画像解析の精度向上方法の模索

東京情報大学　総合情報学科　情報システム学系
知能情報システム研究室　永井ゼミ

目次

使用ソフトウェア

- 開発手順
- 検証手順
- 研究計画

研究に使用する言語・環境・ソフトウェア

- Python
  - 無料で使うことができて機械学習用の素材が大量に使用可能である
- **TensorFlow**
  - 上記で動作するオープンソースな機械学習ライブラリ
- Keras
  - 追加

画像解析のあれこれ

- 現代では機械が写真などを自動的に分類するという技術は各サービスで一般的に提供されている機能である
- 同じ写真を探すだけではなく自身が撮影した写真に自動的にタグやタイトルを付けるサービスもある
- どのサービスも基本的には精度は高く明確なデータを与えた場合9割以上は正確である、しかしごく僅かな機械ゆえの誤検出が問題となっている

- Google社は写真分類サービスで黒人とゴリラを誤検出する問題を起こしゴリラというタグ自体を削除して再発を強制的に封じた
- また、男女の検出を誤るとAIが性差別をするという観念からポリコレに配慮して男性/女性というタグを廃止した

- 人が見れば知識から判別できるものも似ているからという理由で誤るのが現在の画像解析事情である

先行研究たち

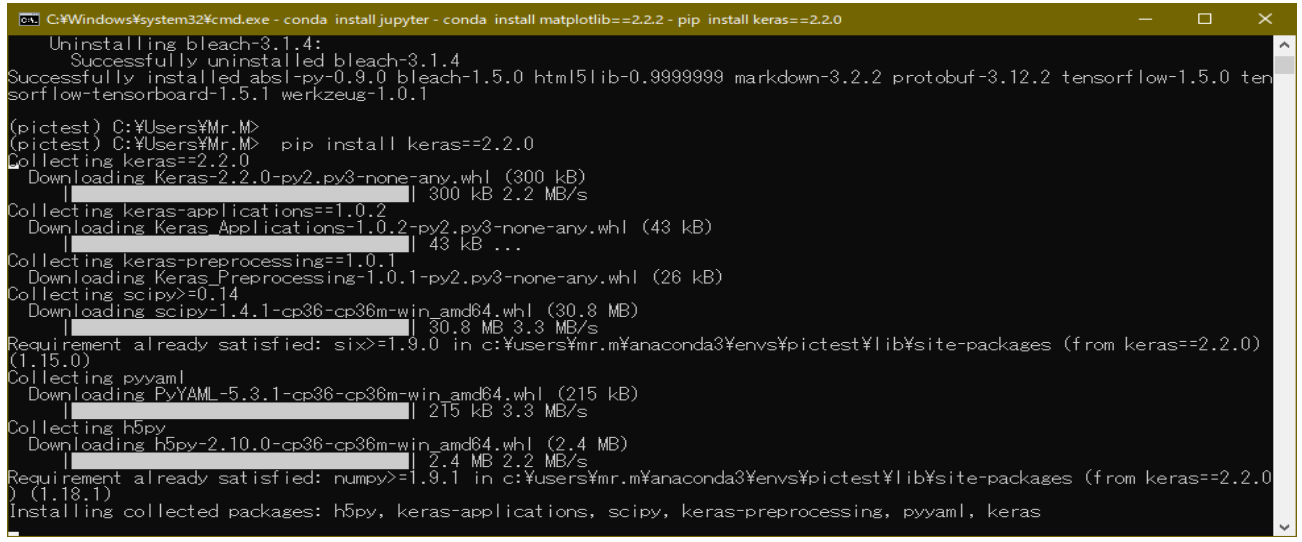
- Google Vision API
  - クラウド上で機械学習モデルを提供するGoogle社のサービス
  - ハイコントラストな画像や物体の輪郭がはっきりしている写真の判別は正確だがピントをずらした風景画や砂浜などの単一背景のような色がはっきりしないものは認識しづらい
- Apple photo
  - 写真の分類をするやつ

画像の学習と判定方法/丙

- A・手動でタグ付けした教師データを元にカテゴリ分類する
  - 画像全体の近似度で判定させる
  - A2・精度向上のために大量のデータを確保するかタグを枝分かれさせる
- B・被写体の輪郭で判定させる
  - グレースケール変換を行い輪郭を強調させた状態で判別させる
- C・被写体の色調で判定する
  - 予め正解のデータを大量に格納した辞書を作り最も近いものを解とする

