GANを用いた新たなアバター生成

藤間裕大　小野佳祐　土井遼太郎　美谷佳寛
千葉大学院　融合理工学府　基幹工学専攻　修士１年

1　研究背景、目的

今年度のLSIデザインコンテストにおける設計課題は「Generative Adversarial Networks（GAN）」である。GANは、画像生成、超解像、スタイル変換、異常検知などに用いられる生成ネットワークである。我々は、画像生成の応用として、様々な髪型、髪色、目、服のアバターの画像を学習されることで、データセットには含まれない新たなアバターを生成するGANの回路設計および実装を行った。本システムの特徴を以下に挙げる。

1. ランダムノイズ（100次元）の入力zに対し、新たなアバター画像を生成するGeneratorネットワーク（CNN）のFPGA実装
2. ランダムノイズzに対し、100次元のランダムベクトル$v1$、$v2$を用いて、zはR画像の学習、$z+v_1$はG画像の学習、$z+v_2$はB画像の学習を行うことで、一つのGeneratorネットワークでカラー画像の出力を実現
3. ハードの工夫点　並列化など？？
4. ハードの工夫点

2章では、本システムを社会実装するにあたって、データセット拡張における有用性を示す。3章では、準備として生成モデルのなかでのGANの位置づけ、およびGANの学習アルゴリズムについて説明する。4章では、ソフトウェアにおける本システムの特徴を示す。5章では、ハード。

2　データセット拡張と本システムの関係

近年、深層学習は医療分野における画像診断や、製造分野における外観検査、さらに自動運転に代表される制御分野の画像処理など、幅広い分野で利用されている。一方で、深層学習モデルの性能は学習データの量および多様性に強く依存することが知られている。特に画像識別タスクにおいては、学習データが不足すると過学習や汎化性能の低下が生じやすく、実運用時の性能劣化を招く要因となる。

この問題に対する一般的な対策として、回転・平行移動・反転などの幾何学的変換によるデータセット拡張が広く用いられている。しかし、これらの手法は既存画像の変形に留まるため、髪型や服装、形状、テクスチャといった新規の意味的バリエーションを十分に増やすことは困難である。

教師あり学習の限界

教師あり学習ではピクセル誤差（L1/L2）を最小化することで画像を学習する。そのため、学習外の入力に対しては、複数の正解候補を同時に満たそうとし、生成結果が画像の平均的な外観となりやすい。特に出力分布が多峰性の場合、この性質は輪郭のぼやけや細部の欠落として顕在化する。したがって、再構成損失中心の生成手法は、データ多様性の拡張という観点では限界がある。

GANの優位性

GANは、特定の入力に対してピクセル単位での一致を目指すのではなく、データ集

合全体の確率分布を再現するように学習する点に特徴がある。このため、既存データの単なる変形に留まらず、学習データの分布に整合した新規サンプルの生成が可能である。結果として、GANは意味的に多様な画像を生成でき、データセット拡張手法として高い有効性を有する。

本システムの応用例（これでいいのか？）

本システムは、任意のランダム入力に対して、学習データ分布に近い新たなカラー画像を生成できる点に特徴がある。これにより、従来の幾何学的拡張では得られなかった意味的バリエーションを含むデータセット拡張が可能となり、学習モデルの汎化性能向上が期待できる。さらに、本システムはFPGAを用いた高速な画像生成を可能としているため、リアルタイムにデータを取得しながら学習を行うような状況においても有効である。特に、取得可能なデータがスパースな状況において、生成画像によるデータセット拡張をリアルタイムに行うことで、学習データ不足を補完できる可能性がある。

3　準備

本章では、3.1において生成モデルにおけるGANの位置づけを示し、3.2においてGANの学習アルゴリズムについて説明する。

3.1　生成モデル

生成モデルは未知の真の分布$p_{data}(x)$をモデル分布p_θで近似することを目標とする。これは、モデルから画像xが生成される確率である周辺尤度$p_\theta(x) = \int p_\theta(z)p_\theta(x|z)dz$を最大化することに等しいが、この$p_\theta(x)$を直接的に計算することは難しい。この問題に対処したアルゴリズムとしてVAEとGANがある。

3.1.1　VAE

画像データxの特徴を潜在変数として次元の圧縮を行い、その潜在変数に基づいて画像を生成する仕組みをオートエンコーダ―(AE)という。AEでは潜在変数と出力画像は一対一対応だが、潜在変数を確率分布とすることで未知データの出力を可能とした技術としてVAE(Variational Auto Encoder)がある。

