平成 30 年度　特別研究報告書

指定画像と類似した画像生成のための データセットの構築

龍谷大学　理工学部　情報メディア学科

T150508　学生氏名 近澤 勇太

指導教員　三好 力　教授

内容概要

本論文では、インターネットや SNS 上の画像を利用するための利便性の向上と知らず知らずに行ってしまう著作権侵害の防止策のシステムの提案を行う。本システムでは 1 枚の画像からインターネット上で多くの類似画像を集め、集めた画像からディープラーニングを用いて画像を生成するものである。本研究では類似画像を集める手段として、画像の特徴を用いて画像検索を行う Content-Based Image Retrieval (CBIR)を用いた実験を行う。実験では CBIR を用いて画像検索を行い、類似画像を集めデータセットを構築する実験を行う。

また、学習に使用する画像の著作権や、生成した画像に対する著作権や利用範囲についての説明と考察を行う。

目次

第 1 章 緒言

1.1 研究の背景と目的

今日、インターネットや SNS の利用者数の増加と発展ともに多くのイラスト、写真（以下、合わせて画像と呼ぶ）が日々、公開され、多くの個人、企業が画像を使用する機会が増えている。そんな中、問題となるのが著作権侵害である。他人の投稿した画像を自分のものとして投稿することや、アニメや漫画のワンシーンをキャプチャして投稿、または改変し投稿するコラージュ画像は著作権侵害にあたる。アニメや漫画は人気商売だから基本的には放置されているが、SNS では世界中に発信しているため違法とされない「私的利用」の範囲を超えている。また、フリー素材であると謳っていても、商用利用が不可である場合もある。SNS やインターネットで多くの画像を目にする機会があるが、自由に使うことのできる画像を探すのは難しい。

そこで本論文では、SNS やインターネットでの安易な著作権侵害の予防を目的とした、自由に利用できる画像の提供を行うシステムの提案を行う。本システムでは自らが、SNS やインターネット上で使用したい画像を検索クエリとし、類似点のある画像をインターネット上で機械的に集めることを想定し、使いたい画像と類似する画像を Deep Learning の手法の一つである DCGAN(敵対的生成ネットワーク)で生成する。一枚の画像から自動で似て非なる画像を Deep Learning の学習によって生成されたネットワーク、つまり人工知能が合成することで問題を解決する。類似する画像の選別には画像特徴を用いた検索を行う。

また、生成した画像に対する法的問題については 1.2 項で述べる。SNS での投稿物に対するユーザーの権利や著作権侵害への対応について、本システムが画像を生成するに当たって、画像データを集める際に著作権に保護された画像が含まれる場合、また、著作権に保護された画像が画像生成に使用される場合、著作権侵害となるかについては本章の次項で述べる。

1.2 著作権法

著作権法は、知的財産権の一つである著作権の範囲について定めたものである。著作権法第一節の第二条より、著作権に守られる、著作物とは思想または感情を創作的に表現したものであって、文芸、学術、美術又は音楽の範囲に属するものをいい、著作者とは著作物を、創作するものをいう。

1.2.1 画像収集に関する著作権法

本プログラムでは生成した画像を提供するにあたって、画像の取集を行う。具体的に提供する部分は画像を取集、学習を行い、画像を生成するプログラムと生成画像と

なる。それに伴い画像を提供するにあたって、著作物である画像を取集及び複製、抽出することは違法ではないかという問題が生じる。その点に関する部分は第二章「著作者の権利」に属する第三節「権利の内容」内の第五款「著作権の制限」に記されている。一部抜粋したものを下記に記す。

「著作物は、電子計算機による情報解析（多数の著作物その他の大量の情報から、当該情報を構成する言語、音、影像その他の要素に係る情報を抽出し、比較、分類その他の統計的な解析を行うことをいう。以下この条において同じ。）を行うことを目的とする場合には、必要と認められる限度において、記録媒体への記録又は翻案（これにより創作した二次的著作物の記録を含む。）を行うことができる。ただし、情報解析を行う者の用に供するために作成されたデータベースの著作物については、この限りでない。」〔情報解析のための複製等 第四十七条の七〕

上記より、考案するプログラムは画像を生成するための情報解析を行うプログラムであり、それを目的とする画像の収集や複製、抽出であるため問題ないといえるだろう。また、そのプログラムを提供することは、プログラムが商用であっても法律上問題とはならない。また、2017 年に行われた内閣府知的財産戦略推進事務局による「知的財産推進計画 2017」では、以下のように述べられている。

「学習用データに著作物が含まれている場合には、データの共有が著作権法上問題となるおそれがあるため、我が国の AI 作成の促進に向け、特定当事者間を越えて学習用データを提供・提示する行為について、前述の「柔軟性のある権利制限規定」に関する制度設計や運用の中で検討を進めることが必要である。」（知的財産推進計画 2017、11 ページより抜粋）

今後の法整備でも、柔軟に対応されていくだろう。

1.2.2 盗作に関する著作権法

前項では画像を収集することは問題ないことを示した。次に問題となるのが本システムで生成した画像を実際に提供できるかである。特に問題点となるのが人工知能で生成した画像をフリー素材として提供した場合、生成画像が学習用データで用いられた画像の盗作にあたるのではないかという点である。盗作の判断基準とされる一般的な要件は以下である。

- 著作物であるか
- 著作権の存在が認められるか
- 依拠性が認められるか
- 類似性が認められるか
- 著作物利用の権限を持っているか
- 創作的な表現がされているか

このうち問題となるのが依拠性と類似性である。依拠性とは既存の著作物を参考に創作されているかということである。Deep Learning などの技術を用いて創作したものに関する著作権の議論については数多く行われている。今回のように Deep Learning で創作したものと、学習で使用したものとの間に依拠性が存在するかという件に関しても 2017 年に行われた内閣府知的財産戦略推進事務局による「知的財産推進計画 2017」で行われている。当議論では、AI 生成物が悪用などの問題となる可

能性の部分で、依拠性について述べられている。内容は、学習用データに含まれる著作物がそのまま出力された場合に依拠性があると考えられるというものである。依拠製があるとされた場合、利用者については出力しただけであれば、私的複製（著作権法第三十条第一項）に該当し、権利侵害とならない。しかし、出力した生成物を SNS 等に投稿する行為は公衆送信権侵害（著作権法第二十三条第一項）となる。また、学習済みモデルの作成者については、侵害行為を幇助したとて共同不法行為（民法七百十九条第二項）に基づく損害賠償責任を負う可能性があるほか、判例によれば複製等の主体として差止請求の対象となる可能性もあると考えられる。ここで、示されているのはあくまで、学習用データに含まれるデータがそのまま出力された場合である。このことから、本論文で提案するシステムを実際に運用する場合は、学習データがそのまま出力されないことや、極度に類似しないことを示すことができれば AI の生成物として認められ画像を提供することが可能となるだろう。

1.3 Twitter での画像使用

SNS での画像使用例として Twitter 上での画像使用について説明する。
Twitter ユーザーの自らの投稿物に対する権利について、Twitter サービス利用規約から一部抜粋する。
「ユーザーは、本サービス上にまたは本サービスを介して自ら送信、投稿または表示するあらゆるコンテンツに対する権利を留保するものとします。ユーザーのコンテンツはユーザーのものです。すなわち、ユーザーのコンテンツ（他のコンテンツに組み込まれたユーザーの音声、写真および動画もユーザーのコンテンツの一部と考えられます）の所有権はユーザーにあります。……

