考慮して  $X, Y$  を 3 階のテンソル、 $n$  チャンネル目の入力を  $X_{n,i,j}$ 、 $m$  チャンネル目の出力を  $Y_{m,i,j}$  とすると次式のようになる。

$$Y_{n,i,j} = \sum_{p=1}^N \sum_{q=1}^P \sum_{r=1}^Q X_{n,(i+p),(j+q)} F_{m,p,q} \quad (2.9)$$

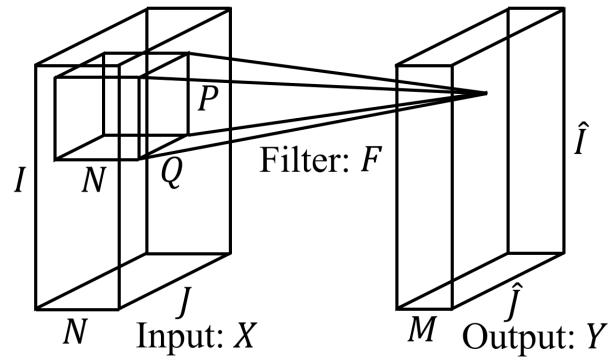


図 2.8 多チャンネル畠み込み

フィルタ適用位置間隔のことを Stride という。また、入力データの周囲を 0 で埋むことで、データサイズを大きくすることを Zero Padding という。

入力データのサイズを  $I \times J$ , 出力データのサイズを  $\hat{I} \times \hat{J}$ , Stride 幅を  $S$ , Padding のサイズを  $K$  としたとき、以下の関係式が成り立つ。

$$\hat{I} = \frac{I - P + 2K}{S} + 1 \quad (2.10)$$

$$\hat{J} = \frac{J - Q + 2K}{S} + 1 \quad (2.11)$$

### 2.3.2 Pooling

計算コストの削減や位置普遍性の獲得のために、小領域に対して特定の操作を行ってデータサイズの縮小を行うことを、Pooling という。特に小領域内の最大のものを選択する手法を Max Pooling という。小領域を  $2 \times 2$  としたときの Max Pooling の例を図 2.9 に示す。

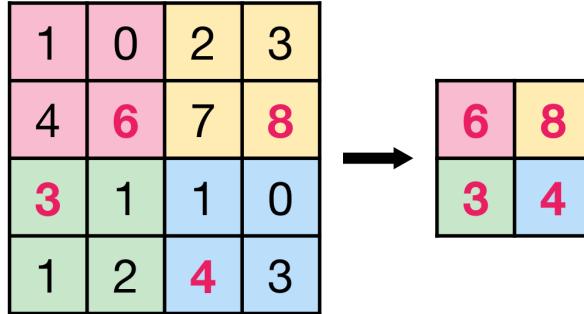


図 2.9 Max Pooling

### 2.3.3 Up Sampling

Stride 幅  $S < 1$  として畠み込みを行うと、入力データサイズよりも出力データサイズが大きくなる。これをを利用してデータサイズを拡大することを Up Sampling という。

### 2.3.4 Batch Normalization

内部共変量シフトの問題を解決するために、バッチ内のデータを、平均  $\mu = 0$ , 分散  $\sigma^2 = 1$  に変換する手法を Batch Normalization[8] という。バッチ入力を  $x = (x_1, x_2, \dots, x_N)$  とすると、変換後  $y = (y_1, y_2, \dots, y_N)$  は次式で表せる。ここで、 $\beta$ ,  $\gamma$  は学習するパラメータ、 $\epsilon$  は計算安定化のための微小な定数である。

$$\mu = \frac{1}{N} \sum_{n=1}^N x_n \quad (2.12)$$

$$\sigma^2 = \frac{1}{N} \sum_{n=1}^N (x_n - \mu)^2 \quad (2.13)$$

$$\hat{x}_n = \frac{x - \mu}{\sqrt{\sigma^2} + \epsilon} \quad (2.14)$$

$$y_n = \gamma \hat{x}_n + \beta \quad (2.15)$$

## 2.4 損失関数

Neural Network の学習では、ネットワークの出力を  $\mathbf{y} = (y_1, y_2, \dots, y_N)$ 、教師データを  $\mathbf{t} = (t_1, t_2, \dots, t_N)$  として、損失関数  $L(\mathbf{t}, \mathbf{y})$  を定義し、これを最小化する操作を行う。