VAEでは$p_\theta(x)$を直接的に計算することは難しいという問題に対して、潜在変数zを介して、推論モデル$q_\phi(z|x)$および生成モデル$p_\theta(x|z)$を導入することで対処している。

VAEの推論モデル$q_\phi(z|x)$は、入力データxに対して潜在変数の分布を近似するものであり、ニューラルネットワークを用いて平均μと分散σ^2を出力することで，zが従うガウス分布$\mathcal{N}(\mu, \sigma^2)$を定義する。生成モデル$p_\theta(x|z)$では、潜在変数$z$を事前分布$p(z)$からサンプリングし、それに基づいて新しい画像を生成することが可能となる。

VAEでは$p(x|z)$をガウス分布としてモデル化するため、再構成誤差は二乗誤差に一致する。しかしながら、二乗誤差は多峰的な分布に対して平均化を促す性質がある。その結果、高周波成分を正確に再現する鋭い画像よりも、平滑化された曖昧な画像の方が誤差が小さくなりやすく、VAEでは生成画像がぼやけやすいというアルゴリズム上の課題が存在する。

3.1.2　GAN

VAEと異なり，GANは明示的な尤度関数$p_\theta(x)$や潜在変数の事後分布$q(z|x)$を定義せず、識別モデルを介して生成分布とデータ分布との差異を評価することで、分布全体を暗黙的に近似する生成モデルである。

具体的には、Generatorとよばれる生成モデル$G(z)$とDiscriminatorと呼ばれる識別モデル$D(x)$を用いて、これらを敵対的に学習させることで画像のデータ分布を学習する。

生成モデル$G(z)$は潜在変数zを画像空間に写像するパラメータを学習し、識別モデル$D(x)$は入力画像xが正解データか$G(z)$かを判別する関数として機能する。VAEとは異なり、p_{data}、p_θのJSダイバージェンスを最小化するように学習が進むため、高周波成分も再現したシャープな画像が生成されやすい。

3.2 GAN (Generative Adversarial Networks) の学習アルゴリズム

GANの目標は画像データxに対する生成モデルの分布p_gを学習することである。そのために、まずノイズ変数zに対する事前分布$p_z(z)$を定義し、これをデータ空間へ移す写像として$G(z;\theta_g)$を学習する。θ_gはNNのパラメータである。これにより、単純な事前分布$z \sim p_z(z)$を関数$G(z)$に入力することで、新たなデータ分布$x = p_g$を生成できる。

次に、単一のスカラー値を出力するパーセプトロン$D(x;\theta_d)$を定義する。$D(x)$は、入力xが生成分布p_gからではなく、実データ分布から得られたものである確率を表す。

GANでは、式(1)に示すminimax問題を解くことにより、$G(z)$はデータ分布を学習し、Dは正解画像のデータ分布とp_gの判別方法を学習する。

$$\min_G \max_D V(D,G) = \mathbb{E}_{x\sim p_{\text{data}}(x)}[\log D(x)] + \mathbb{E}_{z\sim p_z(z)}[\log(1-D(G(z)))] \quad (1)$$

ここで、先にDの最適化を完全に進めてしまうと、$D(G(z)) \approx 0$となってしまい、Gの損失関数$L_G = \mathbb{E}_{z\sim p_z(z)}[\log(1-D(G(z)))]$の勾配は$\partial\mathcal{L}_G/\partial\theta_g \approx 0$となる。その結果、$G$の有効な更新ができず、生成器$G$が$p_z(z)$から$p_g$へと変換する過程を学習できなくなってしまう。そのため、DとGを交互に更新することにより、Dの過度な最適化を避け、Gの学習を有効に進めることができる。

3.2.1、3.2.2にてDiscriminatorおよびGeneratorの最適解について、3.2.3にてGAN全体の学習の流れについて詳細を説明する。

3.2.1 Discrminator 最適解

期待値Eは確率変数xに対して関数$f(x)$の平均をとることで求まるから、離散の確率密度関数$P(x)$に対して$E[f(x)] = \Sigma f(x)P(x)$、連続の確率密度関数$p(x)$場合は$E[f(x)] = \int f(x)p(x)dx$と記述できる。

識別器 D では、式 (1) の最大化を目指すため、式 (1) を変形すると式 (2) のようになる。

$$V(D,G) = \mathbb{E}_{x\sim p_{\text{data}}(x)}[\log D(x)] + \mathbb{E}_{z\sim p_z(z)}[\log(1-D(G(z)))] = \int dx\,[p_{\text{data}}(x)\log D(x) + p_g(x)\log(1-D(x))] \quad (2)$$