ユーザーは、ご自身が必要な許可を得ているまたはその他の理由により素材を投稿し Twitter に上記のライセンスを許諾することができる法的権限を有している場合を除き、当該コンテンツが著作権その他の財産権の対象となる素材を含むものではないことに同意するものとします。」引用

上記より、Twitter ユーザーは投稿物に対する権利を有している。また自身の投稿物は著作権やその他の財産権の対象となる素材を含んでいない、つまり自身の投稿物は著作権を侵害するものではないことに同意するとしている。前章で述べたとおり、著作権侵害は法律的手段をとらず放置されることがほとんどだが、Twitter 社はこの 2017 の利用規約の改定とともに著作権侵害の申し立てに対応するとしている。

第 2 章 既存技術

2.1 画像検索手法

画像検索に用いられる2つの手法を説明する。

2.1.1 Text-Based Image Retrieval（TBIR）

Text-Based Image Retrieval（TBIR）はテキストを検索クエリとし、予めタグ付けされたキーワードをもとに画像検索手法である。検索の精度は画像のタグ付けの精度に大きく影響される。

TBIR の問題点として人手で膨大な量のタグ付けを行うという点であったが、近年では物体認識を用いて、画像のタグ付けを自動で行うことが可能となってきている。しかし、自動でタグ付けを行った場合主観的な表現のタグ付けは難しい。

2.1.2 Content-Based Image Retrieval (CBIR)

Content-Based Image Retrieval (CBIR)はテキストデータによるインデックスから画像を検索するのではなく、画像の色や形状といった画像情報から特徴を抽出し類似画像を検索する手法である。画像から色、テクスチャ、エッジなどを数値として多次元ベクトルで表現し、特徴同士の類似度で画像間の類似度を測る。TBIR のように画像にタグ付けをする必要がないが、単純な特徴量では、特徴量が画像の内容に結びつかないため、色合いといった見た目は近くとも、内容は大きく異なるという問題がある。一般に画像特徴量で用いられる特徴量には、画像全体から抽出される大域特徴と、画像の一部分から抽出される局所特徴量がある。大域特徴は同一画像や、連続写真といったほぼ同一の類似画像を検索することできるが、画像の一部分が隠れるなどの変化をしてしまうと近い特徴量が抽出できないが、局所特徴量では画像の一部から複数の特徴量を抽出するため、部分的に隠れていても特徴量得ることができる。局所的特徴量には、SIFT、SURF、FAST、ORB, KAZE、といったものがある。

近年では、TBIR と同じように物体認識の技術の発展によって画像内容の不一致は減少している

2.2 ヒストグラム

ヒストグラムは、画像中の画素全体的な分布を知るために、横軸を画素、縦軸に画素値の頻出頻度をプロットしたものである。ヒストグラムは一つの画像に対して一つに定まるが、異なる画像であってもそれと同一のヒストグラムをもつ場合もある。

2.3 Features from Accelerated Segment Test (FAST)

Features from Accelerated Segment Test (FAST)は、2つのエッジが交点であるコーナーを特徴点として高速に検出する特徴点検出手法である。FAST ではまず、コーナーかどうかの判定のため、画像中のピクセルpを選び、pの輝度をIpとする。図 2.1 のようにピクセルpを中心とする 16 ピクセルのうちI_pと比べて、予め定めた閾値 tより明るい、または暗いピクセルがn個以上連続して存在する場合pをコーナーとする。

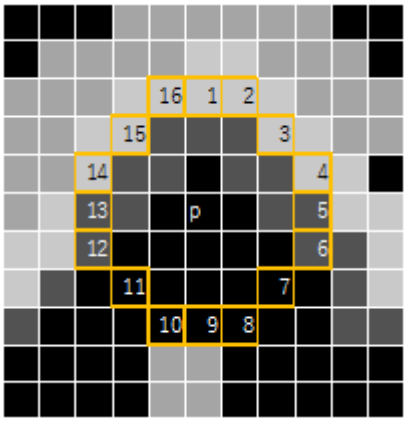


図 2.1 画素 p のコーナー判定
Edward Rosten and Tom Drummond (2006) Figure1 を参考に作成

実際のコーナーの検出には、pの周囲の 16 ピクセルを明るい(brighter)、類似(similar)、暗い(darker)の 3 つに式 2.1 の範囲で分類する。xは円状の画素の位置、$I_{p\to x}$は円周上の画素の輝度値を表している。

$$S_{p\to x}=\begin{cases} d & I_{p\to x}\le I_p-t \quad \text{(darker)} \\ s, & I_p-t<I_{p\to x}<I_p+t \quad \text{(similar)} \\ b, & I_p+t\le I_{p\to x} \quad \text{(brighter)} \end{cases} \qquad (2.1)$$

分類した 16 個の画素を特徴ベクトルとし、画素値が連続してn個以上が bright,または darker の時にコーナーとする。決定木による分類器である ID3 アルゴリズムで候補ピクセルがコーナーかどうかについて最大の情報利得が得られる決定木を作成する。作成した決定機を用いて高速でコーナー検出を行う。

FAST の問題点として隣接して特徴点を多重に検出してしまう。しかし、この問題は非最大値抑制によって解決できる。FAST による決定木を用いたコーナー検出では、応答値を出力しないため、以下の式で応答値Vを次式 2.2 のように算出する。2 つの隣接する特徴点のうちVが低い方を候補から外すことで多重検出を防ぐ。

$$V = \max\left(\sum_{x \in S_{brighter}} \left|I_{p\to x} - I_p\right| - t, \sum_{x \in S_{darker}} \left|I_p - I_{p\to x}\right| - t,\right) \quad (2.2)$$

2.4 Oriented FAST and Rotated BRIEF (ORB)

Oriented Fast and Rotated BRIEF (ORB) は、2 点の画素値の比較によってバイナリコードを生成する Binary Robust Independent Elementary Features (BRIEF)にスケール不変性と回転不変性を加えたものである。BRIEF は、CenSurE や SURF などで検出した特徴点をバイナリコードで記述する手法である。あらかじめ特徴点の周辺領域(パッチ)を平滑化し、パッチ上の 2 点x, yで画素値を比較し$I(x) < I(y)$ であれば 1、それ以外は 0 とする。これをパッチ上のn_d個のペアで行うことでn_d次元のビットコード列を得ることができる。特徴点検出では FAST によって特徴点を算出し、多段階に縮小した画像を生成し、スケール違いの画像にも対応させる。特徴点記述では BRIEF と同様に 2 点の画素の比較でバイナリコードを生成する。

2.5 Scale-Invariant Feature Transform (SIFT)

Scale-Invariant Feature Transform (SIFT)は ORB と同様にスケールの変化や回転に強い特徴量である。SIFT では異なる値のガウシアンフィルタにより平滑化された画像の差分を利用し、スケールの変化によって変動しない特徴量を記述する。

2.6 KAZE,Accelerated-KAZE 特徴

KAZE 特徴とその高速化を加えて改良した、Accelerated は、ORB と同様に特徴点の抽出と特徴点の記述からなる画像マッチング手法である。SIFT では画像の差分を得るために、ガウシアンフィルタを用いていたが、KAZE では非線形拡散フィルタによる平滑化画像から差分をとり、局所的な特徴を検出する。

