**ĐẠI HỌC QUỐC GIA HÀ NỘI**

**ĐẠI HỌC CÔNG NGHỆ**

**BÁO CÁO**

**BÀI TẬP LỚN MÔN TRÍ TUỆ NHÂN TẠO**

**Chủ đề: Phân lớp tàu thuyền**

Thực hiện: ***Nhóm Gâu Gâu***

***Đinh Văn Kiệt*** MSSV: ***14020247***

***Nguyễn Hải Nhật*** MSSV: ***14020716***

Năm học 2017 - 2018

**MUC LỤC**

[**1. Mô tả bài toán 3**](#_Toc501950394)

[**2. Phân tích dữ liệu 6**](#_Toc501950395)

[**3. Một số hướng tiếp cận 11**](#_Toc501950396)

[**3.1. Histogram of Oriented Gradients 11**](#_Toc501950397)

[**3.2. Mô hình Bag Of Visual Words 13**](#_Toc501950398)

[**4. Thực hiện trainning 15**](#_Toc501950399)

[**5. Kết quả 16**](#_Toc501950400)

[**5.1. Kết quả ban đầu (trong buổi báo cáo) 16**](#_Toc501950401)

[**5.2. Kết quả cải tiến (thầy Hưng gợi ý) 17**](#_Toc501950402)

[**5.3. Cross Validation 19**](#_Toc501950403)

[**6. Các vấn đề còn tồn đọng 20**](#_Toc501950404)

# 1. Mô tả bài toán

Cho tập dữ liệu tàu HRSC2016 được các tác giả cắt từ ảnh Google Earth mô tả theo đường link <https://sites.google.com/site/hrsc2016/>

Bộ dữ liệu gồm 2 phần: file annotations và file ảnh (tổng số 1680 cặp). Một số đặc điểm của tập dữ liệu

* Ứng với một file ảnh sẽ có 1 file annotations
* Có ảnh sẽ không có tàu nào được gán nhãn

Các class được chia theo 3 layer như bảng sau

|  |  |  |  |
| --- | --- | --- | --- |
| **Class ID** | **Parent class ID** | **Class layer** | **English name** |
| 100000001 | NULL | 0 | ship |
| 100000002 | 100000001 | 1 | aircraft carrier |
| 100000003 | 100000001 | 1 | warcraft |
| 100000004 | 100000001 | 1 | merchant ship |
| 100000005 | 100000002 | 2 | Nimitz class aircraft carrier |
| 100000006 | 100000002 | 2 | Enterprise class aircraft carrier |
| 100000007 | 100000003 | 2 | Arleigh Burke class destroyers |
| 100000008 | 100000003 | 2 | WhidbeyIsland class landing craft |
| 100000009 | 100000003 | 2 | Perry class frigate |
| 100000010 | 100000003 | 2 | Sanantonio class amphibious transport dock |
| 100000011 | 100000003 | 2 | Ticonderoga class cruiser |
| 100000012 | 100000002 | 2 | Kitty Hawk class aircraft carrier |
| 100000013 | 100000002 | 2 | Admiral Kuznetsov aircraft carrier |
| 100000014 | 100000003 | 2 | Abukuma-class destroyer escort |
| 100000015 | 100000003 | 2 | Austen class amphibious transport dock |
| 100000016 | 100000002 | 2 | Tarawa-class amphibious assault ship |
| 100000017 | 100000003 | 2 | USS Blue Ridge (LCC-19) |
| 100000018 | 100000004 | 2 | Container ship |
| 100000019 | 100000003 | 2 | Command ship A |
| 100000020 | 100000004 | 2 | Car carrier A |
| 100000022 | 100000004 | 2 | Hovercraft |
| 100000024 | 100000004 | 2 | yacht |
| 100000025 | 100000004 | 2 | Container ship A |
| 100000026 | 100000004 | 2 | Cruise ship |
| 100000027 | 100000001 | 1 | Submarine |
| 100000028 | 100000003 | 2 | Warcraft A |
| 100000029 | 100000003 | 2 | Medical ship |
| 100000030 | 100000004 | 2 | Car carrier B |
| 100000031 | 100000002 | 2 | Ford-class aircraft carriers |
| 100000032 | 100000002 | 2 | Midway-class aircraft carrier |
| 100000033 | 100000002 | 2 | Invincible-class aircraft carrier |

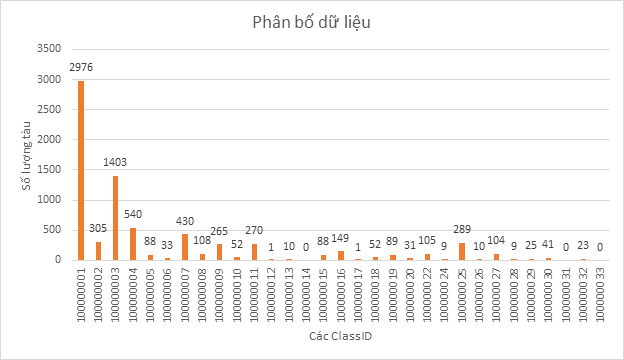
Trong đó có 1 lưu ý của các tác giả: Một tàu bất kì ở layer 2 sẽ được gán 3 class ứng với từng layer 0 1 2.

