NHẬN DIỆN BIỂN BÁO GIAO THÔNG VIỆT NAM SỬ DỤNG FASTER RCNN

VÀ

THỰC NGHIỆM MỘT SỐ PHƯƠNG PHÁP PHÂN LOẠI ẢNH SỬ DỤNG DEEP MODEL

Sinh viên thực hiện:
14520956 – HOÀNG HỮU TÍN, 14520247 – TRẦN XUÂN HẢI, 14520156 – TRẦN QUANG ĐẠT

Lớp Khoa học Tài năng 2014, Khoa Khoa học máy tính, Trường Đại học Công nghệ Thông tin;
14520956@gm.uit.edu.vn, 14520247@gm.uit.edu.vn, 14520156@gm.uit.edu.vn

Tóm tắt:

Object detection là một trong các task quan trọng trong thị giác máy tính, từ một bức ảnh chúng ta sẽ dạy cho máy học thế nào để biết được vị trí và loại đồ vật có bên trong ảnh đó. Chúng ta có thể áp dụng các phương pháp học Object Detection cho nhiều thể loại đồ vật khác nhau. Xuất phát từ thực tế nhu cầu cải thiện giải pháp giao thông ở Việt Nam hiện nay, nhóm đã quyết định chọn đề tài về phát hiện biển báo giao thông đường bộ. Đây là một phần trong dự án lớn thực hiện xe tự lái, hướng tới loại bỏ dần dần những sai lầm có thể có của con người khi tham gia giao thông, đặc biệt là với giao thông phức tạp như ở Việt Nam.

Trong Project lần này sẽ được phân làm hai nhiệm vụ chính tùy thuộc vào năng lực của thành viên trong nhóm:

- *Phần 1:* Nhận diện biển báo giao thông việt nam sử dụng Faster RCNN

  Thực hiện training lại model Faster RCNN dùng để detect biển báo giao thông từ dataset nhóm đã thu thập trước đó.

  Phụ trách bởi: Hoàng Hữu Tín

- *Phần 2:* Khảo sát một số phương pháp phân loại ảnh sử dụng deep model

  Task 1: Thực hiện classification theo dataset biển báo đã thu thập, dựa trên deep features được rút trích từ model VGG16 pretrained on ImageNet dataset. Các phương pháp classification sử dụng: linear, SVM-linear, SVM-RBF.
  Task 2: Thực hiện train lại hoàn toàn model VGG16 với input và output phù hợp với dataset VNTSDB của nhóm (Vietnam Traffic Signs Dataset)

  Phụ trách bởi: Trần Xuân Hải, Trần Quang Đạt

Phần 1:
NHẬN DIỆN BIỂN BÁO GIAO THÔNG VIỆT NAM SỬ DỤNG FASTER RCNN

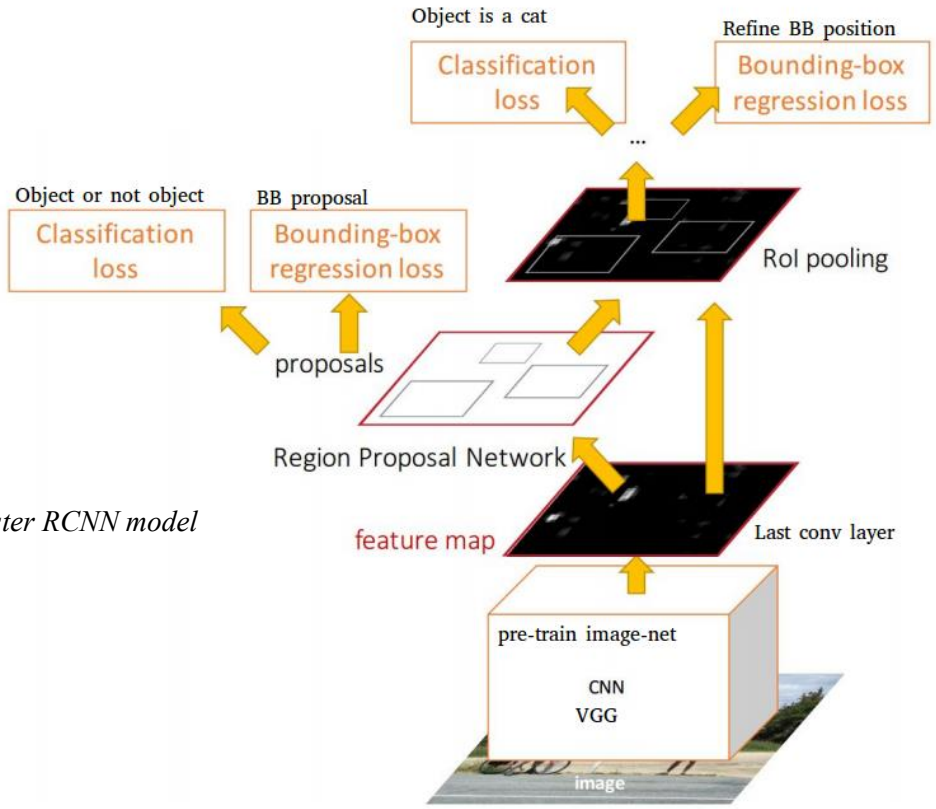
(Biên tập nội dung: Hoàng Hữu Tín)

1. GIỚI THIỆU

- Bài toán: Thực hiện Detection trên một ảnh (hoặc video) chụp tại quang cảnh giao thông đường bộ, cho biết vị trí trên ảnh xuất hiện biển báo giao thông và loại biển báo đó.
- Tầm quan trọng của bài toán: Với tình hình giao thông phức tạp như ở Việt Nam, người lái xe khó có thể vừa tập trung quang sát đường phía trước vừa phải để ý các biển báo giao thông hai bên đường. Thực tế đã có khá nhiều trường hợp vi phạm luật giao thông, thậm chí xảy ra tai nạn do không quan sát được biển báo. Do vậy việc có một công cụ hỗ trợ thông báo cho tài xế biết được trên đoạn đường có biển báo giao thông gì là cần thiết. Xuất phát từ việc đó, nhóm đã chọn tìm hiểu xây dựng dataset về biển báo giao thông Việt Nam và demo một số phương pháp máy học trên dataset này.

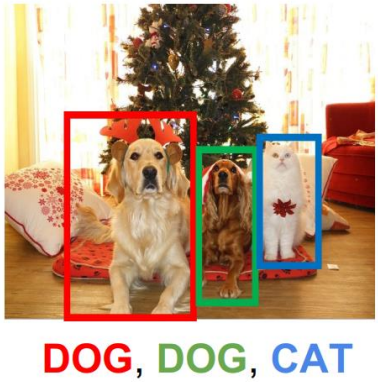
Mã nguồn Project được up tại:
https://github.com/Flavius1996/VNTS-faster-rcnn

2. PHƯƠNG PHÁP FASTER RCNN [1]



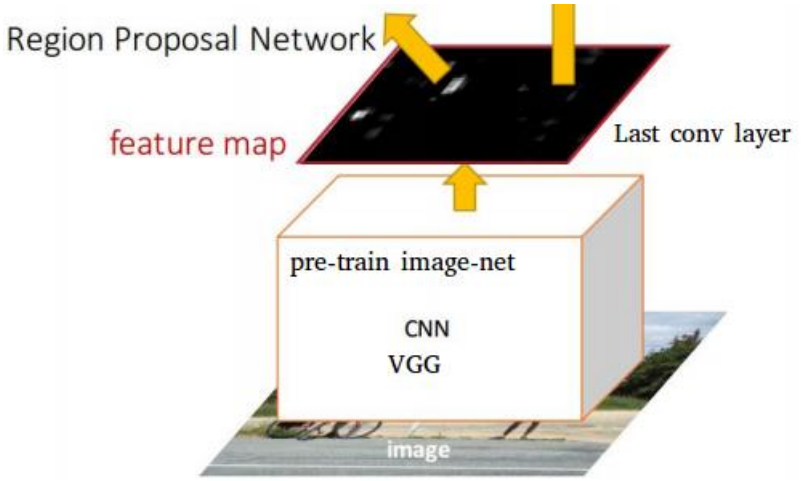
Hình 1.
Tổng quan Faster RCNN model [4]

Faster RCNN là một phương pháp dùng để detect object từ một ảnh đầu vào. Phương pháp sử dụng kết hợp một Convolution Neural Network (CNN) để rút trích đặc trưng của ảnh kết hợp với một mạng xử lý Region Proposal Network (RPN) để đưa ra các gợi ý về vị trí có thể xuất hiện object.