2.7 Generative Adversarial Networks (GAN)

Generative Adversarial Network (GAN)は Deep Learning の学習モデルの一つで、Generator(生成器)と Discriminator(識別機)の二つのモデルを用いて学習させる。生成器は識別機をだませるような本物に近い画像を生成し、識別機は本物を見分けられるように、互いに競うように学習をしていく。GAN の概略図を図 2.2 に示す。生成器ではノイズzを入力とし、偽物の画像を生成する。識別機は生成された偽物と本物の画像を入力として受け取り、本物であるか偽物であるかを判定する。

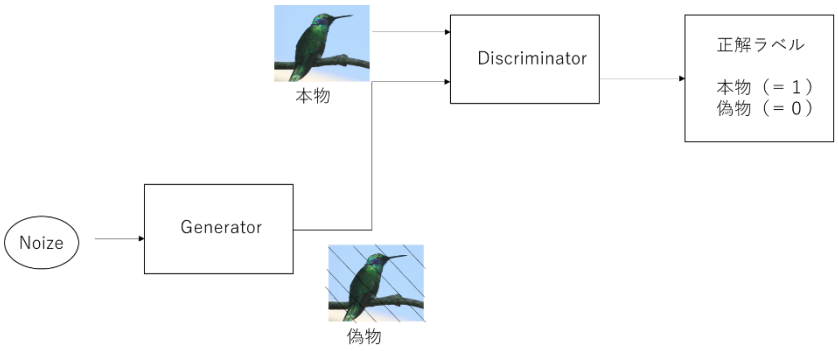


図 2.2GAN の概略図

この互いに競い合う学習構造を生成器 G 識別機 D の 2 プレイヤーによるミニマックスゲームとして定義すると式 2.3 のようになる。ここでxは訓練画像、zは潜在空間上のベクトルとする。

$$\min_G \max_D V(D,G) = E_{x\sim P_{data}(x)}[\log D(x)] + E_{z\sim P_{noize}(z)}\left[\log\left(1-D\left(G(z)\right)\right)\right] \quad (2.3)$$

G はベクトル $z\sim P_{noize}(z)$から画像を生成し、生成画像の確立分布$P_{data}(x)$と一致させることを目的としている。G が訓練データと似ているものを生成できると D の分類が失敗し $D\left(G(z)\right)$ が大きくなり、$\log\left(1-D\left(G(z)\right)\right)$ が小さくなる。識別機側では画像が$P_{data}(x)$、生成器の確率分布である$P_G(x)$のどちらの確率分布により生成された画像であるかを判別することを目的としている。D がうまく分類できると偽物と判断されるため$D\left(G(z)\right)$が小さくなり、$\log\left(1-D\left(G(z)\right)\right)$が大きくなる。また識別機の最適な出力は式 2.4 のようになる。

$$D(x) = \frac{P_{data}(x)}{P_{data}(x) + P_G(x)} \quad (2.4)$$

学習がうまく進行すると、$P_G(x)$と$P_{data}(x)$が一致するので$D(x)$は 0.5 となり識別機は本物か偽物かを識別できない状態となる。この状態が GAN の問題設定における最適化となる。

2.8 Deep Convolutional Generative Adversarial Networks (DCGAN)

Deep Convolution Generative Adversarial Networks (DCGAN) は、GAN に画像関連を中心に成功している畳み込みニューラルネットワークを適用し、安定した画像生成を行うことができる GAN モデルである。DCGAN では識別機では畳み込み層、生成器では図 2.3 のように Deconv.層を中心にネットワークを構築されている。

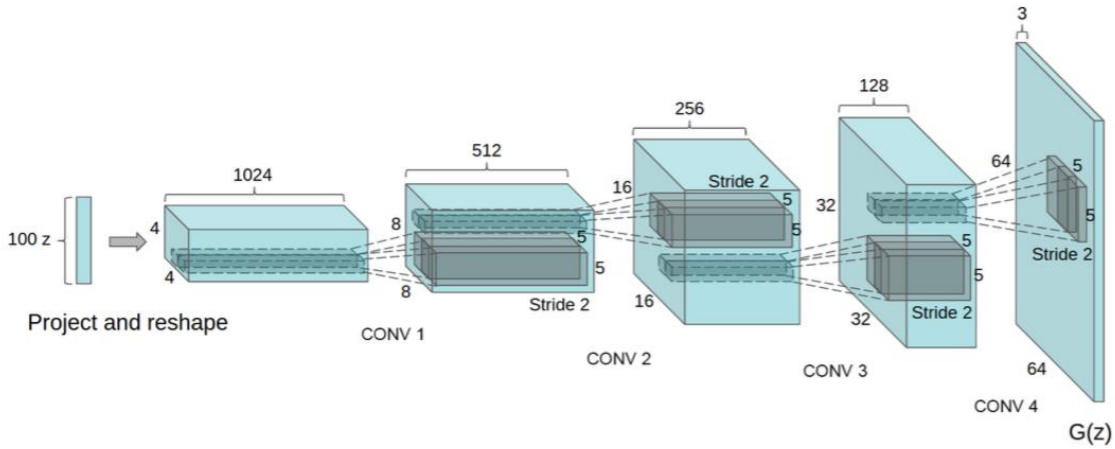
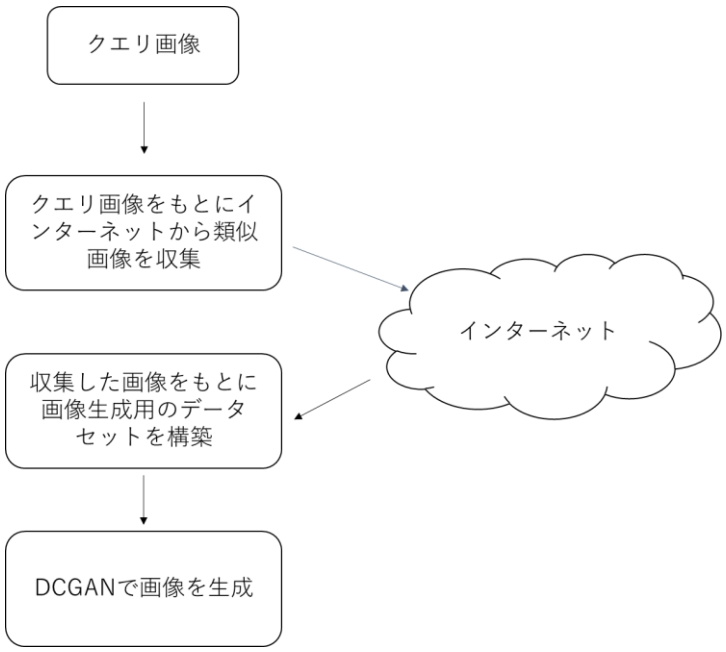


図 2.3 DCGAN の生成器の構造 (Radford ,Metz and Chintala (2015) Figure 1 から引用)

第 3 章 提案手法

SNS やインターネットでの安易な著作権侵害の予防を目的とした、自由に利用できる画像の提供を行うシステムの提案を行う。本システムでは図 3.1 のように自らが SNS やインターネット上で使用したい画像を検索クエリとし、類似点のある画像をインターネット上で機械的に集めることを想定し、使いたい画像と類似する画像を Deep Learning の手法の一つである DCGAN(敵対的生成ネットワーク)で生成をする。