ここで、関数$y = a\log y + b\log(1-y)$は$y = a/(a+b)$で最大値をとることから、式(2)は式(3)で最大値をとる。

$$D_G^*(x) = \frac{p_{\text{data}}(x)}{p_{\text{data}}(x) + p_g(x)} \quad (3)$$

よって、識別器の最適解は$D_G^*(x)$と分かる。GANの学習が理想的に進行した場合、生成分布は実データ分布に一致し$p_{\text{data}} = p_g$となる。このとき、識別器の最適解は$D_G^*(x) = 1/2$である。

3.2.2 Generator の最適解

KL ダイバージェンス

確率分布の類似性を測る指標としてKLダイバージェンスがあり、真の確率分布$p(x)$を$q(x|\theta)$で表現することは、式(4)に示すKLダイバージェンスを最小化する問題に帰着することができる。

$$D_{KL}(p||q) = \int p(x)\log\frac{p(x)}{q(x)}dx \quad (4)$$

JS ダイバージェンス

KLダイバージェンスは非対称性を持ち、$D_{KL}(P||Q) \neq D_{KL}(Q||P)$であるため距離として扱うことはできない。そこで、対称性を持たせるための指標としてJSダイバージェンスがあり、式(5)で示す。

$$D_{JS}(p||q) = \frac{D_{KL}(p||m) + D_{KL}(q||m)}{2} \quad (5)$$

生成器の学習では、式 (2) の $V(D,G)$ の最小化を考える。式 (2) の $V(D,G)$ に対して、式 (3) の最適演算子 D^* および式 (4)、式 (5) を適用すると式 (6) になる。

$$\begin{aligned} V(G) &= \int p_{\text{data}}(x) \log \frac{p_{\text{data}}}{p_{\text{data}} + p_g} dx \\ &\quad + \int p_g(x) \log \left(1 - \frac{p_{\text{data}}}{p_{\text{data}} + p_g} dx\right) \\ &= D_{KL}(p_{\text{data}} || (p_{\text{data}} + p_g)) + \\ &\quad D_{KL}(p_{\text{data}} || (p_{\text{data}} + p_g)) - \log 4 \\ &= 2D_{JS}(p_{\text{data}} || p_g) - \log 4 \end{aligned} \tag{6}$$

式(6)から、生成器の学習はJSダイバージェンスの最小化問題に帰着できることが分かる。JSダイバージェンスは非負性を持つため、$D_{JS} \geq 0$である。また、等式が成り立つのは確率分布が一致するときであるから、$p_g = p_{\text{data}}$のときであり、このときの$V(G)$の最小値は$-\log 4$である。

以上の内容から、識別器を最適解$D^*_G(x) = 1/2$の近傍で保った状態で、生成器に対してJSダイバージェンスの最小化を目指すことで、唯一の最適解$p_g = p_{\text{data}}$に収束することが分かる。

3.2.3 GAN の学習の流れ

式 (1) をもとに、Discriminator および Generator の損失関数を L_D、L_G とすると、式 (7)、式 (8) のようになる。

$$L_D = \frac{1}{m} \sum_{i=1}^{m} \Big[\log D(x) + \log(1 - D(G(z))) \Big] \tag{7}$$

$$L_G = \frac{1}{m} \sum_{i=1}^{m} \log(1 - D(G(z))) \tag{8}$$

BCELoss(Binary Cross Entropy Loss)

交差エントロピー損失 (BCELoss) は式 (9) で定義される。

$$BCE(x, y) = -(y \log x + (1 - y) log(1 - x)) \tag{9}$$

Discriminator の学習

Discriminator の目標は p_{data} と p_g を判別することであるから、学習時のラベルとして Fake=0,True=1 とした場合、$D(G(z)) = 0, D(x) = 1$ となる。

式 (9) を参考にすると、式 (7) の損失は次式で表せる。

$$\begin{aligned} &BCE(D(x \sim p_{\text{data}}), 1) = -\log(D(x)) \\ &BCE(D(G(z)), 0) = -\log(1 - D(G(z))) \\ &L_D = |BCE(D(x), 1) + BCE(D(G(z)), 0)| \end{aligned} \tag{10}$$

