ひとりで作る
画像検索システム

OLTA株式会社 山際 康貴

このセッションについて

セッションについて

- https://docs.google.com/presentation/d/1M4O9lQrrYpuA_daT3uUzoUBnJLg4wcJuaIeq-XhgSSk/ で資料公開中 https://bit.ly/3gy2jWu
- **メタ情報のない大量画像** を検索するアプリケーションを
  一人で作った ときの必要だった知見を共有します
- **得られるもの** : 画像検索システムを作るために必要な知識が手に入ります
- **話すこと** : 画像検索の背景と、実際のPythonコードの紹介
- **話さないこと** : データの集め方、画像検索の理論
- ※画像検索の研究者ではありません

目次

1. このセッションについて
2. 自己紹介
3. 画像検索の背景と仕組み
    a. 背景
    b. 画像検索の構成要素
    c. 特徴量の抽出
    d. 特徴量の検索
4. 画像検索チュートリアルの紹介
5. まとめ

自己紹介

@tamanobi

名前：山際康貴
所属：OLTA株式会社、武蔵野大学客員研究員、個人事業主
技術：Python, Go, PHP, Kubernetes, GCP
趣味：画像収集

画像処理との出会い：
ソニーのレンズ製造現場でレンズの傷検出@学生時代

画像大好きエンジニア

免責：画像処理の研究者ではありません

OLTA株式会社

- 1年前からPyCon JPのスポンサー
- プロダクトはどちらもPython（Django）
  - OLTA
    請求書を現金化するサービス
    中小企業や個人事業主の資金繰りをサポート
  - INVOY
    無料で使える請求書発行サービス
- ミッション「あらゆる情報を信用に変え、あたらしい価値を創出する」
- 累計32億円を調達している、40人程度のFinTech企業
- 詳しくはカルチャーデック

画像検索の背景と仕組み

画像検索の日常

- こんな感じ（見た目）の商品を探しているんですが、この店舗に置いていますか？
- 欲しい製品を見つけたが、売り切れだ。似た見た目のものはないかな？
- 青と白のストライプ柄の傘を忘れたんですが、届いていませんか？
- Twitterで見かけた漫画の1ページ、どの作品かわからない

画像を使った検索は、日常的

画像検索のニーズ

- 衣料品や家具をオンラインで購入する際、
  回答者の85%以上がそれぞれテキスト情報よりも視覚情報を重視
  （Visual Search Wins Over Text as Consumers’ Most Trusted Information Source）
- 55%の消費者が、ビジュアル検索は自分のスタイルや好みを開発するのに役立っていると回答
  （Upgrading Lens for more online to offline inspiration）

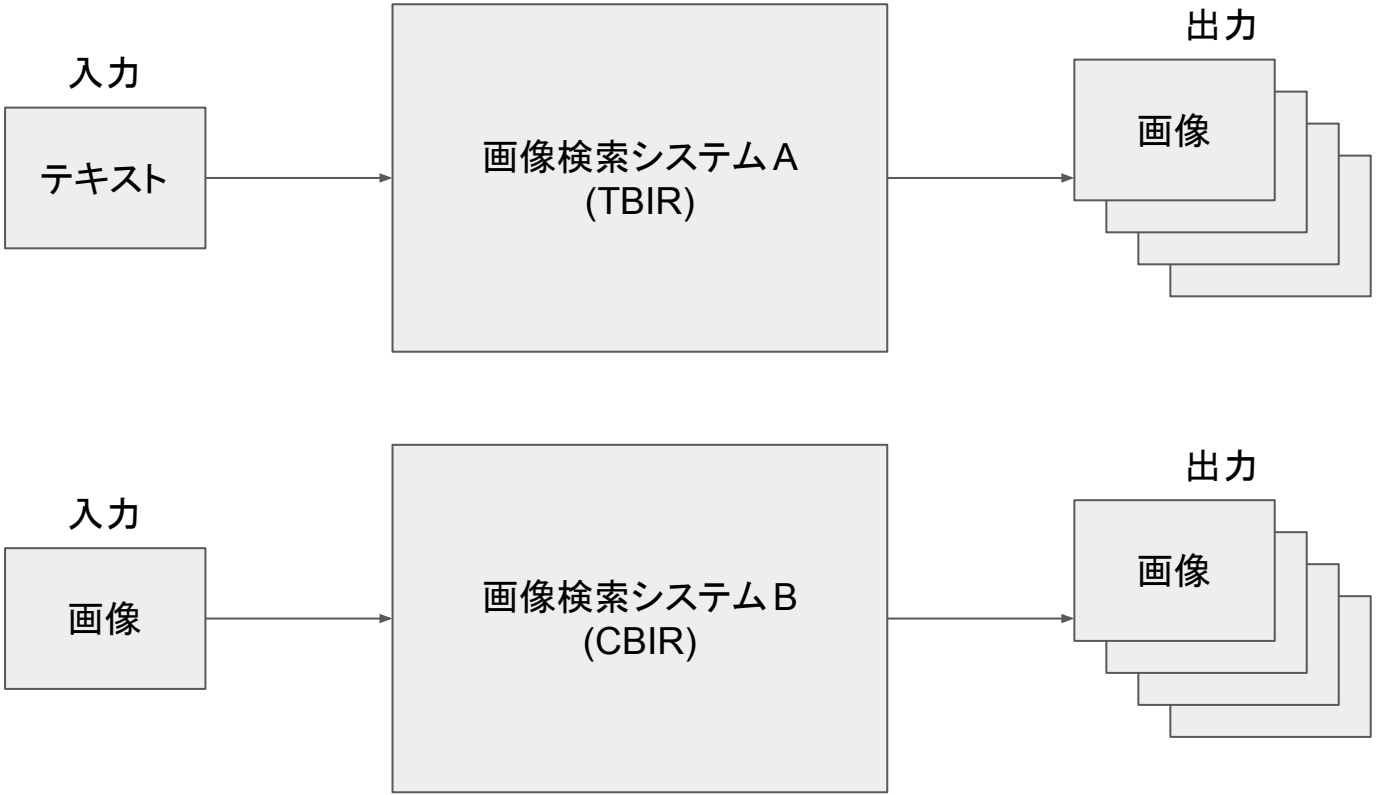
国内外の画像検索事情(ごく一部)