### 2.4.1 平均 2乗誤差

教師データと出力がどの程度離れているのかを計算し、それらを足し合わせる。回帰問題によく用いられる。

$$L = \frac{1}{N} \sum_{n=1}^N (t_n - y_n)^2 \quad (2.16)$$

### 2.4.2 交差 Entropy

教師データと出力の Entropy を計算し、それらを足し合わせる。分類問題によく用いられる。

$$L = - \sum_{n=1}^N t_n \log y_n \quad (2.17)$$

## 2.5 最適化手法

損失関数の値からネットワークの重みを更新する最適化の手法には様々な種類がある。

### 2.5.1 Adaptive Moment Estimation

Adaptive Moment Estimation[9](以下、Adam) は、学習率を可変とし、勾配の平均と分散の指数移動平均をから最適な値に自動で更新する。更新する重みを  $\mathbf{w}$ 、学習率を  $\alpha$ 、 $\beta_1$ 、 $\beta_2$  を定数とすると、更新手順は以下のようになる。

$$\begin{aligned} m &\leftarrow \beta_1 m + (1 - \beta_1) \nabla L(\mathbf{w}) \\ v &\leftarrow \beta_2 v + (1 - \beta_2) \nabla L(\mathbf{w}) \odot \nabla L(\mathbf{w}) \\ \hat{m} &\leftarrow \frac{m}{1 - \beta_1} \\ \hat{v} &\leftarrow \frac{v}{1 - \beta_2} \\ \mathbf{w} &\leftarrow \mathbf{w} - \alpha \frac{\hat{m}}{\sqrt{\hat{v}} + \epsilon} \end{aligned} \quad (2.18)$$

## 2.6 敵対的生成ネットワーク

敵対的生成ネットワーク [1](Deep Convolutional Generative Adversarial Networks 以下、GAN) は、生成器と判別器の 2 つを組み合わせて構成され、互いに敵対しながら学習が進むことからこのように呼ばれている。

### 2.6.1 生成器

図 2.10 は生成器の動作を表したものである。生成器  $G$  は入力としてベクトル  $\mathbf{x}$  を与えると、Neural Network を順伝搬して出力  $Y_g = G(\mathbf{x})$  を生成する。

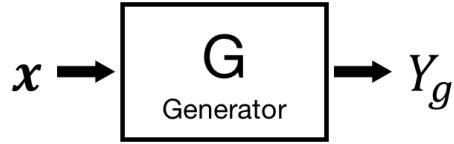


図 2.10 生成器

### 2.6.2 判別器

図 2.11 は判別器の動作を表したものである。判別器  $D$  は入力が生成器で生成したデータ  $Y_g$  であるのか、訓練データ  $Y_t$  であるのかを判別する。判別器の出力  $p$  は、Neural Network によって推定された、入力が  $Y_t$  である確率である。

判別器は判定精度の向上を目指して、 $D(Y_g) = 0$ ,  $D(Y_t) = 1$  となるように、教師あり学習で学習を繰り返す。

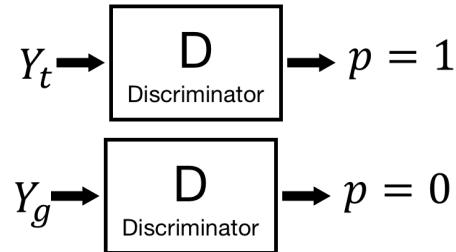


図 2.11 判別器

### 2.6.3 敵対的生成

ネットワーク全体の目的は、生成器が精度の高い出力をするようになることである。従って、 $D(Y_g) = 1$  と判別器が誤判定するような  $Y_g = G(\mathbf{x})$  を生成できるように、生成器のパラメータの調整を繰り返す。これは次式を 0 に近づけるように重みを更新するという意味である。

$$\log(1 - D(G(\mathbf{x}))) \quad (2.19)$$

## 2.7 Deep Convolutional GAN

Deep Convolutional GAN[3](以下、DCGAN) は、GAN に CNN を使い、Batch Normalization 層と活性化関数に Leaky ReLU を取り入れることで、画像生成において大きな成果を上げた手法である。生成器を CNN を用いた Up Sampling, 判別器を CNN を用いた画像分類として学習を進める。図 2.12 に DCGAN の構成図を示す。

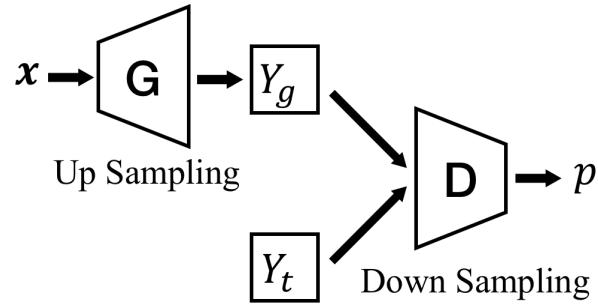


図 2.12 DCGAN 構成図

## 2.8 多段階 GAN

多段階 GAN は画像生成分野において、複数の DCGAN を組み合わせて高解像度の画像を生成する手法である。高解像度の画像を生成しようとしたとき、1 段の DCGAN で画像を生成するよりも、多段階 GAN で生成するほうが、良い結果を得られることが知られている [10]。図 2.13 に 2 段のときの多段階 GAN の構成図を示す。

