深層学習による生成モデリング

海老澤優

慶應義塾大学理工学部数理科学科
小林景研究室

2022 年 2 月 1 日

目次

第 1 章

はじめに

GAN という機械学習モデルがあり，データから特徴を学習することで実在しないデータを生成したり，画像のドメイン変換が行えることを知った．それまで，機械学習は分析の道具という認識をしていたが，生成モデルとしての理論や課題を学びたいと感じた．

2 章では，生成モデルとは何かということを確率分布間の距離を用いながらその概要について説明する．3 章では AE（オートエンコーダ）について解説し，4 章では AE が生成モデルになるように改良した VAE（変分オートエンコーダ）について説明する．4〜7 章では GAN のモデルについて解説を行い，8 章では GAN を用いて人間の顔画像の生成を行う．

第 2 章

生成モデルとは

生成モデルとは，データの生成過程を確率分布でモデル化し，新しいデータを生成できるモデルのことである．モデル化の背景には，観測されているデータは何らかの確率分布に基づき，その生成過程をモデル化できるという考えがある．生成モデルは，ランダムノイズなどの確率変数からデータ x を生成する確率 $p(x)$ を出力する．生成モデルが出力する確率分布をモデル分布とよぶ．また，データ x が従う分布をデータ分布とよぶ．

生成モデルは，データ分布とモデル分布の近さを評価する．確率分布の近さを表す指標はたくさんあるが，まずは KL ダイバージェンス（Kullback-Leibler divergence）を紹介する．2 つの確率分布の密度関数を $p(x), q(x)$ としたとき，KL ダイバージェンス D_{KL} は以下の式で定義される．

$$D_{KL}(p(x)||q(x)) = \int p(x) \log \left(\frac{p(x)}{q(x)} \right) dx.$$

KL ダイバージェンスは非負性を持ち，2 つの確率分布が近いほど小さな値を取り，一致するとき最小値 0 をとる．データ分布の密度 $p_d(x)$ とモデル分布の密度 $p_\theta(x)$（θ はモデルのパラメータ）を KL ダイバージェンスに代入すると，期待値を用いて以下のように書き換えられる．

$$\begin{aligned} D_{KL}(p_d(x)||p_\theta(x)) &= \int p_d(x) \log \left(\frac{p_d(x)}{p_\theta(x)} \right) dx \\ &= \int p_d(x) \log(p_d(x)) dx - \int p_\theta(x) \log(p_\theta(x)) dx \\ &= E_{p_d}[\log(p_d(x))] - E_{p_d}[\log(p_\theta(x))]. \end{aligned} \tag{2.1}$$

ただし，$E_{p_d}[\cdot]$ は p_d を密度とした期待値を表す．式 (2.1) の第 1 項はパラメータ θ に依存しない定数となる．よって，第 2 項目を最大化することで KL ダイバージェンスを最小化でき，データ分布に近いモデル分布を決定することができる．しかし，p_d の形が不明なため，期待値を直接計算することはできない．そこで，L 個のデータ $x^1, \ldots, x^L$ が得られたとき，モンテカルロ近似により $\log(p_\theta(x))$ の平均で期待値を近似することで最尤推定が可能となる．つまり，

$$E_{p_d}[\log(p_\theta(x))] \simeq \frac{1}{L} \sum_{l=1}^{L} \log(p_\theta(x^l))$$

とする．4 章で扱う VAE は，この手法でモデル分布のパラメータを計算する．

KL ダイバージェンスは対称性が成り立たない（つまり，$D_{KL}(p(x)||q(x)) \neq D_{KL}(q(x)||p(x))$）ため，距離ではない．また，$p(x) = 0, q(x) \neq 0$ となる領域がある場合に値が不定となってしまう．そこで，これらの問題点を克服できるように KL ダイバージェンスを改良した JS ダイバージェンス（Jensen-Shannon divergence）というものがある．JS ダイバージェンス $D_{JS}(p(x)||q(x))$ は以下で定義される．

$$D_{JS}(p(x)||q(x)) = \frac{1}{2}D_{KL}\left(p(x)||\frac{p(x)+q(x)}{2}\right) + \frac{1}{2}D_{KL}\left(q(x)||\frac{p(x)+q(x)}{2}\right).$$

JS ダイバージェンスは，5 章で扱う GAN の学習で用いられている．

以下，x は画像データのテンソルもしくはそれをベクトルに変換したものを表すこととする．

第 3 章

AE（Autoencoder）

ニューラルネットワークを用いた次元削減手法として知られる AE について解説する．

3.1 AE の概要

AE（オートエンコーダ）は深層ニューラルネットワークにより，画像の圧縮と復元を通して入力した画像と同じような画像を復元するモデルである．ネットワークを学習する際，低次元の隠れ層を挟むことで情報を圧縮することができる．入力層を隠れ層に圧縮する仮定をエンコーダ，隠れ層から出力層に復元する仮定をデコーダという．また，隠れ層の変数のことを潜在変数とよぶ．

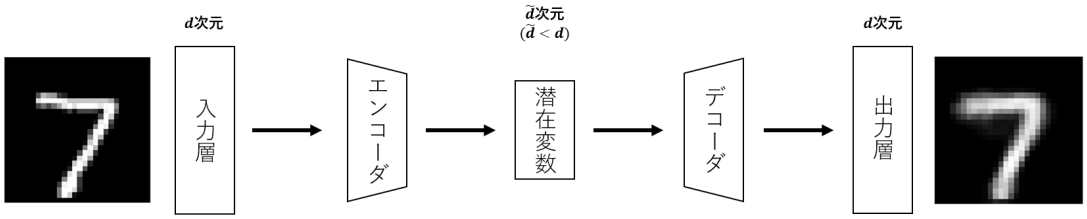

図 3.1: AE の全体像

3.2 AE の損失関数

入力画像 x と出力画像 y の損失関数は再構成誤差 J^{REG} で定義され，これを最小化するようにエンコーダとデコーダの重みを学習する．

$$J^{REC} = -\sum_{i=1}^{d}(x_i \log y_i + (1 - x_i)\log(1 - y_i)).$$

ただし，d は入力層の次元，x_i, y_i はそれぞれ入力層，出力層の第 i 成分である．

3.3　AE は生成モデルになり得ない

AE は画像の圧縮や復元ができるが，生成モデルではない．それは，潜在変数の分布がわかってないので，どんな変数をデコーダに入力すれば意味のある画像を復元できるかが不明だからである．詳しくいうと次の 3 つの理由があげられる．

1. 訓練データの近くのデータから似た画像が生成できる根拠がない．
2. 画像の種類ごとに分布の範囲が異なり，生成される画像が異なる．
3. 潜在空間の範囲が制限されず，疎な空間が存在する．

1 つめの問題点は，訓練データの近くのデータは損失に影響しないからである．2 つめと 3 つめの問題点は，以下の潜在変数のプロットを見ることで確認できる．図 1 は，入力画像を MNIST データとし，隠れ層を 2 次元に設定したときの潜在変数のプロットである．

