

ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
ĐẠI HỌC CÔNG NGHỆ THÔNG TIN
KHOA KHOA HỌC MÁY TÍNH



BÁO CÁO ĐỒ ÁN
MÔN: MÁY HỌC
ĐỀ TÀI: PHÂN LOẠI RÁC THẢI
GIẢNG VIÊN HƯỚNG DẪN:
PHẠM NGUYỄN TRƯỜNG AN
LÊ ĐÌNH DUY
THÀNH VIÊN NHÓM:
PHAN MINH NHẬT - 19521956
LÊ VÕ TIẾN PHÁT - 19521993
HỒ THỊNH - 19522274

MỤC LỤC

I. Tổng quan.....	2
1. Mô tả bài toán.....	2
2. Mô tả dữ liệu.....	2
II. Các nghiên cứu trước.....	2
III. Xây dựng bộ dữ liệu.....	5
1. Quá trình thu thập dữ liệu.....	5
2. Thông số của bộ dữ liệu.....	5
IV. Training và đánh giá model.....	7
1. Giới thiệu về model.....	7
2. Quá trình training.....	7
3. Phương pháp đánh giá.....	10
4. Test model.....	14
V. Ứng dụng và hướng phát triển.....	17
* Các nguồn tham khảo.....	18
* Nguồn các bài nghiên cứu.....	18

I. Tổng quan

1. Mô tả bài toán

Trong thời đại công nghệ 4.0, các công việc lớn nhỏ nặng nhọc hầu như đều được giao hết cho máy làm, con người đã rảnh rỗi hơn trong cuộc sống. Vì vậy việc phân loại rác cũng như thế. Khi đi thu rác từ các hộ gia đình, người ta thường đi một xe tải lớn để gom hết rác rồi mới đem về nhà máy phân loại. Việc phân loại đó ở nước ta trước giờ vẫn do con người làm là chính. Điều đó ảnh hưởng không nhỏ tới sức khỏe của con người khi làm việc trong môi trường khắc nghiệt như vậy. Vì vậy, chúng ta sẽ tạo ra một model dùng để phân loại rác tích hợp vào robot để có thể giải quyết chúng.

- **Input:** 1 bức ảnh có một hoặc nhiều loại rác. Ảnh được đưa vào phải chụp ở nơi đầy đủ ánh sáng, càng gần trung tâm bức ảnh càng tốt.
- **Output:** 1 bức ảnh có bounding box cho từng loại rác có trong ảnh thuộc 1 trong 3 class: Easy Decompose, Hard Decompose, Recyclable.

2. Mô tả dữ liệu

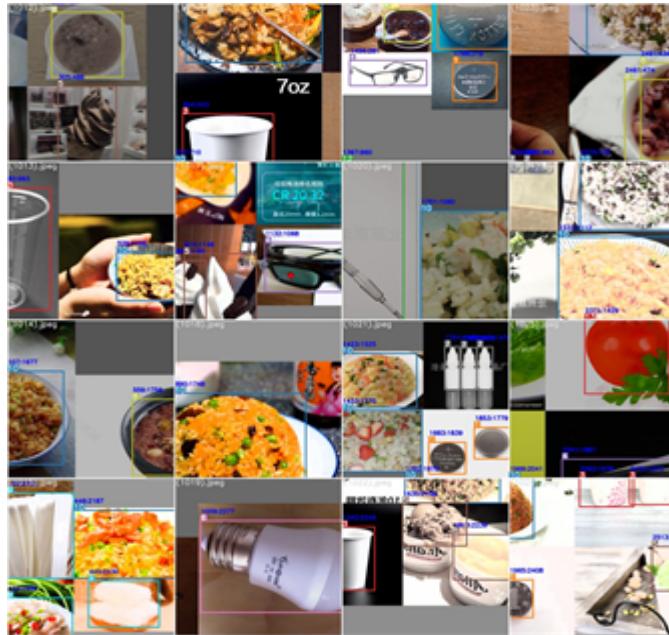
Dữ liệu và cách thu thập: Mục tiêu ban đầu nhóm đặt ra là thu thập khoảng 2000 ảnh thuộc 3 class và data sẽ do nhóm em tự chụp. Một lý do mà nhóm em không sử dụng data có sẵn trên Kaggle vì muốn tạo ra một model có tính thiết thực hơn (data trên kaggle chỉ chia ra các loại rác cụ thể như: paper, plastic, metal, cardboard, glass,... và ảnh không phù hợp với ngữ cảnh đặt ra).

II. Các nghiên cứu trước

• Garbage Classification Detection Based on Improved YoloV4:

Tác giả: Qingqiang Chen, Qianghua Xiong from Shenzhen University and Dongguan University of Technology.

Dataset: 22000 ảnh chia thành 3 nhóm gồm: rác thải nguy hiểm, rác thải từ nhà bếp và các loại rác khác. Trong từng nhóm chia thành 5 loại khác nhau như là pin (1223 ảnh), pin dạng nút (1450 ảnh), băng sửa lỗi (894 ảnh), ly dùng một lần (1500 ảnh), thủy tinh (1866 ảnh), kem (1450 ảnh), hộp thuốc (1635 ảnh), bóng đèn (1168 ảnh), ngũ cốc (1422 ảnh), các loại mì (1976 ảnh), cơm (1670 ảnh), bao tay (1340 ảnh), nhiệt kế (1388 ảnh), cà chua (1233 ảnh), bàn chải đánh răng (1785 ảnh). Chia data ra làm 2 tập train và test với tỉ lệ 7/3. Dữ liệu được đánh nhãn bằng LabelImg.