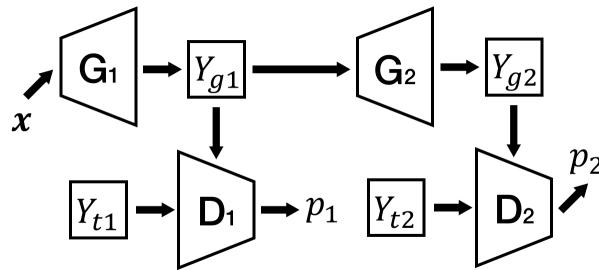


図 2.13 多段階 GAN 構成図

## 2.9 Super Resolution CNN

Super Resolution CNN[4](以下、SRCNN) は、低解像度の画像を拡大してから CNN を用いて補正を行い、高解像度の画像を推定することによって、画像の高精細化(以下、超解像)を実現するネットワークである。学習は、訓練画像  $Y$  と低解像度化した訓練画像  $X$  を対にして与えて、 $X$  から  $Y$  を生成できるようにパラメータを調整する。図 2.14 に SRCNN の構成図を示す。

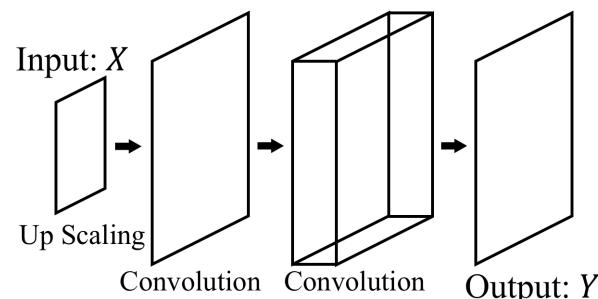


図 2.14 SRCNN 構成図

# 第3章 実験

まず、DCGAN の各種パラメータを調整して、出力精度を比較した。次に、DCGAN と多段階 GAN で高解像度の画像を生成し、それらをアンケート調査により比較した。

## 3.1 データセット

Labeled Faces in the Wild[11] で配布されている、 $250 \times 250$ [pixels] の顔画像データから 4096 枚を無作為に抽出し、これを訓練データとした。取得した画像の一例を図 3.1 に示す。



図 3.1 学習データの一例

## 3.2 DCGAN

図 3.2 に示す構成で DCGAN モデルを作成した。生成器は表 3.1 に示す 3 層の CNN による Up Sampling で、判別器は表 3.2 に示す 3 層の CNN による分類器から構成される。出力画像サイズ  $h \times h$  は  $64 \times 64$ [pixels] とした。損失関数に交差 Entropy 用いて、最適化手法は判別器で Adam( $\alpha = 1 \times 10^{-5}$ ,  $\beta_1 = 0.1$ ,  $\beta_2 = 0.999$ ), GAN 全体で Adam( $\alpha = 1 \times 10^{-4}$ ,  $\beta_1 = 0.5$ ,  $\beta_2 = 0.999$ ) として学習を行った。