メルカリ	メルカリの写真検索を支えるバックエンド
ZOZO	類似アイテム検索機能についてGoogle Cloud Next '19 in Tokyoで技術発表をしました
Pinterest	Building Pinterest Lens: a real world visual discovery system
Sn	

画像検索は注目されており、
実プロダクトがそこそこ出ている

画像検索の仕組み

画像検索システムを2種類に大別

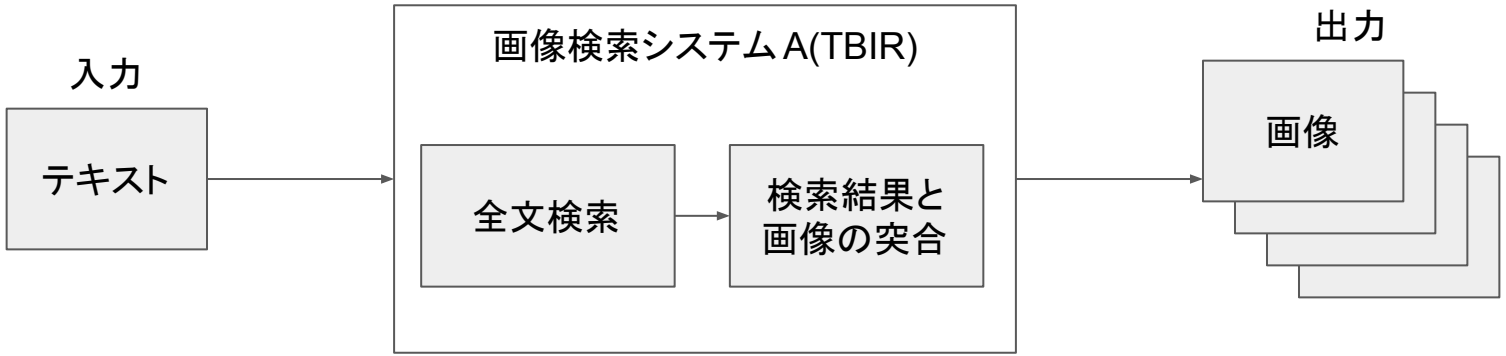


TBIR(Text Based Information Retrieval)

- テキストを入力として、画像(とメタ情報)群を得るシステム
- 画像と紐付けられた、テキスト情報を検索する
- テキスト情報が適切に設定されていないと検索できない

画像にテキスト情報が紐付いている場合

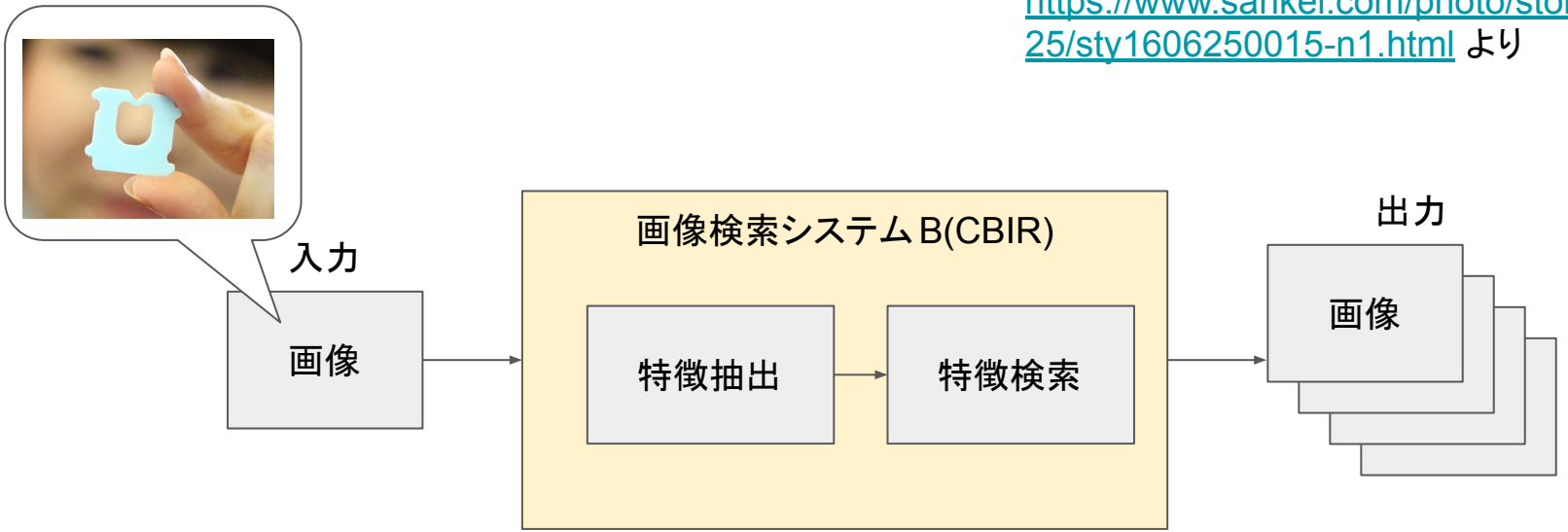
- TBIR = 画像にテキスト情報が紐付いている状態
- すでに良い製品が提供されている
  - 全文検索エンジン: Elasticsearch, Solr
  - 全文検索SaaS: Algolia