Tăng cường dữ liệu: trong nghiên cứu này sử dụng các công cụ có sẵn của Yolov4 để chỉnh sửa hình ảnh (chỉnh độ sáng, độ tương phản, màu sắc, độ bão hoà, độ nhiễu hình ảnh), biến đổi hình học (phóng to, cắt, lật ảnh), biến dạng trắc quang (Random Erase, Cutout, Hide and Seek, Grid Mask, trộn hình ảnh bằng CutMix và Mosaic). Ngoài ra, người ta còn kết hợp hai ảnh tỷ lệ khác nhau để thu được khung hình hợp lý.

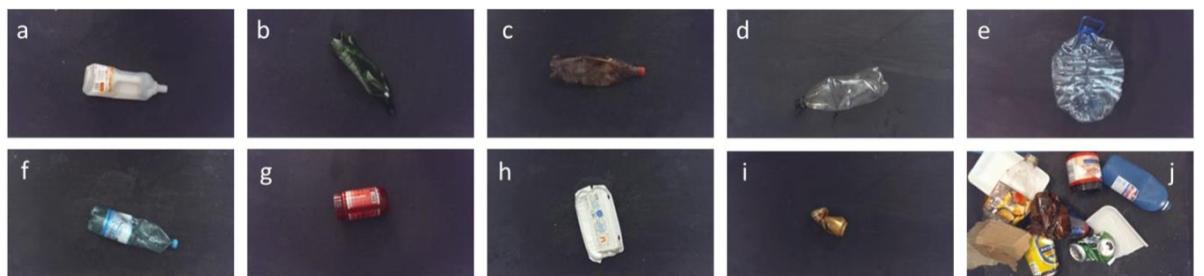
Kết quả train-test: Bài nghiên cứu thống kê được khi iterations từ 200-500 thì Complete-IoU Loss giảm đáng kể và độ chính xác được cải thiện. Từ 700-1200 hiệu suất đào tạo dần ổn định. Và giá trị mAP của model Improved Yolov4 thu được là 64%. Qua quá trình thống kê thì người ta kết luận model Improved Yolov4 có thể nhận diện và phân loại rác trong thời gian thực.

Dù có lượng data khá lớn nhưng chỉ số mAP lại không quá cao, có thể do train chưa đủ lâu vì model Yolov4 với lượng data lớn như thế này thì 1200 vòng train vẫn còn khá ít.

- Development of a method of detection and classification of waste objects on a conveyor for a robotic sorting system:**

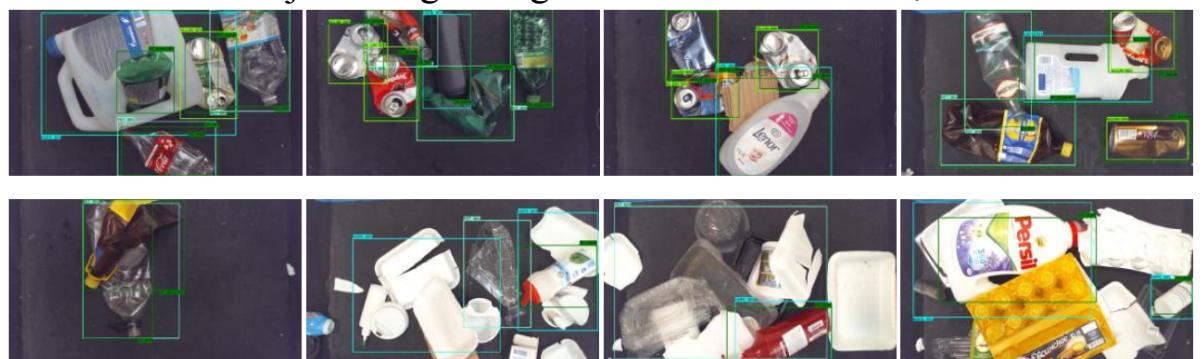
Tác giả: A V SeredKin, M P Tokarev, I A Plohih, O A Gobyzov and D M Markovich from Russia.

Bài nghiên cứu này có ngữ cảnh ứng dụng khá giống với đề tài của nhóm, đều là nhận diện và phân loại rác trên băng chuyền. Họ thu thập data bằng cách bỏ các loại rác lên nền màu đen để chụp, điều này giúp cho bộ data tương đồng với ngữ cảnh ứng dụng đã đề ra hơn.



Dataset: Hơn 14000 ảnh MSW (Municipal solid waste- 1 trong những loại chất thải từ hộ gia đình, công nghiệp,...) được chia làm 3 loại chính polyethylene terephthalate (PET), high-density polyethylene (HDPE), aluminum, và các loại khác. Các object thuộc lớp PET bottles được gán nhãn bằng màu, còn lại các loại khác được gán nhãn dựa vào chất liệu. Chi tiết dataset theo nhóm: HDPE (4978 ảnh), Aluminum(3411), PET Transparent(1749 ảnh), PET Dark/Green/Blue/Teal/Multicolor (1477/511/1041/309/52 ảnh), các loại khác (1007 ảnh).

Kết quả train: sau khi train model bằng R-CNN, họ test model với 729 ảnh có nhiều object trong khung hình thì chỉ số mAP đạt 55%.



Tăng cường dữ liệu: Sử dụng thuật toán watershed segmentation algorithm để cắt những đối tượng đơn lẻ từ ảnh ban đầu theo ranh giới tự nhiên của chúng tay vì bị giới hạn bởi bounding box. Ngoài ra, ở bài nghiên cứu này còn thay đổi độ sáng, độ tương phản, độ bão hòa của ảnh.

Kết quả sau tăng cường: Vẫn còn một số lỗi xảy ra, mAP khi thực hiện nhận diện và phân loại trên ảnh thực tế đạt 64.1%.

Nhận xét: Chỉ số mAp khi test ảnh là không cao bởi các bức ảnh này có nhiều loại rác ở gần nhau, nằm chồng lên nhau khiến cho việc nhận diện trở nên khó khăn hơn, khiến cho model không thể nhận diện được hoặc phân loại sai.