カテゴリ分けAI作成

- tensorflow・Keras・pillowを入れた仮想環境を構築
- コードの中身はGitから持ってきたいくつかの使えそうなやつを結合したやつなので画像解析や学習の過程を追々勉強して卒論に乗せる



画像の学習

- 各カテゴリごとに手動で振り分けた教師データを用いて学習させる
- 現段階では教師データに用いる画像は対象が画像中央にあるフルカラーのPNGデータ限定で入れてある

```
Train on 1 samples, validate on 1 samples
Epoch 1/1
1/1 [==============================] - 0s 280ms/step - loss: 2.1653 - acc: 1.0000 - val
_loss: 2.4864 - val_acc: 0.0000e+00

_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_5 (Conv2D)            (None, 32, 32, 32)        896
_________________________________________________________________
activation_6 (Activation)    (None, 32, 32, 32)        0
_________________________________________________________________
conv2d_6 (Conv2D)            (None, 32, 32, 32)        9248
_________________________________________________________________
activation_7 (Activation)    (None, 32, 32, 32)        0
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 16, 16, 32)        0
_________________________________________________________________
dropout_4 (Dropout)          (None, 16, 16, 32)        0
_________________________________________________________________
conv2d_7 (Conv2D)            (None, 16, 16, 64)        18496
_________________________________________________________________
activation_8 (Activation)    (None, 16, 16, 64)        0
_________________________________________________________________
conv2d_8 (Conv2D)            (None, 16, 16, 64)        36928
_________________________________________________________________
activation_9 (Activation)    (None, 16, 16, 64)        0
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 8, 8, 64)          0
_________________________________________________________________
dropout_5 (Dropout)          (None, 8, 8, 64)          0
_________________________________________________________________
flatten_2 (Flatten)          (None, 4096)              0
_________________________________________________________________
dense_3 (Dense)              (None, 512)               2097664
_________________________________________________________________
activation_10 (Activation)   (None, 512)               0
_________________________________________________________________
dropout_6 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_4 (Dense)              (None, 10)                5130
=================================================================
Total params: 2,168,362
Trainable params: 2,168,362
Non-trainable params: 0
_________________________________________________________________

Test loss: 2.304145097732544
Test accuracy: 0.0
```

学習

- タグ付けした画像を適量用意して学習させる、精度向上のために同一画像を反転・回転させたものを用意して学習させる
- なるべく被写体が中央に映っている画像を用意する
- 解像度や被写体の鮮明さは揃えることなく用意する