提案手法の中での課題は画像を自動での画像収集と画像を生成する手段である。今回は画像を収集する手法についての検討と実験を行う。GAN を用いて画像を生成すには、何千、何万枚の画像が必要となる。研究成果を上げている GAN の分野の論文では、質の高い画像(高解像度や、画像の中心に目的の物体が移っているなど)を何万、何十万と用いて生成を行っている。たとえば、DCGAN の論文では、1 万人の顔画像 300 万枚から、Opencv の顔検出で高解像度の顔画像を検出し、得られた 35 万枚の顔画像をもとに画像を生成している。生成画像の質は格段に落ちるが、技術情報共有サービスの Quiita では数百枚から数千枚での実験結果もある。これらを参考に実験では鳥の画像 1 万枚を含む 2 万枚の画像の中から、画像の類似度の高いものから順に 5,000 枚を抽出しその中に含まれる鳥の画像数から。画像の類似度の比較には ORB 特徴量、A-KAZE 特徴量、カラーヒストグラムを用いる。

提案手法の画像生成の部分では、画像生成の分野で成功を収めている DCGAN での生成を想定する。DCGAN の利用に伴い、検索で得られる画像は同一サイズでなければならない。しかし、インターネット上の画像や今回の実験で使用する画像サイズは統一されていない。

そこで、ORB による特徴点記述を用いて画像を正方形に正規化する。ORB で得ら

れた複数の特徴点の重心から対象物の位置を推定し、その重心を中心に画像を正方形に変更する。図 3.1 のように横長の画像の場合は重心を中心に画像の縦の長さ、縦長の場合は重心を中心に画像の横の長さにあわせてきりとる。図 3.2 のように重心が片側によって重心を中心に画像を切り取れなかった場合は重心が近い方の端からきりとる。

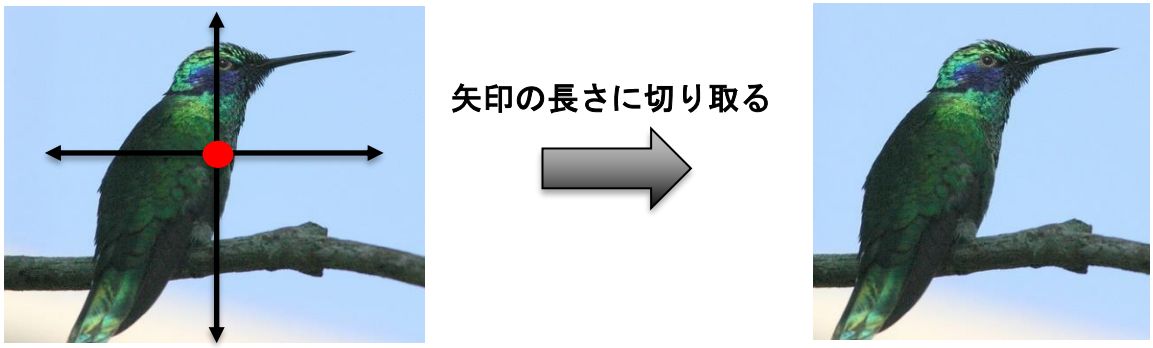


図 3.1 ORB を用いた画像の正規化

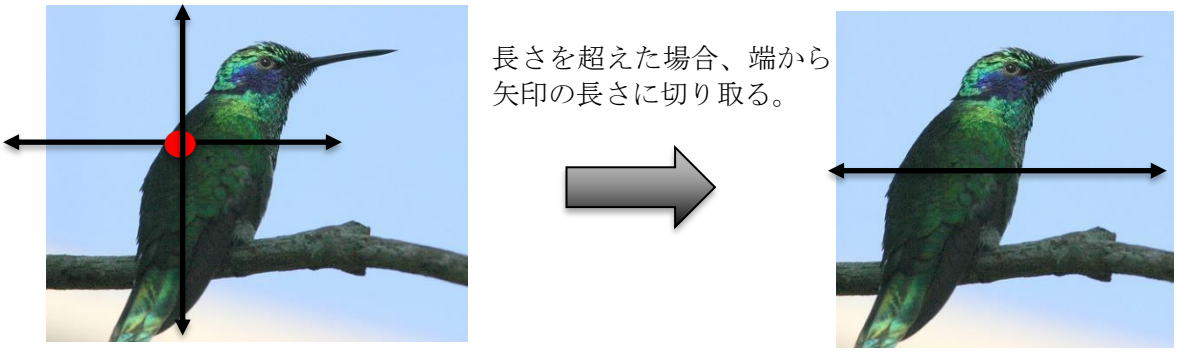


図 3.2 ORB を用いた画像の正規化(重心が片側に寄っている場合)

第 4 章 実験

4.1 実験目的

本実験では多様な画像が含まれるデータベースの中から CBIR による検索を用いて、画像生成用のデータセットを作成することを目的とする

4.2 実験 1

多様な形状の画像に対して、対象物が中心に来るような正方形の画像に変換する手法の有効性を検証する実験を行った。

4.2.1 実験結果 1

図 4.1 の画像を、両端を切り落とすように、正方形の形にトリミングした結果を図 4.2 に、ORB による特徴点の重心をとってトリミングしたものを図 4.3 に示す。下記図のような画像に対して行う場合、対象物が中心になるように正方形の画像に変換できていることがわかる。

図 4.1 加工前の花の画像

図 4.2 両端を切り落とすように正方形にした花の画像

図 4.3 提案手法によって正方形にした花の画像

欠点としては、図 4.4 のように文字など映り込むとそちらに特徴点が検出され、目的の物体以外の部分に重心が傾いてしまう点や、図 4.5 のように画像の全体から特徴点が検出できる画像の場合、図 4.6 のように画像の真ん中よりに重心をとるためうまくいかない場合がある。

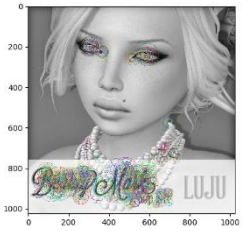


図 4.4 特徴点が文字を中心に検出されている画像

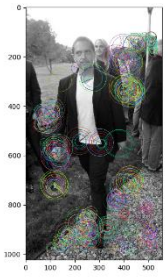


図 4.5 全体的に特徴点が検出できる画像

図 4.6 を提案手法により加工した画像

4.3 実験 2

提案手法では、インターネットからクエリ画像に類似した画像を収集して学習させることで目的画像を生成する。収集した画像が学習用として適切であるかを調べるため以下の実験を行った。

Content-Based Image Retrieval (CBIR)による類似画像検索を行い学習用データセットの作成を行った。提案手法ではインターネットから画像を収集するが、実験では鳥の画像構成されたデータセットである CUB_200_2011 から 1 万枚と、Open Images Dataset V4 の 1 万枚の画像データを合わせた、計 2 万枚の JPEG 形式の画像を検索用のデータベースとして使用した。クエリとなる 4 種類の鳥の画像は実験 1 の手法で正方形に変換し、図 4.7 に示す。同様にデータベースの画像も検索時に正方形への変換を行っている。画像検索ではクエリの画像と全ての画像で特徴量を比較し、類似度の高い画像を取得して学習用データセットとする。まず、類似度の高い画像の上位 5000 枚を取得し、検索手法の有用性を比較した。次に、取得枚数を上位 100,500,1000,2000,3000, 4000,5000 と区切り、その中に含まれる鳥の画像枚数の割合の推移を比較した。