Generator の学習

Generator の目標は $p_{\text{data}} = p_g$ として、Discriminator をだます分布を生成することであるから $D(G(z)) = 1$ を目指す。式 (9) を参考にすると、式 (8) の損失は次式で表せる。

$$\begin{aligned} &BCE(G(z), 1) = -\log(D(x)) \\ &L_G = |BCE(G(z), 1)| \end{aligned} \tag{11}$$

学習アルゴリズム

GAN における学習アルゴリズムを Algorithm1 に示す。

Algorithm 1 Training of GAN

Input: correct Image $p_{\text{data}}(x)$, noise prior $z \sim \mathcal{N}(0, I)$, minibatch size m
Input:learning rates η_D, η_G
Initialize parameters θ_d, θ_g
for iteration $= 1, 2, \ldots$ **do**
  Sample minibatch $\{x^{(i)}\}_{i=1}^m \sim p_{\text{data}}(x)$
  Sample minibatch $\{z^{(i)}\}_{i=1}^m \sim p_z(z)$
  $\tilde{x}^{(i)} \leftarrow G(z^{(i)}; \theta_g)$
  $g_d \leftarrow \nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^m \Big[\log D(x^{(i)}; \theta_d) + \log(1 - D(\tilde{x}^{(i)}; \theta_d)) \Big]$
  $\theta_d \leftarrow \theta_d + \eta_D \, g_d$
  Sample minibatch $\{z^{(i)}\}_{i=1}^m \sim p_z(z)$
  $\tilde{x}^{(i)} \leftarrow G(z^{(i)}; \theta_g)$
  $g_g \leftarrow \nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^m (1 - \log D(\tilde{x}^{(i)}; \theta_d))$
  $\theta_g \leftarrow \theta_g - \eta_G \, g_g$
end for

4　本システムの構成

入力は 100 次元のランダムノイズ z で 32 × 32 の画像を出力するような Generator を学習する。ここで、値が固定の 100 次元のランダムノイズ $v1$、$v2$ を用いることで、z は R 画像の学習、$z + v_1$ は G 画像の学習、$z + v_2$ は B 画像の学習を行う。

Discriminator では、32 × 32 の画像を見てそれが偽物（G が生成）か本物（正解画像）かの確率を 0〜1 で出力する（本物:1、偽物:0）。本システムでは、カラー画像の精度向上のため、2 種類の Discriminator を用意した。1 つ目の Discrininator では、Generator の出力するグレースケール画像（R,G,B）に対し、グレースケールの正解画像と比較を行うことでアバターの形としての正しさを評価する。2 つ目の Discriminator では、z、$z + v1$、$z + v2$ の入力で得た R 画像、G 画像、B 画像を用いて作成した偽物のカラー画像と正解カラー画像との比較で正誤の確率を出力する。このように、Discriminator の役割をアバターの形と色に分担を行うことで精度の向上を図った。

4.1　FPGA 実装時のイメージ

FPGA 実装では Generator ネットワークのみを実装し，100 次元の入力に対して 32 × 32 の画像生成を行うことを目指す。カラー画像を生成するには 3ch（RGB） の出力が必要であるが，メモリ制約のため FPGA 上で $32 \times 32 \times 3ch$ の画像を同時に出力することは困難である。

そこで本システムでは，固定値の 100 次元ベクトル v_1, v_2 を導入し，入力 z に対して z，$z + v_1$，$z + v_2$ の 3 種の入力を作成する。具体的には，z を R 画像の学習，$z + v_1$ を G 画像の学習，$z + v_2$ を B 画像の学習に対応付ける (図 1)。

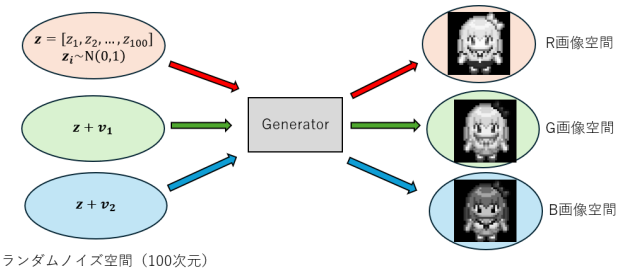


図 1　固定ベクトルを用いた RGB 学習

学習したパラメータおよび固定ベクトル v_1, v_2 を FPGA 実装することで、任意の 100 次元入力 z 入力すると、FPGA 内部で z，$z + v_1$，$z + v_2$ の 3 種のベクトルに対する演算を行うことができ、R・G・B に対応した画像を生成できる (図 2)。

R 用・G 用・B 用に別々のネットワークを学習する場合、FPGA へパラメータを都度転送する必要がある。一方、本方式では入力を z、z+v1、z+v2 のように工夫することで一つのネットワークでカラー画像を出力できることが強みである。