# 2. Phân tích dữ liệu

Tổng số tàu cho từng class được cho trong bảng dưới (chú ý 1 tàu ở layer 2 được tính làm 3 lần cho 3 layer 0 1 2)

|  |  |  |  |
| --- | --- | --- | --- |
| **Class ID** | **sum** | **Class ID** | **sum** |
| 100000001 | 2976 | 100000010 | 52 |
| 100000003 | 1403 | 100000018 | 52 |
| 100000004 | 540 | 100000030 | 41 |
| 100000002 | 305 | 100000006 | 33 |
| 100000027 | 104 | 100000020 | 31 |
| 100000007 | 430 | 100000029 | 25 |
| 100000025 | 289 | 100000032 | 23 |
| 100000011 | 270 | 100000026 | 10 |
| 100000009 | 265 | 100000013 | 10 |
| 100000016 | 149 | 100000028 | 9 |
| 100000008 | 108 | 100000024 | 9 |
| 100000022 | 105 | 100000017 | 1 |
| 100000019 | 89 | 100000012 | 1 |
| 100000015 | 88 | 100000014 | 0 |
| 100000005 | 88 | 100000033 | 0 |
|  |  | 100000031 | 0 |

Để dễ quan sát ta vẽ biểu đồ cho bảng trên



*Biểu đồ phân bố dữ liệu theo từng class*

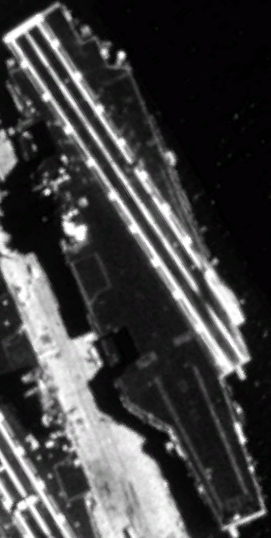
Dựa vào biểu đồ phân bố dữ liệu theo các class, ta có thể thấy đây là bộ dữ liệu rất imbalance. Quan sát một chút vào các class có dữ liệu ít, ta thấy như sau



*class 100000005 (cha là class 100000002)*



*class 100000006 (cha là class 100000002)*



*class 100000012 (cha là class 100000002)*

Nhận xét rằng có một số class cùng cùng 1 class cha có các tàu không thực sự là khác nhau. Điều này đặt ra thử thách khi phân lớp cho layer 2 đó là xử lý imbalance và phân lớp chính xác nhưng class không khác biệt nhau nhiều như này.

Tương tự với class cha là 100000003



*class 100000010*



*class 100000011*



*class 100000015*

Sự khác biệt có thể dễ nhận ra bằng mắt thường nhưng là rất nhỏ. Do đó e quyết định thực hiện phân lớp cho các tàu ở layer 1.

# 3. Một số hướng tiếp cận

Trước khi thực hiện đi sâu vào một hướng e có thử qua các hướng như sau cùng một số vấn đề gặp phải. Các hướng tiếp cận dưới đây chủ yếu là khâu features extraction.

## 3.1. Histogram of Oriented Gradients

Sử dụng thuật toán HOG trích xuất features => đưa vào bộ phân lớp

*Tham số cho HOG (skimage):*

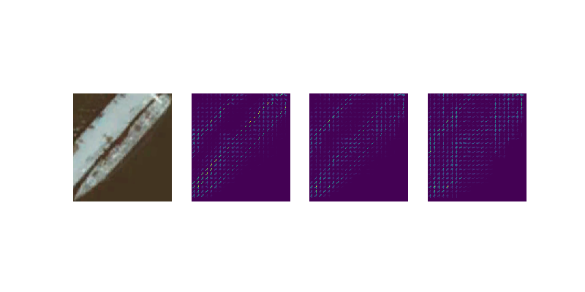
* number\_of\_orientations: 9
* pixels\_per\_cell: 8 (hoặc 16)
* cells\_per\_block': 1 (hoặc 2, 3)
* do\_transform\_sqrt: True

