顔認識 AI とカスケード分類機の関係性

松坂高校 A　藤本雄大・中井千駿

1. はじめに

近年、自動運転が可能な乗用車や iOS 端末に搭載された音声アシスタントの Siri などといった我々の生活において人工知能 (以後 AI) の需要が急激に増加していると考え、我々は本校で導入されている SSH(スーパーサイエンスハイスクール) の一環としてマイクロコンピューターの RaspberryPi3 Model B+ を使った顔認識 AI 作成し、その AI が顔を認識しているのを確認することができた。しかしその時 AI に学習させた顔の定義となったデータはインターネットにて配布されているものであり、自分たち自身で作ったものではなかったので、自分たちで AI に学習させるデータを作り、それとの関係性を知りたいと考え、本実験を開始した。

2. 実験方法

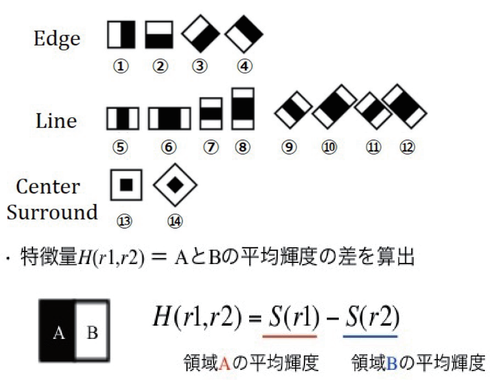
(1) 実験機材

	PC(深層学習用)	PC(マイコン操作用)	RaspberryPi
CPU	Core i7 8700K	Core i5 4258U	BCM2837
実装 RAM	32GB	8GB	1GB
OS	windows10 Pro	MacOS 10.15.3	Raspbian

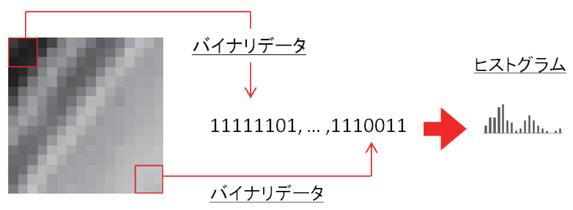
プログラミング言語：C++　　コンパイラ：g++　　ライブラリ :OpenCV

(2) カスケード分類機とは

カスケード分類機とは今回実験に使う Intel 社の OpenCV という Python、C++ 等に向けて開発された画像認識ライブラリにおいて AI が対象物かを判別するために基準となるファイルである。
画像認識 AI に学習させるカスケード分類機を作るにはポジティブデータ (認識させたい物が写っている画像) とネガティブデータ (認識させたい物が写っていない画像) が必要となる。
検出アルゴリズムには物体の局所的な明暗差の組み合わせにより、画像を判別する Haar-Like 特徴量と物体の局所的な輝度の分布の組み合わせにより、画像を判別する LBP（Local Binary Pattern）特徴量などがある。



(図 3)Haar-like 特徴量



(図 4)LBP 特徴量

(3) 計測方法

OpenCV の公式ドキュメントによると一般的に使われている画像認識 AI 並の精度のカスケード分類機を作ろうとするとポジティブデータが 7000 枚、ネガティブデータが 3000 枚必要らしいがデータを集めるのにとても時間がかかるので今回は最大ポジティブデータ数を 1500 枚、最大ネガティブデータを 2000 枚集めてデータの枚数や学習回数を変更して Haar-Like 特徴量と LBP 特徴量の二つのアルゴリズムにおけるカスケード分類機の違いを計測した。
AI に実験者の顔を学習させ学習データを元に実験者の顔を認識出来るようになるのを目標にした。
AI に学習させるための正解画像として 1500 枚の実験者の顔の画像を集めるのは大変なので GitHub で配布されている顔の定義が記載されたのカスケード分類機を AI に読み込ませ、実験者の顔を検知した場合、検知した顔をトリミングした画像を保存しファイルに顔の高さと幅をファイルに出力するプログラムを作成し、それを RaspberryPi 上で実行し、ポジティブデータと確認用の顔データ計 1600 枚を集めた。
処理速度を重視したためプログラミング言語はコンパイル言語である C++ を採用した。
以下はカスケード分類機を作るに当たって使用したコマンドである。