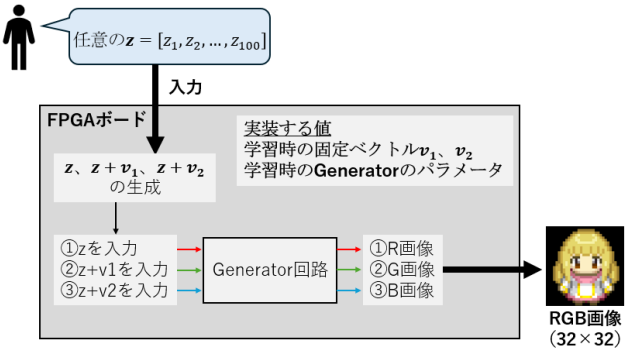


図 2　FPGA 実装時のカラー化演算イメージ

4.2　データセット

キャラメル (CharaMEL0.9.0) というフリーソフトを用いて、32 × 32 のアバターを生成した。本ソフトでは、アバターの目、髪型、髪色、服装、アクセサリーなどの要素を任意にカスタマイズすることが可能である。本実験では、図 3 に示すような多様な外観を持つアバターを 275 種類生成し、これらを学習用データセットとして用いた。

図 3　入力データの一部

4.3　ネットワーク構成

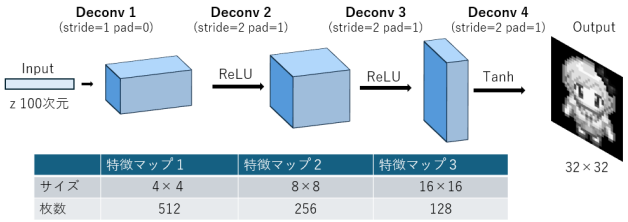
本システムでは、表 1 に示す 3 つのネットワークを用いて学習を行う。各ネットワークの構造については、4.2.1〜4.2.3 節で詳細に説明する。

表 1　各ネットワークの入出力構成

ネットワーク	入力	出力
Generator	100 次元ランダムノイズ	32 × 32 グレー画像（R, G, B）
$\text{Discriminator}_{\text{grey}}$	32 × 32 グレー画像（R, G, B）	本物・偽物の判別確率 (0〜1)
$\text{Discriminator}_{\text{color}}$	32 × 32 × 3 カラー画像（RGB）	本物・偽物の判別確率 (0〜1)

4.3.1　Generator

Generator の構成は、DCGAN[?] のネットワークを参考とした。Generator のネットワーク構造を図 4 に示す。本ネットワークは、入力として 100 次元の潜在変数 z を受け取り、逆畳み込み層 (ConvTranspose2d) を用いて段階的に空間解像度を拡大する構造を有する。具体的には、1 × 1 の潜在特徴マップを 4 × 4、8 × 8、16 × 16 と段階的にアップサンプリングし、最終的に 32 × 32 の画像を生成する。各逆畳み込み層の後には ReLU 関数を適用し、非線形性を導入することで表現能力を高めている。最終層では Tanh 関数を用い、出力値を [-1,1] の範囲に正規化する。

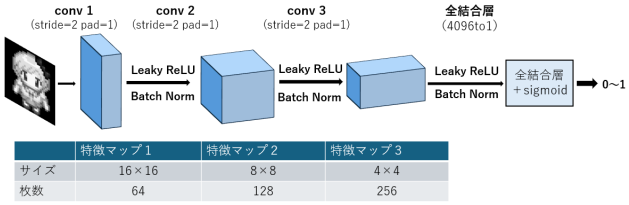


	特徴マップ 1	特徴マップ 2	特徴マップ 3
サイズ	4×4	8×8	16×16
枚数	512	256	128

図 4　Generator のネットワーク構造

4.3.2　Discriminator(grey)

Discriminator(grey) は，32×32（1ch）のグレースケール画像を入力とし、畳み込み層により空間解像度を段階的に縮小しながら特徴量を抽出する。最終的に、全結合層の出力を sigmoid 関数に通すことで 0〜1 のスコア（本物らしさ）を出力する。本ネットワークの構成を図 5 に示す。ここで、Generator とは異なり Leaky ReLU 関数および Batch 正規化を用いているが、これは DCGAN における学習安定性および性能向上を目的としたものであり、参考文献 [?] に基づいている。