CBIR(Content Based Information Retrieval)

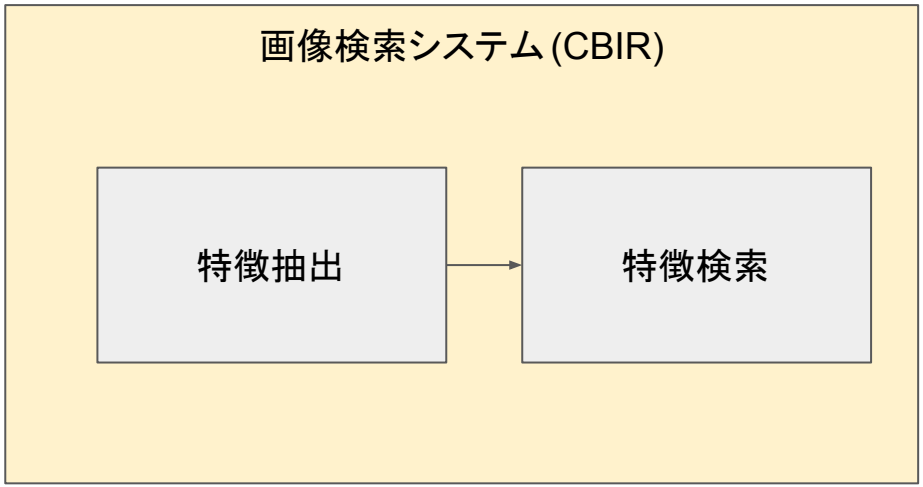
- 画像を入力として、画像(とメタ情報)群を得るシステム
- 入力画像を言葉で表せなくても利用できる
- **似た画像**を見つけ出すシステム

