GAN によるグレースケール画像のカラー化

研究者:阿多　健史郎　　指導教員:濱川　恭央

あらまし　グレースケール画像とカラー画像では，それらの持つ情報量に差がある．特にカラー画像は，一目見ればその場の景観や雰囲気を感じ取ることができる．グレースケール画像を直接カラー化することは困難であり，近年ではニューラルネットワークを用いた自動カラー化が研究されている．本稿では，GAN を使用して自動カラー化を完全に一般化することを試みる．このネットワークは，CIFAR-100 などの一般公開されているデータセットで訓練する．また，GAN と従来のニューラルネットワークの結果を比較する．結果，従来のネットワーク（ベースラインネットワーク）と比較して高精度の画像を生成することができることがわかった．

キーワード:　グレースケール画像，自動カラー化，ニューラルネットワーク，CNN，GAN

1. 序論

カメラが発明されたばかりの写真は全て白黒であり，現代のカラー写真を見慣れている私達が，それを見ても情景がすぐさま思い浮かばない．白黒の写真の詳細な情報は，その写真撮影に関わった人物のみ知っているのである．それに対し現代のカラー写真は，写真撮影に関わっていなくてもその写真を一目見ればその場所の景観だけではなく，雰囲気や匂いまで感じることができる．そこで本稿では，グレースケール画像を現代写真のようにカラー化することで情報量が増え，過去の景観を可能な限り再現することを考えた．再現されれば，当時の感情なども蘇る．このカラー化を機械学習で行う取り組みは，多々研究されている．本稿では，Goodfellow 氏ら[1]の提案した敵対的生成ネットワーク(Generative Adversarial Networks，以下 GAN)を用いて自動カラー化を行う方法を採用する．これを用いることで従来手法を上回る，幅広いカテゴリの画像をより鮮やかなカラー化が期待できる．公開データセットの CIFAR-100 によって，ネットワークを訓練する．

弊研究室では，グレースケール画像から RGB の配合を検討する手法と，畳み込みニューラルネットワーク(Convolutional Neural Network，以下 CNN)などの機械学習を用いる手法でカラー化が行われてきた．前者の手法には，緑成分の多い山や草原を学習し同色の画像をカラー化する[2]，青色成分の多い海や空の画像を学習し同様の画像をカラー化する[3][4]，赤成分の多い紅葉画像を学習し同様の画像を学習する[5]，さらに人物画像[6]や，複数色のカラー化[7]なども行われている．後者の手法には，CNN を用いカラー化[8]，HSV 色空間を用いてカラー化[9]がある．ただしどちらも風景画像に限定している．

そこで本稿では，GAN を用いることで，風景画像に限らず，より一般的なカラー化が可能と考えた．一般的なカラー化が行えることで，より高い精度で情景の再現が可能となる．これは，自動カラー化問題の最大の目標である．それは，車や衣服などの人工物は，色のパターンが風景などの自然物と比べ遥かに多様であり，カラー化の障害となるからである．機械学習においては，学習データセットに紫の車の画像がなければ車が紫にカラー化されることはほとんどない．また，学習データセットにおける車がほとんど赤であったら，車は赤でカラー化されることが多くなってしまう．本稿では，人工物を含めたカラー化を目的に，GAN によるグレースケール画像のカラー化を行う．

2. Generative Adversarial Networks

GAN は，日本語では敵対的生成ネットワークといい，新しいタイプの生成モデルである．GAN は，Generator と Discriminator の 2 つのネットワークからなっており，それぞれが異なる役割を持つ．図 1 に GAN の概要図を示す．