	特徴マップ 1	特徴マップ 2	特徴マップ 3
サイズ	16×16	8×8	4×4
枚数	64	128	256

図 5　Discrminator のネットワーク構造

4.3.3　Discriminator(color)

本ネットワークは、Discriminator(grey) の入力をカラー画像 (3ch) に変更することで構成される。Discriminator(grey) では、アバターの形状に注目して判別を行うのに対し、Discriminator(color) ではアバターの色の相関関係に注目して判別を行うことで、Discriminator の判別性能を向上することを目的として構成した。

4.4　訓練パラメータ

学習に用いたパラメータを表 2 に示す。

表 2　学習パラメータ設定

項目	設定値
学習回数	10000
学習率 (生成器)	1.0×10^{-4}
学習率 (識別器)	1.0×10^{-6}
バッチサイズ	64
Adam β_1	0.5
Adam β_2	0.999
入力ノイズ次元	100

4.5　学習結果（ソフトウェア）

学習済み Generator に新たなランダムノイズを入力したときの出力結果を図 6 に示す。図 6 より、Generator はアバターの特徴を学習しており、髪型や服装の異なる多様なアバターを生成できていることが分かる。図 6 中の 2 番および 5 番は正解画像に近い生成結果となっている。一方で、1 番および 4 番の服装に見られる黄色いラインは正解データセットには存在しない。また、3 番の水色のシャツにネクタイを着用したような画像も正解データセット内には見られない。これらの例から、Generator は単なる学習画像の再現にとどまらず、アバターの

特徴を保持しつつ新規性を含む画像を生成できていると考えられる。なお、6 番の画像では服装がぼやけたように見えるが、これは Discriminator の識別性能が十分でなく、6 番のようなぼやけを含む画像も正解として判別されてしまったと考える。

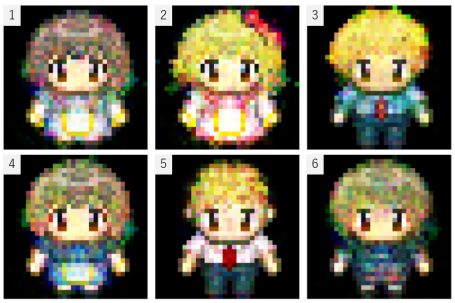


図 6　6 種のランダム入力に対する出力結果

4.5.1　ランダム空間の学習の確認

100 次元ランダム入力 z は、平均 0、分散 1 の独立な正規分布 $\mathcal{N}(0,1)$ に従う乱数として生成される。新たにサンプリングされる z_1, z_2 も同様の正規分布から生成されている。つまり、Generator はこの連続な潜在空間 z から画像空間への連続写像 $G(z)$ を学習することで、多様なアバター画像を生成していると解釈できる。

これを確かめるために、ランダムな 2 つの入力ベクトル z_1, z_2 に対し、その間を線形補間した 3 点の入力ベクトルから生成した画像を図 7 に示す。線形補間により得られるベクトル $z(t)$ は次式で定義される $(N = 3)$。

$$z(t_k) = (1 - t_k)\, z_1 + t_k\, z_2, \tag{12}$$

$$t_k = \frac{k}{N+1}, \quad k = 1, 2, \ldots, N. \tag{13}$$

図 7 の結果から、GAN では入力 z に対して正解画像を学習しているのではなく、z という連続空間に対してアバターらしい画像を学習していることが分かる。また、隣接する画像を比較すると形状が似ているものが多く、画像を確率分布として学習していることが分かる。

画素単位の一致を目的とした教師あり学習では、任意の 2 入力における補間点に対応する出力は、それぞれの教師画像の画素値を平均化したような結果になりやすく、視覚的にぼやけた画像が生成される傾向がある。一方、GAN は画像を確率分布としてモデル化し、潜在空間 z から画像空間への写像

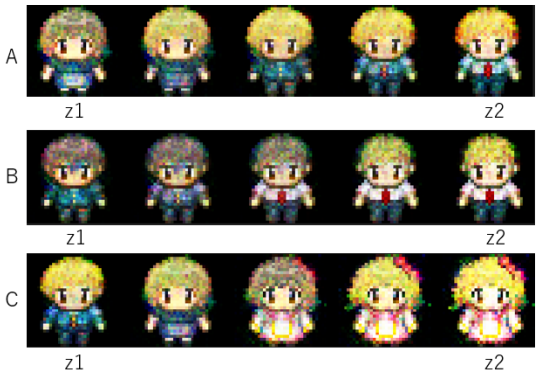


図 7　3 種のランダム入力セットに対する補間結果

$G(z)$ を学習するため、任意の 2 つの潜在変数間の補間点においても、意味的に一貫性を保った新しい画像を生成することが可能である。この性質により、GAN は単なる画素補間では得られない多様な画像を生成でき、データセット拡張の観点において優位性を有するといえる。