https://www.sankei.com/photo/story/news/160625/sty1606250015-n1.html より



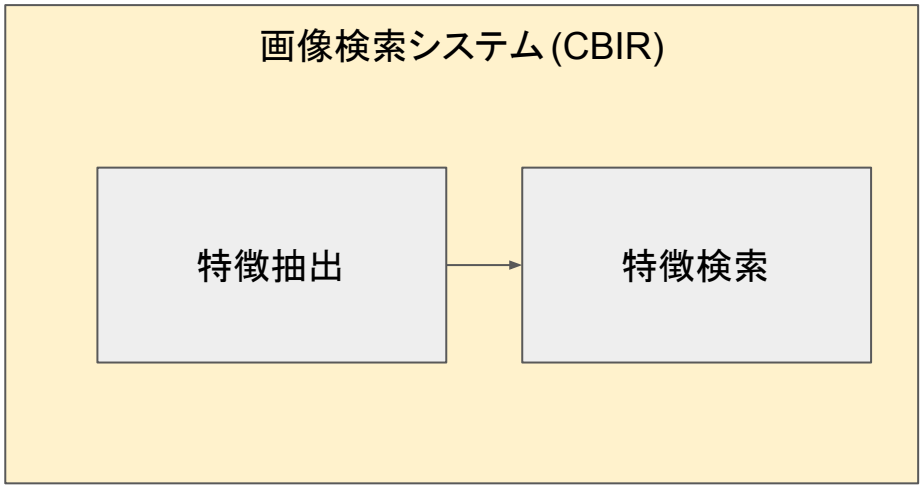
画像検索の肝

- 特徴抽出
- 特徴検索



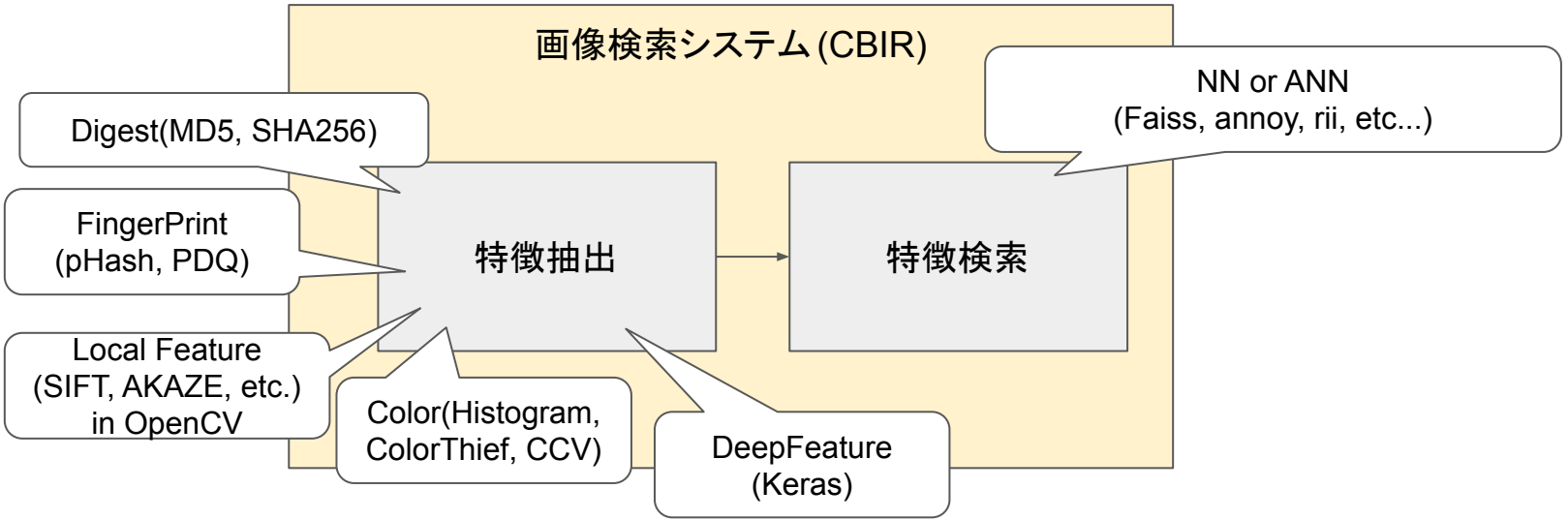
画像検索の肝

- 特徴抽出: 何を似ているとするか
- 特徴検索: いかに高速に似ている画像を見つけるか



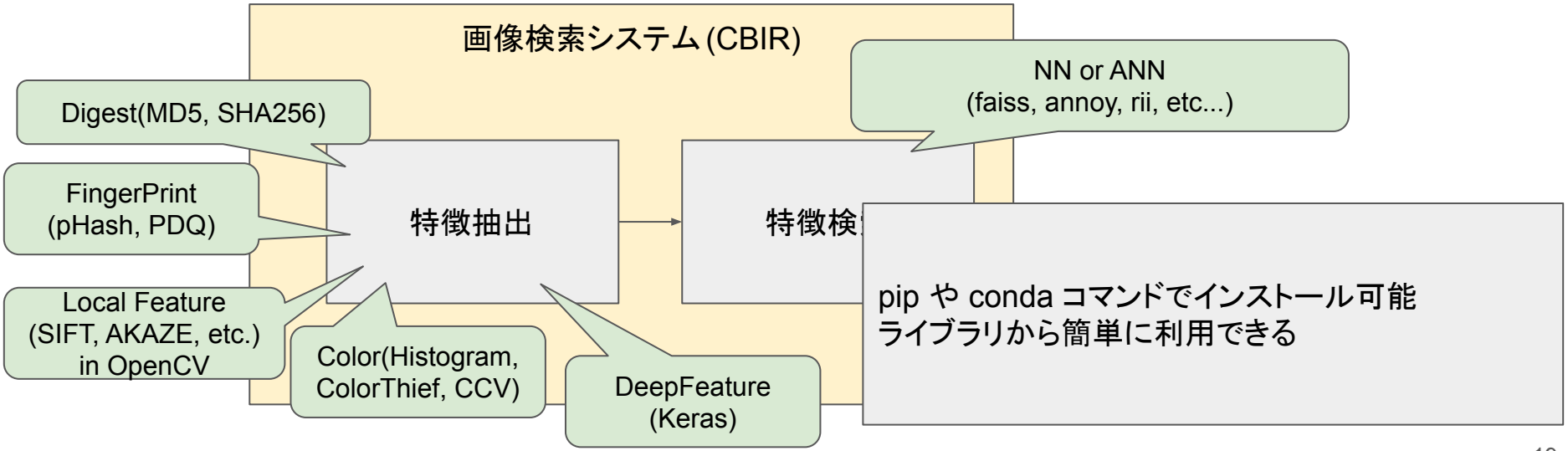
画像検索とPythonモジュール

- OpenCVやScikit-learn
- PyPIにも特徴量抽出、検索モジュールが多く登録されており、利用が簡単



画像検索とPythonモジュール

- OpenCVやScikit-learn
- PyPIにも特徴量抽出、検索モジュールが多く**登録されており、利用が簡単**



大規模でなければ、ひとりで作れる！

ひとりで作ったもの

どれもモジュールの組み合わせで作成

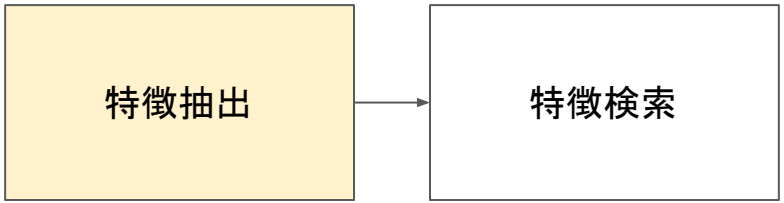