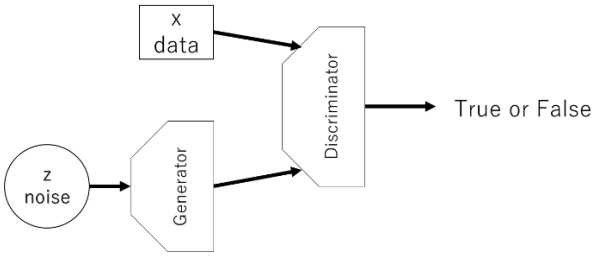


図 1:GAN の概要図

まず図 1 にて左側の六角形ブロックで表されている Generator の役割は，画像を生成することである．通常の GAN では 100 次元程度のノイズから画像を生成する．次に図 1 にて右側の六角形ブロックで表されている Discriminator の役割は，入力の画像が元のデータ分布から与えられているのか，それとも Generator により生成された画像なのか判別することである．この 2 つのネットワークは互いに学習し，成長していく．最終的には，Generator は，Discriminator が本物と間違えるような画像を生成することを目指して学習し，Discriminator は Generator の画像を間違えることなく判別することを目指して学習する．Discriminator の正答率は 50%程度になることが理想である．

GeneratorとDiscriminatorのアーキテクチャは，どちらも多層パーセプトロンモデルに従う．カラー化は画像変換であるため，GeneratorとDiscriminatorはどちらもCNNである．また式(1)及び式(2)に，Generator及びDiscriminatorを数学的に表す．

$$\min_{\theta_G} J^{(G)}(\theta_D, \theta_G) = \min_{\theta_G} \dot{E}_z\left[\log\left(1 - D\left(G(z)\right)\right)\right] \quad (1)$$

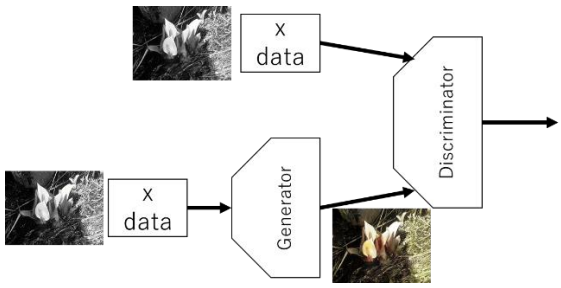
$$\max_{\theta_G} J^{(D)}(\theta_D, \theta_G) = \max_{\theta_G}(\dot{E}_x\left[\log\left(D(x)\right)\right] + \dot{E}_z[\log(1 - D(G(z)))]) \quad (2)$$

式(1)及び式(2)においてGeneratorは写像Gで表され，zはノイズ変数，すなわちGeneratorの入力である．同様にDiscriminatorは，0と1との間のスカラを生成するために写像Dによって表され，xは正解画像である．Gは，Discriminatorが生成データにおいて正しい予測をする確率を最小化するように学習され，Dは正しいラベルを割り当てる確率を最大化するように学習される．

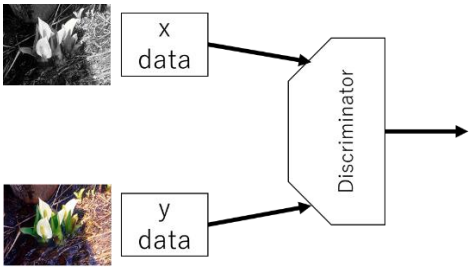
また，本研究で使用するGANはノイズから画像を生成するものではない．GANには他にも様々な種類があるが，本研究で使用するGANについて以下で述べる．

2.1 Conditional GAN

従来のGANでは，Generatorの入力はランダムに生成されたノイズデータである．しかし，グレースケール画像はノイズではないので，従来のGANはカラー化には適用できない．しかし，条件付き敵対的生成ネットワーク(Conditional GAN)と呼ばれるモデルを用いることにより，カラー化などのタスクに対応可能となる．図2にConditional GANの概要図を示す．



(a) Generatorの結果を判断する場合



(b) 正解画像を判断する場合

図2：Conditional GANの概要図

図2より，ノイズが導入されないので，Generatorの入力は画像となる．式(3)及び式(4)にConditional-GANを数学的に表す．数学的にはゼロノイズ，$G(0_z|x)$として扱われる．さらにDiscriminatorの入力は条件付きネットワークに対応するように変更される．yは正解画像を表す．

$$\min_{\theta_G} -J^{(G)*}(\theta_D, \theta_G) = \min_{\theta_G} -\dot{E}_z[\log(D(G(0_z|x)))] \quad (3)$$

$$\max_{\theta_G} J^{(D)}(\theta_D, \theta_G) = \max_{\theta_G}(\dot{E}_x\left[\log\left(D(y|x)\right)\right] + \dot{E}_z[\log(1 - D(G(0_z|x)|x))]) \quad (4)$$