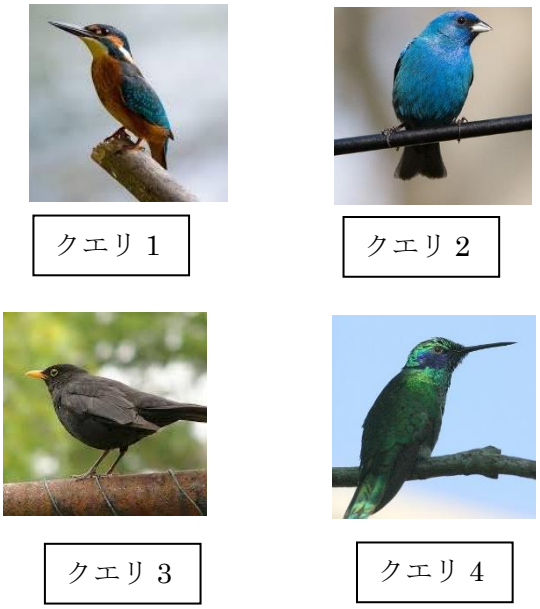


図 4.7 クエリ画像を正規化

4.3.1 実験結果 2

2 万枚の画像の中から類似度の高い 5000 枚を取得し、その中に含まれる鳥の画像の枚数を表 4.1 に示す。

表 4.1 5000 枚に含まれる鳥の画像枚数と割合

	クエリ 1	クエリ 2	クエリ 3	クエリ 4
ORB	2497(50%)	2756(55%)	2698(54%)	2657(53%)
A-KAZE	2748(55%)	2097(42%)	2316(46%)	2473(49%)
ヒストグラム	2979(60%)	3194(64%)	3421(68%)	2165(43%)

表よりヒストグラムによる検索手法が最も良い結果を示している。しかし、ヒストグラムでは画像によって結果に大きな差が生じている。これはヒストグラムでは画像に写る物体ではなく画像全体の色合いによる比較であるため、背景が白や水色といった 1 色の場合、物体の色よりも背景による影響が大きいと考えられる。したがってクエリ 4 のような画像では、画像全体が薄い色の画像が多く混ざってしまいこのような結果になったと考えられる。対して、クエリ 3 の結果が良いのは、黒に近い色の鳥は多く存在し、背景が森や木といった緑色である画像は鳥の画像が持つ特徴であると考えられるため、良い結果を示すのだと思われる。

局所特徴量である ORB、A-KAZE では、鳥の画像枚数は約 50%となってしまっている。データベース 2 万枚に含まれる鳥の画像は 50%の 1 万枚であるため、鳥の特徴を得ることができておらず、ほぼランダムに画像を取得してしまっていることがわかる。

次に総取得枚数ごとに分けて、鳥の画像の割合の推移を散布図にしたものを検索手法ごとに図 4.8、図 4.9、図 4.10 に示す。グラフよりすべての特徴量において右肩下がりとなり 50%に近付いていることがわかる。特に局所特徴量である ORB、A-KAZE では上位 2000 枚以降の取得枚数では、50%付近へ緩やかに近づいていることから画像の特徴を捉えることができず、ほぼランダムに画像を選んでしまっていることがわかる。対してヒストグラムでは 50%に近づいてはいるものの比較的高い割合を保てている。

また、局所特徴量である ORB、A-KAZE では、特徴を捉えることができている上位 2000 枚手前だと、クエリ 4 がともに最も良い結果を示している。ヒストグラムでも上位 100 枚なら良い結果が出ているが、そこから急激に下がっていることからも背景の色が単調だと目的の画像以外が混ざりやすいことがわかる。