目的	権利侵害画像の発見
収集方法	とある会社保有データ
規模	100万枚
抽出	image-match, pHash
検索	Elasticsearch(image-match)

※権利関係上すべて一般非公開プロダクト

目的	画像の推薦
収集方法	10年で集めた画像＋クローラー
規模	300万枚
抽出	Keras(VGG中間層)
検索	annoy

目的	実験目的
収集方法	10年で集めた画像
規模	10万枚
抽出	Keras(VGG中間層), AutoEncoder
検索	Elasticsearch

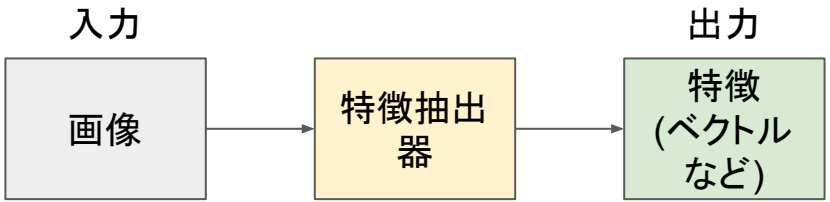
特徴抽出



特徴抽出: 画像をどう解釈するか？

- 800x600のRGB画像は、1.44M次元のベクトルとも解釈できる
  - N枚画像がD次元のベクトルになったとき、O(ND)
  - でかすぎて検索が現実的ではない
  - 1ピクセルの画素値ごとの似ている似ていないに関心がない場合が多い

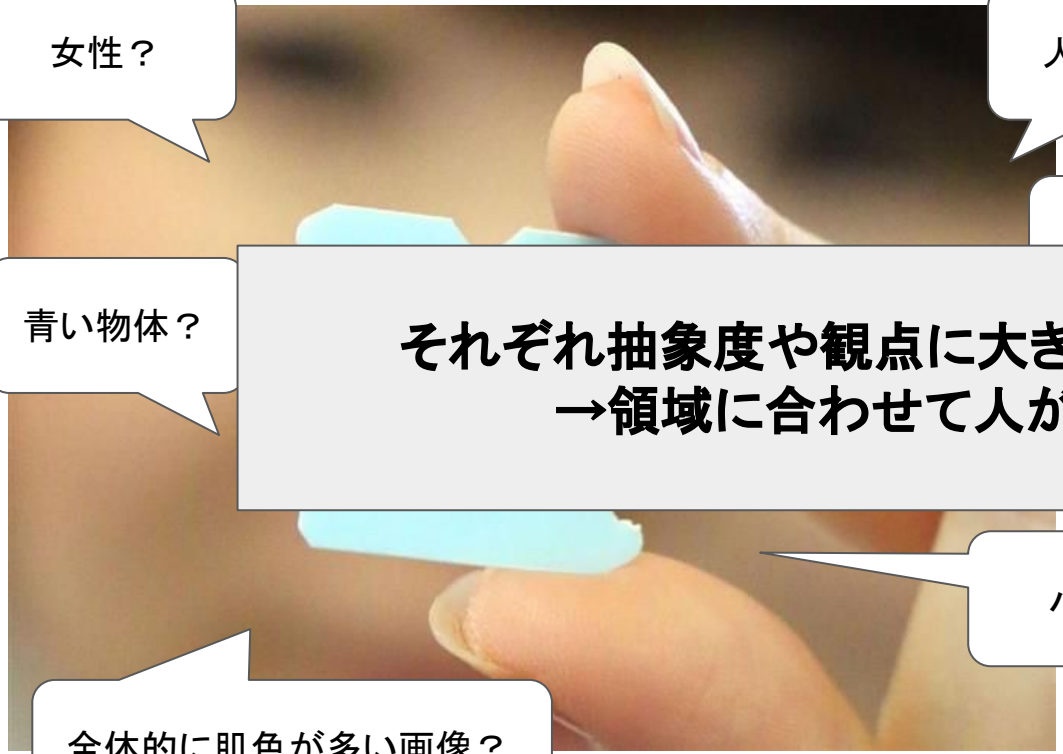
- 画像1枚からなんらかの方法で検索可能なベクトルに次元を落とす
  ＝特徴抽出