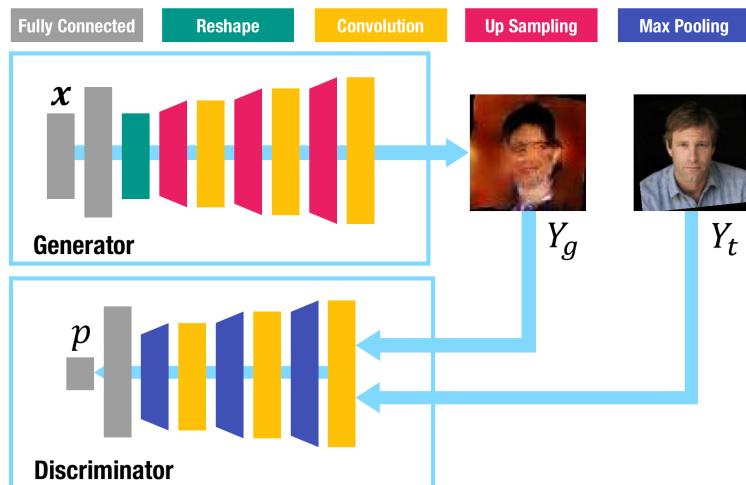


図 3.2 作成した DCGAN の構成

表 3.1 生成器の各種構成

Layer (type)	Output Shape	Parameters
Input	100	
Dense	512	
Dense	$256 \times h/8 \times h/8$	
Batch Normalization		
Leaky ReLU		
reshape	( $h/8, h/8, 256$ )	
Up Sampling	( $h/4, h/4, 256$ )	Size: (2, 2)
Convolution	( $h/4, h/4, 128$ )	Kernel: (3, 3)
Batch Normalization		
Leaky ReLU		
Up Sampling	( $h/2, h/2, 128$ )	Size: (2, 2)
Convolution	( $h/2, h/2, 64$ )	Kernel: (3, 3)
Batch Normalization		
Leaky ReLU		
Up Sampling	( $h, h, 64$ )	Size: (2, 2)
Convolution	( $h, h, 3$ )	Kernel: (3, 3)
Sigmoid		

表 3.2 判別器の各種構成

Layer (type)	Output Shape	Parameters
Input	( $h, h, 3$ )	
Convolution	( $h, h, 64$ )	Kernel: (3, 3)
Leaky ReLU		
Max Pooling	( $h/2, h/2, 64$ )	Size: (2, 2)
Convolution	( $h/2, h/2, 128$ )	Kernel: (3, 3)
Leaky ReLU		
Max Pooling	( $h/4, h/4, 128$ )	Size: (2, 2)
Convolution	( $h/4, h/4, 256$ )	Kernel: (3, 3)
Leaky ReLU		
Max Pooling	( $h/8, h/8, 256$ )	Size: (2, 2)
Flatten	$256 \times h/8 \times h/8$	
Dence	512	
Leaky ReLU		
Dence	1	
Sigmoid		

### 3.2.1 Hyperparameter 調整

Neural Network の学習を行うときに、予め設定しておく必要があるパラメータのことを Hyperparameter という。層数、ユニット数、学習係数などがこれにあたる。

表 3.1, 3.2 に示すモデルを出力画像サイズ  $h \times h = 64 \times 64$ [pixels] として作成し、以下のように条件を変えて出力精度の変化を観察した。

- 1) 判別器、生成器の中間層で活性化関数を LeakyReLU から tanh, ReLU と変化させた。
- 2) 畳み込み層のユニット数を増減させた。
- 3) 畳み込み層の層数を増減させた。
- 4) 全結合層のユニット数を増減させた。

5) 判別器学習プロセスで勾配更新を行うタイミングを変化させた。

### 3.2.2 出力解像度による変換

表 3.1, 3.2 に示すモデルを出力画像サイズ  $h \times h$  を  $32 \times 32$ ,  $64 \times 64$ ,  $128 \times 128$ [pixels] と変化させて作成し, 出力精度, 学習コストの変化を観察した。

## 3.3 多段階 GAN

図 3.3 に示す構成で多段階 GAN モデルを作成した。Stage1 の生成器は表 3.1 と同じ構成, Stage1 と Stage2 の判別器は共に, 表 3.2 と同じ構成, Stage2 の生成器は表 3.3 に示す構成としてモデルを作成した。Stage1 出力画像サイズ  $h \times h$  を  $64 \times 64$ [pixels], Stage2 出力画像サイズ  $2h \times 2h$  を  $128 \times 128$ [pixels] として, 損失関数に交差 Entropy 用いて, 最適化手法は Stage1, Stage2 共に, 判別器で Adam( $\alpha = 1 \times 10^{-5}$ ,  $\beta_1 = 0.1$ ,  $\beta_2 = 0.999$ ), GAN 全体で Adam( $\alpha = 1 \times 10^{-4}$ ,  $\beta_1 = 0.5$ ,  $\beta_2 = 0.999$ ) として学習を行った。