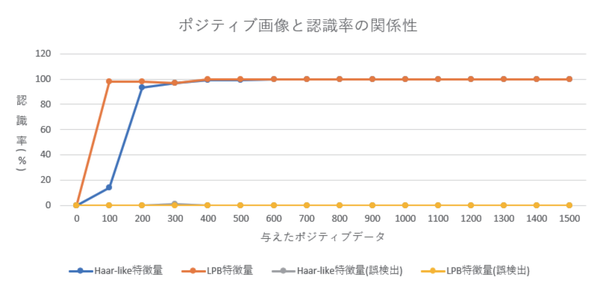
```
$opencv_createsamples.exe -info ./pos.lst -num 1000 -vec -pos_samples.vec
$opencv_traincascade.exe -data ./cascade -vec ./pos_samples.vec -bg ./neg.lst -numPos 1300
  -numNeg 2000 -featureType HAAR -numStages 20
```

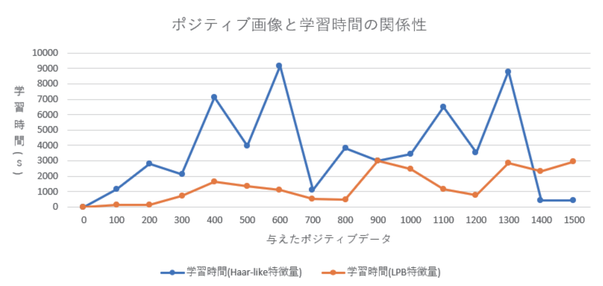
前者が与えられたポジティブデータを回転させた場合などのパターンをベクトルファイルと呼ばれるファイルに書きこむ処理である第二引数に’-num 1000’を指定したのでこの場合は 1000 枚分のベクトルファイルが作成されたことになる。続いて後者の命令は集めたネガティブ画像とポジティブ画像から作られたベクトルファイルを基にカスケード分類機を作成する命令である。
第二、第三引数にはポジティブデータ、ネガティブデータの相対パスを記載し、第四、第五引数ではポジティブデータ、ネガティブデータのサンプル数を設定し第 6 引数の’-featureType HAAR’で作成するカスケード分類機のタイプを指定している (今回は Haar-like 特徴) 、第七引数では’-numStages 20’と AI の学習回数を設定している。
そうして作られたカスケード分類機を AI に読み込ませ、同じ実験者の学習するときに与えなかった顔データ 100 枚のスクリーンショットを AI がどのように捉えるかを計測する。

3. 実験結果

ポジティブデータを変更した場合

ポジティブデータ :0~1500　ネガティブデータ：2000　学習回数：20

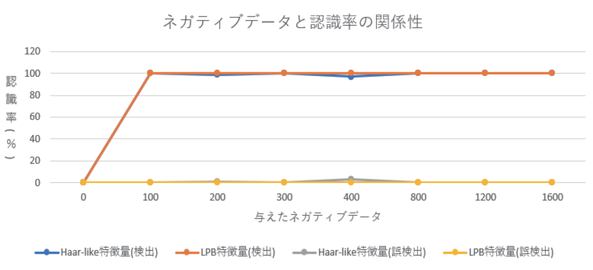


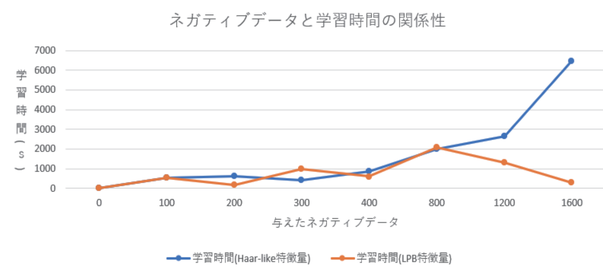


検出率については LBP 特徴量が 100 枚 ,Haar-like 特徴量が 200 枚で 90% 代に到達したので Haar-like 特徴量よりも LBP 特徴量の方が少ないポジティブデータでの検出が可能と考えられる。
学習時間についてはほとんどのデータ数で LPB 特徴量が Heer-like 特徴量の半分ほどの学習時間で学習を終えているのが観察出来る。