図2よりDiscriminatorは，グレースケール画像を条件として，Generatorからの生成画像とオリジナルデータからの正解画像の2種類のカラー画像を入力とし，どのペアにオリジナルデータからの正解画像が含まれているかを判別する．

2.2 Convolutional GAN

GeneratorモデルとDiscriminatorモデルでは，DCGAN(Deep Convolutional GAN)[11]のガイドラインに従い，両方のモデルでCNNを採用する．このようにGeneratorやDiscriminatorにCNNを用いるものをConvolutional GANと呼ぶ．また，条件付きDCGANとしての変更も加える．Generatorのいくつかのレイヤーにドロップアウトを設け，過学習を抑制させる．Generatorの構造は，ReLU関数ではなく傾きが0.2のLeakyReLU関数を用いる点と，出力レイヤーにtanh関数を用いる点，活性化関数の前にバッチ正規化を行うなどの特徴がある．Discriminatorの構造はベースラインのエンコーダモデルと同様であり，LeakyReLU関数を用い，加えてバッチ正規化を行う．出力レイヤーにはsoftmax関数を用いて，一次元にマッピングした入力を確率値に変換する．Discriminatorの入力は，グレースケール画像と連結されたGeneratorの生成画像またはオリジナルデータからのカラー画像である．

2.3 Patch GAN

Patch GANとは，ピクセルセグメントごとにDiscriminatorによる真偽判定を行うGANのことである．例えば，32×32の画像を8×8に切り出して，その少領域(パッチ)ごとにDiscriminatorによる判定を行う．また，L1ノルムは画像をぼかし低周波成分を捉える性質を持つ．これにより，Discriminatorで高周波成分を捉えれば一度で大域的及び局所的な判定を行うことができる．

3. 手法

画像のカラー化は，高次元の入力を高次元の出力にマッピングする，画像から画像変換のタスクである．これは，入力における構造が出力における構造と高度に位置合わせされているピクセルワイズ回帰問題として見ることができる．ベースラインとして完全畳み込みネットワーク(Fully Convolutional Networks，以下 FCN)を使用し，GAN モデルと比較する．

3.1 ベースラインネットワーク

従来(ベースライン)のネットワークとして，FCN を使用する．これにおいて，通常の CNN で使用されるプーリングレイヤーは，使用されない．これは，プーリングを行うことにより生じる必要情報の欠損や解像度の低下を防ぐためである．

モデルのアーキテクチャはエンコーダユニットとデコーダユニットを持ち，対称的である．エンコーダは 4×4 畳み込みレイヤーで構成され，デコーダは 4×4 転置畳み込みレイヤーで構成されている．活性化関数には ReLU 関数を使用する．また，損失関数は平均二乗誤差を使用し，オプティマイザは Adam を使用する．

3.2 バッチ正規化

Generator 及び Discriminator にそれぞれバッチ正規化を適用する．バッチ正規化とは，バッチデータからバッチデータの平均値を引き，標準偏差で割ることによりデータの平均を 0，分散を 1 とすることである[12]．パラメータ更新の際には前の層の出力に影響を受けて重みが更新されるので，深いネットワークであれば前の層の出力は毎回変わることが想定される．そこでバッチ正規化を行いそれぞれの入力データの分布を 0 平均 1 分散のガウス分布に変換することで，入力データが学習のたびに変わることを防ぐ効果がある．これにより中間層の学習が安定化し，過学習が抑制される．本研究では，Generator 及び Discriminator の全ての畳み込み層の後かつ活性化関数の前でバッチ正規化を行う．

3.3 ドロップアウト

ドロップアウトは，ニューロンをランダムに消去しながら学習する手法である．これにはバッチ正規化と同様に，過学習抑制の効果がある訓練時に隠れ層のニューロンをランダムに選び出し，その選び出したニューロンを消去する．消去されたニューロンは信号の伝達が行われなくなる．

機械学習では，アンサンブル学習を使用することが多い．これは，複数のモデルを個別に学習させ，推論時には，その複数の出力を平均するというものである．アンサンブル学習を行うことで，ニューラルネットワークの認識精度が数%向上する事が実験的にわかっている．