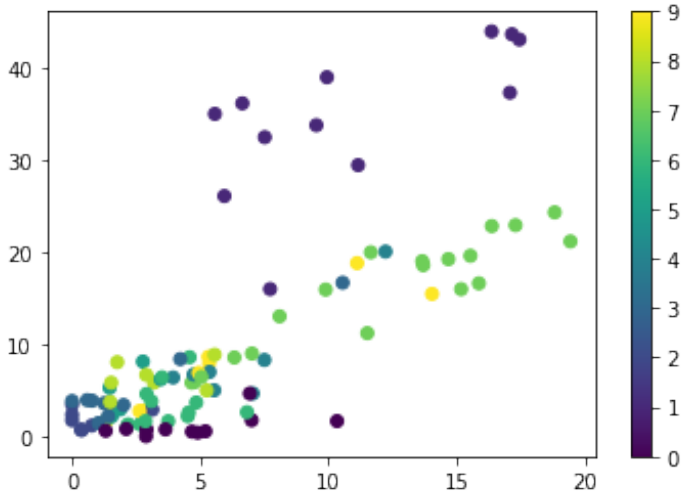

図 3.2: AE における MNIST データの潜在空間

第 4 章

VAE（Variational Auto Encoder）

AE を応用した生成モデルである VAE について解説する．

4.1 VAE の概要

VAE（変分オートエンコーダ）は AE と似た手法のモデルであるが，先ほど列挙した AE が生成モデルになりえない理由を克服させた生成モデルの一種である．VAE は潜在変数を標準正規分布に従う確率変数でモデル化する．このモデル化により，入力画像の特徴量を標準正規分布に押し込むことができ，標準正規分布から得られた乱数をデコーダに入力して画像を生成できるようになる．学習後はエンコーダは不要となり，標準正規分布に従う確率変数から画像を確率的に生成できる．

VAE は AE と同様にエンコーダとデコーダで構成されている．エンコーダにより入力画像を $\tilde{d}$ 次元の平均ベクトル $\boldsymbol{\mu}$ と分散ベクトル $\boldsymbol{\sigma^2}$ に変換する．続いて，平均ベクトルと分散ベクトルの線形和で正規分布に従う潜在変数 z を作成する．具体的には，$\boldsymbol{\varepsilon} \sim N(\mathbf{0}, I)$ とし，$z_i = \mu_i + \sqrt{\sigma_i^2} \cdot \varepsilon_i$ とする．ただし，$z_i, \mu_i, \sigma_i^2, \varepsilon_i$ はそれぞれ $\boldsymbol{z}, \boldsymbol{\mu}, \boldsymbol{\sigma^2}, \boldsymbol{\varepsilon}$ の第 i 成分である．最後に，デコーダにより潜在変数 $\boldsymbol{z}$ を画像に変換する．

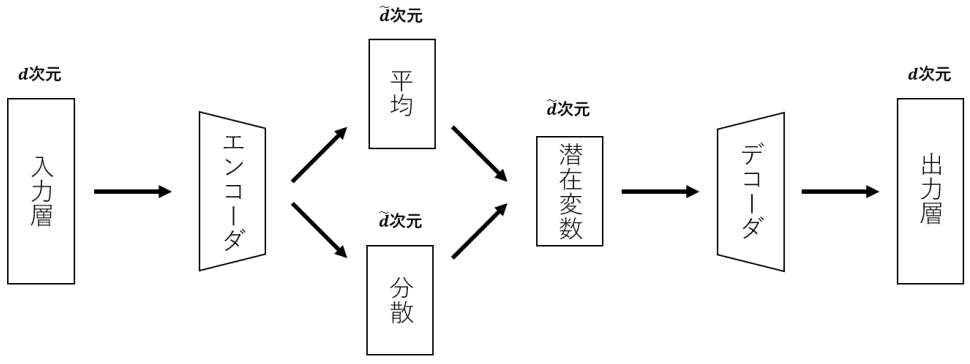

図 4.1: VAE の全体像

4.2 VAE の損失関数

VAE の損失関数を導出する．データ分布 p_d とモデル分布 p_θ の近さを KL ダイバージェンスで測ると，2 章で扱ったようにモデル分布 $p_\theta(x)$ の尤度が最大となるパラメータ θ を計算することに帰着される．$p_\theta(x)$ は以下のように潜在変数 $\boldsymbol{z}$ の事前分布 $p_\theta(\boldsymbol{z})$ を用いて以下のように書きなおせる．

$$p_\theta(x) = \int p_\theta(x|\boldsymbol{z})p_\theta(\boldsymbol{z})d\boldsymbol{z}.$$

ただし，潜在変数 $\boldsymbol{z}$ のサンプリングは困難である．この問題を解決するため，新たに条件付き密度関数 $q_\phi(\boldsymbol{z}|x)$ を導入し，画像 x から潜在変数 $\boldsymbol{z}$ をサンプリングできるようにする．つまり，$q_\phi(\boldsymbol{z}|x)$ はエンコーダ，$p_\theta(x|\boldsymbol{z})$ はデコーダとなる．

対数尤度 $\log p_\theta(x)$ の計算に $q_\phi(\boldsymbol{z}|x)$ を追加し，$\log p_\theta(x)$ の最小値を計算すると以下のようになる．

$$\begin{aligned}
\log p_\theta(x) &= \log \int p_\theta(x|\boldsymbol{z})p_\theta(\boldsymbol{z})d\boldsymbol{z} \\
&= \log \int q_\phi(\boldsymbol{z}|x)\frac{p_\theta(x|\boldsymbol{z})p_\theta(\boldsymbol{z})}{q_\phi(\boldsymbol{z}|x)}d\boldsymbol{z} \\
&\geq \int q_\phi(\boldsymbol{z}|x)\log\left(\frac{p_\theta(x|\boldsymbol{z})p_\theta(\boldsymbol{z})}{q_\phi(\boldsymbol{z}|x)}\right)d\boldsymbol{z}.
\end{aligned}$$

ここで，2 行目から 3 行目の不等式では Jensen の不等式を利用している．この最小値を変分下限 $\mathcal{L}(x;\theta,\phi)$ と定義し，$\log p_\theta(x)$ の最大化の代わりに $\mathcal{L}(x;\theta,\phi)$ の最大化を考える．変分下限は以下のように書きなおせる．

$$\begin{aligned}
\mathcal{L}(x;\theta,\phi) &= \int q_\phi(\boldsymbol{z}|x)\log\left(\frac{p_\theta(x|\boldsymbol{z})p_\theta(\boldsymbol{z})}{q_\phi(\boldsymbol{z}|x)}\right)d\boldsymbol{z} \\
&= \int q_\phi(\boldsymbol{z}|x)\log(p_\theta(x|\boldsymbol{z}))d\boldsymbol{z} - \int q_\phi(\boldsymbol{z}|x)\log\left(\frac{q_\phi(\boldsymbol{z}|x)}{p_\theta(x)}\right)d\boldsymbol{z} \\
&= E_{q_\phi}[\log(p_\theta(x|\boldsymbol{z}))] - D_{KL}(q_\phi(\boldsymbol{z}|x)||p_\theta(\boldsymbol{z})). \qquad (4.1)
\end{aligned}$$

式 (4.1) の第 1 項目を最大化，第 2 項目を最小化すれば，変分下限の最大化が従う．第 1 項目はエンコーダ $q_\phi(\boldsymbol{z}|x)$ でサンプリングしたときのデコーダ $p_\theta(x|\boldsymbol{z})$ の対数尤度の期待値であるが，この期待値は計算することが難しいため，潜在変数の L 個のサンプル $\boldsymbol{z}^1,\ldots,\boldsymbol{z}^L$ でモンテカルロ近似を行うと，

$$E_{q_\phi}[\log(p_\theta(x|\boldsymbol{z})] \simeq \frac{1}{L}\sum_{l=1}^{L}\log(p_\theta(x^l|\boldsymbol{z}^l))$$