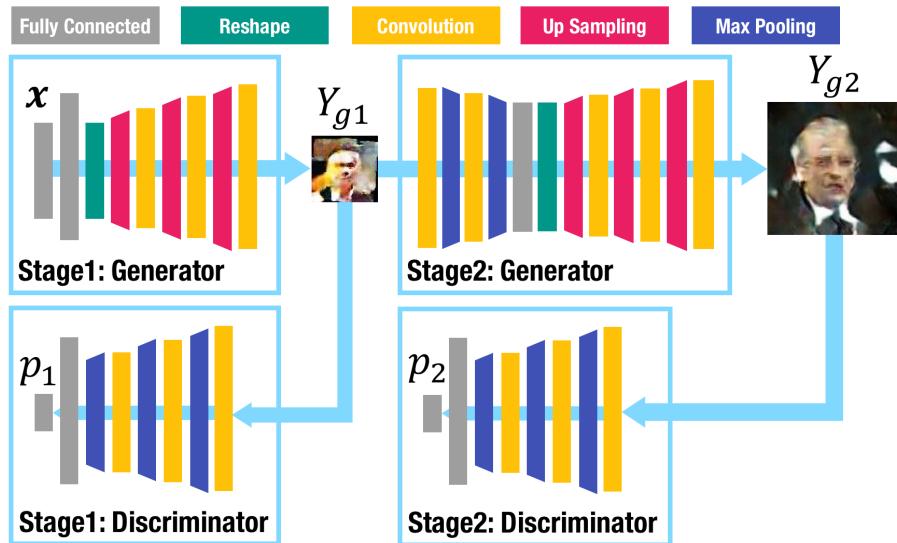


図 3.3 作成した多段階 GAN の構成

表 3.3 Stage2 生成器の各種構成

Layer (type)	Output Shape	Parameters
Input	( $h, h, 64$ )	
Convolution	( $h, h, 64$ )	Kernel: (3, 3)
Leaky ReLU		
Max Pooling	( $h/2, h/2, 64$ )	Size: (2, 2)
Convolution	( $h/2, h/2, 128$ )	Kernel: (3, 3)
Leaky ReLU		
Max Pooling	( $h/4, h/4, 128$ )	Size: (2, 2)
Flatten	$128 \times h/4 \times h/4$	
Dence	256	
Leaky ReLU		
Dence	$256 \times h/4 \times h/4$	
Batch Normalization		
Leaky ReLU		
Reshape	( $h/4, h/4, 256$ )	
Up Sampling	( $h/2, h/2, 256$ )	Size: (2, 2)
Convolution	( $h/2, h/2, 128$ )	Kernel: (3, 3)
Batch Normalization		
Leaky ReLU		
Up Sampling	( $h, h, 128$ )	Size: (2, 2)
Convolution	( $h, h, 64$ )	Kernel: (3, 3)
Batch Normalization		
Leaky ReLU		
Up Sampling	( $2h, 2h, 64$ )	Size: (2, 2)
Convolution	( $2h, 2h, 3$ )	Kernel: (3, 3)
Sigmoid		

## 3.4 SRCNN

図 3.4 に示す構成で SRCNN モデルを作成した。表 3.4 に示すように、入力画像を Nearest Neighbor 法で 2 倍に拡大した後、3 層の補正 CNN に通して出力する。入力画像サイズ  $h \times h$  を  $64 \times 64$ [pixels]、出力画像サイズ  $2h \times 2h$  を  $128 \times 128$ [pixels] として、損失関数に平均 2 乗誤差を用いて、最適化手法は Adam( $\alpha = 1 \times 10^{-3}$ ,  $\beta_1 = 0.9$ ,  $\beta_2 = 0.999$ ) で学習を行った。

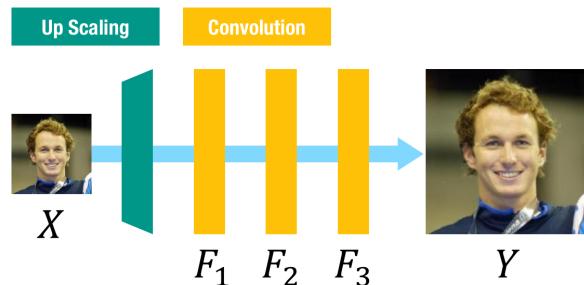


図 3.4 作成した SRCNN の構成

表 3.4 SRCNN モデルの各種構成

Layer (type)	Output Shape	Parameters
Input	( $h, h, 3$ )	
Up Scaling	( $2h, 2h, 3$ )	Nearest Neighbor
Convolution	( $2h, 2h, 256$ )	Kernel: (9, 9)
ReLU		
Convolution	( $2h, 2h, 128$ )	Kernel: (3, 3)
ReLU		
Convolution	( $2h, 2h, 3$ )	Kernel: (5, 5)
Sigmoid		

### 3.5 DCGAN+SRCNN

3.2 節の DCGAN で生成した  $h \times h$ [pixels] の画像を 3.4 節の SRCNN によって超解像することで  $2h \times 2h$ [pixels] の画像を生成した。図 3.5 に構成を示す。