*Nhận xét:*

* Cần chọn tham số thích hợp cho HOG, ảnh quá to dẫn đến HOG quá chi tiết => số features rất lớn => mô hình phức tạp
* Ảnh kích cỡ khác nhau cho ra số features khác nhau (có cách lách cái này dựa vào công thức tính HOG rồi chọn cặp số chiều rộng – chiều cao ảnh, e đã thử nhưng kết quả không tốt)
* Resize ảnh để có cùng số features sẽ làm ảnh bị biến dạng
* Vì đây là layer 1, trong cùng 1 class có thể có một số tàu khác nhau đôi chút => em thấy HOG chỉ hợp với dữ liệu mà trong đó các dữ liệu phải na ná nhau về đường nét (như HOG hay dùng cho bài toán nhận diện LOGO hay người đi bộ, mặt người,….)
* HOG lấy được rất nhạy cảm với nhiễu
* https://www.learnopencv.com/histogram-of-oriented-gradients/

*Kết quả lấy HOG*

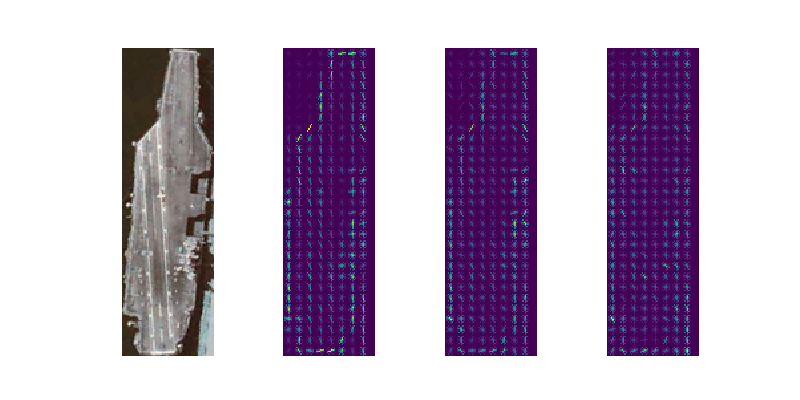
Từ trái qua phải ứng với 3 loại màu HLS, HSV và YUV



*Ảnh lấy HOG chưa resize, nhiều nhiều*



*Ảnh lấy HOG chưa resize, không chứa nhiễu*



*Ảnh lấy HOG đã resize và không chứa nhiếu*

## 3.2. Mô hình Bag Of Visual Words

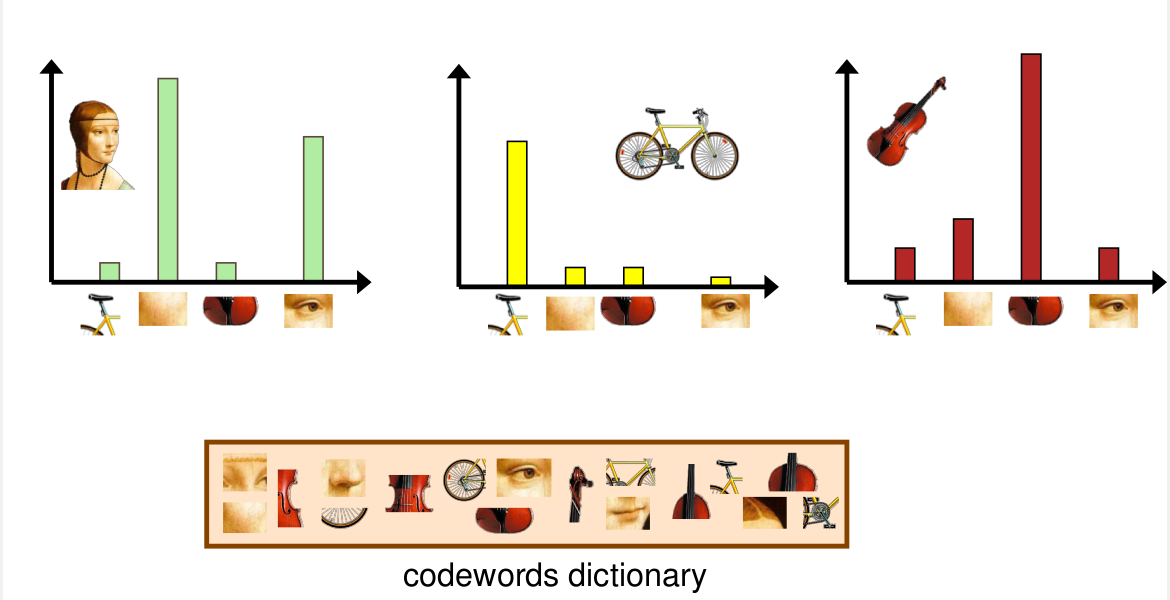
Các bước thực hiện như sau:

* Thực hiện preprocessing gồm các bước nhỏ sau:
  + Dựa vào tọa độ tâm tàu và đầu tàu => lập thành 1 vector chỉ hướng đầu tàu, tiến hành quay tất cả các tàu về chiều Ox.
  + Để triệt tiêu nhiễu hoàn toàn, từ tọa độ tâm và đầu tàu mới, mở rộng sang bề ngang 25% độ dài vector lúc đầu (cái này phải thực nghiệm nhiều lần)
  + Thu được bộ dữ liệu sạch
* Với mỗi ảnh ta sẽ sử dụng thuật toán SIFT trích xuất ra các patches đặc trưng (vùng ảnh vuông chứa các góc). Tham số SIFT được lựa chọn là
  + Số features tốt nhất giữ lại (nfeatures): 250 (ứng với 250 điểm)
  + Ngưỡng cạnh (edgeThreshold): 50
  + Ngưỡng độ tương phản (contrastThershold): 0.02

Các tham số còn lại để mặc định.

https://docs.opencv.org/3.3.0/da/df5/tutorial\_py\_sift\_intro.html

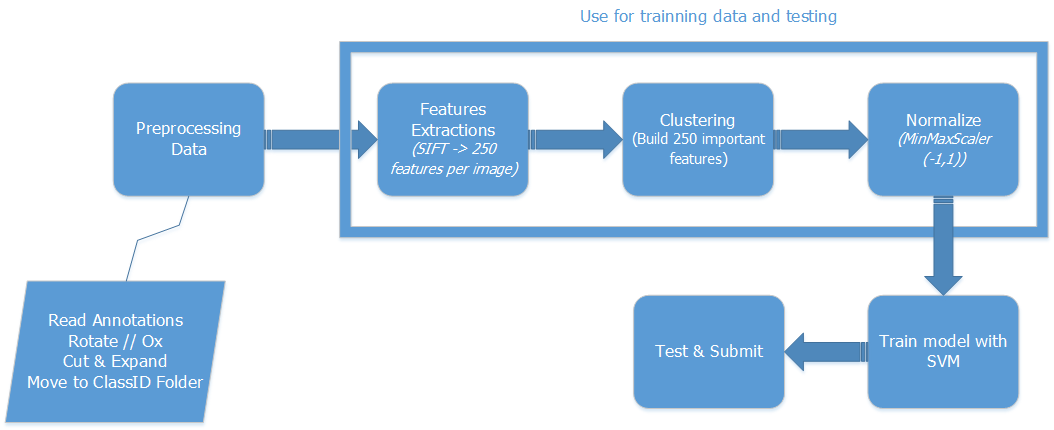
* 
* Nhận thấy các features do SIFT lấy được dễ bị ảnh hưởng bởi nhiễu (phà liền sát tàu) => nếu có phà bên cạnh sẽ rất đến kết quả rất xấu.
* Gom tất cả các patches này lại rồi tiến hành phân ra K cụm bằng Kmeans (đây cũng có thể hiểu là bước giảm chiều dữ liệu)
* Với mỗi ảnh, tiến hành tính histogram của các patches => tạo thành features của mỗi ảnh
* Đem các feautes này vào các bộ phân lớp để thử



*(nguồn trên mạng)*

# 4. Thực hiện trainning

E lựa chọn mô hình Bag Of Visual Words như đã trình bày ở trên. Sơ đồ model của e như sau:



Source code tại https://github.com/dinhvankiet124/HRSC

# 5. Kết quả

Tập train/val gồm 1260 file xml ứng với

|  |  |  |
| --- | --- | --- |
| **ClassID** | **Số lượng ảnh** | **Mô tả** |
| 100000002 | 168 | Ảnh tàu sân bay cỡ lớn và cỡ bé |
| 100000003 | 686 | Ảnh tàu quân sự có sân bay trực thăng |
| 100000004 | 413 | Ảnh tàu thương mại, bán hàng |
| 100000027 | 45 | Tàu ngầm |