III. Xây dựng bộ dữ liệu

1. Quá trình thu thập dữ liệu

Ý tưởng của nhóm khi bắt đầu thu thập data là làm sao cho ảnh để train có background giống như băng chuyên ở nhà máy phân loại rác. Nên mọi người trong nhóm đã mua decal đen để làm background chụp ảnh. Nhóm đặt ra các tiêu chí chung khi chụp hình là ảnh phải chụp nơi có đủ ánh sáng, nhìn thấy rõ được vật thể, góc chụp thẳng đứng từ trên xuống, khoảng cách chụp khoảng 50cm.

2. Thông số của bộ dữ liệu

Bộ data của nhóm gồm 2138 ảnh được chia làm 3 class:

- + Easy Decompose : 522 ảnh
- + Hard Decompose : 654 ảnh
- + Recyclable : 962 ảnh

Một số ảnh trong bộ dữ liệu:

Easy Decompose

Hard Decompose

Recyclable

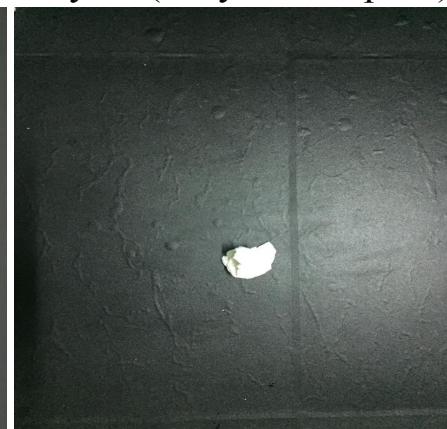
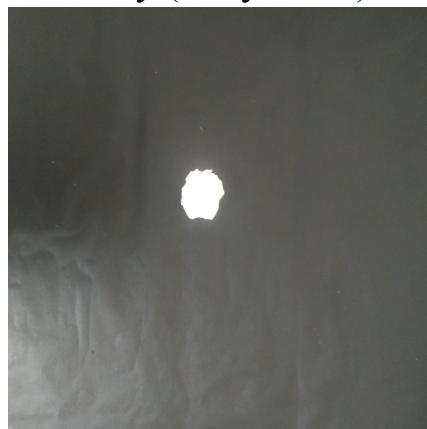


Các trường hợp khó xử lý trong bộ dữ liệu:

Khi giấy và giấy ăn vo tròn lại nhìn khá giống nhau:

Giấy (Recyclable)

Giấy ăn (Easy Decompose)



Lọ thủy tinh và lọ nhựa trong suốt cũng khá khó để phân biệt:

Thủy tinh (Hard Decompose)

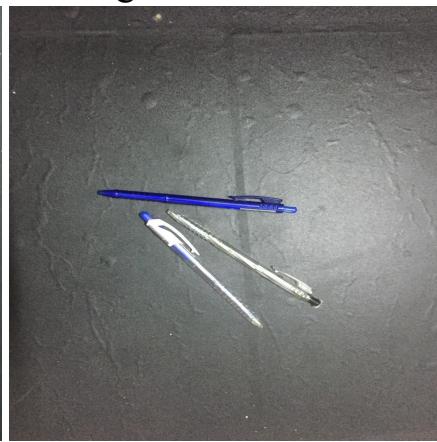
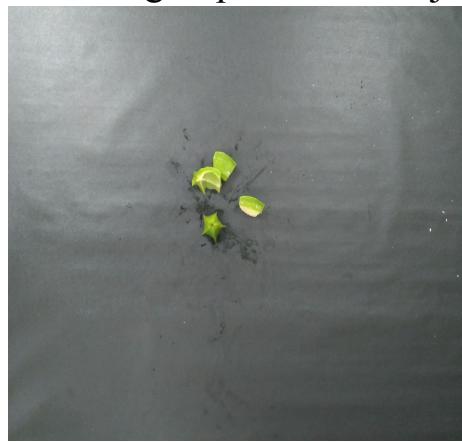


Nhựa (Recyclable)



Các trường hợp trên khá khó phân biệt bằng mắt thường, trong bộ dữ liệu có khá ít các trường hợp như vậy, chỉ 4 cặp ảnh nhìn giống nhau.

Các trường hợp có nhiều object nằm gần nhau:



Có khoảng hơn 80 ảnh như vậy trong bộ dữ liệu, nằm ở chủ yếu ở hai class Easy Decompose và Hard Decompose.

Nhìn chung, trong thực tế có rất nhiều loại rác khác nhau và mỗi loại rác cũng có nhiều hình dạng khác nhau, chúng còn nằm sát nhau hay nằm chồng lên nhau. Đây cũng là một khó khăn nếu muốn áp dụng model vào thực tế, ta sẽ cần một lượng data rất lớn và đa dạng.