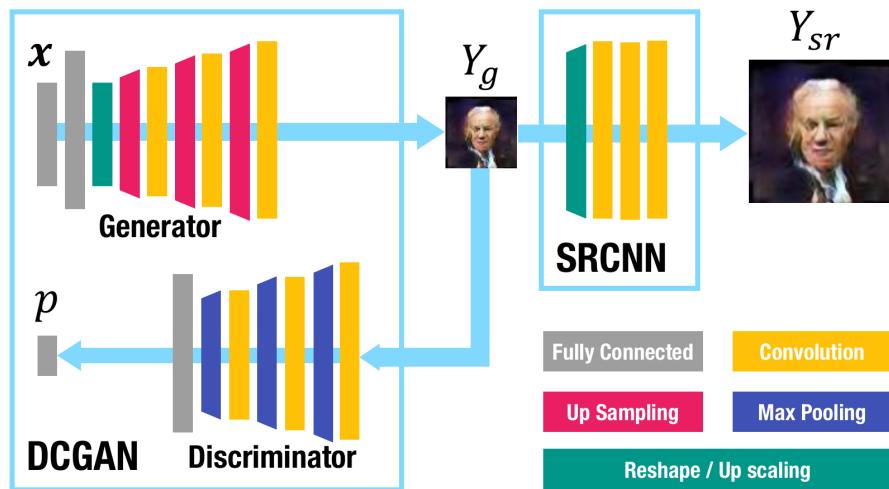


図 3.5 DCGAN+SRCNN の構成

### 3.6 画像評価

GAN を高解像度に対応させる手法として、以下の 2 つの結果について、学習反復数を共通としたときと、学習時間を共通としたときの生成画像を比較した。

- 1) 多段階 GAN を用いて  $2h \times 2h$ [pixels] の画像を作成する方法(以下、手法 A)。
- 2) DCGAN により  $h \times h$ [pixels] の画像を作成し、それを SRCNN により  $2h \times 2h$ [pixels] に拡大する方法(以下、手法 B)。

#### 3.6.1 アンケート調査

画像評価において定量的評価と人間の主観の間には相違があることが知られている。本研究では利用目的に基づき、人間の主観による評価を得るために、アンケート調査を行った。

複数人に対して以下のような調査を行い、各モデルの出力精度を評価した。実際の調査ページの Screenshot を図 3.6 に示す。

- 1) 手法 A, 手法 B の生成画像のうち判別器の出力値が比較的高いものから, 各 24 枚, 全 48 枚の画像を抽出した。
- 2) 抽出した手法 A, 手法 B の画像から各 3 枚ずつ, 全 6 枚の画像を表示し, その中から最も良いと感じた画像を選択させた。
- 3) 上記の操作を 8 回行わせ, 評価された画像の情報を収集した。



図 3.6 調査ページの Screenshot

# 第4章 結果

## 4.1 DCGAN

3.2節のDCGANモデルを反復数6000[epochs]で訓練した。出力画像の一例を図4.1に示す。ここで、反復数とはすべての訓練データに対して勾配更新を行う回数のことである。



図 4.1 DCGAN 出力画像の一例

### 4.1.1 Hyperparameter調整

3.2節のDCGANモデルを以下のように条件を変えて作成し、反復数3000[epochs]で訓練した。

- 1) 生成器、判別器共に中間層の活性化関数をtanhとしたとき、生成器の中間層の活性化関数にReLU、判別器にLeakyReLUとしたとき、生成器、判別器共に中間層の活性化関数をLeakyReLUとしたときの結果を図4.2に示す。
- 2) 畳み込み層のユニット数を半分にしたモデルと、もとのモデルの比較を図4.3に示す。
- 3) 畳み込み層を1層増やし、4層CNNとしたモデルと、もとのモデルの比較を図4.4に示す。
- 4) 全結合層のユニット数を2倍にしたモデルと、もとのモデルの比較を図4.5に示す。
- 5) 判別器学習プロセスで1回の勾配更新の間に生成画像と教師画像を混在させるか否かによる結果を図4.6に示す。