Hình 2. Thực hiện detect cho 2 class: DOG, CAT [4]

❖ **Bước 1:** Sử dụng CNN để xây dựng feature map

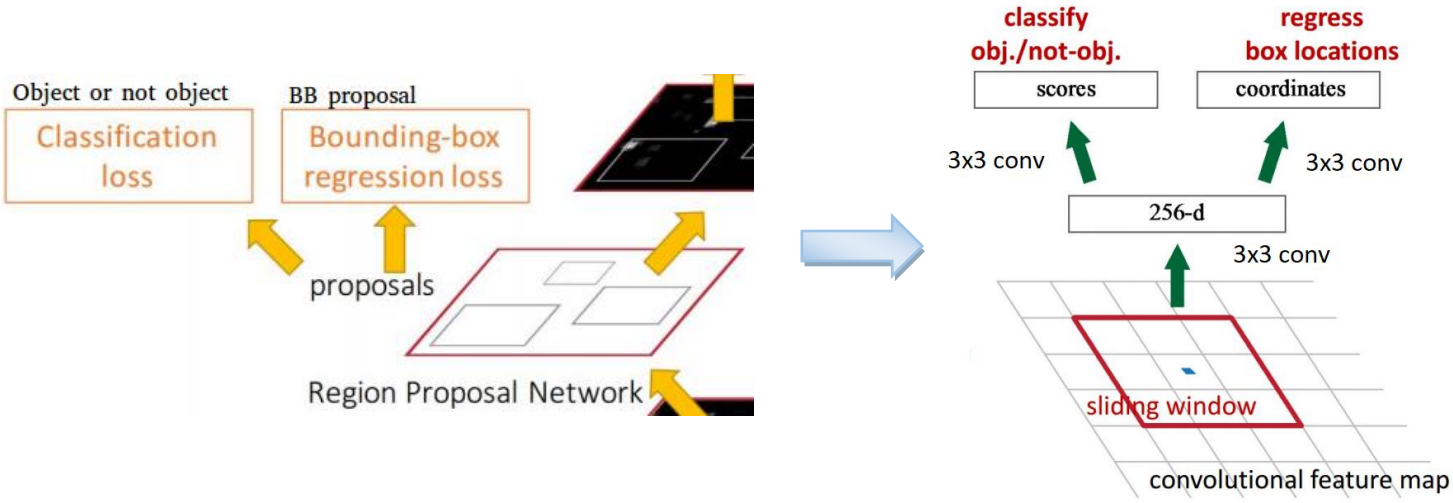


Hình 3. Using CNN extract feature map [4]

Áp dụng một mạng CNN chuyên dùng để tính feature của ảnh thông thường như VGG16, ZF, GoogleNet, Resnet-101, …
Mạng CNN này được áp dụng lên toàn bộ ảnh để cho ra feature map – tương ứng với các điểm trên ảnh.
Để thuận tiện không phải train lại, CNN ở đây dùng sẵn các trọng số đã train trên Image-net dataset.

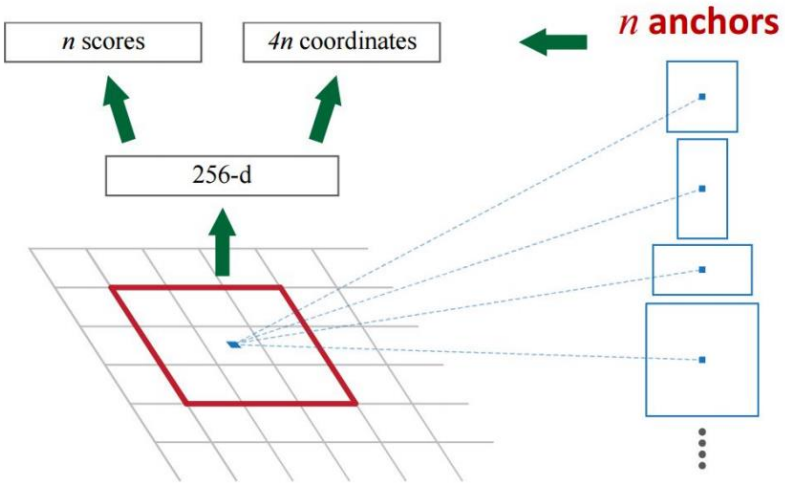
Trong thử nghiệm lần này do hạn chế về phần cứng nên nhóm chỉ chọn ZF model để thực hiện bước này.

❖ **Bước 2:** Region Proposal Network (RPN)



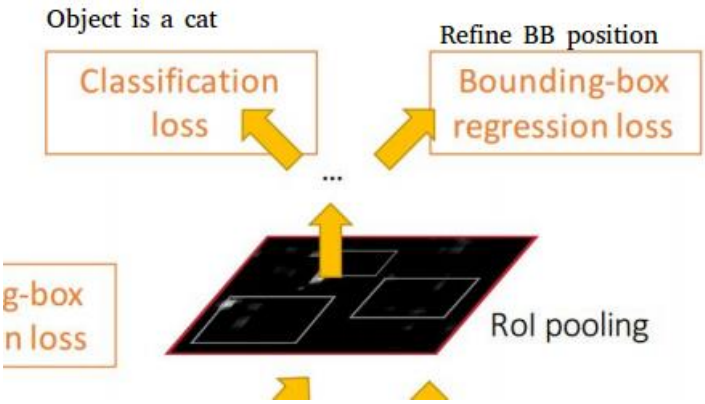
Hình 4. Region Proposal Network [4]

Sau khi đã có được feature map, ta cần phải thực hiện để biết được vùng nào trong ảnh có khả năng chứa object.
Về cơ bản RPN thực hiện sliding window qua tất cả điểm trên feature map, với mỗi điểm ta áp dụng một bộ anchors gồm các cửa sổ kích thước khác nhau sẽ áp dụng lên điểm này để ra được 256-dimensional vector sau đó tính classify cost (object / not object) và regressing bbox locations cost (tọa độ bounding box được quy định theo top-left-bottom-right)



Hình 5. RPN với các anchors khác nhau [4]

❖ **Bước 3**: Classification



Hình 6. Classification trên proposal region [4]

Nhận về Bounding Box từ RPN áp dụng để crop feature map từ đó thực hiện classification như thông thường để tính loss cho các class cần detect (DOG, CAT, BIRD, …)
Và tính loss của Bounding-Box dựa trên Bounding-Box groundtruth đã cung cấp trong training data.

Vậy kết quả trả về cuối cùng của Faster RCNN là 4 giá trị BoundingBox và giá trị confidence (classification loss) cho mỗi proposal được đưa ra bởi RPN.

3. TẬP DỮ LIỆU

Toàn bộ Dataset được lưu trữ tại:
https://drive.google.com/drive/u/0/folders/0B9hMAZTpHpyCclFwT2NFWTRYSjg

Chi tiết báo cáo thu thập và format Dataset xem tại:
https://docs.google.com/document/d/1u5nGoa2GwW-G6TkxRJYGDG-TXE3RK7cUGGXZgJKQFac

Quá trình thu thập dữ liệu

Cách thức thu thập dữ liệu:

- Sử dụng xe máy để đi quay videos các tuyến đường trên địa bàn thành phố Hồ Chí Minh.
- Sử dụng tool do nhóm tự tạo để tách các ảnh và đánh annotation (sign id, sign flag, bounding box) cho từng biển báo từ các videos thu thập được.

Thông tin về tập dữ liệu

Thống kê:

- Tổng số lớp có dữ liệu: 69
- Lớp cao nhất có: 164 ảnh
- Lớp thấp nhất có: 1 ảnh
- Trung bình: 22 ảnh / lớp
- Định dạng lưu trữ: JPG

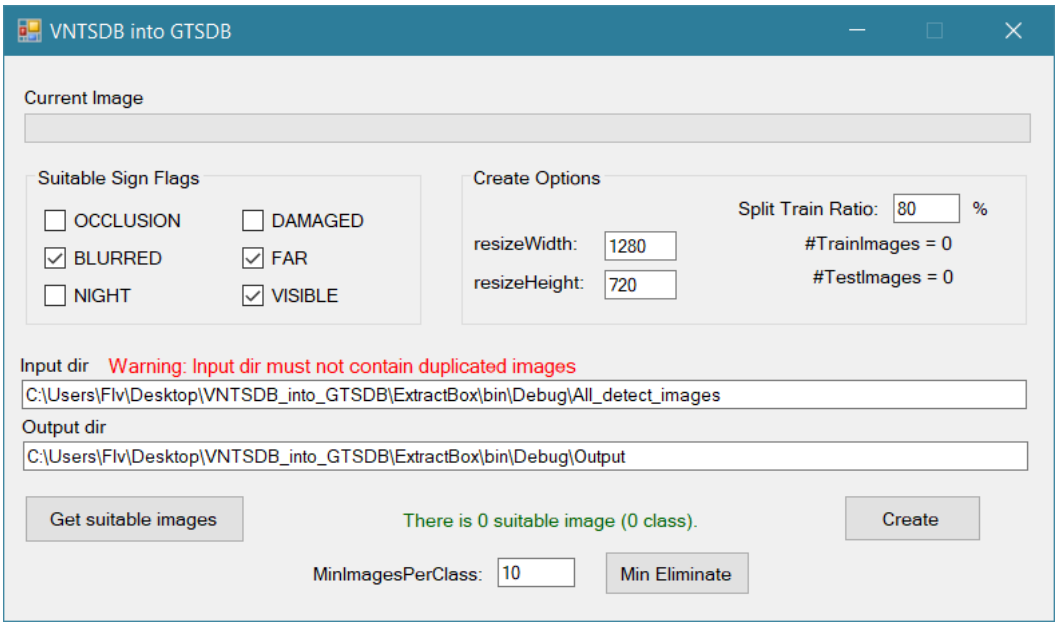
Báo cáo thống kê cho từng loại biển báo:
https://drive.google.com/file/d/0B9hMAZTpHpyCN0l1WU9mQ2RxT1U/view

Hình 7. Minh họa một số ảnh của các lớp khác nhau trong tập dữ liệu.