IV. Training và đánh giá model

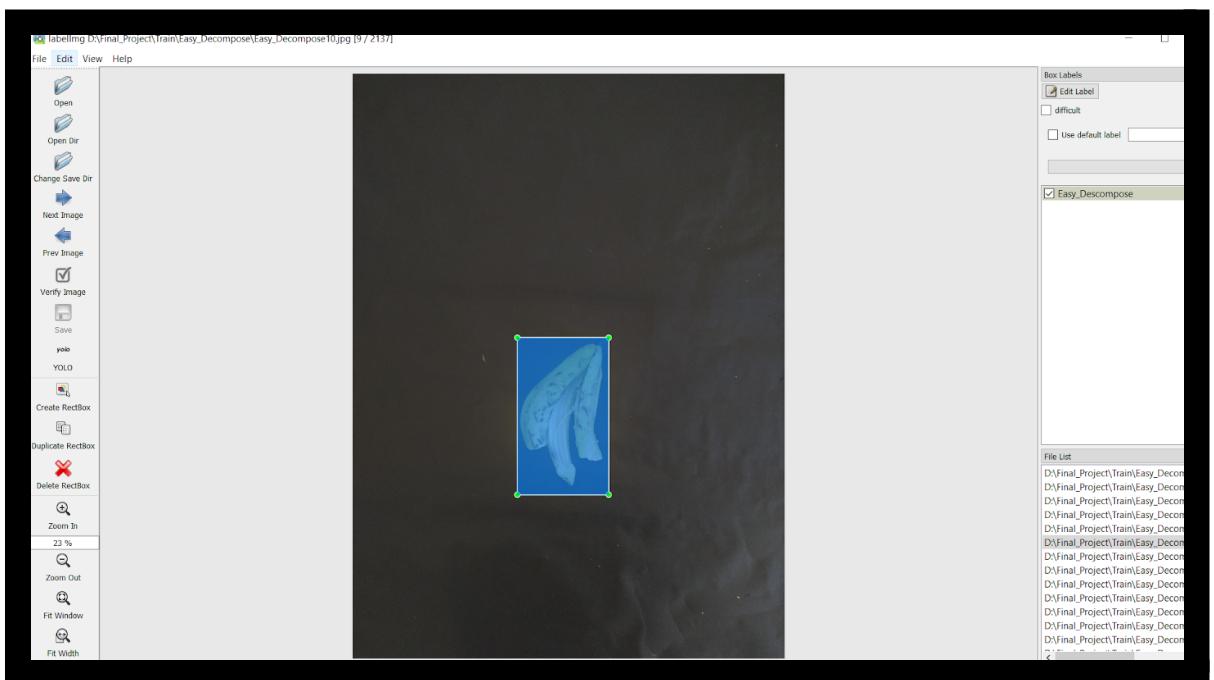
1. Giới thiệu về model

Nhóm quyết định sử dụng giải thuật Object Detection khá tốt hiện nay là: YOLOv4 (You Only Look Once) .

Yolo là một mô hình mạng CNN cho việc phát hiện, nhận dạng, phân loại đối tượng. Yolo được tạo ra từ việc kết hợp giữa các convolutional layers và connected layers. Trong đó các convolutional layers sẽ trích xuất ra các feature của ảnh, còn full-connected layers sẽ dự đoán ra xác suất đó và tọa độ của đối tượng.

2. Quá trình training

Sau khi thu thập data thì nhóm tiến hành gán nhãn cho đối tượng bằng LabelImg Tool (phần mềm gán nhãn cho hình ảnh, phục vụ cho mục đích đào tạo các mô hình Deep learning).



Việc train model sẽ thực hiện trên Google Colab nên việc đầu tiên sẽ là liên kết colab với Google Drive.

```
▶ from google.colab import drive  
drive.mount('/content/gdrive')  
  
↳ Mounted at /content/gdrive
```

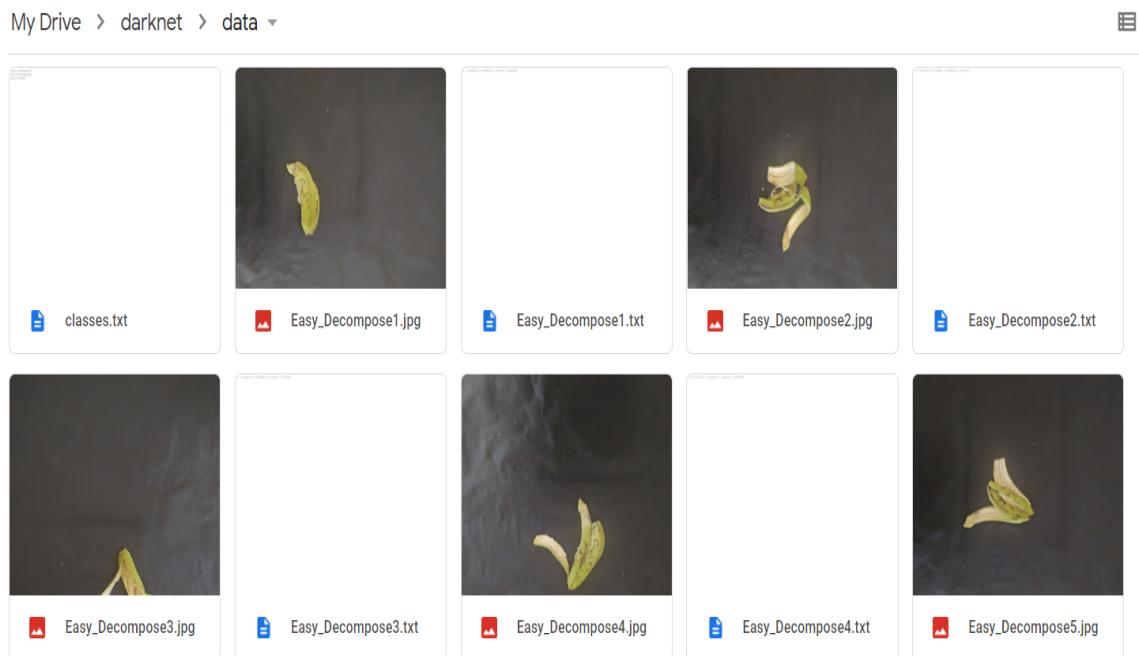
Tiếp theo tải model về drive:

```
%cd /content/gdrive/MyDrive/  
!git clone https://github.com/AlexeyAB/darknet
```

Tạo thêm thư mục backup trong thư mục darknet để lưu kết quả train:

My Drive > darknet > backup ▾

Đưa toàn bộ ảnh và label vào thư mục darknet/data trên Google Drive sau khi đã xóa dữ liệu trong thư mục data có sẵn:



Tạo file train.txt (chứa địa chỉ của các file ảnh đưa vào quá trình train model) và file val.txt (chứa địa chỉ của các file ảnh đưa vào quá trình test và đánh giá model).

Chia tập train và validation với tỉ lệ 8/2.