ドロップアウトはこのアンサンブル学習と近い関係がある．それは，ドロップアウトは学習時にニューロンをランダムに消去することで，毎回異なるモデルを学習させていると解釈できるからである．つまりドロップアウトは，アンサンブル学習と同様の効果を擬似的に 1 つのネットワークで実現していると考えることができる．本研究においても，これらのような効力を期待するためにドロップアウトを採用している．

3.4 LeakyReLU

Generator のダウンサンプリング及び Discriminator において，LeakyReLU 関数を使用する．これは負数を全て 0 として扱う通常の ReLU 関数とは異なり，ある程度の負数を許可するものである．式(5)に数式を示す．またαはパラメータを表し，本研究では 0.2 を使用する．

$$f = \begin{cases} x, & x > 0 \\ \alpha x, & x \leq 0 \end{cases} \tag{5}$$

3.5 ラベル平滑化

Discriminator の学習において，ラベル平滑化を用いる．通常では Discriminator はデータからの真の画像であれば 1，Generator の生成画像すなわち偽の画像であれば 0 を出力するよう学習される．また，GAN の学習を促進させることを目的とし，そのラベルにランダムに微少量を加えることがある．その他にも，1 の代わりに 0.9 を使用する，0 の代わりに 0.1 を使用するという手法もある．適切なラベリングをシミュレーションによって検討する．

4. シミュレーション

表 1 に，シミュレーションの実行環境について示す．プログラムは Python を用いて作成され，ライブラリは Keras を使用する．また，バックエンドは Tensorflow である．この環境に基づき，種々のシミュレーションを行う．表 2 に，行うシミュレーション内容を示す．

4.1 ネットワークモデル

使用するネットワークモデルの構造を示す．図 3，図 4 にそれぞれ Generator 及び Discriminator のモデル図を示す．カーネルサイズが 4×4 でストライドが 2 である．これは，チェス盤ひずみを防ぐためにカーネルサイズがストライドで割り切れることが必要だからである[12]．チェス盤ひずみとは，生成された画像においてピクセル空間が均一にカバーされないことであり，GAN の比較的一般的なトラブルである．