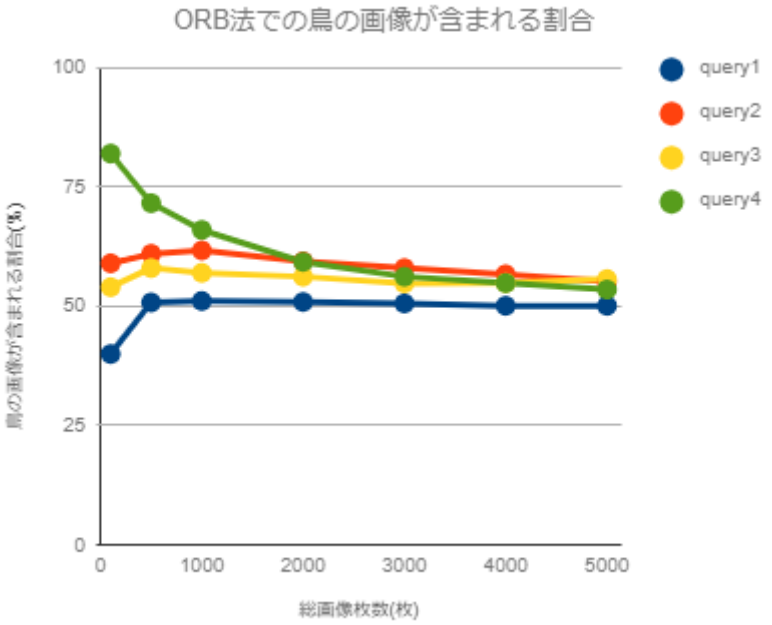


図 4.8 ORB 法での鳥の画像が含まれる割合

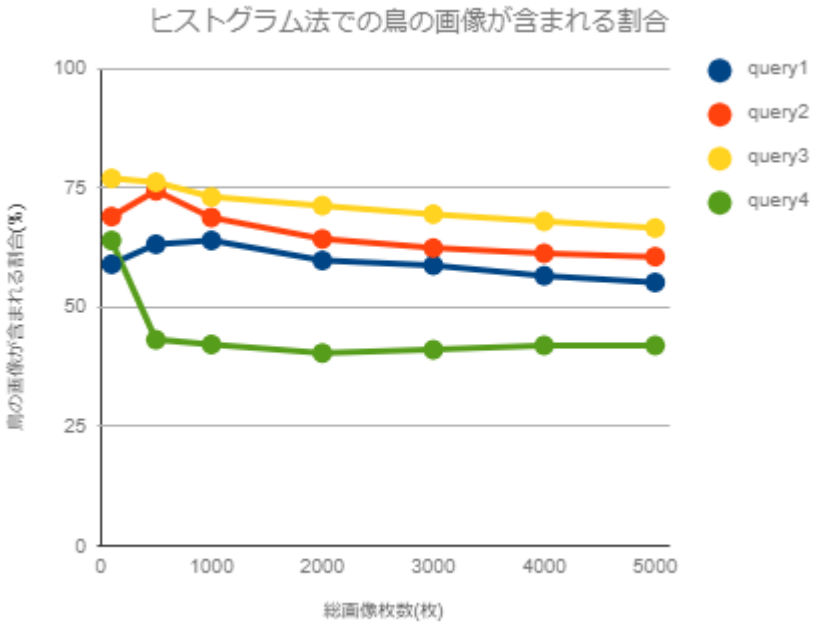


図 4.9 A-KAZE 法での鳥の画像が含まれる割合

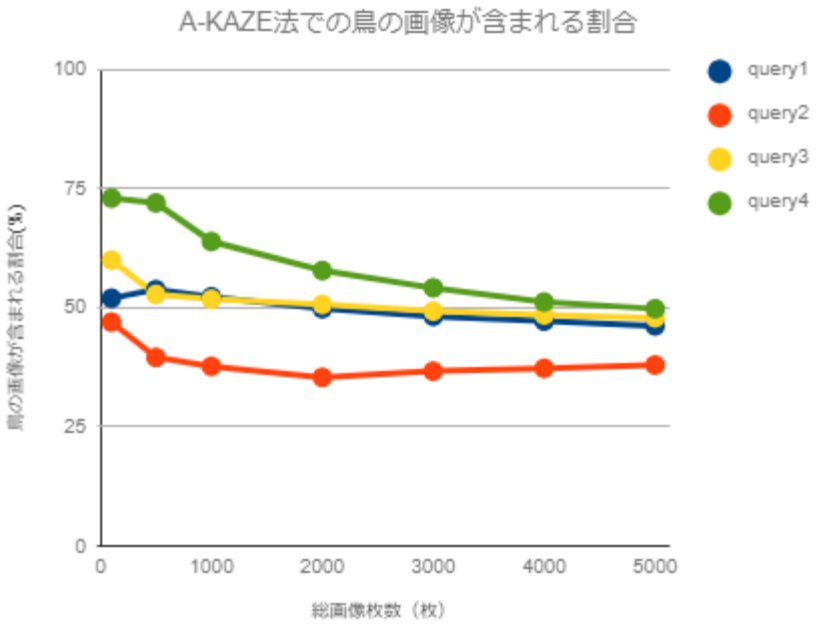


図 4.10 ヒストグラム法での鳥の画像が含まれる

4.4 実験 3

画像内の対象物の特徴をとらえる局所特徴と、画像全体の色合いをとらえる大域特徴を合わせることで、より鳥の画像の特徴を捉えることができるかを検証する実験を行う。複数の特徴量で取得した画像に共通する画像を調べ、その中に含まれる鳥の画像の割合を調べる。また、実験 2 と同様に取得枚数を上位 100、500、1000、2000、3000、4000、5000 枚と区切り、その中に含まれる鳥の画像の枚数を調べる。

4.4.1 実験結果 3

複数の特徴量の取得画像の中で、共通する画像とその中の鳥の画像枚数の比を表 4.2 に示す。表より、ORB とヒストグラムの組み合わせが最も高い結果となっている。また、局所特徴と大域特徴の組み合わせのほうが、同じ局所特徴である組み合わせよりも高いことから、局所特徴と大域特徴を組み合わせ、鳥の画像にみられる特徴と色合いの二つで判断したほうが、局所特徴を増やし鳥にみられる特徴を選んでいくより鳥の画像を多く選出できることが分かった。

しかし、複数の特徴量を組み合わせると、画像の総数が半分以下になってしまっている。ORB と A-KAZE の組み合わせでは鳥の画像の割合 50%を下回るが、共通している画像数は大域特徴と局所特徴の組み合わせより多い。

表 4. 2　複数の特徴量に共通する画像とその中の鳥の画像枚数比

	クエリ 1	クエリ 2	クエリ 3	クエリ 4	鳥の画像の平均枚数	共通画像数の平均
ORB と A-KAZE	50%	44%	49%	54%	1002	2020
ORB とヒストグラム	60%	72%	72%	50%	881	1402
A-KAZE とヒストグラム	59%	58%	67%	48%	858	1518
すべてに共通する画像	63%	67%	67%	53%	381	629

次に総取得枚数ごとに分けて鳥の画像の割合の推移を散布図にしたものを図 4.11、図 4.12、図 4.13 、図 4.14、に示す。図より、2000 枚までに含まれる鳥の画像の割合は高く、それ以降では鳥の画像の割合が一定になっていることが、実験 2 より際立って見える。しかし、ほとんどの検索結果において上位 1000 枚ほどの取得枚数のうち、共通する画像は 10%を下回っていた。