Tập train: 1748 ảnh

Tập validation: 390 ảnh

```

files = []
for ext in ["*.png", "*.jpeg", "*.jpg"]:
    image_files = glob2.glob(os.path.join("data/", ext))
    #tìm kiếm các file có đuôi dạng: png, jpeg, jpg trong thư mục data
    print(image_files)
    files += image_files
    #lưu tất cả các file vào biến files
    print(len(files))

nb_val = math.floor(len(files)*0.2)
#Biến nb_val được gán bằng 0.2 độ dài của biến files(chứa tất cả các tên ảnh)
rand_idx = np.random.randint(0, len(files), nb_val)
#Chọn ngẫu nhiên các ảnh trong biến files

# Tạo file train.txt
with open("/content/gdrive/My Drive/darknet/train.txt", "w") as f:
    for idx in np.arange(len(files)):
        if (os.path.exists(files[idx][:-3] + "txt")):
            f.write(files[idx]+'\n')

# Tạo file vali.txt
with open("/content/gdrive/My Drive/darknet/val.txt", "w") as f:
    for idx in np.arange(len(files)):
        if (idx in rand_idx) and (os.path.exists(files[idx][:-3] + "txt")):
            f.write(files[idx]+'\n')

```

Để cho model yolov4 có thể chạy được, ta cần thêm và chỉnh sửa một số file cấu hình.

Tạo 2 file cấu hình:

yolo.data	yolo.names
<pre> 1 classes=3 2 train=train.txt 3 valid=val.txt 4 names=yolo.names 5 backup=backup </pre>	<pre> 1 Easy_Decompose 2 Hard_Decompose 3 Recyclable </pre>

Sửa file Makefile:

Makefile
<pre> 1 GPU=1 2 CUDNN=1 3 CUDNN_HALF=1 4 OPENCV=1 </pre>

Sửa file yolov4-custom.cfg theo hướng dẫn của tác giả model:

- Sửa batch = 64 và subdivisions = 16 ở dòng 6,7.
- Sửa width và height ở dòng 8,9 thành 416 để không bị lỗi Out of Memory khi train.
- Sửa max_batches = 6000 (max_batches = classes*2000) ở dòng 20.
- Sửa step = 4800, 5400 (80% và 90% của max_batches) ở dòng 22.
- Sửa classes = 3 ở dòng 610,696 và 783.
- Sửa filters = 24 ở dòng 603,689 và 776 (filters = (classes + 5)*3).

Tiếp theo, tải pretrain cho model:

```
%cd /content/gdrive/MyDrive/darknet  
!wget https://github.com/AlexeyAB/darknet/releases/download/darknet_yolo_v3_optimal/yolov4.conv.137
```

Chuyển darknet thành folder executable :

```
%cd /content/gdrive/MyDrive/darknet  
!make clean  
!make  
!chmod +x ./darknet
```

Tiến hành train model:

```
%cd /content/gdrive/MyDrive/darknet  
./darknet detector train yolodata cfg/yolov4-custom.cfg yolov4.conv.137 -dont_show -map  
v3 (iou loss, Normalizer: (iou: 0.07, obj: 1.00, cls: 1.00) Region 161 Avg (IOU: 0.608476), count: 5, class_loss = 0.999325, iou_loss = 0.232689, total_loss = 1.2320%  
total_bbox = 267533, rewritten_bbox = 0.102791 %  
v3 (iou loss, Normalizer: (iou: 0.07, obj: 1.00, cls: 1.00) Region 139 Avg (IOU: 0.394697), count: 1, class_loss = 0.787041, iou_loss = 0.295404, total_loss = 1.0820%  
v3 (iou loss, Normalizer: (iou: 0.07, obj: 1.00, cls: 1.00) Region 150 Avg (IOU: 0.700689), count: 13, class_loss = 7.477744, iou_loss = 3.597215, total_loss = 11.01%  
v3 (iou loss, Normalizer: (iou: 0.07, obj: 1.00, cls: 1.00) Region 161 Avg (IOU: 0.577378), count: 9, class_loss = 2.690634, iou_loss = 0.356108, total_loss = 3.046%  
total_bbox = 267556, rewritten_bbox = 0.102782 %  
  
Tensor Cores are disabled until the first 3000 iterations are reached.  
(next mAP calculation at 1000 iterations)  
569: 2.539852 2.863127 avg loss, 0.000105 rate, 3.395169 seconds, 36416 images, 39.660037 hours left  
Loaded: 21.198427 seconds - performance bottleneck on CPU or Disk HDD/SSD  
v3 (iou loss, Normalizer: (iou: 0.07, obj: 1.00, cls: 1.00) Region 139 Avg (IOU: 0.345547), count: 1, class_loss = 0.791665, iou_loss = 2.849281, total_loss = 3.640%  
v3 (iou loss, Normalizer: (iou: 0.07, obj: 1.00, cls: 1.00) Region 150 Avg (IOU: 0.620138), count: 11, class_loss = 8.654719, iou_loss = 1.774378, total_loss = 10.41%  
v3 (iou loss, Normalizer: (iou: 0.07, obj: 1.00, cls: 1.00) Region 161 Avg (IOU: 0.546257), count: 9, class_loss = 5.060030, iou_loss = 0.608354, total_loss = 5.663%  
total_bbox = 267577, rewritten_bbox = 0.102774 %  
v3 (iou loss, Normalizer: (iou: 0.07, obj: 1.00, cls: 1.00) Region 139 Avg (IOU: 0.759391), count: 3, class_loss = 2.300085, iou_loss = 4.655926, total_loss = 6.956%  
v3 (iou loss, Normalizer: (iou: 0.07, obj: 1.00, cls: 1.00) Region 150 Avg (IOU: 0.623784), count: 18, class_loss = 11.641280, iou_loss = 4.964728, total_loss = 16.11%  
v3 (iou loss, Normalizer: (iou: 0.07, obj: 1.00, cls: 1.00) Region 161 Avg (IOU: 0.592269), count: 7, class_loss = 3.156475, iou_loss = 0.573509, total_loss = 3.723%  
total_bbox = 267605, rewritten_bbox = 0.102763 %  
v3 (iou loss, Normalizer: (iou: 0.07, obj: 1.00, cls: 1.00) Region 139 Avg (IOU: 0.795095), count: 3, class_loss = 1.776867, iou_loss = 3.259384, total_loss = 5.036%  
v3 (iou loss, Normalizer: (iou: 0.07, obj: 1.00, cls: 1.00) Region 150 Avg (IOU: 0.618387), count: 9, class_loss = 4.960433, iou_loss = 2.809111, total_loss = 7.769%  
v3 (iou loss, Normalizer: (iou: 0.07, obj: 1.00, cls: 1.00) Region 161 Avg (IOU: 0.596817), count: 5, class_loss = 1.873576, iou_loss = 0.781204, total_loss = 2.654%  
total_bbox = 267622, rewritten_bbox = 0.102757 %  
v3 (iou loss, Normalizer: (iou: 0.07, obj: 1.00, cls: 1.00) Region 139 Avg (IOU: 0.539666), count: 2, class_loss = 1.644429, iou_loss = 1.739868, total_loss = 3.384%  
v3 (iou loss, Normalizer: (iou: 0.07, obj: 1.00, cls: 1.00) Region 150 Avg (IOU: 0.509552), count: 11, class_loss = 6.441649, iou_loss = 1.829018, total_loss = 8.271%  
v3 (iou loss, Normalizer: (iou: 0.07, obj: 1.00, cls: 1.00) Region 161 Avg (IOU: 0.583963), count: 7, class_loss = 2.938732, iou_loss = 0.328335, total_loss = 3.267%  
total_bbox = 267642, rewritten_bbox = 0.102749 %  
v3 (iou loss, Normalizer: (iou: 0.07, obj: 1.00, cls: 1.00) Region 139 Avg (IOU: 0.539968), count: 3, class_loss = 1.828908, iou_loss = 2.095903, total_loss = 3.924%  
v3 (iou loss, Normalizer: (iou: 0.07, obj: 1.00, cls: 1.00) Region 150 Avg (IOU: 0.648215), count: 14, class_loss = 7.587945, iou_loss = 4.342549, total_loss = 11.9%
```