5　ハード

6　まとめ

7　補足資料

7.1　固定ベクトル $v1$、$v2$ の頑健性について

高次元の z は球殻上

7.2　ニューラルネットワーク

ニューラルネットワークは複数の層から構成され，それぞれの層にはニューロンと呼ばれる計算ユニットが存在する。各ニューロンは他のニューロンと結合されており，その結合には重みが付けられている。この重みがニューラルネットワークの学習の鍵となり，入力データに対する適切な出力を生成するために最適化される。層の構造は入力層，中間層（隠れ層），および出力層に分かれており，隠れ層の数が大きいほどより複雑なデータのパターンを学習する。ニューラルネットワークの構造を図 8 に示す。

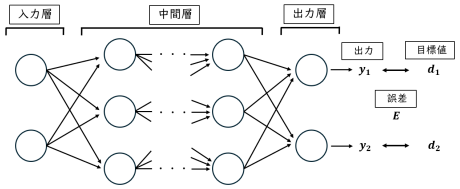


図 8　ニューラルネットワークの構造

(1) ニューロン

ニューロンの概略図を図 9 に示す。ここで，z_j^m は第 m 層における j 番目のニューロンの出力値，w_{ij}^m は第 $m-1$ 層 i 番目のニューロンから第 m 層 j 番目のニューロンへの重み，u_j^m は第 m 層の j 番目のニューロンの入力値，b_j^m はバイアス，f は活性化関数である。

各ニューロンは，前層の出力値 z^{m-1} に重み w^m をかけた重み付き線形和を計算し，その総和にバイアス b_j^m を加えた値を入力値 u_j^m とし式 (14) で定義する。ニューロンはこの入力値 u_j^m を活性化関数 f に通すことで出力値 $z_j^m = f(u_j^m)$ を得る。

$$u_j^m = \sum_i w_{ij}^m z_i^{m-1} + b_j^m \tag{14}$$

ここで，活性化関数 $f(u_j^m)$ は，入力された値に非線形変換を行い，出力を生成する。代表的な活性化関数に，式 (15) で表される ReLU 関数がある。ReLU 関数の微分は，入力値が正のとき 1，負のとき 0 となるため，勾配消失問題を緩和する効果がある。

$$f(u_j^m) = \begin{cases} u_j^m & (u_j^m > 0) \\ 0 & (u_j^m \leq 0) \end{cases} \tag{15}$$

(2) 損失関数

損失関数は，ニューラルネットワークの出力値と目標値の差分を表す関数である。ニューラルネットワークは，目標値に対する損失関数 E の値を最小にするためにユニット間の重みとバイアスを調整する。損失関数の代表的なものに二乗誤差があり，ニューラルネットワークの出力 y_i と，目標値 d_i を用いて式 (16) で定義する。

$$E = \frac{1}{2} \sum_i (y_i - d_i)^2 \tag{16}$$

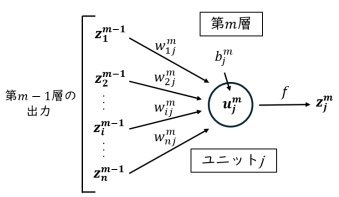


図 9　ニューロンの概略図

(3) 最急降下法

最急降下法は，損失関数を最小化するために，損失関数の勾配を用いて重み w を更新する手法である。最急降下法は，パラメータ w を式 (17) で更新する。

$$w \leftarrow w - \eta \frac{\partial E}{\partial w} \tag{17}$$

(4) 誤差逆伝播法

誤差逆伝播法は，出力層から入力層に向かって，各層の重みを更新する手法である。誤差逆伝播法は，出力層の誤差を計算し，その誤差を入力層に向かって逆伝播させることで，各層の重みを更新する。誤差逆伝播法の手順を以下に示す。

出力層

L 層のニューラルネットワークにおける出力層の重みの更新量を考える。損失関数 E の w_{ij}^L に関する偏微分を連鎖律を用いて式 (18) に示す。

$$\frac{\partial E}{\partial w_{ij}^L} = \frac{\partial E}{\partial u_j^L} \cdot \frac{\partial u_j^L}{\partial w_{ij}^L} = \delta_j^L z_i^{L-1} \tag{18}$$