となる．さらに，バッチサイズが十分に大きい場合を考え，$L=1$ とする．デコーダ $p_\theta(x|\boldsymbol{z})$ が多変量ベルヌーイ分布に従う，つまり，各ピクセルが 0 から 1 の間の値をとると仮定すれば，

$$\log(p_\theta(x|\boldsymbol{z}) = \sum_{i=1}^{d} x_i \log y_i + (1-x_i)\log(1-y_i)$$

と近似できる．次に, 式 (4.1) の第 2 項目の最小化を考える．エンコーダ $q_\phi(\boldsymbol{z}|x)$ が標準正規分布に従うようにモデル化するので, $q_\phi(\boldsymbol{z}|x)$ は平均ベクトル $\boldsymbol{\mu}$, 分散共分散行列 Σ ($\Sigma = diag(\sigma_1^2, \ldots, \sigma_{\tilde{d}}^2)$) の $\tilde{d}$ 変量正規分布の密度関数，$p_\theta(z)$ は $\tilde{d}$ 変量標準正規分布の密度関数となる．つまり，

$$q_\phi(\boldsymbol{z}|x) = (2\pi)^{-\frac{\tilde{d}}{2}} |\Sigma|^{-\frac{1}{2}} \exp\left(-\frac{1}{2}(\boldsymbol{z}-\boldsymbol{\mu})^\top \Sigma^{-1}(\boldsymbol{z}-\boldsymbol{\mu})\right)$$
$$p_\theta(\boldsymbol{z}) = (2\pi)^{-\frac{\tilde{d}}{2}} \exp\left(-\frac{1}{2}\boldsymbol{z}^\top \boldsymbol{z}\right)$$

よって，KL ダイバージェンスは以下のように計算できる（付録 A）．

$$D_{KL}(q_\phi(\boldsymbol{z}|x)||p_\theta(\boldsymbol{z})) = -\frac{1}{2}\sum_{j=1}^{\tilde{d}}(1 + \log\sigma_j^2 - \mu_j^2 - \sigma_j^2)$$

この式は，$\mu_j = 0, \sigma_j^2 = 1$ $(1 \leq j \leq \tilde{d})$ のとき最小値 0 をとる．

以上より，変分下限は次の式になる．

$$\mathcal{L}(x;\theta,\phi) \simeq \sum_{i=1}^{d}(x_i \log y_i + (1-x_i)\log(1-y_i)) + \frac{1}{2}\sum_{j=1}^{\tilde{d}}(1 + \log\sigma_j^2 - \mu_j^2 - \sigma_j^2)$$

この式の符号を反転させた式を VAE の損失関数とする．つまり, AE の損失関数である再構成誤差に，潜在変数と標準正規分布の差を計上する正則化誤差 $J^{REG} = -\frac{1}{2}\sum_{j=1}^{\tilde{d}}(1 + \log\sigma_j^2 - \mu_j^2 - \sigma_j^2)$ を足し合わせたものが VAE の損失関数であり，これを最小化するようにパラメータを更新する．

4.3 AE との違い

VAE は AE が生成モデルになり得ない 3 つの理由が克服されている．一つ目の「訓練データの近くのデータから似た画像が生成できる根拠がない」という点は，入力画像を一点ではなく，分布にエンコードすることで，同じ画像が毎回少しずれてプロットされるので，それが訓練データの近くのデータから似た画像が生成できる根拠となる．二つ目と三つ目「画像の種類ごとに分布の範囲が異なり，生成される画像が異なる」「潜在空間の範囲が制限されず，疎な空間が存在する」という点は，エンコードされた潜在変数の分布と標準正規分布の差を損失として計上することで克服されている（図 4.2）．

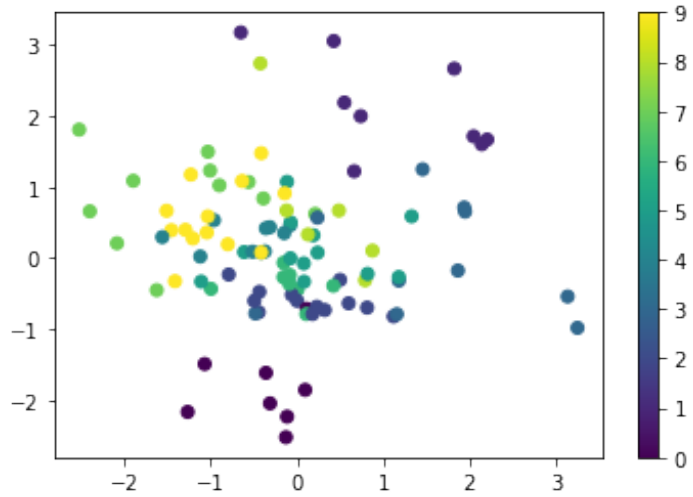

図 4.2: VAE の MNIST データの潜在空間

4.4 VAE の実装

MNIST データを用いて VAE を実装した．潜在変数を 10 次元として学習させ，標準正規分布の乱数をデコーダに入力すると図 4.3 のような生成画像が得られた．

図 4.3: 標準正規分布の乱数から得られた生成画像

どの画像も手書き数字の特徴を持っていることが分かり，生成モデルとして機能していることが分かる．次に，手書き数字画像をエンコーダとデコーダに通し生成画像を作成した．図 4.4 の 1 行目が実際の MNIST データで，2 行目が生成画像である．本物画像と比べると，生成画像は全体的にぼやけていることがわかる．VAE で生成した画像がぼやけてしまうのは，潜在変数が正規分布に従うという明示的な仮定をしたことや，ピクセル単位で最適化を行ったため，実際の画像に存在する複雑性を表現できないためである．

3 行目は潜在変数の値を変化させ，最初の 2 つの「7」と「2」の画像を連続的に変化させたものである．今回学習したモデルでは，「7」と「2」の間に「9」「4」「6」の特徴を持った画像が生成されていることが分かる．

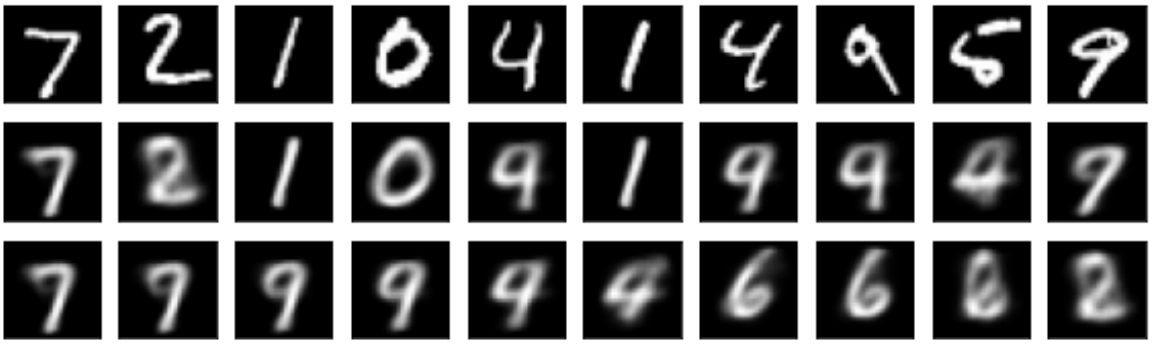

図 4.4: 本物画像（1 行目）と生成画像（2 行目），「7」と「2」を連続的に変化させた生成画像（3 行目）

第 5 章

GAN（Generative Adversarial Networks）

本論文の主たるテーマである GAN の解説を理論を中心に行う．

5.1 GAN の概要

GAN（Generative Adversarial Networks）は日本語では「敵対的生成ネットワーク」という生成モデルの一種である．VAE では潜在変数に確率分布を明示的に仮定し，画像をピクセル単位で最適化したため生成画像がぼやける問題点があったが，GAN は潜在変数の確率分布に明示的な仮定をせず，ニューラルネットワークを用いて確率分布を最適化することで，より複雑な確率分布を扱えるようになる．結果，GAN は VAE に比べて鮮明な画像を生成できるようになる．