Khi model train được 2033 iterations trong khoảng 16 tiếng, nhận thấy avg loss không có dấu hiệu giảm nữa nên nhóm đã dừng việc train.

3. Phương pháp đánh giá

Nhóm thực hiện các phương pháp đánh giá model trên tập validation.

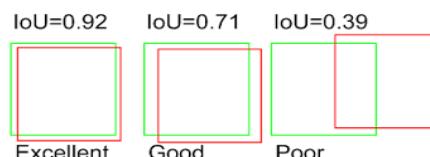
- IOU(Intersection Over Union):

Intersection Over Union là chỉ số đánh giá được sử dụng để đo độ chính xác của phát hiện đối tượng trên tập dữ liệu cụ thể. Chỉ số này thường được gặp trong các bài toán Object Detection. IOU thường được đánh giá hiệu năng của các bộ phát hiện đối tượng như: R-CNN, Fast R-CNN, YOLO,...

Để áp dụng IOU vào việc đánh giá cần:

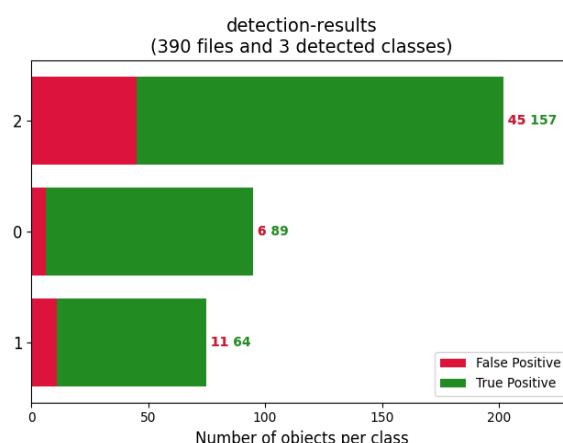
- Đường bao thực (ground-truth bounding box): là đường bao mà chúng ta gán cho vật thể bằng LabelImg Tool.
- Đường bao dự đoán (predicted bounding box): là đường bao chúng ta sử dụng file Weights sau khi đào tạo để nhận dạng.

IOU là tỉ lệ giữa diện tích giao nhau giữa hai đường bao (thường là đường bao dự đoán và đường bao thực) để nhằm xác định hai khung hình có đè chòng lên nhau không. Tỷ lệ này được tính dựa trên phần diện tích giao nhau giữa 2 đường bao với phần tổng diện tích giao nhau và không giao nhau giữa chúng.



Các tiêu chí đánh giá:

- Đối tượng được nhận dạng đúng với tỉ lệ $\text{IOU} > 0.5$ (True positive : TP)
- Đối tượng được nhận dạng sai với tỉ lệ $\text{IOU} < 0.5$ (False positive : FP)
- Đối tượng không được nhận dạng (False negative: FN)



(0: Easy Decompose, 1: Hard Decompose, 2: Recyclable)

Nhìn chung, Precision của từng class khá tốt: 78% ở class Recyclable, 94% ở class Easy Decompose, 85% ở class Hard Decompose.