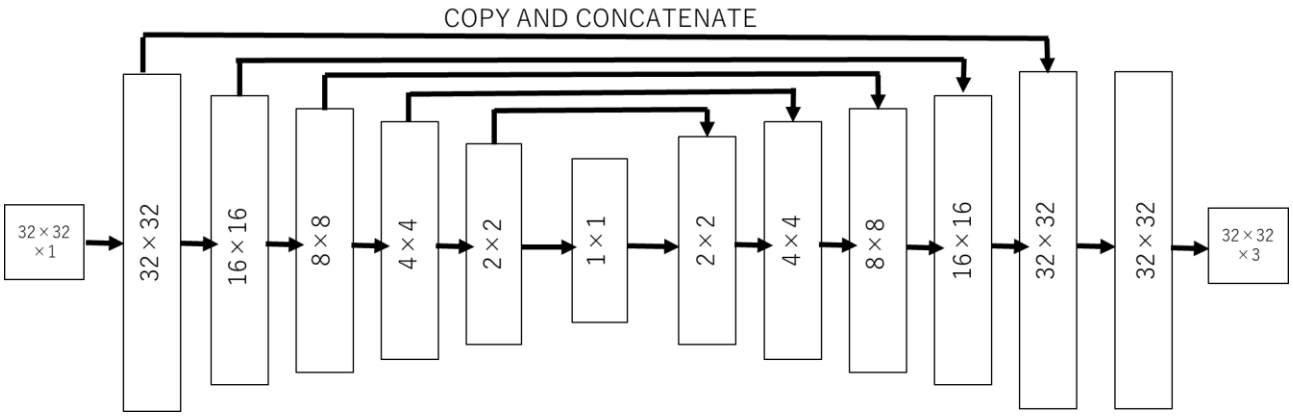


図 3： Generator のモデル図

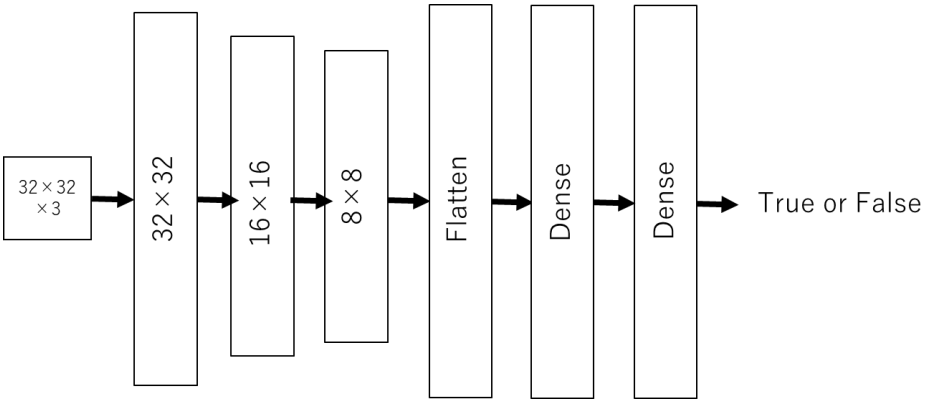


図 4： Discriminator のモデル図

4.2 学習

学習には公開データセットの CIFAR-100 を使用する. 訓練データ 5 万枚テストデータ 1 万枚を含むデータセットであり, 100 クラスに分類される画像を含む. また, サイズは 32×32 であるのでネットワークもそれに応じた形状となっている. バッチサイズは 128 で 100epoch 学習を行う.

表 1: 実行環境

名称	バージョン
CPU	Inter Core i7-8700, 3.20GHz
GPU	NVIDIA GeForce RTX 2070
Memory	32.0GB
OS	Windows 10 Pro
Python	3.6.7
Keras	2.2.4
Tensorflow	1.12.0
Tensorflow-gpu	1.12.0
Scikit-image	0.14.1
Numpy	1.15.4
Pillow	5.3.0
Opencv-python	3.4.2
h5py	2.8.0
Matplotlib	3.0.2
CUDA	9.0
cuDNN	7.1.4

表 2: シミュレーション

番号	シミュレーション内容
1	損失関数の検証
2	オプティマイザの検証
3	ラベリングの検証
4	パッチサイズの検証
5	学習率の検証
6	クラス数ごとの検証
7	ベースラインネットワークとの比較

5. 結果

結果の画像の性能を評価するために, SSIM を用いる検証を行う. なお, 今回は用いた画像のサイズが 32×32 と小さいため, ユーザアンケート等による主観的な評価は実施しない.

SSIM とは, Structural Similarity の略で, 画像の類似度を示す画像評価式である. 式(6)に, 数式を示す. なお, SSIM は画像内の局所領域毎に算出され, μ_xとμ_yは領域内の画素値の平均を表し, σ_xとσ_yは画素値の標準偏差, σ_{xy}はその共分散を表す. また, c_1及びc_2は分母の値が非常に小さくなった際に評価値が不定とならないための定数である.

$$\mathrm{SSIM} = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)} \tag{6}$$

SSIM は, 画素値, コントラスト, 構造の 3 点の変化を捉える. これにより, PSNR や MSE などの画像評価式よりも主観的で人間の感覚に近い値を算出する性質を持つ. 値の範囲は 0 から 1 である. また, 結果の表には測定データの平均を示す.

5.1 損失関数の検証

Generator の損失関数以外の条件を全て一致させ, 適切な損失関数の検証を行う. 表 3 に結果を示す. 表 3 より, Generator の損失関数は 10×L1+Binary が最適となる. L1 損失とは画素の差分を足し合わせたものであり, Binary とは 2.1 式(3)で示したものである.

5.2 オプティマイザの検証

オプティマイザ以外の条件を全て一致させ, 適切なオプティマイザの検証を行う. 表 4 に結果を示す. 表 4 より, 最適なオプティマイザは RMSprop であることがわかる. 式(7)に数式を示す. wが重みを表し, Eは誤差を表す.