3.1 Tổ chức dữ liệu

Xem cách tổ chức dữ liệu của VNTSDB tại:
https://docs.google.com/document/d/16Fe2-b_eLCv-YRgcm3exkJzLIDgldua4pcFKYUe4rs8

- **Chuyển đổi định dạng Vietnam Traffic Sign Dataset (VNTSDB) sang German Traffic Sign Dataset (GTSDB):**
GTSDB là một bộ dataset nổi tiếng về traffic signs của Đức. Đã có khá nhiều bài báo cũng như source code được thực hiện dựa trên dataset này. Do vậy cần xây dựng tool để tự động chuyển đổi từ VNTSDB sang định dạng của GTSDB để có thể dễ dàng sử dụng lại các source code có sẵn.



Hình 8. Tool chuyển đổi VNTSDB sang GTSDB

Tool cho phép chọn VNTSDB sign flags cần extract, giới hạn chỉ lấy các class với số lượng ảnh tối thiểu, resize ảnh, randomly split train-test data, lưu trữ ảnh với định dạng .PPM
Sau khi chuyển đổi, Dataset sẽ có tổ chức như sau:

+ Folder: **Train**
    + Gồm các sub folder với số ID của class (theo số integer – không phải sign ID của VNTSDB) chứa các ảnh chỉ gồm biển báo đã crop từ ảnh lớn.
    + Các ảnh training .PPM (ảnh lớn toàn cảnh)
+ Folder: **Test**
    + Các ảnh testing .PPM (ảnh lớn toàn cảnh)
+ File **test_annotation.txt:** Thông tin về annotation của biển báo trên ảnh ở Test folder
+ File **test_annotation.txt:** Thông tin về annotation của biển báo trên ảnh ở Train folder
+ File **SignID_mapping.txt:** Thông tin về chuyển đổi VNTSDB sign ID (ID theo quy định của bộ giao thông Việt Nam) sang ID theo thứ tự số nguyên.

Định dạng trong **annotation.txt** ví dụ như sau:

00000.ppm;693;11;798;115;14
00001.ppm;949;214;990;254;13
00002.ppm;309;195;345;232;0
00002.ppm;967;173;1003;209;4

Ý nghĩa:

<tên ảnh> ; <4 giá trị bounding box của biển báo> ; <ID của biển báo>

Dataset được dùng để chạy Faster RCNN lần này được tách từ VNTSDB gồm các thông số:

+ Checked sign flags: Visible, far, blurred
+ MinImagesPerClass: 10
+ Resize: 1280x720
+ Train-Test ratio: 80-20

Như vậy Dataset chính thức sử dụng cho Faster RCNN có thống kê như sau:

+ Số loại class: 28
+ Số ảnh training: 424 (là ảnh toàn bộ khung cảnh đường phố)
+ Số ảnh testing: 106 (là ảnh toàn bộ khung cảnh đường phố)
+ Số lần xuất hiện trung bình trong một biển báo trong training: 23.5

4. CHƯƠNG TRÌNH

Ngôn ngữ: Python 2.7
Thư viện chính: caffe gpu + CUDA + cuDNN
Môi trường: Ubuntu 16.04 – Card Nvidia GTX 1050 (4GB)

Chi tiết source code, cách cài đặt xem ở folder Github của Project đã nêu ở trên.
Model đã train được và file log:
https://drive.google.com/drive/u/0/folders/1nG5X4M_D3mPi29hz-VdipfnzYArwrfDP

Sử dụng model ZF cho phần CNN, tổng số lần lặp 70000.

Cấu hình Config chạy training (Ý nghĩa từng cái xem tại Github/lib/fast_rcnn/config.py):

```
{'DATA_DIR': '/home/flv/Desktop/tsr-py-faster-rcnn/data',
 'DEDUP_BOXES': 0.0625,
 'EPS': 1e-14,
 'EXP_DIR': 'faster_rcnn_end2end',
 'GPU_ID': 0,
```

'MODELS_DIR': '/home/flv/Desktop/tsr-py-faster-rcnn/models/vntsdb',
'PIXEL_MEANS': array([[[102.9801, 115.9465, 122.7717]]]),
'RNG_SEED': 4,
'ROOT_DIR': '/home/flv/Desktop/tsr-py-faster-rcnn',

'TRAIN': {'ASPECT_GROUPING': True,
'BATCH_SIZE': 128,
'BBOX_INSIDE_WEIGHTS': [1.0, 1.0, 1.0, 1.0],
'BBOX_NORMALIZE_MEANS': [0.0, 0.0, 0.0, 0.0],
'BBOX_NORMALIZE_STDS': [0.1, 0.1, 0.2, 0.2],
'BBOX_NORMALIZE_TARGETS': True,
'BBOX_NORMALIZE_TARGETS_PRECOMPUTED': True,
'BBOX_REG': True,
'BBOX_THRESH': 0.5,
'BG_THRESH_HI': 0.5,
'BG_THRESH_LO': 0.0,
'FG_FRACTION': 0.25,
'FG_THRESH': 0.5,
'HAS_RPN': True,
'IMS_PER_BATCH': 1,
'MAX_SIZE': 1000,
'PROPOSAL_METHOD': 'gt',
'RPN_BATCHSIZE': 256,
'RPN_BBOX_INSIDE_WEIGHTS': [1.0, 1.0, 1.0, 1.0],
'RPN_CLOBBER_POSITIVES': False,
'RPN_FG_FRACTION': 0.5,
'RPN_MIN_SIZE': 16,
'RPN_NEGATIVE_OVERLAP': 0.3,
'RPN_NMS_THRESH': 0.7,
'RPN_POSITIVE_OVERLAP': 0.7,
'RPN_POSITIVE_WEIGHT': -1.0,
'RPN_POST_NMS_TOP_N': 2000,
'RPN_PRE_NMS_TOP_N': 12000,
'SCALES': [600],
'SNAPSHOT_INFIX': '',
'SNAPSHOT_ITERS': 10000,
'USE_FLIPPED': True,
'USE_PREFETCH': False},
'USE_GPU_NMS': True}

5. KẾT QUẢ

Một số ảnh kết quả khi chạy trong Test folder (boundingbox đỏ gồm sign ID và confidence value):