5.2 GAN の識別器と生成器

GAN は画像を生成するネットワークである生成器 G と訓練画像と生成画像を判別するネットワークである識別器 D が競い合うように学習が進む．

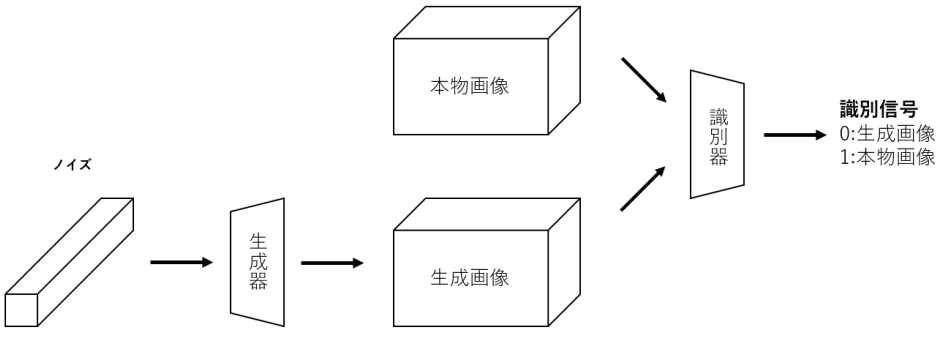

図 5.1: GAN の全体像

識別器 D は入力された画像が本物画像か，生成器 G が作った生成画像かを判別する識別信号を出力する．識別信号は 0 と 1 の間の値を取り，入力画像が本物である確率を表す．識別器 D は正

解ラベルに 1 と 0 の 2 値を用い，本物画像を入力したときに 1 を，生成画像を入力したときに 0 を出力するように学習する．このとき，生成器のパラメータは更新しないことに注意する．

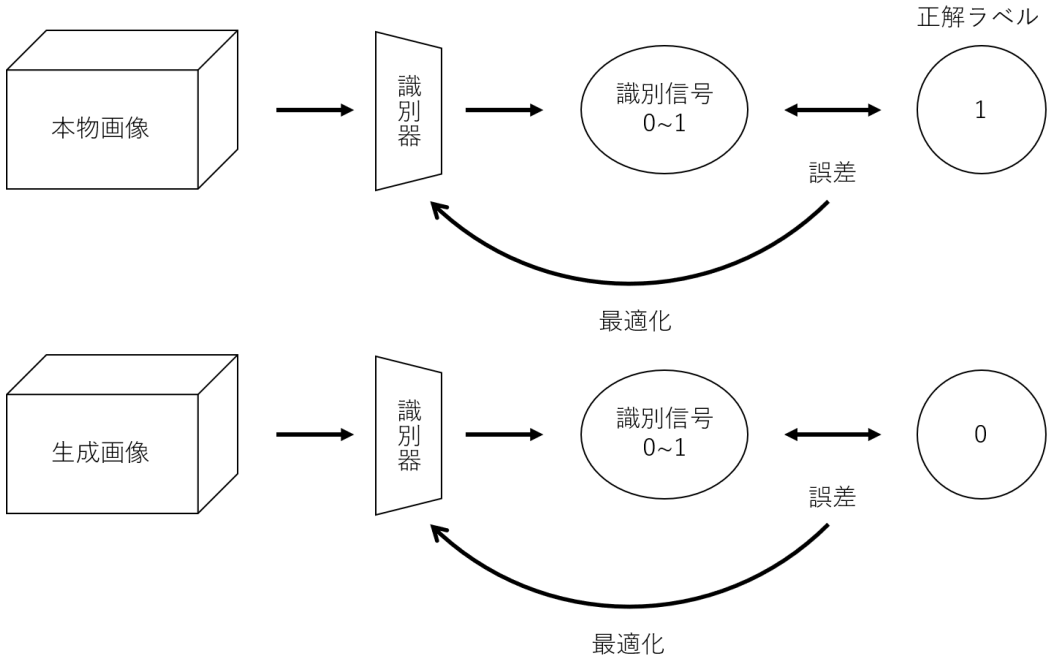

図 5.2: 識別器の学習

生成器 G は VAE のデコーダと似ており，乱数ベクトルから生成画像 x' を出力する生成モデルである．生成画像を識別器に通し，正解ラベル 1 を出力するように学習を進める．つまり，生成器は識別器を騙すように学習を進める．このとき，識別器のパラメータは更新しないことに注意する．

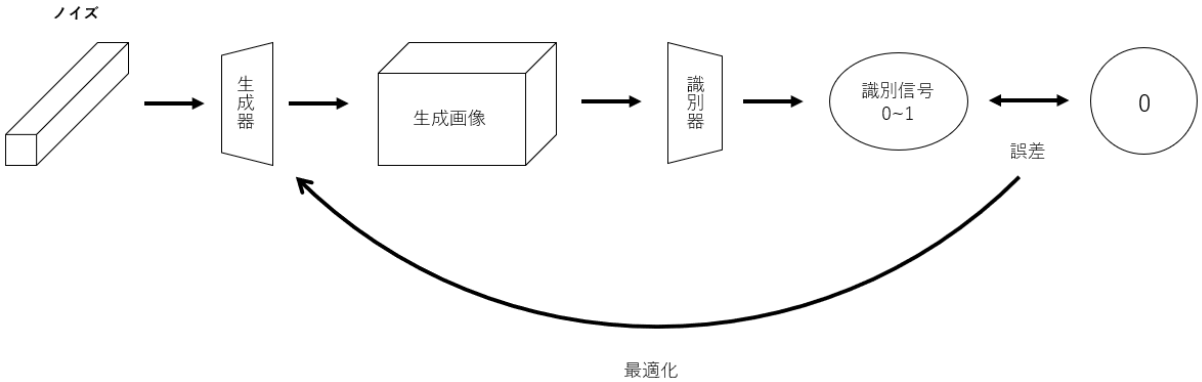

図 5.3: 生成器の学習

このように学習を繰り返すことにより生成器 G と識別器 D は共に賢くなり，生成器 G はノイズから本物のような画像を生成できるようになる．

5.3 GAN の損失関数

GAN の学習では生成器と識別器のパラメータを交互に最適化する．つまり，GAN の学習は以下の 2 つのステップからなる．

1. 生成器のパラメータを固定して，識別器のパラメータを最適化
2. 識別器のパラメータを固定して，生成器のパラメータを最適化

GAN の損失関数を次の式で定める．

$$\min_G \max_D L(D,G) = E_{p_d}[\log D(x)] + E_{p_{\boldsymbol{z}}}[\log(1 - D(G(\boldsymbol{z})))]$$

ただし，それぞれの記号の意味は以下の通りである．

$$x : \text{本物画像}, \quad \boldsymbol{z} : \text{ノイズベクトル}, \quad D(\cdot) : \text{識別器の出力},$$
$$G(\cdot) : \text{生成器の出力} \quad p_d(\cdot) : \text{データ分布の密度}, \quad p_{\boldsymbol{z}}(\cdot) : \text{ノイズの分布の密度}$$

損失関数は，生成器 G に対して $L(D,G)$ を最小化，識別器 D に対しては $L(D,G)$ を最大化する．損失関数 $L(D,G)$ はの期待値を積分を用いて書くと以下のようになる．

$$L(D,G) = \int p_d(x) \log D(x) dx + \int p_{\boldsymbol{z}}(\boldsymbol{z}) \log(1 - D(G(\boldsymbol{z}))) d\boldsymbol{z}$$