- Precision và Recall:

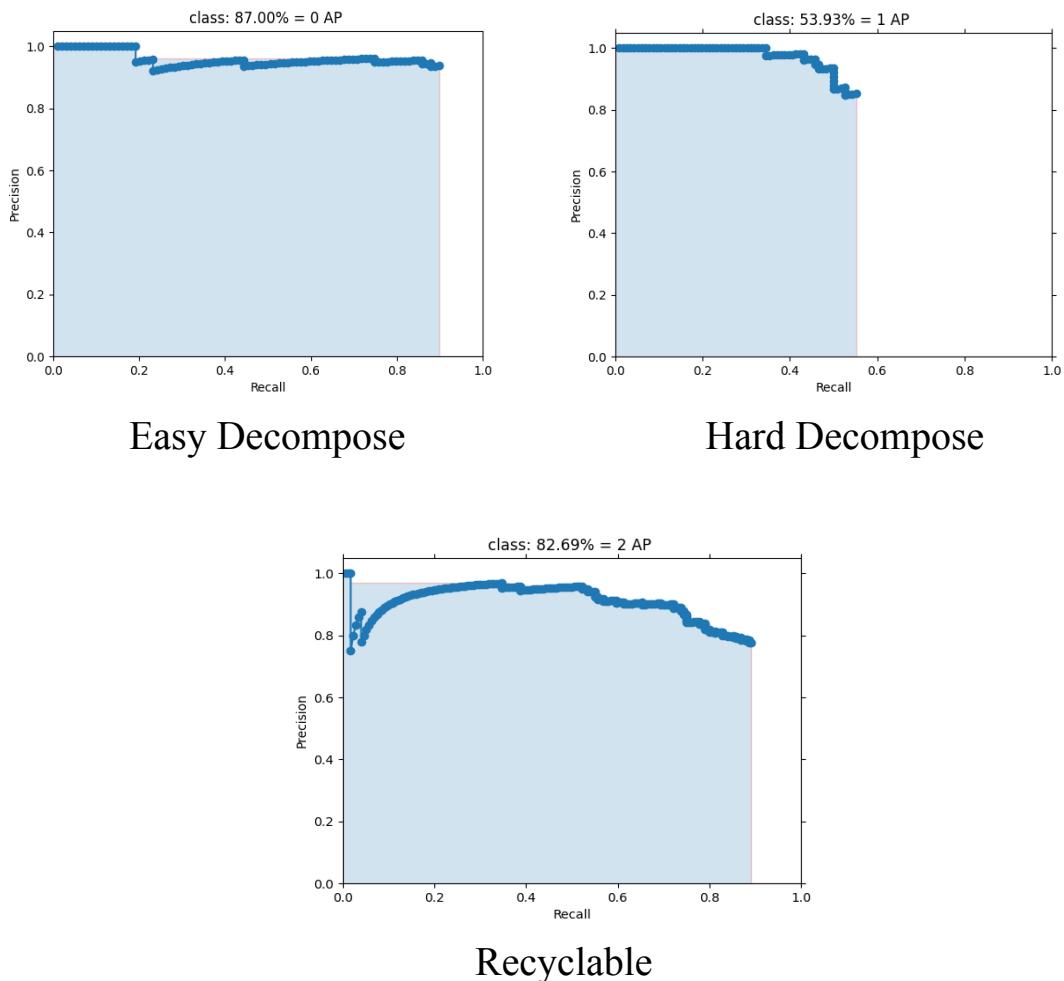
Precision được định nghĩa là tỉ lệ số điểm true positive trong số những điểm được phân loại là positive.

$$\text{Precision} = \text{TP} / (\text{TP} + \text{FP})$$

Recall được định nghĩa là tỉ lệ số điểm true positive trong số những điểm thực sự là positive.

$$\text{Recall} = \text{TP} / (\text{TP} + \text{FN})$$

Precision-Recall Curve:



Một model tốt là model có sự tradeoff giữa Precision và Recall (Precision giảm khi Recall tăng). Trong các đồ thị trên khi Recall tăng thì Precision có xu hướng giảm dù cho ở đồ thị của class Easy

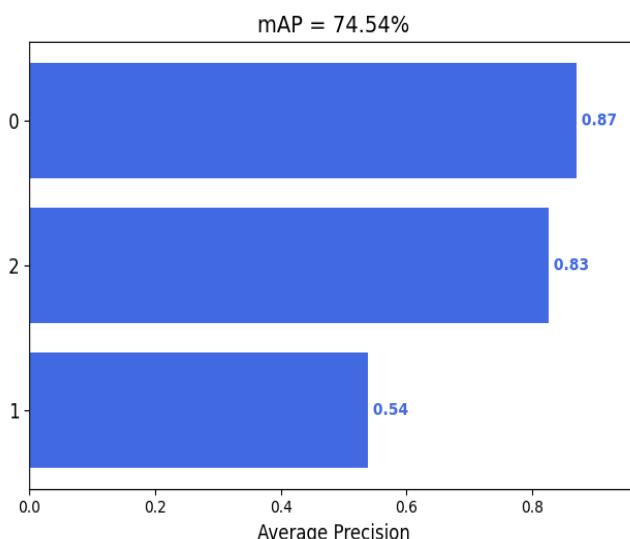
Decompose và Recyclable có nhiều điểm khiến cho Precision tăng khi Recall tăng. Có thể rút ra nhận xét chung là model không quá tốt.

- mAP(mean Average Precision):

Từ precision và recall đã được định nghĩa ở trên chúng ta cũng có thể đánh giá mô hình dựa trên việc thay đổi một ngưỡng và quan sát giá trị của Precision và Recall. Khái niệm Area Under the Curve (AUC) đối với Precision-Recall Curve Average Precision (AP). Giả sử có N ngưỡng để tính precision và recall, với mỗi ngưỡng cho một cặp giá trị precision, recall là P_n, R_n với $n=1,2,\dots,N$. Precision-Recall curve được vẽ bằng cách vẽ từng điểm có toạ độ (R_n, P_n) trên trực toạ độ và nối chúng với nhau. AP được xác định bằng:

$$AP = \sum_n (R_n - R_{n-1})P_n$$

Còn mAP là trung bình của AP của tất cả các lớp.