特徴抽出: 画像をどう解釈するか？

画像出典: https://www.sankei.com/photo/story/news/160625/sty1606250015-n1.html より

特徴抽出: 画像をどう解釈するか？



女性？

人の顔？

指？

横長の画像

青い物体？

それぞれ抽象度や観点に大きな違いがある
→領域に合わせて人が決める

バッグ・クロージャー？

全体的に肌色が多い画像？

0625/sty1606250015-n1.html より

特徴抽出: 似ているとは？

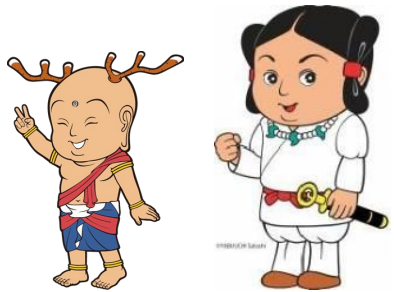
元画像は同
じっぽい

特徴の決め方で、似ている似ていないが決まる

画像出典
- https://www.sankei.com/photo/story/news/160625/sty1606250015-n1.html
- https://plus.amanaimages.com/items/FYI00226761?item_type=all&query=%E6%8C%87%E3%81%A7%E3%81%A4%E3%81%BE%E3%82%80&whole_price_zones=false
- https://withnews.jp/article/f0180926001qq000000000000000W00o10101qq000016896A

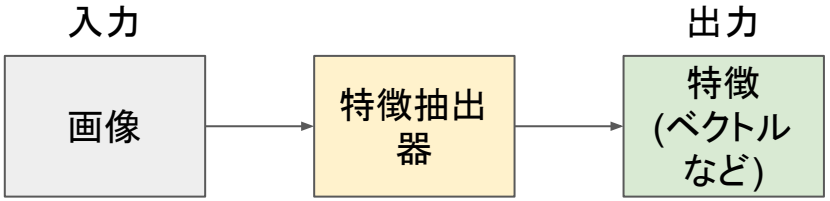
画像における類似

- 具体から抽象まで幅広い
  - 画素レベルの近い（ノイズ有り画像と、ノイズ無し画像）
  - 色合いが似ている（りんごと、苺）
  - 形や模様が似ている（ボールと、テニスボール）
  - とある図式が埋め込まれている（車載カメラの風景と、標識）
  - 同じ分野の物体が写っている（ダックスフント、チワワ）
  - 写っているものとは違うが、なんとなく似ている（せんとくんと、サカイタケルくん）



画像出典
・http://www.pref.nara.jp/36906.htmより
・https://www.city.sakai.lg.jp/kanko/hakubutsukan/sakaitakeru-kun.htmより

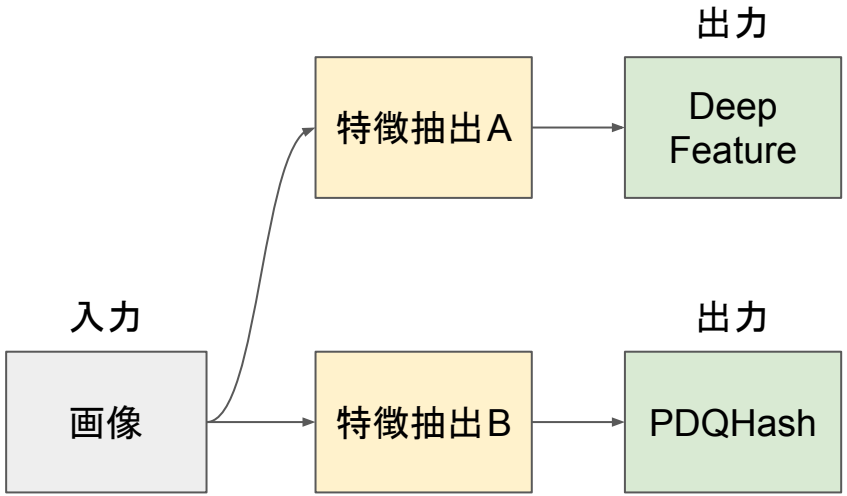
特徴の取り出し方



用途	抽出手法（例）	抽出された特徴量の形式
完全一致 重複検出	MD5, SHA256	バイナリ
画像が同一かどうか 犯罪性のある画像検出	PDQHash, pHash	バイナリ
色合い、色味	Color Histogram, Color Coherence Vector, 代表色	実数ベクトル
写っている物体 一般物体やドメインを絞った物体や特徴	ディープラーニングを使った特徴抽出 （DeepFeature）	実数ベクトル
ドメイン特化（後述）	ドメインに依る	ドメインに依る

実例：特徴を取り出す

- Google Colaboratory
  - 機械学習に便利なモジュールがインストール済みの環境
- デモ（特徴抽出から、特徴検索まで）
  - PDQHashの例
  - DeepFeatureの例



実例：特徴を取り出す（PDQ）

犯罪性が高い画像の指紋を得るために作成されたHash(256 bit)。
Facebookが作成。
[Open-Sourcing Photo- and Video-Matching Technology to Make the Internet Safer]

JPEG-PNG相互変換や修正（透かし、ロゴ）に強い。トリミングには弱い。

pip install pdqhashでインストール可能

```
import pdqhash
import cv2
import os
import numpy as np

def compute(img):
  hash, _ = pdqhash.compute(img)
  return hash

hash_list = [compute(img) for img in imgs]
```