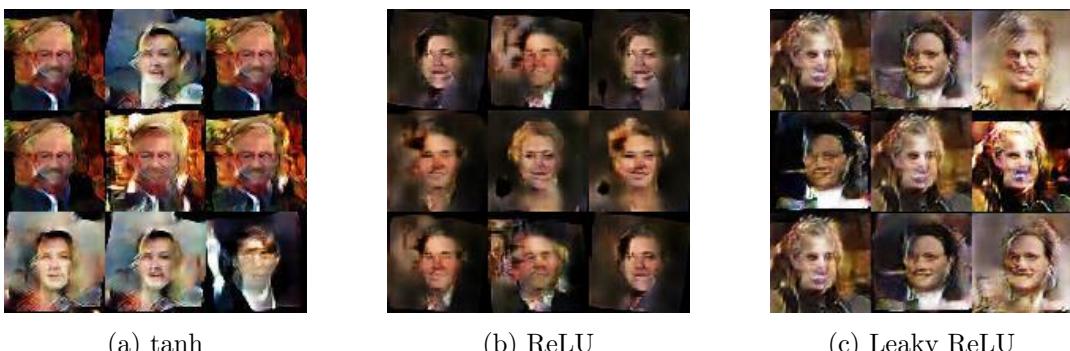


図 4.2 活性化関数による出力の変化



(a) ユニット数半分



(b) もとのモデル

図 4.3 畳み込みユニット数による出力の変化



(a) もとのモデル



(b) 畳み込み層追加

図 4.4 畳み込み層数による出力の変化



(a) もとのモデル



(b) ユニット数 2 倍

図 4.5 全結合層ユニット数による出力の変化



(a) 混在



(b) 分離

図 4.6 教師画像と生成画像の混在・分離による出力の変化

#### 4.1.2 出力画像サイズによる変化

3.2 節の DCGAN モデルを反復数 3000[epochs] で訓練した。出力画像サイズによる変化を図 4.7 に示す。



(a)  $32 \times 32$

(b)  $64 \times 64$

(c)  $128 \times 128$

図 4.7 出力画像サイズによる変化

## 4.2 多段階 GAN

3.3 節の多段階 GAN モデルを反復数 6000[epochs] で訓練した。4000[epochs] 時と 6000[epochs] 時の出力画像の一例を図 4.8 に示す。



(a) 反復数 4000

(b) 反復數 6000

図 4.8 多段階 GAN 出力画像の一例

### 4.3 SRCNN

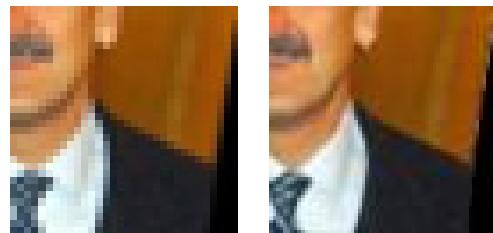
3.4 節の SRCNN モデルを反復数 2000[epochs] で訓練した。3.1 節のデータセットから訓練用画像とは別に取得した画像を超解像したときの結果の一例を図 4.9, 4.10 に示す。



(a) 入力画像

(b) 超解像後画像

図 4.9 SRCNN 出力画像の一例



(a) 入力画像

(b) 超解像後画像

図 4.10 一部を拡大した SRCNN 出力画像

#### 4.4 DCGAN+SRCNN

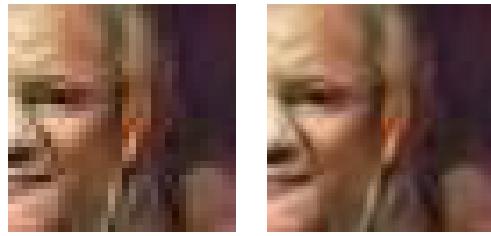
4.1 節の訓練された DCGAN モデルと、4.3 節の訓練された SRCNN モデルを組み合わせて、DCGAN で生成した画像を SRCNN で超解像することにより、拡大を行った。出力結果の一例を図 4.11, 4.12 に示す。



(a) DCGAN 出力

(b) SRCNN 適用後

図 4.11 DCGAN+SRCNN 出力画像の一例



(a) DCGAN 出力      (b) SRCNN 適用後

図 4.12 一部拡大した DCGAN+SRCNN の出力画像

## 4.5 画像評価

評価人数 50[人] で、画像評価調査を行った。このとき母集団の内訳は、男性 42[人]、女性 6[人]、その他 2[人] で、10 代 13[人]、20 代 34[人]、30 代 1[人]、40 代 2[人] だった。

### 4.5.1 反復数共通のとき

手法 A の多段階 GAN と、手法 B の DCGAN の学習反復数を 6000[epochs] 共通としたときの結果を表 4.1 に示す。このとき手法 B の SRCNN の学習反復数は 2000[epochs] とした。

表 4.1 反復数共通のときの結果

方法	選択率 [%]	反復数 [epochs]	学習時間 [hours]
手法 A	50.0	6000	113.8
手法 B	50.0	6000, 2000	$50.9 + 24.3 = 75.2$

### 4.5.2 学習時間共通のとき

手法 A と手法 B の学習時間を約 75[hours] 共通としたときの結果を表 4.2 に示す。

表 4.2 学習時間共通のときの結果

方法	選択率 [%]	反復数 [epochs]	学習時間 [hours]
手法 A	22.7	4000	76.8
手法 B	77.3	6000, 2000	$50.9 + 24.3 = 75.2$

# 第5章 結論

## 5.1 考察

### 5.1.1 Hyperparameter 調整

DCGAN で出力精度を向上させるには、生成器、判別器共に活性化関数に Leaky ReLU を使い、畳み込み層のユニット数を増やし、判別器学習プロセスで生成画像と教師画像を別バッチに分けると良いことがわかった。また、畳み込み層の層数や、全結合層のユニット数を増やしすぎると、最適化に時間がかかり十分に学習が行えないため、同じ反復数では、出力精度が低下することもわかった。