ここで，u_j^L は式 (19) で表されるから，u_j^L を w_{ij}^L で偏微分すると z_i^{L-1} となる。

$$\begin{aligned} u_j^L &= \sum_i w_{ij}^L z_i^{L-1} + b_j^L \\ &= w_{1j}^L z_1^{L-1} + w_{2j}^L z_2^{L-1} + \ldots + w_{ij}^L z_i^{L-1} \\ &\quad + \ldots + w_{nj}^L z_n^{L-1} + b_j^L \end{aligned} \tag{19}$$

次に，δ_j^L を連鎖律を用いて式 (20) に示す。

$$\delta_j^L = \frac{\partial E}{\partial u_j^L} = \frac{\partial E}{\partial z_j^L} \cdot \frac{\partial z_j^L}{\partial u_j^L} \tag{20}$$

ここで，損失関数 E が式 (16) で表せることと，z_j^L が出力値 y_j であることから，損失関数 E の z_j^L に関する偏微分は式 (21) で表される。

$$\frac{\partial E}{\partial z_j^L} = \frac{\partial}{\partial y_j} \left(\frac{1}{2} \sum_i (y_i - d_i)^2 \right) = y_j - d_j \tag{21}$$

また，活性化関数 f を ReLU 関数とすると，$z_j^L = f(u_j^L) = u_j^L$ であるから，z_j^L に関する u_j^L の偏微分は 1 となる。したがって，式 (18) は式 (22) に変形できる。

$$\frac{\partial E}{\partial w_{ij}^L} = (y_j - d_j) z_i^{L-1} \tag{22}$$

式 (22) において，y_j は出力値，d_j は目標値，z_i^{L-1} は前層の出力値であるから，第 L 層において計算可能な値である。

中間層

第 l 層の重みの更新量を考える。損失関数 E の w_{ij}^l に関する偏微分を連鎖律を用いて式 (23) に示す。

$$\frac{\partial E}{\partial w_{ij}^l} = \frac{\partial E}{\partial u_j^l} \cdot \frac{\partial u_j^l}{\partial w_{ij}^l} = \delta_j^l z_i^{l-1} \tag{23}$$

中間層における誤差分 δ_j^l を考えるにあたって，第 l 層と第 $l+1$ 層におけるニューロン間の入出力関係を図 10 に示す。

損失関数 E の定義は式 (16) であるから，第 l 層の重み更新には第 $l+1$ 層の出力値が必要となる。第 $l+1$ 層の入力値 u_i^{l+1} は第 l 層の出力値 u_j^l の関数でもあるから，損失関数 E は $E(u_1^{l+1}(u_j^l), u_2^{l+1}(u_j^l), \ldots, u_k^{l+1}(u_j^l))$ の関数としてみることができる。したがって，損失関数 E の u_j^l に関する偏微分は多変数関数の連鎖律を用いて式 (24) となる。

$$\begin{aligned}\delta_j^l &= \frac{\partial E}{\partial u_j^l} = \sum_k \frac{\partial E}{\partial u_k^{l+1}} \cdot \frac{\partial u_k^{l+1}}{\partial u_j^l} \\ &= \sum_k \delta_k^{l+1} \cdot \left[\frac{\partial u_k^{l+1}}{\partial z_j^l} \cdot \frac{\partial z_j^l}{\partial u_j^l} \right] = \sum_k \delta_k^{l+1} \cdot w_{kj}^{l+1}\end{aligned} \tag{24}$$

ここで，最後の式変形において $\frac{\partial u_k^{l+1}}{\partial z_j^l} = w_{kj}^{l+1}$，$\frac{\partial z_j^l}{\partial u_j^l} = 1$ を用いた。δ_k^{l+1} および w_{kj}^{l+1} の値は第 $l+1$ 層の値であるため，出力層側から更新を行うことで既知の値となる。つまり，第 l 層目の δ_j^l の値は第 $l+1$ 層目の $\delta_k^{l+1}(k = 1, 2, \ldots)$ から求まる。このように，誤差逆伝播法では出力層から入力層に向かって誤差を逆伝播させることで，各層の重みの更新を行う。

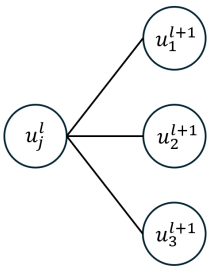


図 10　第 l 層と第 $l+1$ 層におけるニューロン間の入出力関係

7.3　CNN

7.4　逆畳み込み演算