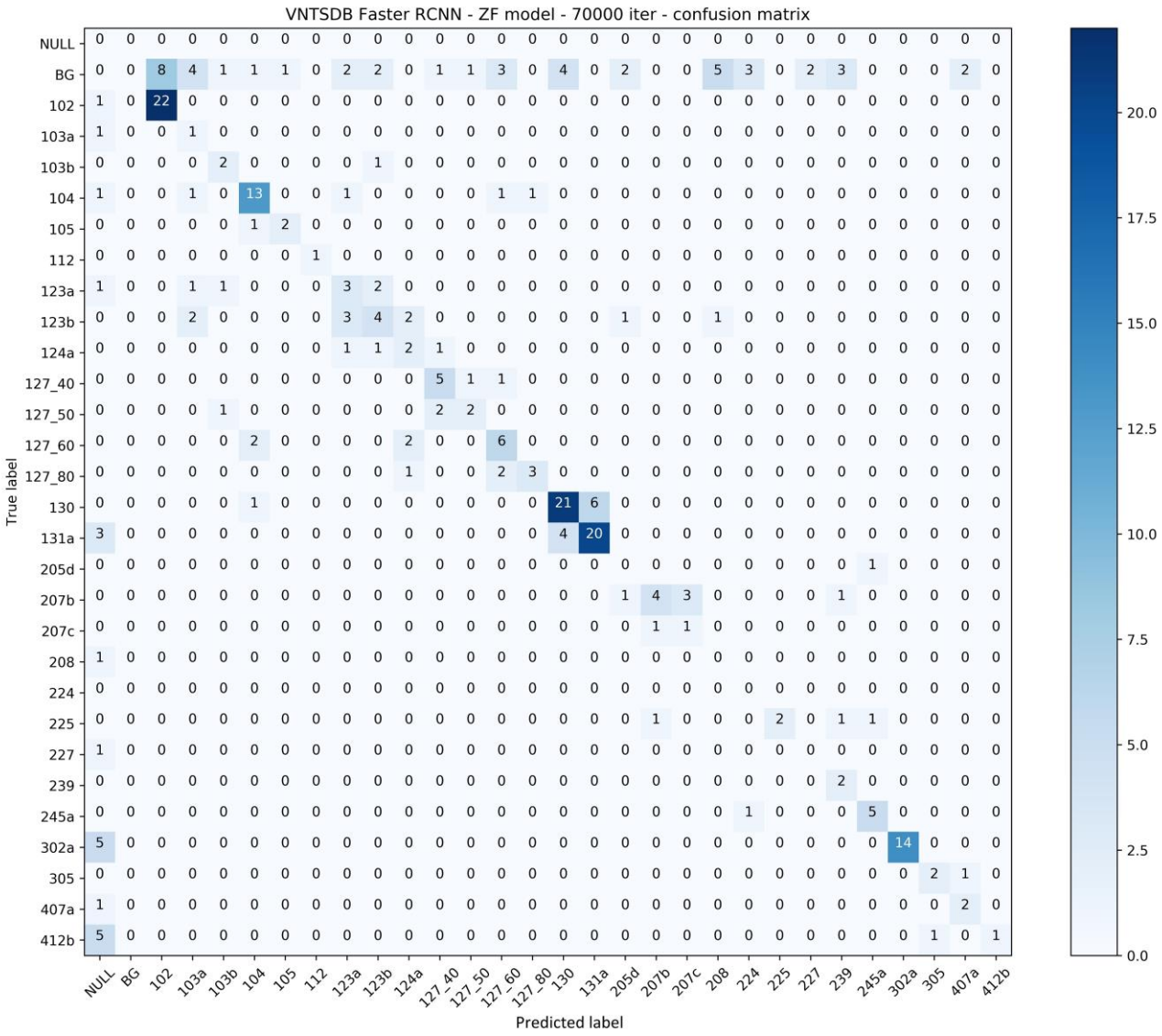





Xem toàn bộ kết quả cho 106 ảnh test tại:
https://drive.google.com/drive/u/0/folders/1vhNnT7uVkNGb3aKUdCvq3Lfejch5dUsu

- Độ chính xác của model: **53.23**%
  (Tính bằng sklearn.metrics.accuracy_score)

- Confusion matrix (theo số lượng):
  Ý nghĩa: X Y
  102 NULL – Groundtruth có biển 102 nhưng không được detect
  BG 102 – Background của ảnh bị detect nhầm thành biển 102



Confusion matrix trên model ZF – 70000 iter

6. KẾT LUẬN

❖ **Đánh giá:**

- Dựa theo confusion matrix cho thấy các biển báo có số lượng ảnh training càng lớn thì càng chính xác như các biển 130, 131a, 102
- Biển 130, 131a thường hay bị detect lộn cho nhau do sự giống nhau giữa 2 biển này.
- Các biển 205d, 208, 224, 227 có độ chính xác là 0%, nguyên nhân nhận định ban đầu: do số lượng ảnh train ít, biển báo có nhiều chi tiết nhỏ, phức tạp.

❖ **Các hướng cải tiến:**

- Sử dụng model CNN tính feature map tốt hơn như: VGG16, resnet-101
- Thay đổi anchors với kích thước nhỏ hơn (vì các biển báo thường kích thước nhỏ) ở RPN
- Cải thiện dataset: thu thập thêm ảnh training.
- Tăng số lần lặp epoch. Sử dụng learning rate biến đổi: lớn ở các epoch đầu, giảm dần ở các epoch cuối.

7. TÀI LIỆU THAM KHẢO

[1] Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks - Shaoqing Ren, Kaiming He, Ross Girshick, Jian Sun

http://arxiv.org/abs/1506.01497

[2] Source: Faster RCNN by Python + caffe

https://github.com/rbgirshick/py-faster-rcnn

[3] Source: Faster RCNN on GTSDB

https://github.com/sridhar912/tsr-py-faster-rcnn

[4] Stanford CS231 course:

http://cs231n.stanford.edu/slides/2016/winter1516_lecture8.pdf

[5] Caffe installation Guide

https://github.com/BVLC/caffe/wiki/Ubuntu-16.04-or-15.10-Installation-Guide

Phần 2:
THỰC NGHIỆM MỘT SỐ PHƯƠNG PHÁP PHÂN LOẠI ẢNH SỬ DỤNG DEEP MODEL

(Biên tập nội dung: Trần Xuân Hải)

Phần 2 - Task 1

THỰC NGHIỆM: ĐỘ CHÍNH XÁC CỦA CÁC THƯ VIỆN LIBLINEAR, LIBSVM_LINEAR, LIBSVM_RBF TRÊN DATASET VIETNAM TRAFFIC SIGNS VỚI VGG16 DEEP FEATURES

Source code:

https://github.com/hanghi/Survey-LibSVM-LibLinear

1. RÚT TRÍCH ĐẶC TRƯNG ẢNH

Sử dụng thư viện Keras[1] với model VGG16[2] có weights được pre-trained trên ImageNet.

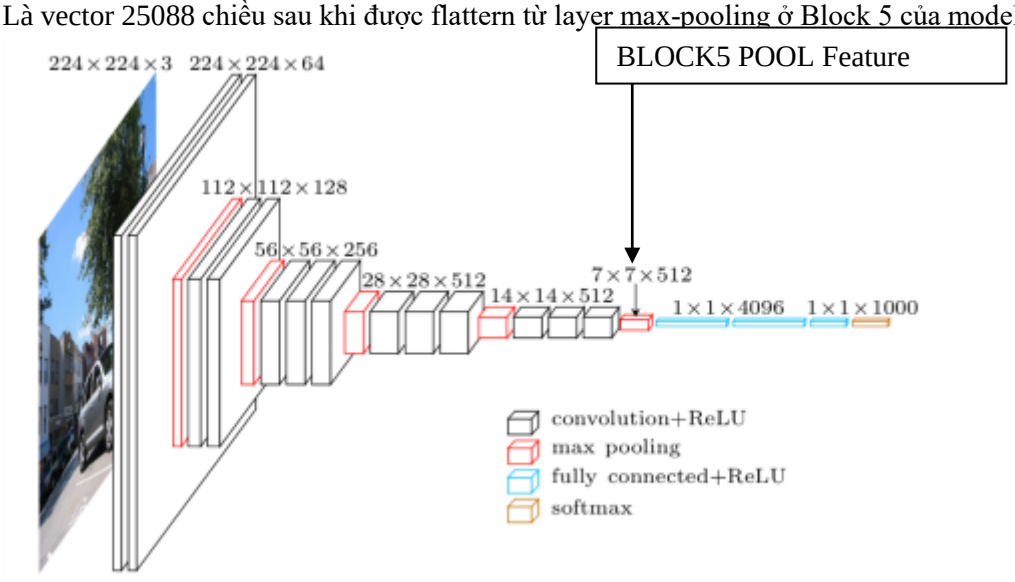
Input: Một bức ảnh

Output: Vector Deep feature (dữ liệu trên Block 5 pool hoặc, Fully-connected 2)

Đặc trưng VGG16 – BLOCK5_POOL [6]

Đặc trưng

Là vector 25088 chiều sau khi được flattern từ layer max-pooling ở Block 5 của model VGG16.



(Source image: http://www.cs.toronto.edu/~frossard/post/vgg16/)

Phương pháp rút trích đặc trưng cho ảnh

Với model VGG16 đã được pre-trained trên ImageNet, sử dụng thư viện Keras để lấy ra đặc trưng học sâu (MaxPooling, FC) của model khi predict một hình ảnh input.

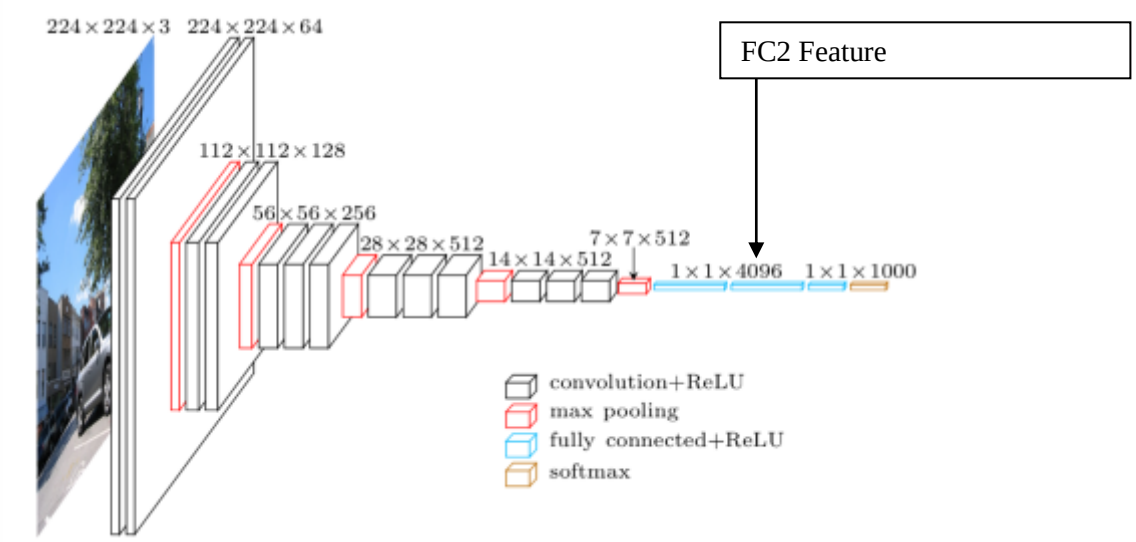
[1] https://keras.io/applications/#vgg16

[2] Very Deep Convolutional Networks for Large-Scale Image Recognition - Karen Simonyan, Andrew Zisserman

Đặc trưng VGG16 – FC2 [6]

Đặc trưng

Là vector 4096 chiều được lấy ra từ … fully connected 2 của model VGG16



(Source image: http://www.cs.toronto.edu/~frossard/post/vgg16/)

Phương pháp rút trích đặc trưng cho ảnh

Với model VGG16 đã được pre-trained trên ImageNet, sử dụng thư viện Keras để lấy ra đặc trưng học sâu (MaxPooling, FC) của model khi predict một hình ảnh input.