また, αやh, η及びϵはパラメータで, これらの値をもとに重みの更新を行う.

$$h_t = \alpha h_{t-1} + (1-\alpha)\nabla \mathrm{E}(w_t)^2$$

$$\eta_t = \frac{\eta_0}{\sqrt{h_t} + \epsilon}$$

$$w^{t+1} = w^t - \eta_t \nabla E(w^t) \tag{7}$$

式(5)より, αによって過去の勾配の影響を抑えると共に, hを優先して反映させるという効果を生み出す.

これは, より直近の勾配情報を優先して計算するということである. 本研究においては, このような性質を持つ RMSprop が適していることが分かった.

5.3 学習率の検証

学習率以外の条件を全て一致させ, 適切な学習率を検証する. 表 7 に結果を示す. 表 7 より, 最適な学習率は Generator 及び Discriminator 共に 2e-4 であることがわかる.

表 3:損失関数の検証結果

	L1	Binary	MSE	L1+Binary	10×L1+Binary	100×L1+Binary	MSE+Binary
SSIM	0.8957	0.8752	0.9075	0.8804	0.9083	0.9057	0.8601

	10×MSE+Binary	100×MSE+Binary
SSIM	0.9043	0.8948

表 4:オプティマイザの検証結果

	SGD	RMSprop	Adagrad	Adadelta	Adam	Adamax
SSIM	0.8515	0.9084	0.8593	0.6767	0.8768	0.8981

表 5:ラベリングの検証結果

	0.0to1.0	0.0to0.9	0.1to1.0	0.1to0.9	random+0.0to1.0	random0.1to0.9
SSIM	0.9084	0.9035	0.9057	0.9090	0.9013	0.9055

表 6:パッチサイズの検証結果

	4×4	8×8	16×16	32×32
SSIM	0.9030	0.8867	0.8993	0.9036

表 7:学習率の検証結果

	2e-3	2e-4	2e-5	2e-6	gen:2e-6 dis:2e-4	gen:2e-4 dis:2e-6
SSIM	0.9034	0.9053	0.8954	0.8687	0.8654	0.9042

表 8:クラス数ごとの検証結果

	10	100
SSIM	0.9313	0.9084

表 9:ベースラインネットワークとの比較結果

	ベースライン	GAN
SSIM	0.7070	0.9084

表 10:彩度の比較結果

	オリジナル	ベースライン	GAN
SSIM	0.3216	0.1952	0.2561

表 11:過去研究との比較結果

	CNN	HSV 空間	GAN
SSIM	0.8430	0.8500	0.9084

5.4 クラス数ごとの検証

クラス数以外の条件を全て一致させ，クラス数による性能を比較する．表 8 に結果を示す．表 8 より，クラス数が多いほど生成画像の精度は下がることがわかる．

5.5 ラベリングの検証

ラベル以外の条件を全て一致させ，Discriminator の適切なラベリングを検証する．表 5 に結果を示す．表 5 の random は，ラベルの値に微小量を加えることである．表 5 より，最適なラベリングは 1 を 0.9 に置き換え，0 を 0.1 に置き換える場合であることがわかる．

5.6 パッチサイズの検証

パッチサイズ以外の条件を全て一致させ，適切なパッチサイズを検証する．表 6 に結果を示す．表 6 より，最適なパッチサイズは32×32 であり，学習画像と同じサイズである．このことから，Patch GAN は不要であることがわかる．

5.7 ベースラインネットワークとの比較

ベースラインネットワークとの比較を行い，表 9 に結果を示す．表 9 より，GAN での生成画像の方がベースラインでの生成画像よりも高精度であることがわかる．図 5 に正解画像，ベースラインによる生成画像及び GAN による生成画像の比較画像を示す．

図 5(a)より，左端の茶色い木の画像では GAN は緑にカラー化している．ベースラインは何色にカラー化しているのかはっきりとはわからないが，緑にはカラー化していない．また，中心の夕焼けのような画像はベースライン，GAN 共にカラー化に失敗している．右端の林檎の画像では，ベースラインは赤黒くなっておりエッジを捉えられていないが，GAN は正確に赤色を塗っている．また，図 5(b)より，右端から 2 番目の木の画像を GAN は正確にカラー化することができている．右端の車の画像も，ベースラインよりも正確なカラー化に成功している．