実例：特徴を取り出す（DeepFeature）

- 画像分類をする分類器の
  中間表現をそのまま特徴とする
- 分類器が学習したデータセットによって
  得られる特徴が異なる
  分類器の対象ではない画像が入力されると、うまく特徴を取り出せないことがある
- Kerasがあれば、少しコードを書いて試せる

```
class FeatureExtractor:
    def __init__(self):
        base_model = VGG16(weights='imagene
        self.model = Model(inputs=base_model.i

    def extract(self, img):
        """
        Extract a deep feature from an input ima
        Args:
            img: from PIL.Image.open(path) or ten
        Returns:
            feature (np.ndarray): deep feature wit
        """
        img = img.resize((224, 224))  # VGG mus
        img = img.convert('RGB')  # Make sure ir
        x = image.img_to_array(img)  # To np.arr
        x = np.expand_dims(x, axis=0)  # (H, W, (
        x = preprocess_input(x)  # Subtracting a
        feature = self.model.predict(x)[0]  # (1, 4
        return feature / np.linalg.norm(feature)
```

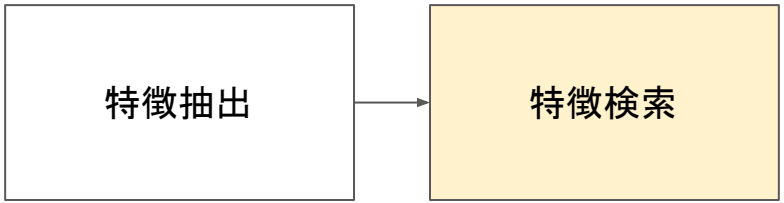
コード出典: https://github.com/matsui528/sis/blob/ad5d4bfd548c8b327ae9d87262e1ff6b2c66b7d7/feature_extractor.py

もう一つの特徴の取り出し方

- 画像中に写っているテキストが似ていれば、類似とみなす
  - 事例: OCRと全文検索を使えば画像検索が実装できるという話 | 株式会社NoSchool
  - OCRで取り出した文字列を全文検索に登録するため、
    全文検索エンジンに任せることができる
  - 記事ではGoogle の Cloud Vision APIでOCRしているが
    pytesseract 0.3.5 というPythonモジュールを使うこともできる（精度は期待できない）

画像→テキストができるなら
ベクトルやバイナリにこだわる必要はない

特徴検索



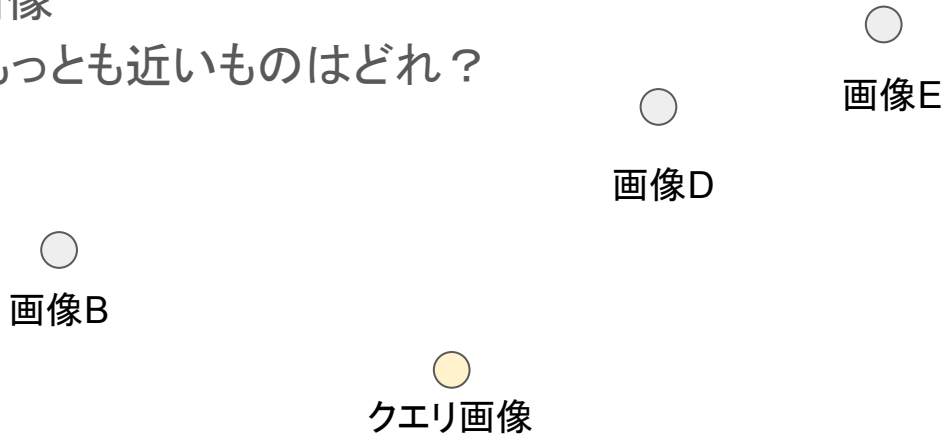
特徴検索：特徴の近さや遠さの計算方法

特徴量の種類によって、距離や類似度の計算方法が違う

特徴	種類	距離・類似度計算	例
PDQHash, pHash	バイナリ	ハミング距離 https://ja.wikipedia.org/wiki/%E3%83%8F%E3%83%9F%E3%83%B3%E3%82%B0%E8%B7%9D%E9%9B%A2	numpy.count_nonzero https://colab.research.google.com/drive/197q_34nsJrkYX5oXoE9tJYdt1CzhgXwV?usp=sharing#scrollTo=PI2t0vPsvPth
Color Histogram, Color Coherence Vector	ベクトル	ヒストグラム一致度 https://docs.opencv.org/4.4.0/d8/dc8/tutorial_histogram_comparison.html	cv2.compareHist https://colab.research.google.com/drive/1xQtv1EsyXj-8LlkO1afTIk5LuvrR3wEZ?usp=sharing
ディープラーニングを使った特徴抽出（DeepFeature）	ベクトル	ユークリッド距離, コサイン類似度 https://enjoyworks.jp/tech-blog/2242	numpy.linalg.norm https://colab.research.google.com/drive/1z3ohcLIY6UmIGPV9Uq6ImuyEsCTWlbGH?usp=sharing#scrollTo=RiBuWjqx_APq