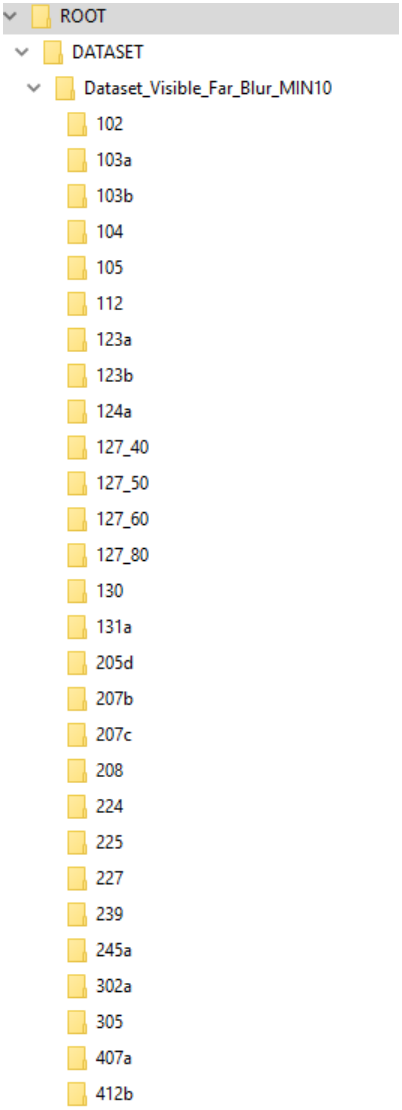
2 TỔ CHỨC DỮ LIỆU

a. *Dataset*

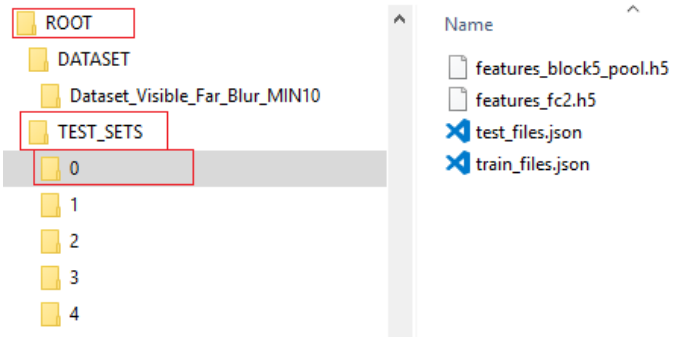
- Dataset được thu thập từ **Bài tập Thu thập Dataset** – là bộ các ảnh biển báo giao thông Việt Nam.
- Trong Dataset sử dụng cho bài tập này, gồm có:
  - *28 Lớp* – Thuộc 28 loại biển báo giao thông khác nhau thu thập được
  - Mỗi lớp chứa ít nhất 10 ảnh

b. *Tổ chức dữ liệu*

- **Dataset:** Dataset được lưu trong thư mục *ROOT\DATASET*, mỗi lớp là một thư mục con khác nhau, trong thư mục lớp chứa ảnh thuộc lớp đó.



- **Train-Test:** Các bộ train, test được lưu trong thư mục *ROOT\TEST_SETS*, mỗi bộ train, test được lưu trong thư mục con với tên là ID của bộ train, test. Trong đó gồm có:
  - `feature_.h5`: Binary file – Lưu các đặc trưng được rút trích
  - `train_files.json`: Lưu trữ index, nhãn, đường dẫn các ảnh dùng để train.
  - `test_files.json`: Lưu trữ index, nhãn, đường dẫn các ảnh dùng để test.



Cấu trúc file JSON lưu index:

```
[…
{
"id": 10,
"label_num": 3,
"label_str": "104",
"path": "104_IMG_000067.JPG"
}…
]
```

- **Phân chia dữ liệu:** Phân chia bộ train, test được thực hiện như sau.
  - Chạy qua số bộ train, test
  - Mỗi lớp random các ảnh và lấy số lượng train:test = 80:20
  - Lưu vào 2 mảng chính train, test
  - Lưu ID, Nhãn, Đường dẫn các ảnh train vào file `train_files.json`; Lưu ID, Nhãn, Đường dẫn các ảnh test vào file `test_files.json`

3 CHƯƠNG TRÌNH

- Dùng ngôn ngữ Python (version: 3.6) với các thư viện: `os, json, h5py, numpy, keras, operator, shutil, math`
- Chạy trên môi trường Windows 10 x64
- Chương trình bao gồm các file:
  - `helper.py` : Random các bộ train-test, rút trích deep features và xuất ra kết quả sau khi predict.
  - `using_liblinear.py`: sử dụng thư viện liblinear để khảo sát độ chính xác việc phân lớp
  - `using_libsvm.py`: sử dụng thư viện libsvm để khảo sát độ chính xác việc phân lớp svm_linear, svm_rbf
  - `survey.py`: file chạy chính
- Source Code: https://github.com/hanghi/Survey-LibSVM-LibLinear

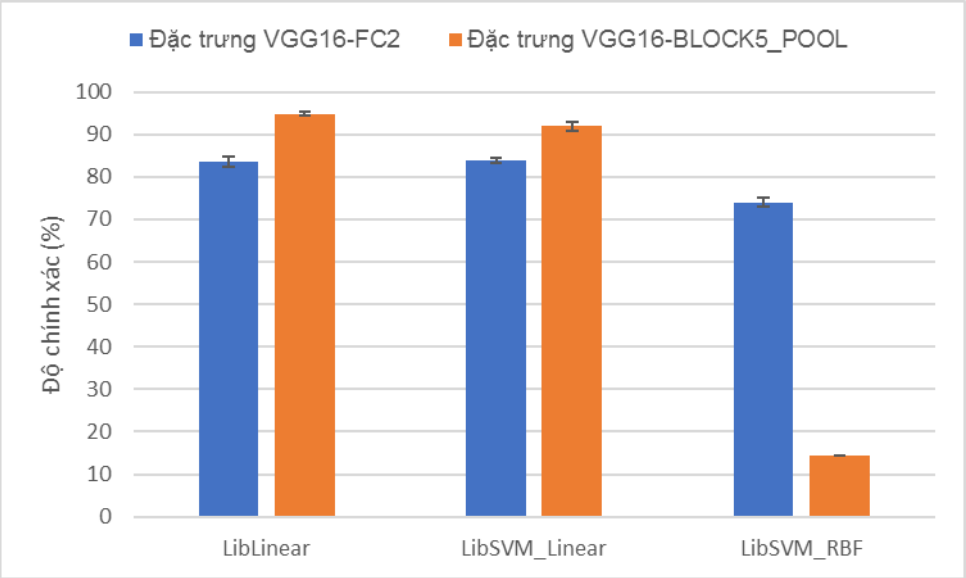
4 KẾT QUẢ

4.1 Bảng kết quả

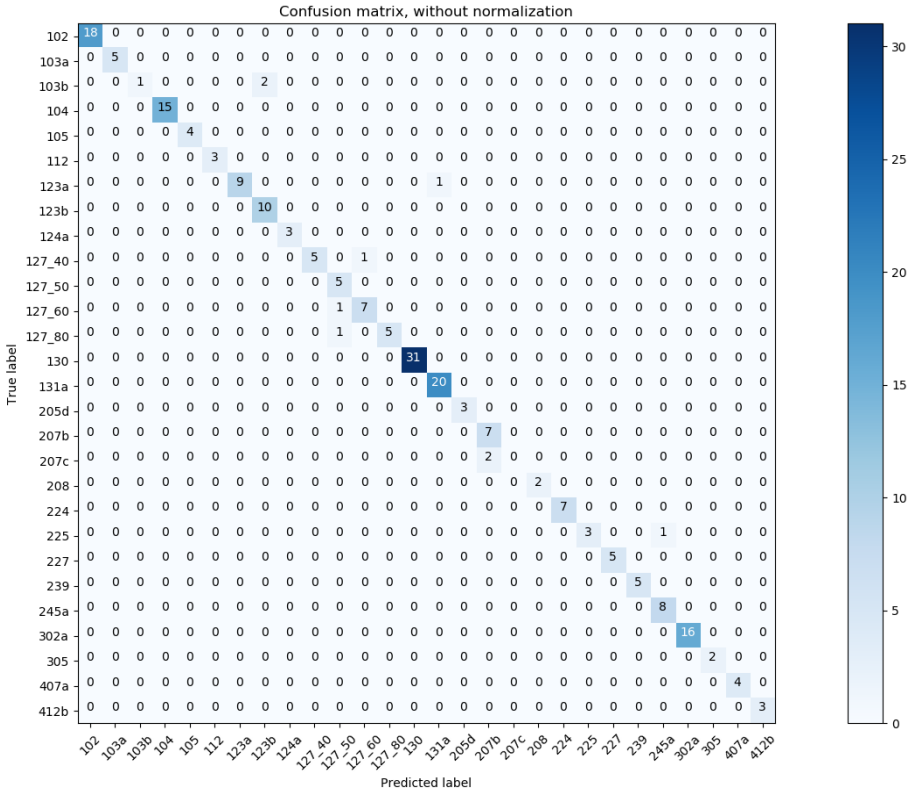
Sau khi chạy 5 bộ train-test.
Kết quả được trình bày trong bảng 1 như sau và được minh họa trong hình 2:

Phương pháp	Đặc trưng VGG16 – FC2	Đặc trưng VGG16 – BLOCK5_POOL
LibLinear	83.44 ± 1.22	94.88 ± 0.56
LibSVM_Linear	83.81 ± 0.59	91.90 ± 1.08
LibSVM_RBF	73.95 ± 1.11	14.42 ± 0

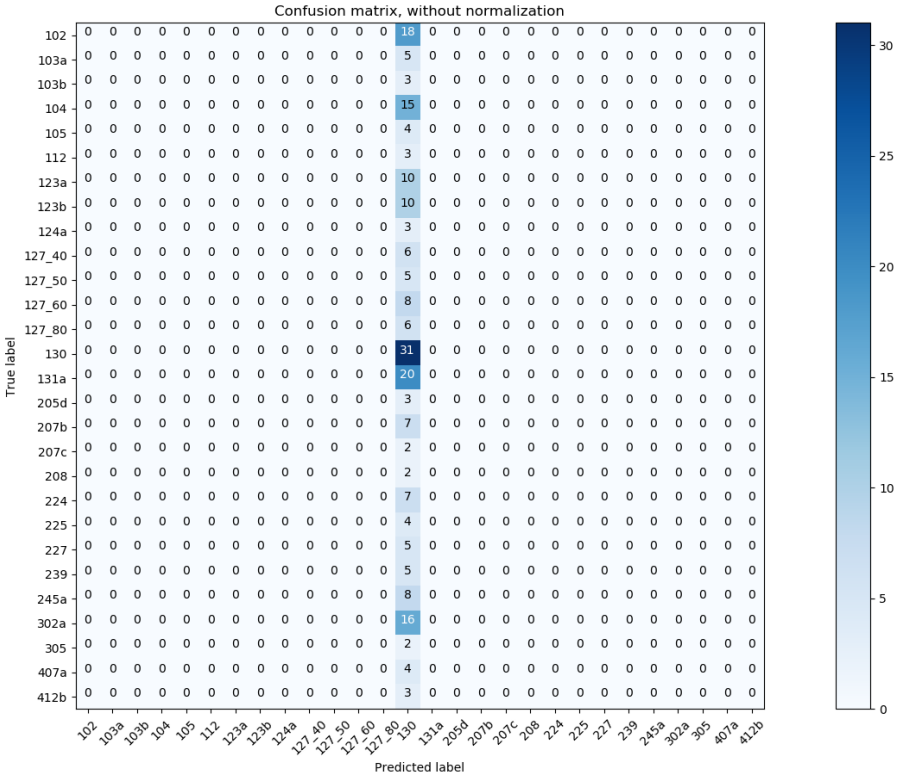
Bảng 1: Độ chính xác phân lớp trung bình sau khi chạy 5 bộ train-test



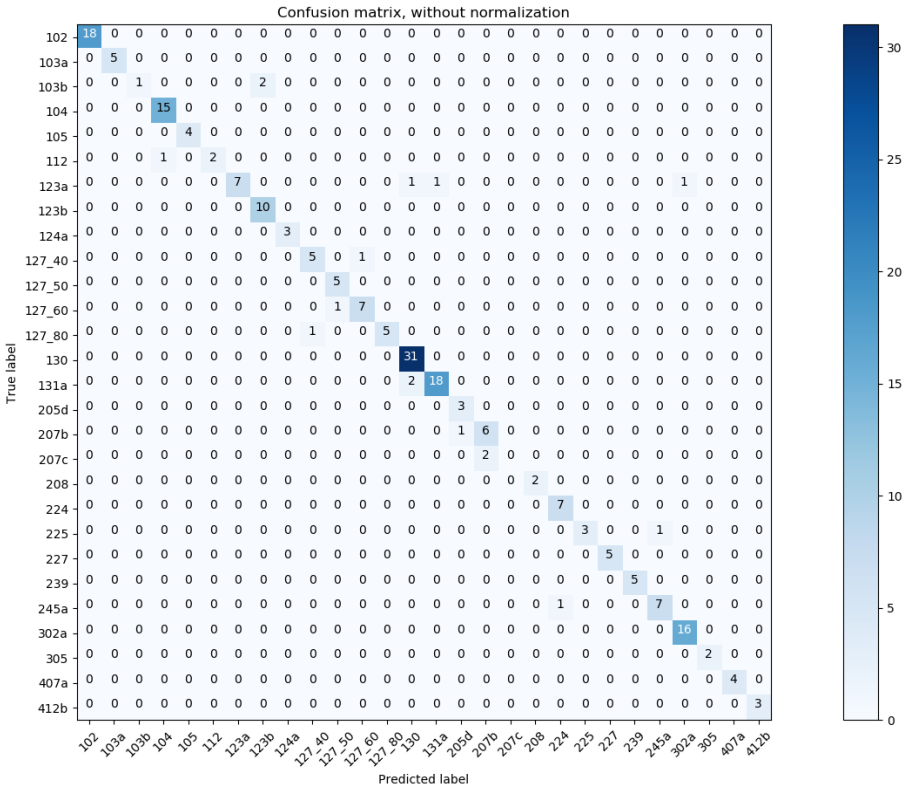
Hình 2: So sánh độ chính xác của các phương pháp.



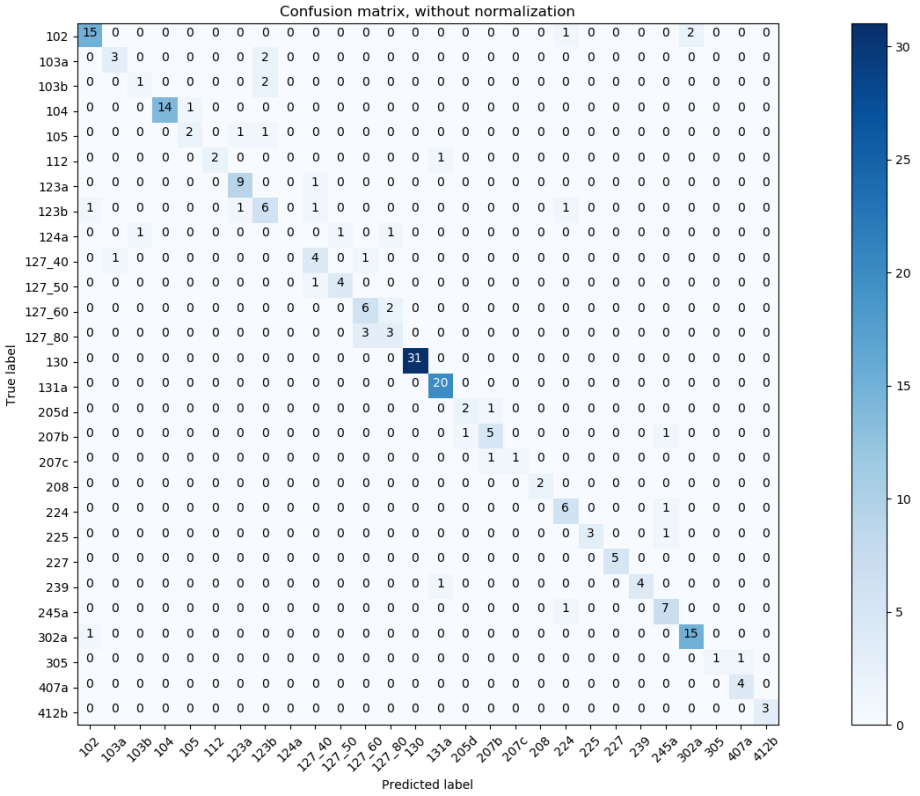
Hình 3: Confusion-matrix trong 1 lần test với feature Block5-Pool và LibLinear