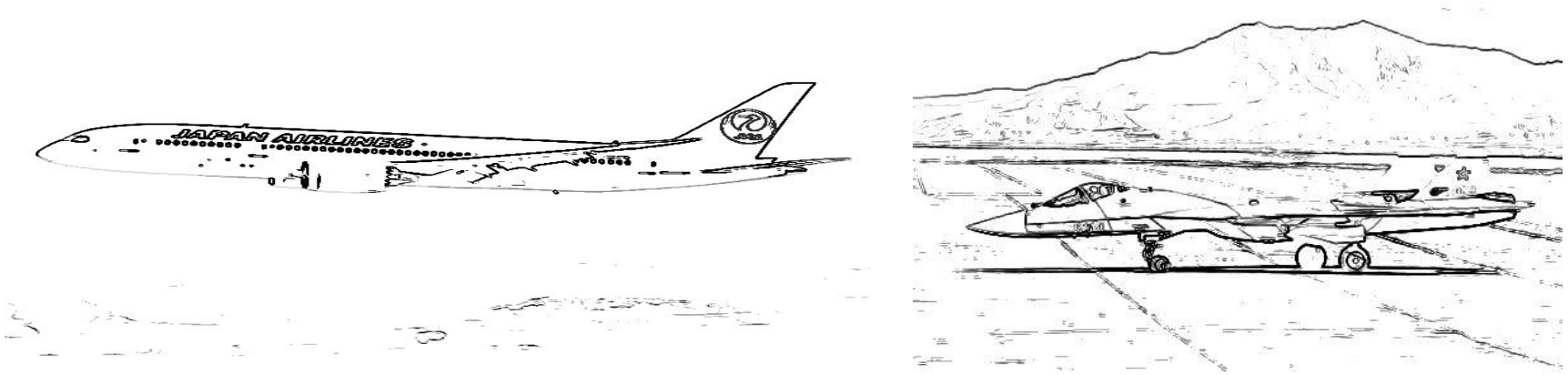
タグ：Aircraft

- この学習方法を用いた場合は同型機や同じようなロケーションで飛行している航空機はおおよそ航空機と判別できた
- しかし同型機かつ同じように駐機されているが背景と同化している、迷彩が施されている、モノクロの写真である場合などは判別精度が低かった
- 人間の目では似ているので航空機の写真であると判別可能だがこのAIは色調を比べてしまうので判別精度がめぐるしく下がってしまった

風景と誤認・航空機の輪郭が潰れて見えない

改善方法

- 誤認識の改善方法として風景の色調などを取り去って被写体の輪郭のみを強調する前処理を写真に行った
- それと同様に判別する写真も輪郭のみにした



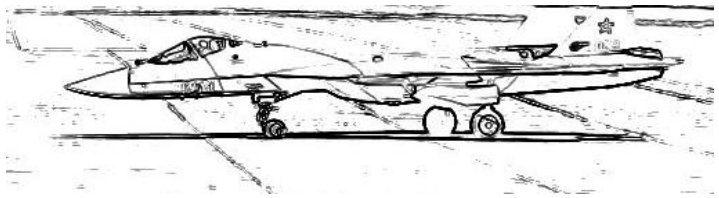
- このような前処理をすることにより被写体が撮影された環境が晴天・曇り・夜間などに関わらず鮮明に判別可能となった
- また機体に描かれた航空会社の塗装・ロービジ迷彩も平坦化することができる

- このように輪郭だけを残すことで判別精度は大いに向上した

```
def main():
    # ラベル
    folder = ["000", "001", "002", "003"
          , "004", "005" , "006", "007" , "008", "009" ]

    # 画像の読み込み（32×32にリサイズ）
    img = load_img("C:/Users/Mr.M/Desktop/AI/pictest/001.png", target_size=(32, 32))
    ## tomato
    img = img_to_array(img)


    # 画像データの正規化
    img = img.astype('float32')/255.0
    img = np.array([img]) # 4次元配列にしないと入力できない

    # モデルの読み込み
    model = model_from_json(open('cifar10.json', 'r').read())

    # 重みの読み込み
    model.load_weights('cifar10.h5')

    # コンパイル
    model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])

    # 10枚（0～9番目）のテスト画像を入力し、予測結果を出力
    y_pred = model.predict(img)

    # 出力をクラスベクトルから整数値に変換
    y_pred = np.argmax(y_pred, axis=1)

    # 予測結果の表示
    print('予測結果：', folder[int(y_pred)])

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

```
予測結果： 002
```

```
In [ ]: 002 風景
```



```
    # モデルの読み込み
    model = model_from_json(open('cifar10.json', 'r').read())

    # 重みの読み込み
    model.load_weights('cifar10.h5')

    # コンパイル
    model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])

    # 10枚（0～9番目）のテスト画像を入力し、予測結果を出力
    y_pred = model.predict(img)

    # 出力をクラスベクトルから整数値に変換
    y_pred = np.argmax(y_pred, axis=1)

    # 予測結果の表示
    print('予測結果：', folder[int(y_pred)])

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

```
予測結果： 004
```

```
In [ ]: 004 航空機(輪郭検出)
```

- AI画像判別はネット上の画像管理やフィルタリングにも使用されている。
- しかし砂漠や砂浜などの滑らかかつ肌色が多い画像などがヌードやポルノと誤検出される問題が多発している、この問題も輪郭強調で対処した