しかし，図 5(c)の車両や花，オレンジの画像や，図 5(d)の右端の体温計の画像は，ベースライン及び GAN 共に正解画像との差が大きい．更に図 5(e)の右端及び図 5(f)の左端の人物画像も正確にカラー化しているとは言えない．後者は肌色の薄く塗ってはいるが，前者に至ってはほぼグレースケールである．また，図 5(j)の右端から 2 番目の林檎の画像は，図 5(a)の右端の林檎の画像と比べて赤色が薄い．

これらより，ベースラインによる生成画像及び GAN による生成画像共にクラスによっては色味が薄く，正解画像との差が大きい画像があることがわかる．しかし，GAN による生成画像はベースラインによる生成画像よりもより鮮やかで正解画像に近いカラー化に成功している．

5.8 過去研究との比較

弊研究室の過去の研究結果と本研究の結果を比較する．CNN を用いた風景画像の自動カラー化[8]，HSV 空間における白黒画像の自動カラー化[9]より，結果の SSIM を抜粋し GAN による SSIM と比較する．表 11 に比較結果を示す．表 11 より，過去研究を上回る精度の画像を生成することができている．

6. 考察

本研究では GAN を用いた自動カラー化についての研究を行った．表 9 より，GAN により生成された画像の方がベースラインネットワークにより生成された画像よりも高精度である．このことから，GAN を用いるとカラー化の精度が高くなることがわかった．また，より精度を上げるためにハイパーパラメータの調整を行い，最適なパラメータを求める．

次に，ベースラインネットワークによる生成画像と GAN によるによる生成画像で， それぞれの鮮やかさを比較する．表 10 にその結果を示す．表 10 より，GAN による生成画像の彩度がベースラインネットワークの生成画像と比べ高いことがわかる．

また，生成された画像の中で平野や山や木などの広大な自然物の SSIM の測定を行ったところ，平均が 0.9189 となり，全てのクラスを含めるときの SSIM である 0.9089 を上回る．このことから広大な自然物の SSIM がその他人工物の SSIM を上回ることがわかる．これは，広大な自然物は人工物と比べカラー化が容易であることを示す．それは自然物の一元性に起因する．例えば，海であれば青系，森であれば緑系というように，それぞれの色は概ね決まっている．これに対し，車のような人工物にはそのような性質はない．これらの理由により，自然物の SSIM がその他人工物の SSIM を上回ったと考えられる．

(a) 結果 1

(f) 結果 6

(b) 結果 2

(g) 結果 7

(c) 結果 3

(h) 結果 8

(d) 結果 4

(i) 結果 9

(e) 結果 5

(j) 結果 10

図 5: CIFAR-100 での生成結果
各画像の上段:オリジナル画像
各画像の中段:ベースラインの生成画像
各画像の下段:GAN の生成画像

7. 結論

GANを用いたグレースケール画像の自動カラー化を高精度で行うことができた．SSIM 及び彩度ともに，GAN で生成したカラー画像がベースラインネットワークで生成したカラー画像を上回った．

今後の課題として，人工物のカラー化においては異なる色付けをしている場合も一定の数見られたので，超大規模なデータセットで何か月もかけて学習するか，新たな手法を発案するなどが挙げられる．

また，本研究では SSIM による性能評価を行っているが，256×256 などのサイズが比較的大きい画像で学習を行い，ユーザによる評価を行うことでよりよいデータが得られ，さらなる性能強化に繋がるだろう．また，GAN はカラー化以外にも様々なタスクを実現可能であり，今後より一層発展が期待できる．本研究のような画像変換タスクに限ってもその応用は多岐に渡る．その一例として，物体のクラス分類だけではなくその位置情報まで求める Semantic Segmentation や，画像のエッジから推定して画像を生成する Edges to Photo，航空写真から地図を生成する Aerial to Map など様々である．付録にて，本研究で使用した GAN で，応用の 1 つであるラベルからの画像生成を行った結果を示す．

謝辞

本研究を進めるにあたり，熱心なご指導をしてくださいました濱川恭央教授に感謝致します．また，日常の議論を通じて多くの示唆を与えてくださった濱川研究室の皆様に感謝致します．

参考文献

[1] Ian Goodfellow， Jean Pouget-Abadie， Mehdi Mirza， Bing Xu， David Warde-Farley，Sherjil Ozair， Aaron Courville， and Yoshua Bengio．Generative adversarial nets.