ステップ 1 の識別器 D の学習では，生成器 G のパラメータを固定し，生成器 G は画像 x を生成するので $x = G(z)$ と考える．さらに，乱数 $\boldsymbol{z} \sim p_{\boldsymbol{z}}(\boldsymbol{z})$ から生成器 G が生成するデータの分布（以下「生成分布」と呼ぶ）の密度 $p_g(x)$(つまり $p_g(x) = p(x|\boldsymbol{z};\theta)$) と定め，$p_{\boldsymbol{z}}(\boldsymbol{z})d\boldsymbol{z} = p_g(x)dx$ を仮定すると，

$$L(D,G) = \int (p_d(x) \log D(x) + p_g(x) \log(1 - D(x))) dx \tag{5.1}$$

と書き換えられる．

定理 5.3.1. 最適な識別器 D^* は $\dfrac{\partial L(D,G)}{\partial D} = 0$ の解であり，それは以下の形となる．

$$D^*(x) = \frac{p_d(x)}{p_d(x) + p_g(x)}$$

Proof. $L(D,G)$ を最大化するには，式 (5.1) の被積分関数 f_L を最大化すればよい．

$$\frac{\partial f_L}{\partial D} = \frac{p_d(x)}{D(x)} - \frac{p_g(x)}{1 - D(x)}$$
$$\frac{\partial^2 f_L}{\partial D^2} = -\frac{p_d(x)}{D(x)^2} - \frac{p_g(x)}{(1 - D(x))^2} < 0$$

より，$f_L(D,G)$ は D に関して凹関数である．よって，$\dfrac{\partial f_L}{\partial D}=0$ の解が求める D^* である．これを解くと，

$$D^*(x)=\frac{p_d(x)}{p_d(x)+p_g(x)}$$

となる． □

もし生成器が最適な状態に学習されているとき，後で示すが p_d と p_g が一致する．このとき，$D^*(x)=1/2$ となる．これは，最適化された識別器をもってしても，本物画像と生成画像の識別ができないことを表している．

続いて，ステップ 2 の生成器の学習に進む．

定理 5.3.2. 最適化された識別器 D^* において，$L(D^*,G)$ が最小値をとるとき，$p_d=p_g$ が成り立つ．

Proof.

$$\begin{aligned}
L(D^*,G)&=\int(p_d(x)\log D^*(x)+p_g(x)\log(1-D^*(x)))dx\\
&=\int p_d(x)\log\left(\frac{p_d(x)}{p_d(x)+p_g(x)}\right)dx+\int p_g(x)\log\left(\frac{p_d(x)}{p_d(x)+p_g(x)}\right)dx\\
&=-\log 4+\int p_d(x)\log\left(\frac{p_d(x)}{\frac{p_d(x)+p_g(x)}{2}}\right)dx+\int p_g(x)\log\left(\frac{p_d(x)}{\frac{p_d(x)+p_g(x)}{2}}\right)dx\\
&=-\log 4+D_{KL}\left(p_d(x)||\frac{p_d(x)+p_g(x)}{2}\right)+D_{KL}\left(p_g(x)||\frac{p_d(x)+p_g(x)}{2}\right)\\
&=-\log 4+2D_{JS}(p_d(x)||p_g(x))
\end{aligned}$$

よって，最適化された識別器のもとでの GAN の損失関数は p_d と p_g の JS ダイバージェンスに依存する．よって，p_d と p_g が一致するとき，$L(D^*,G)$ は最小化できる． □

$L(D^*,G)$ の最小値を達成する生成器を G^* とすると，識別器と生成器が共に最適化されたとき，$L(D^*,G^*)=-\log 4$ となり，この状態が GAN の損失関数の最適解である．

ここまで，数式を用いて GAN の学習の最適化を述べたが，学習の初期は本物と偽物の判別が簡単なので，生成器の学習の際，$D(G(\boldsymbol{z}))$ は 0 付近での勾配がほとんど 0 になり，学習が停滞する勾配消失問題や，生成器の学習が識別器に対して進みすぎると，識別器を騙しやすい特定の画像のみを生成するようになり多様な画像を生成できなってしまうモード崩壊と呼ばれる課題が存在する．

第 6 章

DCGAN

GAN のネットワークに畳み込み層を用いたモデルである DCGAN について解説する．

6.1 DCGAN の概要

本節で紹介する DCGAN(Deep Convolutional Generative Adversarial Networks) は生成器と識別器に CNN（畳み込みニューラルネットワーク）を利用した GAN の発展版のひとつである．CNN を用いたことで，通常の GAN よりも画像が鮮明になり，安定した学習ができるようになった．現在では一般的な GAN のモデルとして知られており，単に GAN と呼ぶ場合もある．モデルは以下の指針で作成する．

1. 識別器にストライド 2 の畳み込み層，生成器にストライド 2 の転置畳み込み層を利用する．
2. 全結合層は利用しない．
3. 出力層と生成器の入力層を除き，バッチ正規化を利用する．
4. 生成器は活性化関数に ReLU を利用し，出力層は tanh 関数を利用する．
5. 識別器はすべての活性化関数に Leaky ReLU を利用する（勾配消失を防ぐため）．
6. 本物画像を -1 から 1 の間で正規化する．

6.2 DCGAN の損失関数

識別器の損失関数 $L(D)$ は，$L(D, G)$ の符号を反転させた以下の式を用い，これの最小化をするように最適化する．

$$L(D) = -E_{p_d}[\log D(x)] - E_{p_{\boldsymbol{z}}}[\log(1 - D(G(\boldsymbol{z})))]$$

第 1 項は本物画像を入力したときの損失を表しており，$D(x) = 1$ のとき損失は 0 となる．第 2 項は生成画像を入力したときの損失を表しており，$D(G(\boldsymbol{z})) = 0$ のとき損失は 0 になる．つまり，$L(D)$ は識別器 D が本物画像と生成画像を正しく分類したときに損失が 0 になり，誤った分類をすると損失が発生する．識別器 D の損失関数は識別器の 2 クラス分類と考えると，バイナリーク

ロスエントロピーで定式化でき，以下のように定式化できる．これを新しく V_D と表す．

$$V_D = -y \log D(x) - (1-y) \log(1 - D(G(\boldsymbol{z}))).$$

本物画像を識別器に入力するとき，正解ラベルは $y = 1$ となり，V_D の第 1 項が識別信号 $D(x)$ の誤差計算に利用される．生成画像を入力するときは正解ラベルは $y = 0$ となり，第 2 項が識別信号 $D(G(\boldsymbol{z}))$ の誤差計算に利用される．なお，$D(x)$ と $D(G(\boldsymbol{z}))$ は識別器の最終層の sigmoid 関数により 0 と 1 の間の数値をとる．

生成器 G の学習では $L(D, G)$ を最小化するよう最適化するが，第 1 項はノイズに対して定数なので生成器の最小化には関係のない項である．よって，第 2 項のみを生成器の損失関数 $L(G)$ として利用する．つまり，

$$L(G) = E_{p_{\boldsymbol{z}}}[\log(1 - D(G(\boldsymbol{z})))]$$

生成器は，識別器が生成画像と誤判別したとき，識別信号 $D(G(\boldsymbol{z})) = 1$ となり，このとき損失は $-\infty$ となり最小化される．. 逆に騙すことができなかったら，識別信号 $D(G(\boldsymbol{z})) = 0$ となり，損失関数は 0 となり最大化される．