特徴検索

- 点 = 特徴 = 画像
- クエリ画像 ともっとも近いものはどれ？



近傍探索（Nearest Neighbor Search）
というCS分野

特徴検索（小規模）

全件探索でOK=最近傍探索

N = 100,000件
D = 1,024次元程度で
numpy.argpartition　で0.40[sec]

動くコード:https://colab.research.google.com/drive/1dKfBNNmLkEcGjfiJMHKIto2ttoCLMS_7R?usp=sharing

```
import numpy as np

N = 100_000  # ベクトル数
D = 1_024  # ベクトルの次元数
top_k = 3  # 下位何件
features = np.array([
  np.random.rand(D) for i in range(N)  # ランダムなベクトル
])

query = np.random.rand(D)

import time
start = time.time()
dists = np.linalg.norm(features - query, axis=1)
ids = np.argpartition(dists, top_k)[:top_k]  # 距離が近いもののインデックス
process_time = time.time() - start


print(f"ids={ids}")
print(f"dists[ids]={dists[ids]}")
print(process_time)
```

```
ids=[50857 80652 57572]
dists[ids]=[12.20031129 12.20729024 12.21524199]
0.40415453910827637
```

特徴検索（大規模）

- 近似でいいからそこそこ近い近傍を取り出せばいい
  →近似近傍探索(Approximate Nearest Neighbor)が発展
- ライブラリが多数ある（https://github.com/erikbern/ann-benchmarks ）
  - annoy(spotify)
  - faiss(facebook)
  - NGT(Yahoo! Japan)
  - etc…
- 自作する必要はない

ライブラリの選び方や、理論背景は研究のサーベイを見よう

特徴検索（大規模）

ANNSライブラリを使う

登録→インデックス構築→検索の流れ

精度と速度を設定するパラメーター有

右の例は、annoy(Spotify製)

N=100,000, D=1,024で 1[ms]

動くコード:
https://colab.research.google.com/drive/132JCrRhAFZuIrj35gzbeME3gC8dcwikq?usp=sharing

```
# クエリとなるベクトルからtop_k件取り出す
query = vectors[0]

import time
start = time.time()
# 距離が近いものの距離とインデックスを取得
ids, dists = u.get_nns_by_vector(query, top_k,
                                 include_distances=True)
process_time = time.time() - start

print(f"ids={ids}")
print(f"dists={dists}")
print(process_time)
```

```
ids=[0, 66431, 43721]
dists=[0.0, 12.34458065032959, 12.403461456298828]
0.0010340213775634766
```

大規模な検索に使うライブラリの選び方

- [CVPR20 Tutorial] Billion-scale Approximate Nearest Neighbor Search
  - 近傍探索の理論的背景から、近似近傍探索ライブラリの比較
- 東京大学 講師の 松井 勇佑 さんが発表
  - KerasとFlask を用いた画像検索のチュートリアルを公開　Image Retrieval in the Wild

部分集合を対象にした近似近傍検索は少ない

- 「かばんカテゴリから探したい」を単独で達成するANNSはあまりない
  - Riiができる（松井さんが作成）
  - Milvusが今後サポート予定
- Elasticsearch のfilterとQuery DSLを使って近傍探索を実現できる
  - Elasticsearchにはメタデータを入れることができる
  - メタデータで絞り込んだあとに、近傍探索するという発想
  - 近傍探索の実現方法
    - Script score query | Elasticsearch Reference [7.x]
    - cosineSimilarity, l2normなど

検索まとめ

- 1万件程度でメモリに乗るなら、numpyでも簡単
- 約100万件以上ならANNSが必要。
  PyPIなどにライブラリがあるので自作不要
- ANNSの選び方は、松井さんの発表スライドを参考

画像検索のチュートリアル紹介

画像検索のチュートリアル紹介

- 画像検索のチュートリアルがいくつかあるので、紹介します。
- Image Retrieval in the Wild
- Milvus × VGG: Building a Content-based Image Retrieval System
- EdjoLabs/image-match: Quickly search over billions of images
- Visual Wordsを用いた類似画像検索
-

[時間があれば] これまでの知識を使って画像検索

- DeepFeatureとannoyを利用した画像検索をColaboratoryで再現
- https://colab.research.google.com/drive/158mTE5a7ZXV5Fa9rdmbq77Rqr_a7c89k

まとめ

まとめ

- 画像検索の需要は高まっている
- 画像検索は、特徴抽出と検索にフェーズが分かれる
  - 見つけ出したい画像やドメインによって特徴抽出の方法は違う
  - 大規模な検索には、ANNSライブラリが必要になる
- どちらもPythonモジュールが提供されており、簡単に扱える

画像検索システムを作ってみてください

おわり

参考資料

- PhotoDNA
- メルカリの写真検索を支えるバックエンド
- 類似アイテム検索機能についてGoogle Cloud Next '19 in Tokyoで技術発表をしました
- ビジュアル検索のトレンドがまとめられている
  - Visual Search – The Ultimate Guide: Statistics, News, Trends, and Tips
- Pinterest Lens
  - Building Pinterest Lens: a real world visual discovery system