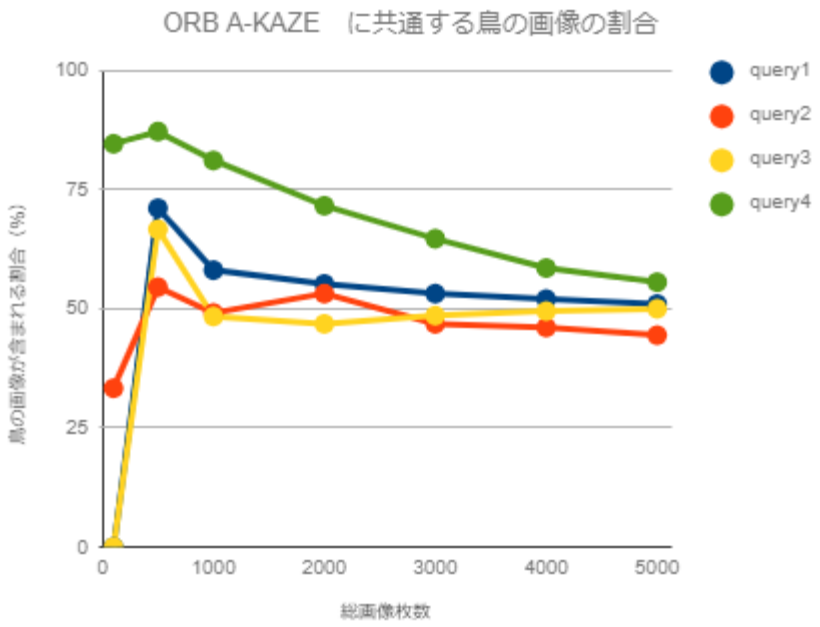


図 4.11 ORB A-KAZE に共通する画像とその中の鳥の画像の比

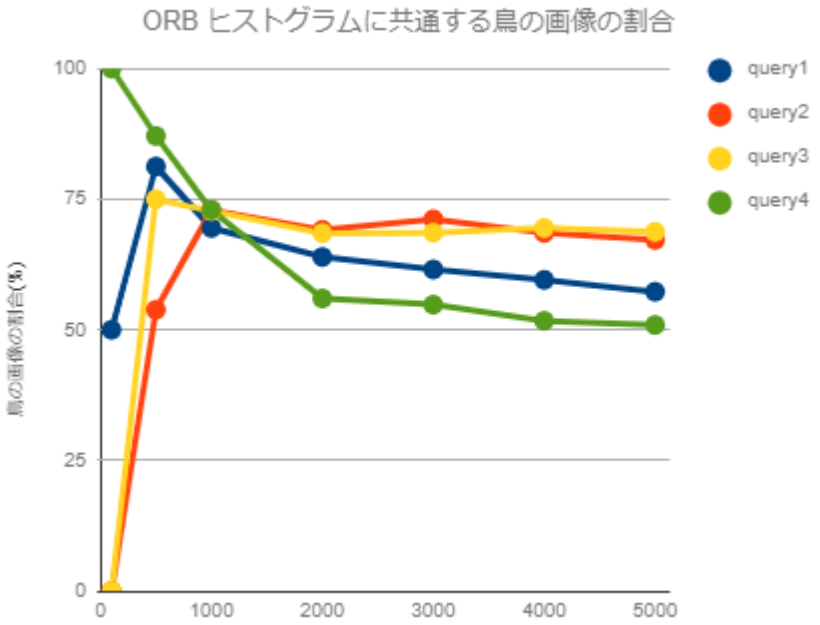


図 4.12 ORB ヒストグラムに共通する画像とその中の鳥の画像の比

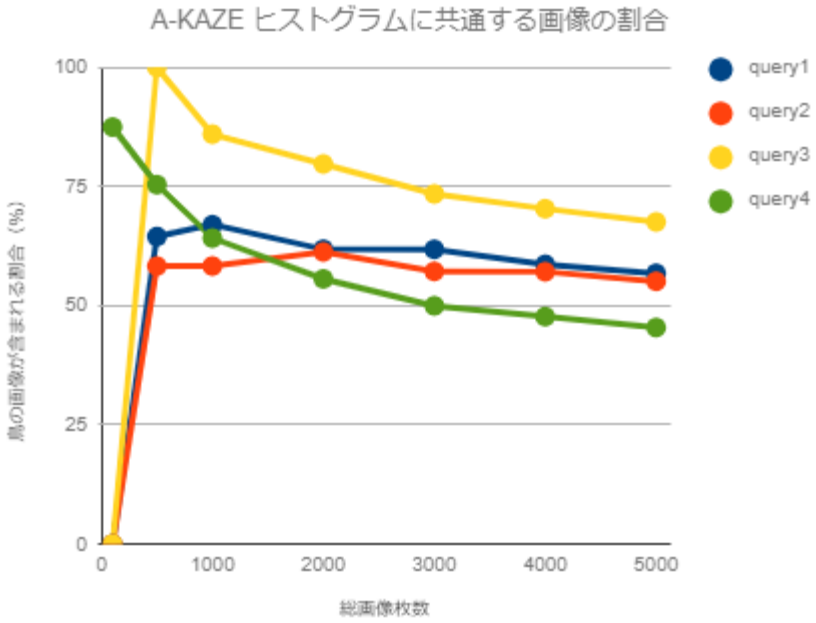


図 4.13 A-KAZE ヒストグラムに共通する画像とその中の鳥の画像の比

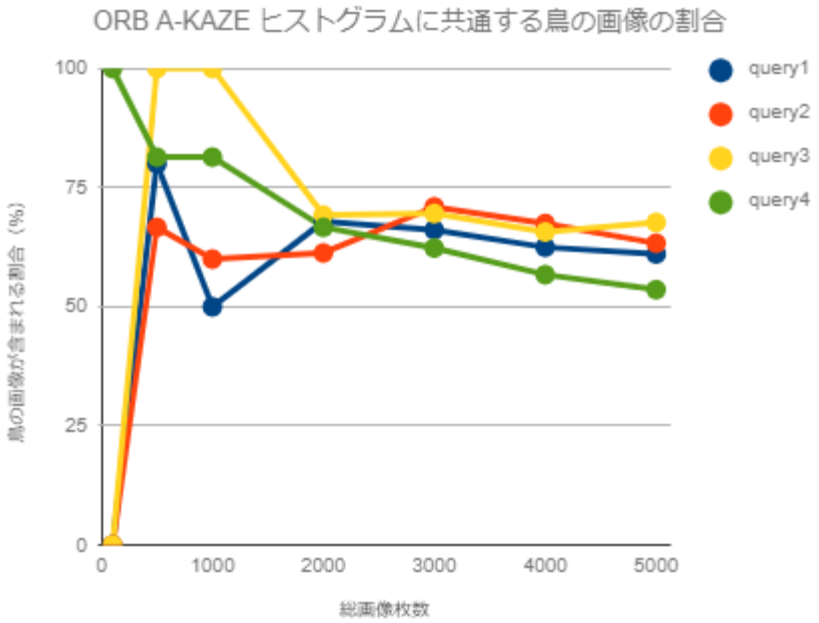


図 4.14 ORB A-KAZE ヒストグラムに共通する画像とその中の鳥の画像の比

第 5 章 考察

今回、画像生成用のデータを集めることを目的として、実験では鳥の画像で実験を行い、鳥の画像がどの程度含まれているかという指標で、画像を収集した結果の評価を行った。その結果、単一の手法を用いて画像を 5000 枚取得した結果 50%ほどしか鳥の画像が含まれておらず、また、複数の画像を用いた場合も 70%を下回るものばかりであった。質、量ともに画像生成用のデータセットとして使用するのは難しいだろう。

今回の実験では 2 万枚から検索により取得を行ったので類似度の高い上位 5000 枚までにしぼって実験を行ったが、本来の提案手法では何枚の画像から取得するかはわからないため、閾値を設けてその類似度より高い低いで画像を収集したほうが良いはずである。実験結果より閾値を考えると 1000 枚から 2000 枚の間あたりの類似度を参考に設けると画像特徴をよくとらえた検索結果になると考えられる。

また今回は収集した画像に含まれる鳥の画像数のみを画像生成用のデータセットの質として評価したが、鳥が映り込んでいない画像であっても、画像特徴によって類似すると判断された画像であれば、学習による画像生成に良い影響を与える可能性が考えられる。今後、今回収集した画像をもとに画像生成を行い、生成された画像でデータセットを評価する必要があると考える。

謝辞

本研究を進めるにあたり、多忙の中様々なご指導をいただきました三好 力教授に深く感謝いたします。また、研究において多くの助言をいただいた研究室に皆様に感謝します

参考文献

1)著作権法 | 国内法令 | 著作権データベース | 公益社団法人著作権情報センター CRIC,
http://www.cric.or.jp/db/domestic/a1_index.html#2_3e ,2018-8-20

2)内閣府　知的財産推進事務局,知的財産推進計画 2017,
http://www.kantei.go.jp/jp/singi/titeki2/kettei/chizaikeikaku20170516.pdf
,2018-8-20

3)内閣府　知的財産推進事務局　,AI に関して残された論点（討議用）,
<https://www.kantei.go.jp/jp/singi/titeki2/tyousakai/kensho_hyoka_kikaku/2017/johozai/dai6/siryou4.pdf,2018-8-22

4)吉原政幸,超図解　ビジネス mini ネット時代の著作権と意匠権,エクスメディア,2006
5) Twitter,Twitter サービス利用規約,
https://twitter.com/tos?lang=ja#yourrights,2018-10-30

6) Alexander Mordvintsev & Abid K,OpenCV3-Python チュートリアル,2018-11-25

7)Edward Rosten and Tom Drummond,(2006)"Machine learning for high-speed corner detection" in 9th European Conference on Computer Vision, vol. 1, 2006, pp. 430–443.

8)Wah C., Branson S., Welinder P., Perona P., Belongie S. "The Caltech-UCSD Birds-200-2011 Dataset." Computation & Neural Systems Technical Report, CNS-TR-2011-001.