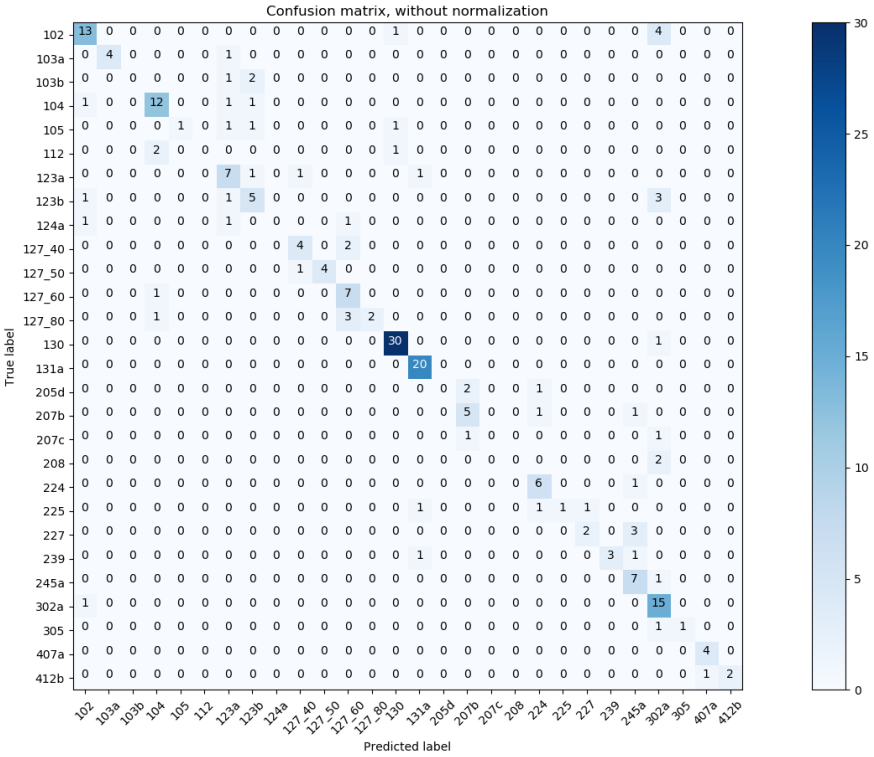
Hình 4: Confusion-matrix trong 1 lần test với feature Block5-Pool và LibSVM-RBF



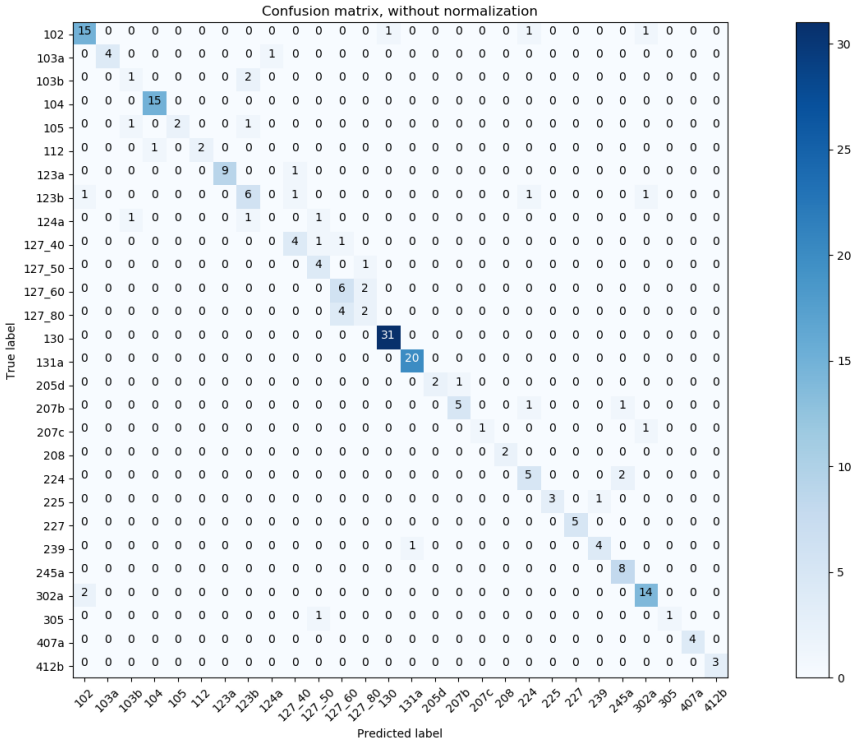
Hình 5: Confusion-matrix trong 1 lần test với feature Block5-Pool và LibSVM-Linear



Hình 6: Confusion-matrix trong 1 lần test với feature FC2 và LibSVM-Linear



Hình 7: Confusion-matrix trong 1 lần test với feature FC2 và LibSVM-RBF



Hình 8: Confusion-matrix trong 1 lần test với feature FC2 và LibLinear

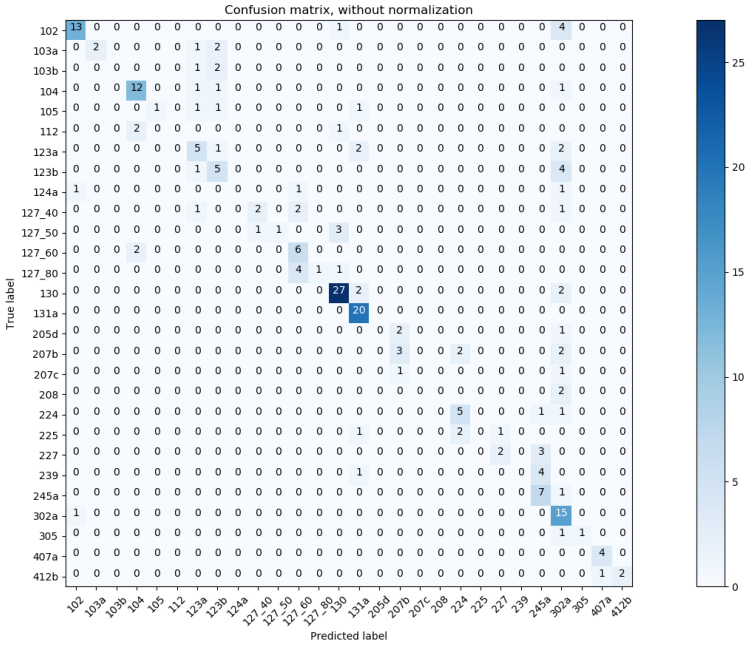
4.2 Thảo luận

- Rút trích đặt trưng Block 5 Pool nhanh nhưng tốc độ train chậm. Vì Block 5 Pool ở trước FC2 nên lấy đặt trưng nhanh còn vì Block 5 Pool tới 25088 chiều nên phân lớp chậm.
- Rút trích đặt trưng FC2 chậm nhưng tốc độ train nhanh. Vì FC2 ở gần cuối model nên lấy đặt trưng chậm còn vì FC2 chỉ có 4096 chiều nên phân lớp nhanh.

- Với đặc trưng FC2 cho kết quả khá tốt ở cả 3 thuật toán phân lớp Liblinear, LibSVM-Linear, LibSVM-RBF.
- Còn đặc trưng Block5-Pool cho kết quả tốt ở 2 thuật toán phân lớp LibLinear, LibSVM-Linear còn - cho kết quả tồi ở thuật toán LibSVM-RBF. Nguyên nhân LibSVM-RBF cho kết quả thấp là gamma quá nhỏ (gamma = 1/số chiều features = 1/25088).

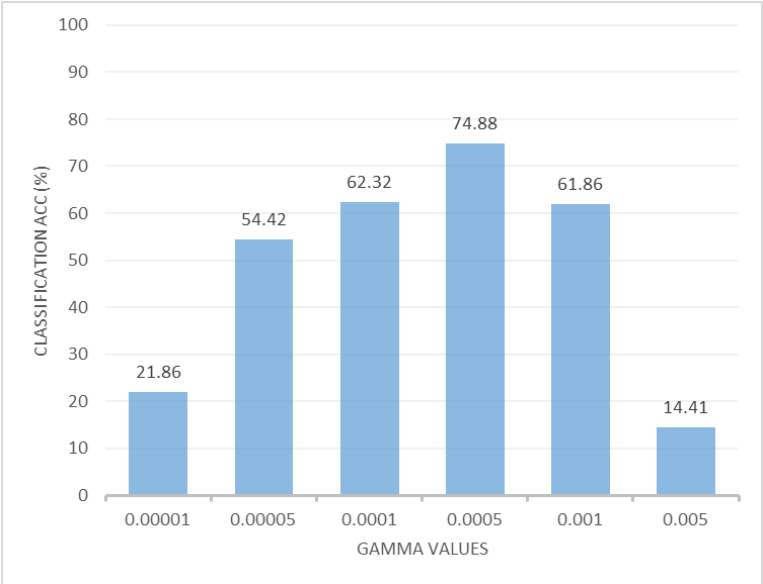
4.3 Hướng cải tiến

- Thực hiện lại LIBSVM-RBF với gamma bằng 0.0001
- Kết quả sau khi cải thiện:
  - Độ chính xác của LIBSVM-RBF tăng lên từ **14.42%** đến **62.33%**
- Confusion Matrix:



Hình 10: Confusion-matrix của LibSVM-RBF sau khi dùng gamma = 0.0001

- Một vài giá trị gamma khác cho độ chính xác khác nhau, cụ thể ở biểu đồ sau:



Hình 11. Biểu đồ tương quan giữa gamma và độ chính xác của thuật toán phân lớp LibSVM-RBF trên tập dữ liệu của phần này.