[2] 大栄大志，青色画像の自動カラー化に関する研究，鹿児島工業高等専門学校，2015

[3] フドゥルムル・オヤンガ，ニューラルネットワークを用いた紅葉画像のカラー化に関する研究，鹿児島工業高等専門学校，2013

[4] 下留諒，ニューラルネットワークを用いた人物画像のカラー化に関する研究，鹿児島工業高等専門学校，2014

[5] 早稲田凌亮， 誤差逆伝播法による複数色画像のカラー化に関する研究， 2016

[6] 飛佐洋平，誤差逆伝播法による色識別に関する基礎研究，鹿児島工業高等専門学校，2009

[7] 中尾真一郎，ニューラルネットワークを用いた白黒画像のカラー化に関する研究，鹿児島工業高等専門学校，2010

[8] 池田伊織，CNNを用いた風景画像の自動カラー化，鹿児島工業高等専門学校，2017

[9] 下吉賢信，HSV空間における白黒画像のカラー化，鹿児島工業高等専門学校，2017

[10] Ian Goodfellow，NIPS 2016 Tutorial: Generative Adversarial Networks，2017

[11] Alec Radford， Luke Metz， and Soumith Chintala． Unsupervised representation learning with deep convolutional generative adversarial networks． 2015.

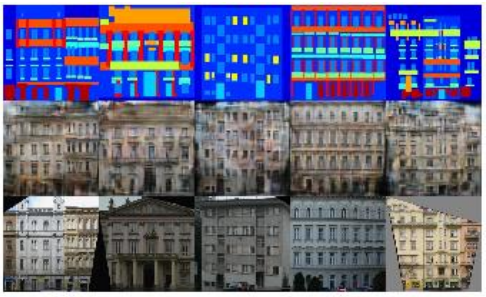
[12] François Chollet， PythonとKerasによるディープラーニング，株式会社マイナビ出版，2018

付録

窓などを色分けしたラベル画像から，建築画像の生成を行う．図 6 に生成結果を示す．

(a) 結果 1

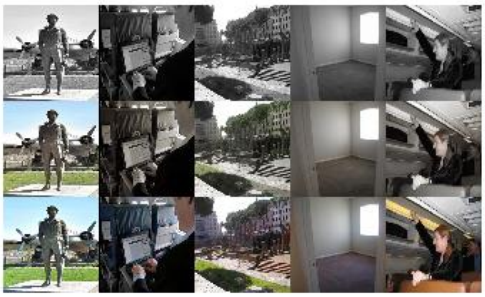
(b) 結果 2



(c) 結果 3

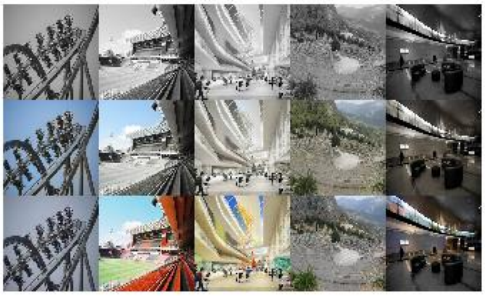
図 6: ラベル画像から建築画像の生成結果
各画像の上段:ラベル画像
各画像の中段:GAN の生成画像
各画像の下段:正解画像

大規模データセットである Places365 で学習を行う. 総画像数 180 万で, クラス数は 365 である. 図 7 に生成結果を示す. 学習回数が CIFAR-100 でのシミュレーションの約 3 分の 1 であるが, 自然物は高精度で生成できた.



(a) 結果 1

(b) 結果 2



(c) 結果 3

(d) 結果 4

(e) 結果 5

(f) 結果 6

(g) 結果 7

(h) 結果 8

図 7: Places365 での生成結果
各画像の上段:ラベル画像
各画像の中段:GAN による生成画像
各画像の下段:正解画像