### 5.1.2 解像度による変化

DCGAN では出力画像の解像度が高くなるに伴い、学習時間も跳ね上がる。更に、ある一定の解像度を超えると画質が向上しにくくなる。従って、一段の DCGAN では出力精度の向上に限界があり、最も性能を発揮できるのは  $64 \times 64$ [pixels] 程度であると考えられる。

高解像度の画像を出力したいとき、多段階 GAN を用いると質の高い画像を得られることがわかった。しかし、学習コストが跳ね上がる問題は維持されたままであった。

### 5.1.3 DCGAN+SRCNN

DCGAN で出力した低解像度の画像を SRCNN で超解像することにより、質の高い高解像度の画像を出力できることがわかった。また、同じ反復数では多段階 GAN よりも短い学習時間で訓練が完了することも確認できた。

### 5.1.4 画像評価調査の結果

アンケート調査の結果より得た、人間の感性による評価では、多段階 GAN を用いて高解像度の画像を出力するよりも、DCGAN と SRCNN を組み合わせて出力する方が、優れた結果を得られることがわかった。

## 5.2 今後の課題

現在、GAN を用いて高解像度の画像を生成する様々な方法が発表されている。本研究では、基本的にはモデルである、多段階 GAN と DCGAN を比較したが、その他の手法についても比較する価値があると思われる。

また、時間の都合により、満足な画像評価調査ができなかつたため、多人数に対して、複数の条件での画像評価調査を行いたいと考えている。

# 参考文献

- [1] Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A. and Bengio, Y.: Generative Adversarial Networks (2014).
- [2] 川井颯斗：自動生成モデルを用いた画像生成に関する研究 (2017).
- [3] Radford, A., Metz, L. and Chintala, S.: Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks (2015).
- [4] Dong, C., Loy, C. C., He, K. and Tang, X.: Image Super-Resolution Using Deep Convolutional Networks (2014).
- [5] Wasserman, P.: ニューラル・コンピューティング—理論と実際, 森北出版, 11-38 pp. (1993).
- [6] 馬場則夫, 小島史男, 小澤誠一 : ニューラルネットの基礎と応用, 共立出版, 4-18 pp. (1999).
- [7] 伊庭齊志 : 進化計算と深層学習, オーム社, 102-108 pp. (2016).
- [8] Ioffe, S. and Szegedy, C.: Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift (2015).
- [9] Kingma, D. P. and Ba, J.: Adam: A Method for Stochastic Optimization (2014).
- [10] Zhang, H., Xu, T., Li, H., Zhang, S., Wang, X., Huang, X. and Metaxas, D.: StackGAN: Text to Photo-realistic Image Synthesis with Stacked Generative Adversarial Networks (2016).
- [11] Zheng, T., Deng, W. and Hu, J.: Cross-Age LFW: A Database for Studying Cross-Age Face Recognition in Unconstrained Environments (2017).

# 謝辞

本研究を進める上でご指導と助言を下さった藤本健司准教授に深く感謝いたします。自堕落な私を見守り、時には発破をかけてくださったおかげで、何とか研究成果を出すことができました。もし、自分の能力を過信し、余裕で締め切りに間にあうと思い続けていたと考えると、恐ろしくて言葉も出ません。また、受験対策からプレゼン資料作成術まで、有益な情報を与えて頂き、本当にお世話になりました。

藤本研究室の卒業生の方々。膨大な資料を残してくださりありがとうございました。研究に行き詰まったときや、締め切りが緊急事態になったとき、どれだけ救われたかは計り知れません。私ができるかぎり後輩に資料を残そうと決意したきっかけでもありました。先輩方には足を向けて寝られません。

お忙しい中、アンケート調査にご協力頂いた皆様。本当にありがとうございました。本論文を執筆できたのは皆様のおかげです。

藤本研究室の皆様。有意義な議論から無意義な議論まで、多くの知識や示唆を頂いたことに深く感謝します。居心地のいい研究室で研究を行え、ギークな話題やサブカルチャーに明るくなったのは皆様のおかげです。

GitHub, Inc. 御中。無償で非公開リポジトリを利用させて頂きました。資料管理が簡単になり、作業効率を上げることができました。

家族。学生生活を温かい目で見守ってくれたことに深く感謝します。おかげで研究に没頭することができました。

友人達。SNS で楽しそうな写真を見るたび心が癒やされました。私は誘われていませんが、研究進捗を心配して誘わなかつたのだと信じています。お気遣いありがとうございました。