しかし，$L(D)$ の最小化問題は $D(G(\boldsymbol{z})) = 0$ 付近の勾配が小さく学習が進まないモード崩壊という問題があった．そこで，新しく生成器の損失関数として

$$\tilde{L}(G) = -E_{p_{\boldsymbol{z}}}[\log D(G(z))]$$

を用いる．$\tilde{L}(G)$ は $D(G(\boldsymbol{z})) = 0$ 付近で大きな勾配を持つようになるため，生成器の学習が可能となる．また，$D(G(\boldsymbol{z})) = 1$ のとき，損失は 0 となり最小化される．

生成器の損失関数も識別器と同様にバイナリークロスエントロピーで定式化できる．ただし，識別器の損失関数 V_D とは符号が逆となり，

$$V_G = -V_D = y \log D(x) + (1-y) \log(1 - D(G(\boldsymbol{z})))$$

となる．ここで，正解ラベルの $y = 0$ の生成画像のみを使用するので，第 2 項のみを残して

$$V_G = (1-y) \log(1 - D(G(\boldsymbol{z})))$$

とする．さらに，$L(G)$ と同じ操作をして

$$V_G = -\log D(G(\boldsymbol{z}))$$

と書きなおし，これを生成器の損失関数として用いる．このとき，正解ラベルには $y = 1$ を用いる．つまり，識別信号 $D(G(\boldsymbol{z})) = 1$ が出力され，生成器が識別器をだますことができたとき，損失関数 V_G は 0 となり最小化される．

6.2.1 DCGAN の実装 1

VAE のときと同様，MNSIT データを DCGAN で実装した．イテレーション数は 12000，バッチサイズは 50 とした．生成した画像は本物のような鮮明な画像が生成されていることが分かる．

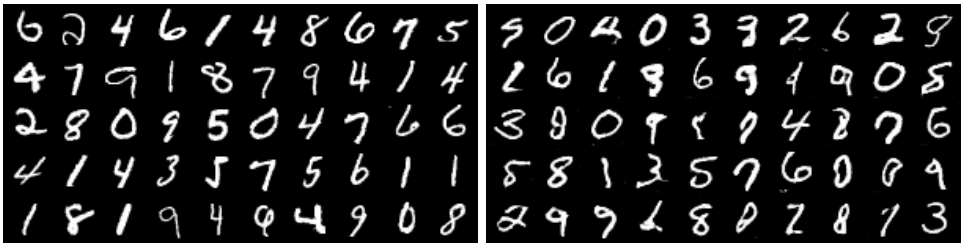

図 6.1: MNIST の本物画像 (左) と DCGAN で生成した画像 (右)

乱数から数字画像が生成できるようにはなったが，生成したい数字を指定することができない．次節で紹介する CGAN ではそれが可能となる．

第 7 章

CGAN

GAN の生成器がラベルに対応した画像が生成できるよう改良した CGAN について解説する．

7.1 CGAN の概要

前節の DCGAN で MNIST データを生成した際，数字画像の生成はランダムで，どの数字が出力されるかは生成画像を見ないとわからなかった．本節で紹介する CGAN（Conditional GAN）はその欠点を克服した GAN であり，クラス指定が可能である．CGAN のネットワークは GAN と同じく生成器と識別器で構成される．GAN との相違点はクラス c を指定する条件ベクトル $\boldsymbol{c}$ が存在することである．本物画像を x，条件ベクトルを $\boldsymbol{c}$，ランダムノイズを $\boldsymbol{z}$ としたとき，損失関数は次の式になる．

$$\min_G \max_D L(D,G) = E_{p_d}[\log D(x|\boldsymbol{c})] + E_{p_{\boldsymbol{z}}}[\log(1 - D(G(\boldsymbol{z}|\boldsymbol{c})))]$$

$D(x|\boldsymbol{c})$ は条件 c を満たす本物画像 x の識別器の出力，$D(G(\boldsymbol{z}|\boldsymbol{c}))$ は条件 c を満たすランダムノイズ $\boldsymbol{z}$ の生成器の出力を表す．損失関数の最適化の手法はこれまでと同じである．

7.2 CGAN のネットワーク

CGAN はノイズ $\boldsymbol{z}$ にクラスを指定する One-Hot ベクトルを結合させたベクトル（これが条件ベクトル $\boldsymbol{c}$）を生成器に入力する．同様に，識別器に入力する画像にもクラスを指定する情報が必要になるが，画像は 3 次元配列なので One-Hot ベクトルをそのまま結合することはできない．そこで，One-Hot ベクトルを画像サイズに拡大した条件画像を入力画像のチャネル方向に結合させる．たとえば，MNIST データでラベル 1 を表すとき，生成器への入力でノイズベクトルに結合する One-Hot ベクトルは $(1, 0, \ldots, 0)^\top \in \mathbb{R}^{10}$ になる．識別器への入力で入力画像に結合する条件画像は $28 \times 28 \times 10$ のテンソルで，すべての成分が 1 である 28×28 の行列の後ろにすべての成分が 0 である 28×28 の行列が 9 つ並んだものとなる．

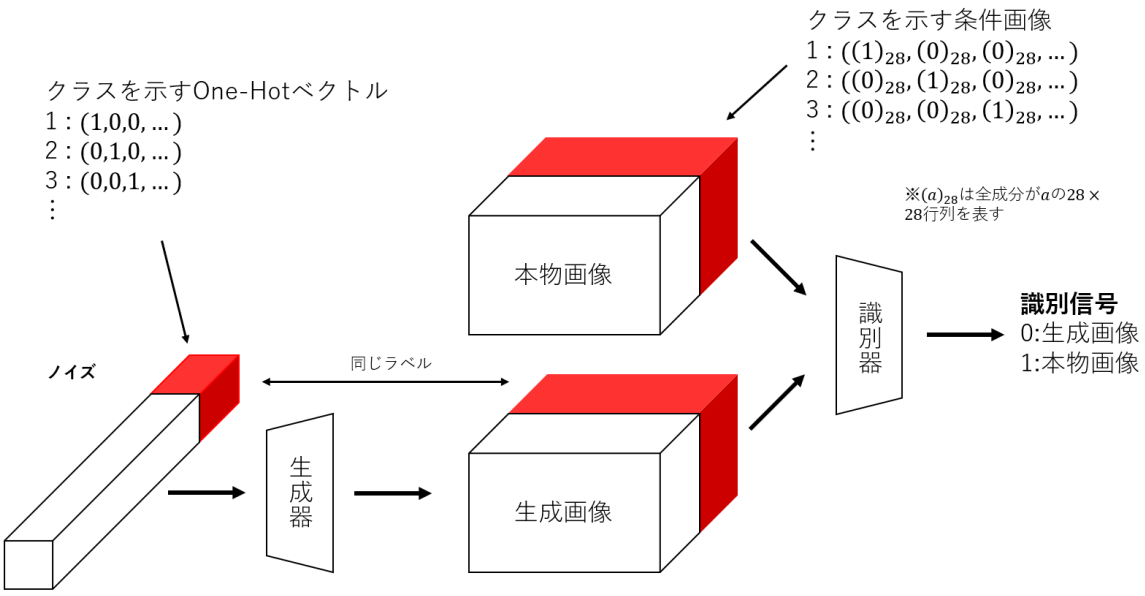

図 7.1: CGAN のイメージ

7.2.1 CGAN の実装

CGAN を MNIST データで実装した．ネットワークは DCGAN と同じものを用いている．以下の図 19 は，ラベル 0 から 9 までの値を使い，画像を 5 枚ずつ生成したものである．ラベルに応じた画像が正しく生成されていることが分かる．