- Gamma từ 0.00001 đến 0.0005 cho thấy gamma tăng thì Accuracy tăng.
- Gamma từ 0.0005 đến 0.005 cho thấy gamma tăng nhưng Accuracy lại giảm.

5 KẾT LUẬN

- Thực hiện được cách rút trích deep feature của model VGG16.
- Thực nghiệm được 3 thuật toán phân lớp Liblinear, LibSVM-Linear, LibSVM-RBF.
- Thấy được sức mạnh và độ tốt của đặc trưng học sâu.

6 TÀI LIỆU THAM KHẢO

[6] "Keras Application," [Online]. Available: https://keras.io/applications/

[7] A. Z. Karen Simonyan, "Very Deep Convolutional Networks for Large-Scale Image Recognition".

[8] "LibLinear Python," [Online]. Available: https://github.com/ninjin/liblinear/tree/master/python

[9] "LibSVM Python," [Online]. Available: https://github.com/cjlin1/libsvm/tree/master/python

Phần 2 - Task 2

THỰC NGHIỆM: ĐỘ CHÍNH XÁC CỦA VGG16 MODEL KHI TRAIN BẰNG DATASET VIETNAM TRAFFIC SIGNS

Source Code:
https://github.com/hanghi/Train-VGG16-With-Custom-Dataset

1. TỔ CHỨC DỮ LIỆU

- Dataset được thu thập từ **Bài tập Thu thập Dataset** – là bộ các ảnh biển báo giao thông Việt Nam.
- Trong Dataset sử dụng cho bài tập này, gồm có:
  - *28 Lớp* – Thuộc 28 loại biển báo giao thông khác nhau thu thập được
  - Mỗi lớp chứa ít nhất 10 ảnh

Tổ chức dữ liệu

- **Dataset:** Dataset được lưu trong thư mục *ROOT\DATASET*, mỗi lớp là một thư mục con khác nhau, trong thư mục lớp chứa ảnh thuộc lớp đó.

- **Phân chia dữ liệu:** Phân chia 1 bộ train, test được thực hiện như sau.
  - Mỗi lớp random các ảnh và lấy số lượng train:test = 4:1
  - Lưu vào 2 mảng chính train, test
  - Lưu ID, Nhãn, Đường dẫn các ảnh train vào file `train_files.json`; Lưu ID, Nhãn, Đường dẫn các ảnh test vào file `test_files.json`
    Cấu trúc file JSON

```
[…
{
"id": 16,
"label_num": 0,
"label_str": "102",
"path": "102_IMG_000024.JPG"
}…
]
```

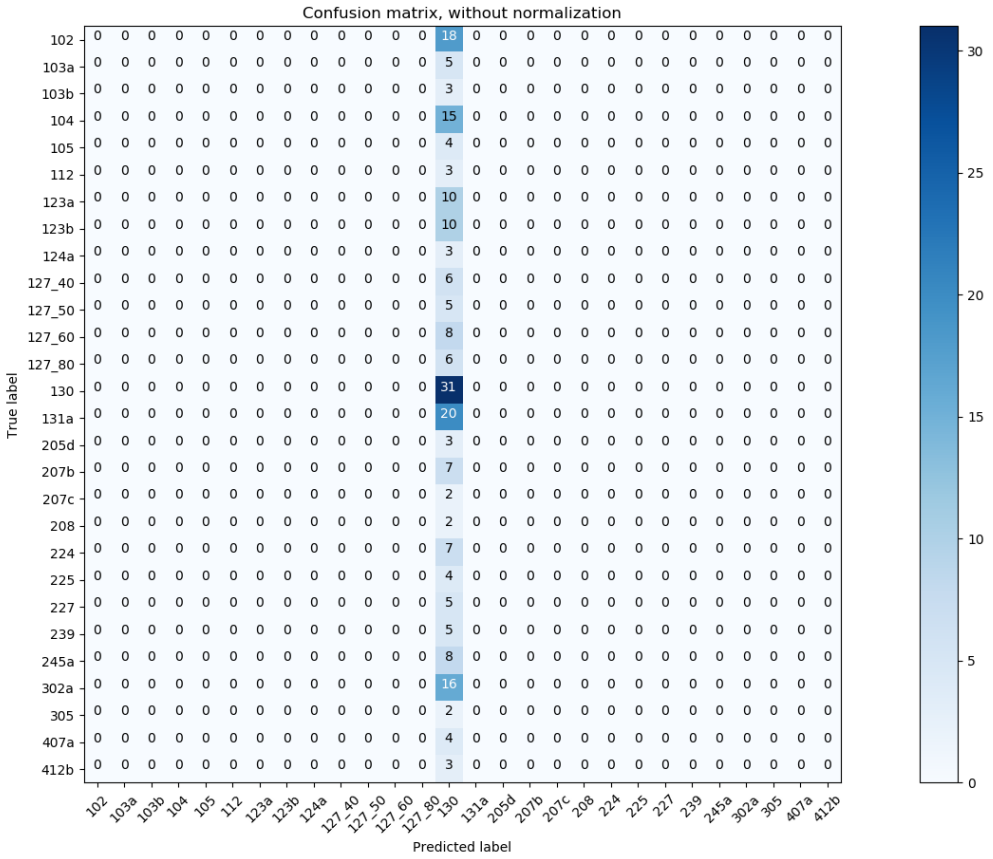
2. CHƯƠNG TRÌNH

- Dùng ngôn ngữ Python (version: 3.6) với các thư viện: `os, json, h5py, numpy, keras, operator, shutil, math`
- Chạy trên môi trường Windows 10 x64
- Chương trình bao gồm các file:
  - `helper.py` : Random các bộ train-test
  - `retrain.py`: File chương trình chính
- Khởi tạo model VGG16 với Keras, với tham số Learning-Rate = 0.05; Epochs = 1000; Batch Size = 64, Image Size = 50x50

```
adam = keras.optimizers.Adam(lr=0.05, beta_1=0.9, beta_2=0.999, epsilon=1e-08, decay=0.0)
pretrained_model.compile(loss='sparse_categorical_crossentropy', optimizer=adam, metrics=['accuracy'])
```

3. KẾT QUẢ

- Sau 1000 epoch, model thu được cho kết quả predict trên tập test là 2.28%



Hình 9: Confusion-matrix sau khi predict bằng model được train

- Kết quả thực sự thấp

4. THẢO LUẬN

- Kết quả cho thấy model đã train không được tốt
- Nguyên nhân có thể do dữ liệu quá ít, và dữ liệu không cân đối giữa các lớp. Riêng lớp 130 thì có nhiều hình ảnh hơn.

5. KẾT LUẬN

- Biết cách train VGG16 với dữ liệu tự tạo

6. TÀI LIỆU THAM KHẢO

[10] "Keras Application," [Online]. Available: https://keras.io/applications

[11] A. Z. Karen Simonyan, "Very Deep Convolutional Networks for Large-Scale Image Recognition".

[12] "LibLinear Python," [Online]. Available: https://github.com/ninjin/liblinear/tree/master/python

[13] Using pre trained VGG16 for another classification task: https://github.com/fchollet/keras/issues/4465#issuecomment-292745650

THÔNG TIN VỀ NHÓM THỰC HIỆN

Mã sinh viên	*Họ tên sinh viên*	*Nội dung thực hiện*	*Đánh giá hoàn thành công việc*
14520956	*Hoàng Hữu Tín*	• *Xây dựng cấu trúc, tổ chức Dataset.* • *Phân công công việc.* • *Kiểm tra kết quả của các thành viên khác* • *Thực hiện đi quay video, tách annotation* • *Upload, quản lý dataset trên Google Drive.* • *Thực hiện Phần 1: VNTS detection using Faster RCNN* • *Viết báo cáo Phần 1* • *Xây dựng các tool phụ trợ*	*100%*
14520247	*Trần Xuân Hải*	• *Đưa ra ý tưởng chọn biển báo giao thông làm đề tài.* • *Thực hiện đi quay video, tách annotation.* • *Thực hiện Phần 2: Classification on Deep Model* • *Viết báo cáo Phần 2* • *Xây dựng các tool phụ trợ*	*88%*
14520156	*Trần Quang Đạt*	• *Thực hiện đi quay video, tách annotation.* • *Liên hệ cấp UIT-Cloud* • *Thực hiện Phần 2: Classification on Deep Model* • *Đếm evaluation cho từng ảnh kết quả của model Faster RCNN*	*78%*