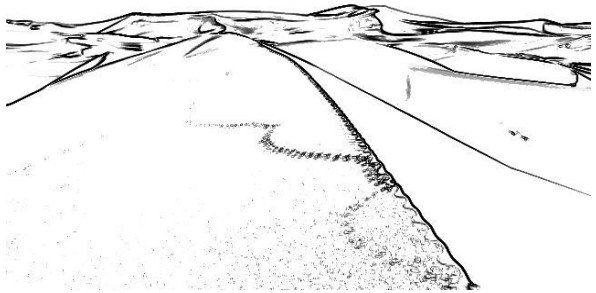
```
    # 出力をクラスベクトルから整数値に変換
    y_pred = np.argmax(y_pred, axis=1)

    # 予測結果の表示
    print('予測結果：', folder[int(y_pred)])

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

予測結果： 003

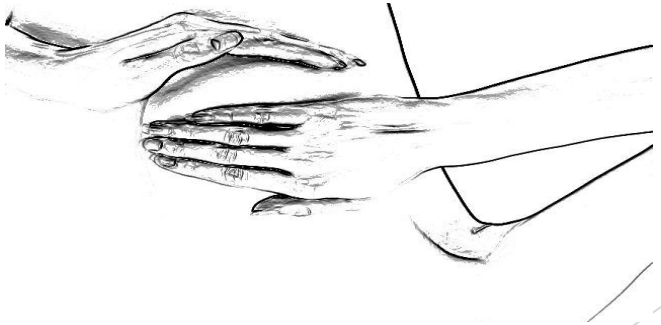
```
In [ ]: 003  ヌード写真集
```

- 航空機と同じような手法で画像の前処理を行った
- しかし、砂漠や砂浜の作り出す波模様となだらかな輪郭は人間の輪郭と似ているようで服を着ていない女性などのヌード画像と区別することが難しかった

- また、被写体が平坦すぎるために輪郭を検出しても判別の鍵となる特徴が見つけづらいため、新たな手法を考えなければいけない。



≠



テスト結果

- 現在はテスト段階なのでラベル(カテゴリ)割当は仮番号である

　　各カテゴリ20枚～ほどの画像を学習させたところ、人が見て似ていると思われる画像であれば正しい判定が返ってくるので正常に動作していると思われる

```
def main():
    # ラベル
    folder = ["000", "001", "002", "003"
          , "004", "005" , "006", "007" , "008", "009" ]

    # 画像の読み込み（32×32にリサイズ）
    img = load_img("C:/Users/Mr.M/Desktop/AI/pictest/003.png", target_size=(32, 32))
    ## tomato
    img = img_to_array(img)



    # 画像データの正規化
    img = img.astype('float32')/255.0
    img = np.array([img]) # 4次元配列にしないと入力できない

    # モデルの読み込み
    model = model_from_json(open('cifar10.json', 'r').read())

    # 重みの読み込み
    model.load_weights('cifar10.h5')

    # コンパイル
    model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])

    # 10枚（0～9番目）のテスト画像を入力し、予測結果を出力
    y_pred = model.predict(img)

    # 出力をクラスベクトルから整数値に変換
    y_pred = np.argmax(y_pred, axis=1)

    # 予測結果の表示
    print('予測結果：', folder[int(y_pred)])

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

```
予測結果： 000
```

問題と改善点

- 現段階ではサンプル数が少ないことと教師データの選定からか学習精度が低く物体の輪郭などよりも色合いで判断されてしまうことが多いのでそれを回避する方法を考える。
- また、仕様上例外(学習させたカテゴリ以外)を無理やりどれかのカテゴリに振り分けてしまうので例外データを弾けるようにもする。

※カテゴリには含まれていない野菜:トマトを判定させたところ最も色合いが似ている写真が多かったカテゴリ2:車と判定された。
(学習に用いたマツダ車は赤色車が多いため)

開発手順

- 画像全体解析・カテゴリ判別・教師データに基づく成分分析など複数のアレから最適な手法を考える
- 学習させて得られたデータの精度を検証する
  - 追記・学園祭までに完成させてアンケートをとることにより精度の検証を行う
- 数種類のAIを比較して機能別に最適な学習方法を検証する
  - 追記・数種類の比較は時間的に余裕がなさそうなので精度向上の面を重点的にする計画に変更

画像の選定　学習データの調整を繰り返して精度を向上させる

開発計画

- 現在の実験結果を精査しつつ項目ごとの実験をブラッシュアップしつつ卒業論文として纏める作業にかかる。