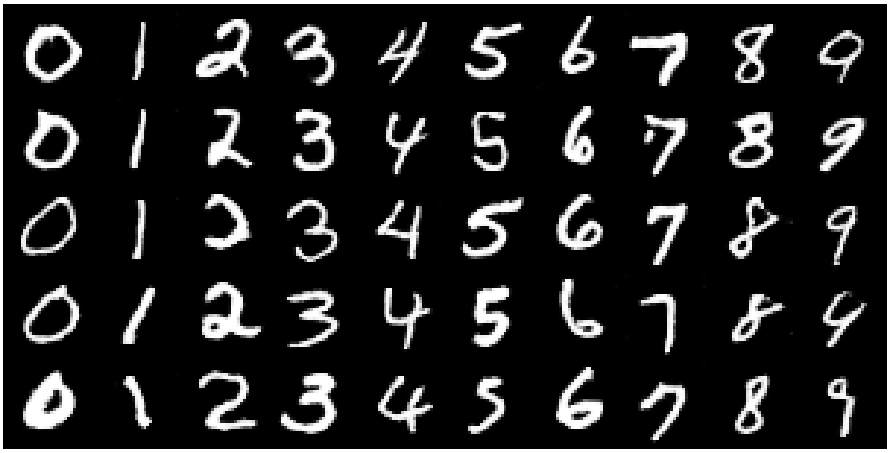

図 7.2: CGAN で生成した画像

第 8 章

顔画像の生成

celebA データセットという有名人の顔画像を集めたデータセットを DCGAN で学習し，顔画像の生成を行った．イテレーション数を 50000，バッチサイズを 100 として学習した．結果として，図 8.1 のような生成画像が得られた．顔が崩れてしまった画像もいくつか見受けられるが，ほとんど本物の顔画像のような鮮明な画像が得られていることが分かる．

図 8.1: celebA の本物画像（左）と DCGAN で生成した画像（右）

生成画像が本物画像の分布の中にどのように広がっているのかを調べた．本物画像の中から 1000 枚を抜き出し多様体学習のモデルの 1 つである UMAP を学習させ 2 次元にプロットした．その後，その UMAP のモデルに生成画像 64 枚を入れて散布図に描き加えたところ図 8.2 のような散布図が得られた．赤い点が本物画像のプロット，青い点が生成画像のプロットである．生成画像は本物画像の分布の中に散らばっているものの，3 分の 1 ほどのデータは本物画像の散布図の境界付近に位置している．

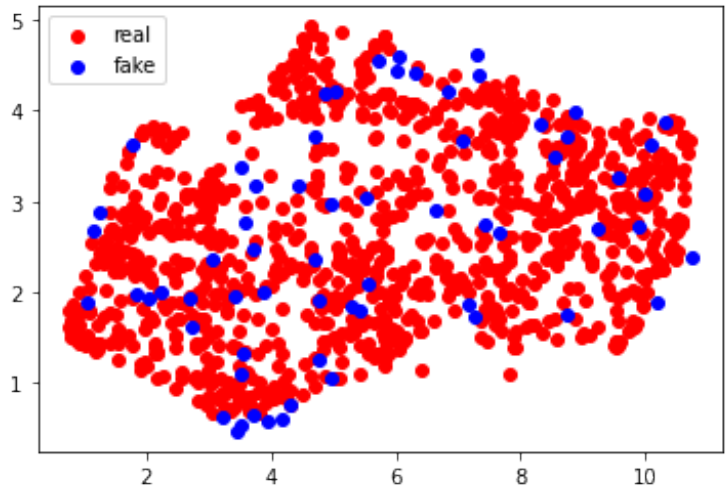

図 8.2: UMAP による可視化

本物画像の散布図の境界付近に射影された生成画像と内側に射影された画像を見比べると，境界付近に射影された生成画像はくずれた画像が多いことがわかる．

図 8.3: 本物画像の散布図の境界に射影された生成画像

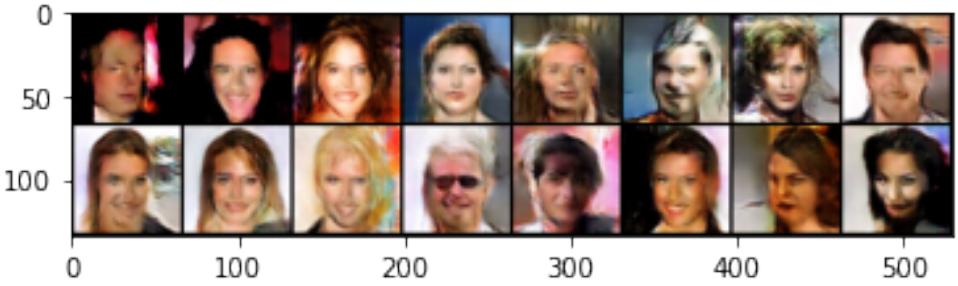

図 8.4: 本物画像の散布図の境界に射影された生成画像

よって，UMAP を用いたことで画像の生成がうまくいっているかどうかを可視化できていることになる．散布図を局所的に見たとき，生成画像は近くのデータの特徴を併せ持っているように見えるので，本物画像の散布図の境界や外側にいる生成画像は，類似度が高い画像が少ない．したがって崩れた画像が多く境界に集まるのだろう．