ネガティブデータを変更した場合

ポジティブデータ：1500　ネガティブデータ：0~1600　学習回数：20

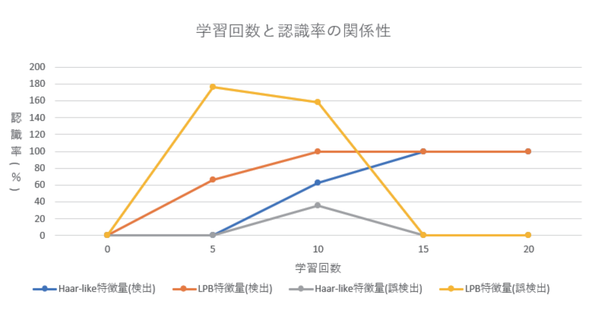


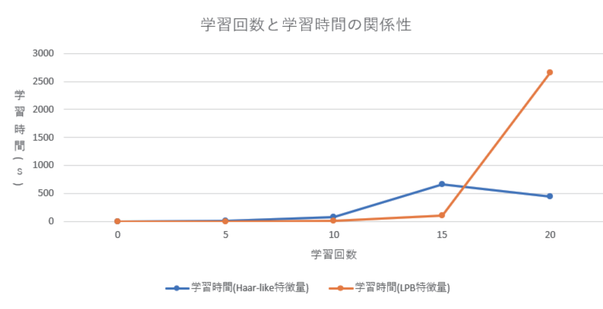


認識率は Heer-like 特徴量、LBP 特徴量共にネガティブデータ 100 でほぼ 100% に達しているのでネガティブデータと認識率はあまり関係性がないように見える。
学習時間は LPB 特徴量の 1200,1600 が下がり気味であるが全体的に正の相関があるのでネガティブデータの量によって学習時間が変化することがわかる。

学習回数を変更した場合

ポジティブデータ：1500　ネガティブデータ：2000　学習回数：0~20

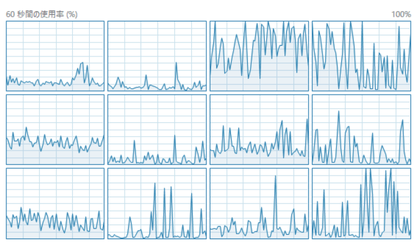




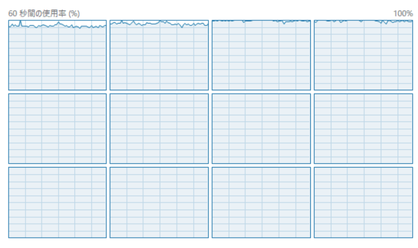
Haar-Lile 特徴量は 10 回目から検出が可能となり 15 回で認識率 100% に収束する。LBP 特徴量は Haar-Like 特徴量よりも少ない 5 回の学習で検出が可能となるが検出量の 3 枚にもおよぶ誤検出が出ることになるが、回数を重ねていく毎に誤検出も少なくなり最終的には 15 回で認識率 100%、誤検出 0% に収束する。収束するときの回数は Haak-Like 特徴量、LPB 特徴量共に同じである。
学習時間いついては Haak-Like 特徴量、LPB 特徴量で真逆の結果となっている。

学習中の CPU の負荷

下の二つのグラフはディープラーニング中の CPU の一分間の使用割合を表すグラフである。
カスケード分類機を 1 つ作る場合特定のコアに負荷がかかり、並行していくつも作る場合は全部のコアの使用率が 100% になることが確認できた。



(図) 単独処理時の CPU 負荷



(図) 並行処理時の CPU 負荷

4. まとめ

今回の実験で Haar-Like 特徴量は LBP 特徴量に比べて学習時間が長く検出するのに LBP 特徴量よりも多くのサンプルデータがいる分、認識精度が安定しており LBP 特徴量は Haar-Like 特徴量に比べて学習時間が短く少ないサンプル数で検出が可能だが誤検出が多いと言った欠点を持っている。
そして双方の特徴量が認識率が 100% に収束するにのは同じポジティブデータ、ネガティブデータ、学習回数のときだということがわかった。
またディープラーニングは CPU に大きな負荷がかかり、多くの時間を要することがわかった。
次はより多くのサンプルを用いてより正確な数値を出したり、今回得られた CPU の負荷のデータを基にした解決策や、CPU だけでなく NVIDIA CUDA などの GPU による並列処理を行ったディープラーニングの軽量化についての実験を行っていきたい。