9)Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, Yoshua Bengio,(2014) "Generative Adversarial Networks". arXiv:1406.2661

10)Alec Radford, Luke Metz, Soumith Chintala,(2015)."Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks". arXiv:1511.06434

11)島田直希,大浦健志 ,Chainer で学ぶディープラーニング, 技術評論社, 2007

付録

```
import cv2
import os
import shutil
import matplotlib.pyplot as plt
def main():
  IMG_NAME = [0]
  IMG_RET = [500]
  c_bird = 0
  bird = [0] * 5001
  total = [0] * 5001
  x_c = [0] * 5001
  m = 1
  mstop = 5
  while m < mstop:
   g = 1
   TARGET_FILE=
'/home/mjuser02/o_test/test'+str(m)+'.jpg'
    IMG_DIR =
 '/home/mjuser02/test_resize_20000/'
   bf= cv2.BFMatcher(cv2.NORM_HAMMING)
   # 特徴点算出のアルゴリズムを決定(コメントアウト
で調整して切り替え)
   detector = cv2.ORB_create()
   #detector = cv2.AKAZE_create()
   (target_kp,target_des) =
 calc_kp_and_des(TARGET_FILE, detector)
    files = os.listdir(IMG_DIR)
     for file in files:
      if file == '.DS_Store' or file ==
TARGET_FILE:
       continue
       comparing_img_path = IMG_DIR + file
         try:
          (comparing_kp, comparing_des) =
 calc_kp_and_des(comparing_img_path,
detector)
           #画像同士をマッチング
           if(len(target_des) <=
 len(comparing_des)):
            matches=
bf.match(target_des,comparing_des)
           else:
            matches =
bf.match(comparing_des,target_des)
            dist = [m.distance for m in matches]
            #類似度を計算する
            ret = sum(dist) / len(dist)
            i = 0
            stop = 5000
            while i < stop:
              if ret <= IMG_RET[i]:
               IMG_NAME.insert(i, file)
               IMG_RET.insert(i,ret)
               break
              i += 1
         except cv2.error:
          ret = 100000
      j = 0
      stop = 5000
      while j < stop:
        ret = 'bird' in IMG_NAME[j]
        if ret == True:
          c_bird += 1
          bird[j+1] = c_bird
          total[j+1] = j
        else:
          bird[j+1] = c_bird
          total[j+1] = j

          if j < 100:
           shutil.copyfile(IMG_DIR +
IMG_NAME[j],'/home/mjuser02/o_test/100_'+s
tr(m)+'/'+IMG_NAME[j])
           if j < 500:
            shutil.copyfile(IMG_DIR +
IMG_NAME[j],'/home/mjuser02/o_test/500_'+s
tr(m)+'/'+IMG_NAME[j])
           if j < 1000:
            shutil.copyfile(IMG_DIR +
IMG_NAME[j],'/home/mjuser02/o_test/1000_'+
str(m)+'/'+IMG_NAME[j])
           if j < 2000:
            shutil.copyfile(IMG_DIR +
IMG_NAME[j],'/home/mjuser02/o_test/2000_'+
str(m)+'/'+IMG_NAME[j])
           if j < 3000:
            shutil.copyfile(IMG_DIR +
IMG_NAME[j],'/home/mjuser02/o_test/3000_'+
str(m)+'/'+IMG_NAME[j])
           if j < 4000:
            shutil.copyfile(IMG_DIR +
IMG_NAME[j],'/home/mjuser02/o_test/4000_'+
str(m)+'/'+IMG_NAME[j])
           if j < 5000:
             hutil.copyfile(IMG_DIR +
IMG_NAME[j],'/home/mjuser02/o_test/5000_'+
str(m)+'/'+IMG_NAME[j])


          if j  > 0:
           x_c[j] = c_bird / j * 100
           g += 1

          if j > 0 and j % 100 == 99:
            print(j+1,c_bird)
            print(j+1, x_c[j])
            j += 1

      m += 1
      IMG_NAME.clear()
      IMG_NAME = [0]
      IMG_RET.clear()
      IMG_RET = [500]
      c_bird = 0
```

```
def calc_kp_and_des(img_path, detector):
  """
  特徴点と識別子を計算する
  :img_path: 画像のファイル名とパス
  :param detector: 算出の際のアルゴリズム"""
  IMG_SIZE = (200, 200)
  img=cv2.imread(img_path,
  cv2.IMREAD_GRAYSCALE)
  img = cv2.resize(img, IMG_SIZE)
  return detector.detectAndCompute(img,
   None)

if __name__ == '__main__':
  main()
```

```
import cv2
import os
import shutil
import matplotlib.pyplot as plt
def main():
  IMG_NAME = ['0']
  IMG_RET = [-2.0]
  c_bird = 0
  bird = [0] * 5001
  total = [0] * 5001
  IMG_SIZE = (200,200)
  m = 1
  x_c = [0] * 5001
  mstop = 5
  while m < mstop:
   g = 1
   TARGET_FILE =
'/home/mjuser02/o_test/test'+str(m)+'.jpg'
   IMG_DIR =
 '/home/mjuser02/test_resize_20000/'
   bf= cv2.BFMatcher(cv2.NORM_HAMMING)
   # 特徴点算出方法
   target_img = cv2.imread(TARGET_FILE,0)
   target_img  =  cv2.resize(target_img,
 IMG_SIZE)
   target_hist = cv2.calcHist([target_img],[0],
None,[256], [0,256])
   print('TARGET_FILE: %s' %
(TARGET_FILE))

   files = os.listdir(IMG_DIR)
   for file in files:
    if file ==' .DS_Store' or file ==
TARGET_FILE:
     continue
    comparing_img_path= IMG_DIR + file
    try:
     comparing_img =
 cv2.imread(comparing_img_path,0)
     comparing_img =
 cv2.resize(comparing_img, IMG_SIZE)
     comparing_hist = cv2.calcHist
([comparing_img],[0], None,[256], [0,256])
     #類似度を計算する
     result = cv2.compareHist(target_hist,
comparing_hist, 0)
     i = 0
     stop = 5000

     while i < stop:
      if result >= IMG_RET[i]:
       IMG_NAME.insert(i,file)
       IMG_RET.insert(i,result)
       break
       i += 1

    except cv2.error:
     ret = 100000

   j = 0
   stop = 5000
   while j < stop:
    ret =  'bird' in IMG_NAME[j]
    if ret == True:
     c_bird += 1
     bird[j+1] = c_bird
     total[j+1] = j

     else:
      bird[j+1] = c_bird
      total[j+1] = j
      if j < 100:
       shutil.copyfile(IMG_DIR                +
IMG_NAME[j],'/home/mjuser02/h_test/100_'+s
tr(m)+'/'+IMG_NAME[j])
       if j < 500:
        shutil.copyfile(IMG_DIR               +
IMG_NAME[j],'/home/mjuser02/h_test/500_'+s
tr(m)+'/'+IMG_NAME[j])
        if j < 1000:
         shutil.copyfile(IMG_DIR              +
IMG_NAME[j],'/home/mjuser02/h_test/1000_'+
str(m)+'/'+IMG_NAME[j])
        if j < 2000:
         shutil.copyfile(IMG_DIR              +
IMG_NAME[j],'/home/mjuser02/h_test/2000_'+
str(m)+'/'+IMG_NAME[j])
        if j < 3000:
         shutil.copyfile(IMG_DIR              +
IMG_NAME[j],'/home/mjuser02/h_test/3000_'+
str(m)+'/'+IMG_NAME[j])
         if j < 4000:
          shutil.copyfile(IMG_DIR             +
IMG_NAME[j],'/home/mjuser02/h_test/4000_'+
str(m)+'/'+IMG_NAME[j])
          if j < 5000:
           shutil.copyfile(IMG_DIR            +
IMG_NAME[j],'/home/mjuser02/h_test/5000_'+
str(m)+'/'+IMG_NAME[j])

          if j > 0:
           x_c[j] = c_bird / j * 100
          if j > 0 and j % 100 == 99:
           print(j+1,c_bird)
           print(j+1, x_c[j])
            j += 1

   m += 1
   IMG_NAME.clear()
   IMG_NAME = ['0']
   IMG_RET.clear()
   IMG_RET = [-2.0]
   c_bird = 0
```

```
if __name__ == '__main__':
  main()
```