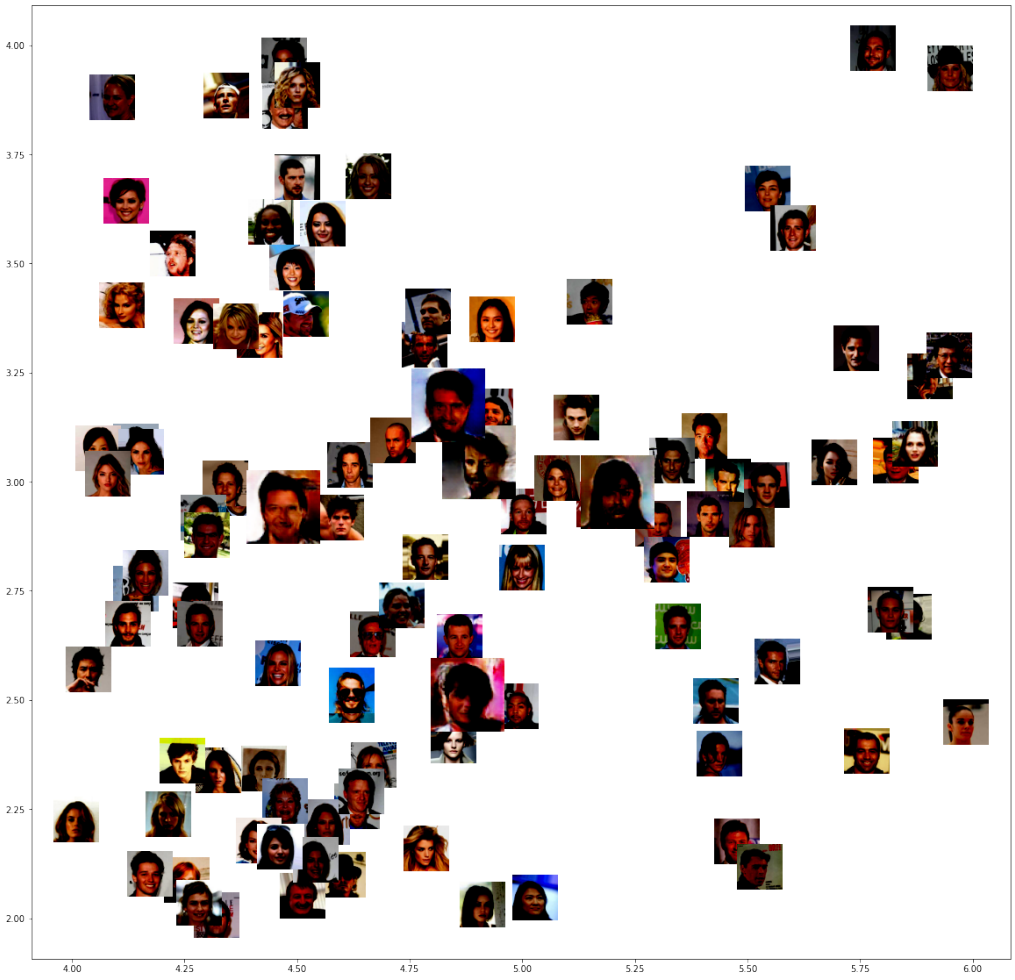

図 8.5: 散布図の一部

しかし，今回用いた UMAP のモデルは画像の類似度をユークリッド距離で計算しているため，背景色の影響を大きく受けていることが考えられる．画像に特化した距離を用いた次元削減法を使う必要があると感じた．

付録 A

多変量正規分布間の KL ダイバージェンス

2 つの d 次元正規分布 $N_d(\boldsymbol{\mu}_1, \Sigma_1), N_d(\boldsymbol{\mu}_2, \Sigma_2)$ の KL ダイバージェンスを導出する．$N_d(\boldsymbol{\mu}_1, \Sigma_1), N_d(\boldsymbol{\mu}_2, \Sigma_2)$ の密度をそれぞれ f_1, f_2 とすると，

$$
\begin{aligned}
D_{KL}(f_1||f_2) &= \int f_1(\boldsymbol{x}) \log \frac{f_1(\boldsymbol{x})}{f_2(\boldsymbol{x})} dx \\
&= E_{f_1}[\log f_1(\boldsymbol{x}) - \log f_2(\boldsymbol{x})] \\
&= E_{f_1}\left[\frac{1}{2}\left(\log \frac{|\Sigma_2|}{|\Sigma_1|} - (\boldsymbol{x}-\boldsymbol{\mu}_1)^\top \Sigma_1^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_1) + (\boldsymbol{x}-\boldsymbol{\mu}_2)^\top \Sigma_2^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_2)\right)\right] \\
&= \frac{1}{2}\left(\log \frac{|\Sigma_2|}{|\Sigma_1|} - E_{f_1}\left[(\boldsymbol{x}-\boldsymbol{\mu}_1)^\top \Sigma_1^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_1)\right] + E_{f_1}\left[(\boldsymbol{x}-\boldsymbol{\mu}_2)^\top \Sigma_2^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_2)\right]\right)
\end{aligned}
$$

$(\boldsymbol{x}-\boldsymbol{\mu}_1)^\top \Sigma_1^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_1) \in \mathbb{R}$ であることと行列のトレースの可換性を用いると，第 2 項目は

$$
\begin{aligned}
E_{f_1}\left[(\boldsymbol{x}-\boldsymbol{\mu}_1)^\top \Sigma_1^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_1)\right] &= E_{f_1}\left[\mathrm{tr}\left(\Sigma_1^{-1}(\boldsymbol{x}_1-\boldsymbol{\mu}_1)(\boldsymbol{x}_1-\boldsymbol{\mu}_1)^\top\right)\right] \\
&= \mathrm{tr}\left(E_{f_1}\left[\Sigma_1^{-1}(\boldsymbol{x}_1-\boldsymbol{\mu}_1)(\boldsymbol{x}_1-\boldsymbol{\mu}_1)^\top\right]\right) \\
&= \mathrm{tr}\left(\Sigma_1^{-1} E_{f_1}\left[(\boldsymbol{x}_1-\boldsymbol{\mu}_1)(\boldsymbol{x}_1-\boldsymbol{\mu}_1)^\top\right]\right) \\
&= \mathrm{tr}\left(\Sigma_1^{-1}\Sigma_1\right) \\
&= \mathrm{tr}(I_d) \\
&= d
\end{aligned}
$$

となる．第 3 項目は，同様な計算をすることで

$$
\begin{aligned}
E_{f_1}\left[(\boldsymbol{x}-\boldsymbol{\mu}_2)^\top \Sigma_2^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_2)\right] &= E_{f_1}\left[(\boldsymbol{x}-\boldsymbol{\mu}_1+\boldsymbol{\mu}_1-\boldsymbol{\mu}_2)^\top \Sigma_2^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_1+\boldsymbol{\mu}_1-\boldsymbol{\mu}_2)\right] \\
&= E_{f_1}\left[(\boldsymbol{x}-\boldsymbol{\mu}_1)^\top \Sigma_2^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_1)\right] + 2\underbrace{E_{f_1}\left[(\boldsymbol{x}-\boldsymbol{\mu}_1)^\top \Sigma_2^{-1}(\boldsymbol{\mu}_1-\boldsymbol{\mu}_2))\right]}_{=0} \\
&\quad + E_{f_1}\left[(\boldsymbol{\mu}_1-\boldsymbol{\mu}_2)^\top \Sigma_2^{-1}(\boldsymbol{\mu}_1-\boldsymbol{\mu}_2)\right] \\
&= \mathrm{tr}\left(\Sigma_2^{-1}\Sigma_1\right) + (\boldsymbol{\mu}_1-\boldsymbol{\mu}_2)^\top \Sigma_2^{-1}(\boldsymbol{\mu}_1-\boldsymbol{\mu}_2)
\end{aligned}
$$

となる．以上より，

$$D_{KL}(f_1(\boldsymbol{x})||f_2(\boldsymbol{x})) = \frac{1}{2}\left(\log\frac{|\Sigma_2|}{|\Sigma_1|} - d + \mathrm{tr}\left(\Sigma_2^{-1}\Sigma_1\right) + (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)^\top \Sigma_2^{-1}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)\right)$$

ここで，第 3 章の $D_{KL}(q_\phi(\boldsymbol{z}|x)||p_\theta(\boldsymbol{z}))$ を計算すると，

$$\begin{aligned} D_{KL}(q_\phi(\boldsymbol{z}|x)||p_\theta(\boldsymbol{z})) &= \frac{1}{2}\left(\log\prod_{i=1}^{d}\sigma_i^2 - d + \sum_{i=1}^{d}\sigma_i^2 + \boldsymbol{\mu}^\top\boldsymbol{\mu}\right) \\ &= -\frac{1}{2}\sum_{j=1}^{\tilde{d}}(1 + \log\sigma_j^2 - \mu_j^2 - \sigma_j^2) \end{aligned}$$

謝辞

本研究を行うにあたり，今室丁寧にご指導をいただいた慶應義塾大学理工学部数理科学科小林景准教授に深く感謝いたします.

参考文献

[1] 毛利拓也, 大郷友海, 嶋田宏樹, 大政孝充, むぎたろう, 寅蔵, もちまる (2021). GAN ディープラーニングハンドブック. 秀和システム

[2] Diederik P Kingma, Max Welling(2014). Auto-Encoding Variational Bayes https://arxiv.org/abs/1312.6114

[3] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, Yoshua Bengio(2014) . Generative Adversarial Networks. https://arxiv.org/abs/1406.2661

[4] Alec Radford, Luke Metz, Soumith Chintala(2016)Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks. https://arxiv.org/abs/1511.06434

[5] Mehdi Mirza, Simon Osindero(2014). Conditional generative adversarial nets. https://arxiv.org/abs/1411.1784