Tập test gồm có 420 file xml ứng với

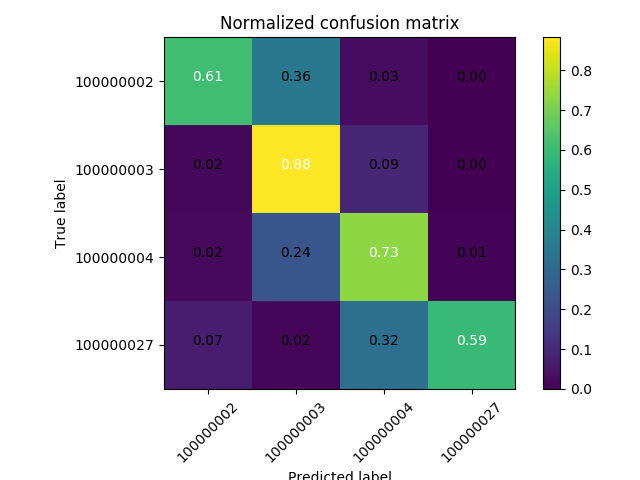
|  |  |
| --- | --- |
| **ClassID** | **Số lượng ảnh** |
| 100000002 | 137 |
| 100000003 | 717 |
| 100000004 | 127 |
| 100000027 | 59 |

## 5.1. Kết quả ban đầu (trong buổi báo cáo)

Do dữ liệu bị imbalance nên e chọn ảnh từ mỗi class sao cho số ảnh ko chênh nhau quá nhiều => số lượng 71, 71, 71, 45 tương ứng. Do resample nên có lúc sẽ chọn đc data đẹp, có lúc data xấu nên e chạy thử nhiều lần rồi lưu model cho kết quả tốt nhất.

Kết quả số tàu phân lớp đúng / số tàu phân lớp sai: **81.25%**

Confusion Matrix



Nhận xét:

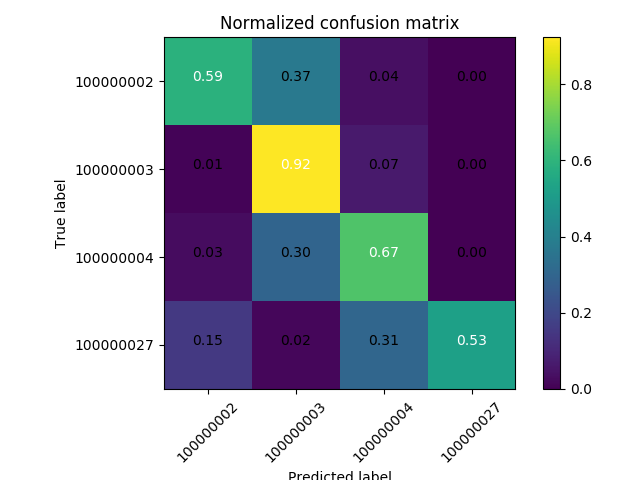
* Class 3 là class quân sự với đặc trưng có sân bay cho trự thăng ở đít, màu sắc, dễ nhận biết giúp cho độ chinh xác khá cao
* Class 2 là tàu sân bay, một số tàu sân bay bị nhầm thành tàu quân sự do tàu sân bay đó có dạng dẹp.
* Class 4 gồm các tàu thương mại cho kết quả cao hơn mong đợi, do class 4 nhiều tàu không thực sự giống nhau lắm
* Class 27 là tàu ngầm với kích thước bé hay bị nhận nhầm thành tàu class 4. Class 27 có thể được nhận diện bởi thân thon và màu tối

## 5.2. Kết quả cải tiến (thầy Hưng gợi ý)

Chọn tham số class\_weight cho SVM là “balanced” <http://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html#sklearn.svm.SVC> và fit toàn bộ data thay vì resample

Kết quả số tàu phân lớp đúng / số tàu phân lớp sai: **82.59%**

Confusion Matrix



Nhận xét:

* Cách làm này giúp fit được toàn bộ dữ liệu
* Khi fit lượng dữ liệu lớn, tổng số patches thu được sẽ rất lớn => kmeans phân cụm rất lâu, cân nhắc sử dụng MiniBatchesKmeans.

## 5.3. Cross Validation

Test bằng phương pháp kiểm chứng chéo với 5 folds, full dữ liệu (train + test) và sử dụng MiniBatchKmeans thu được kết quả như sau:

|  |  |  |
| --- | --- | --- |
| Phương pháp | None | class\_weight=”balanced” |
| Độ chính xác | [ 0.88322718 0.88535032 0.87473461 0.87659574 0.87206823] | [ 0.84713376 0.86624204 0.87048832 0.86382979 0.88486141] |
| Mean | 0.88 (+/- 0.01) | 0.86 (+/- 0.02) |

# 6. Các vấn đề còn tồn đọng

* Chưa xử lý imbalance một cách tốt nhất
* Chưa phân lớp được cho layer 2