(0: Easy Decompose, 1: Hard Decompose, 2: Recyclable)

AP của class Easy Decompose là 87%

AP của class Recyclable là 83%

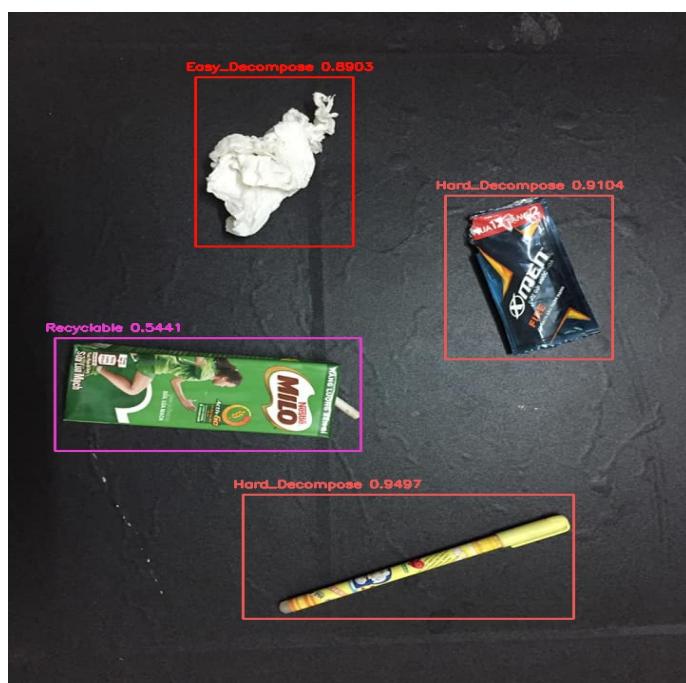
AP của class Hard Decompose là 54%

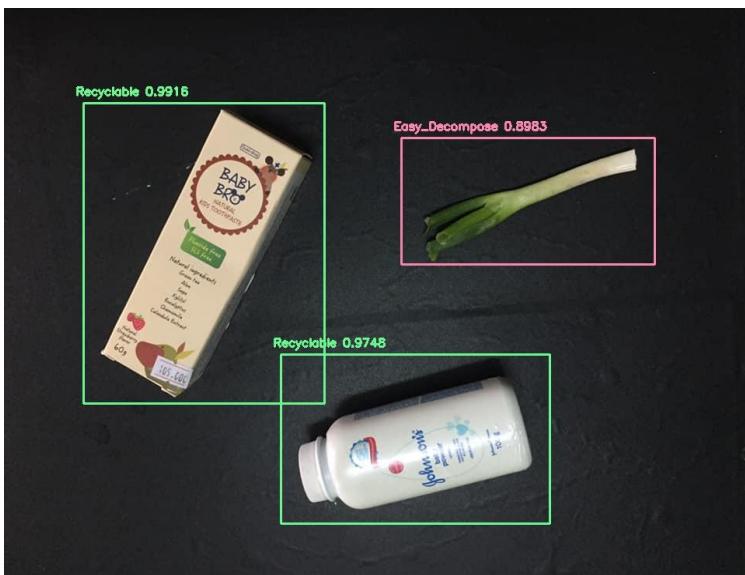
Average Precision của class Hard Decompose khá thấp, nên dù cho AP của hai class còn lại ở mức ổn thì mAP của model chỉ là 74.54%.

Nhận xét: Vì giống ngữ cảnh ứng dụng nên em sẽ so sánh model này với model trong bài nghiên cứu “Development of a method of detection and classification of waste objects on a conveyor for a robotic sorting system”. Dù với lượng data ít hơn nhưng model này lại có chỉ số mAP cao hơn model trong nghiên cứu trên, lý do là vì dữ liệu khi đánh giá của họ là ảnh có nhiều object trong khung hình, còn nhóm em chỉ đánh giá model với hầu hết các ảnh có một object.

4. Test model

Một số hình ảnh khi test model:



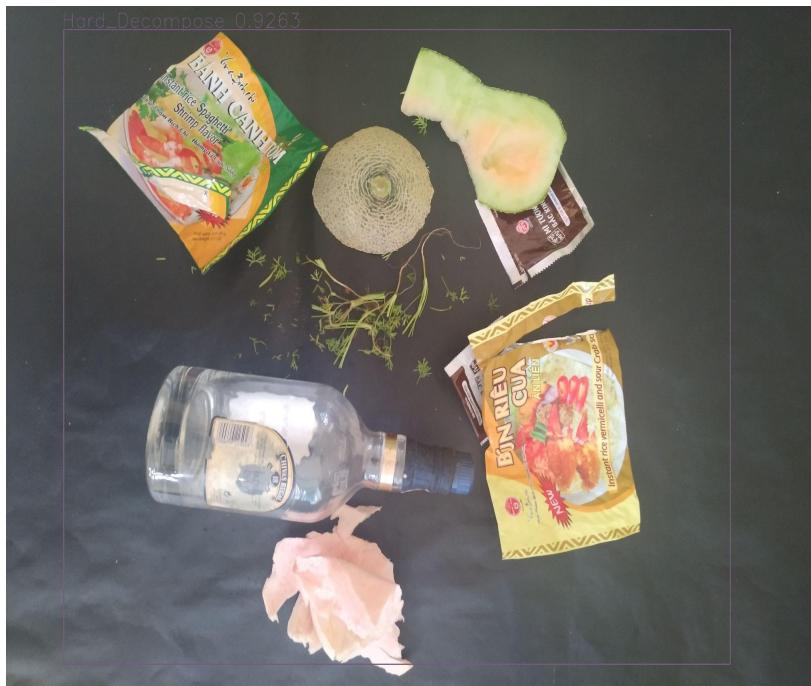


Các trường hợp trên khá dễ khi rác nằm cách xa nhau và model đã nhận diện và phân loại đúng hết.



Trong trường hợp này, model nhận diện thiếu 1 object. Có thể là do khoảng cách các object khá gần hoặc model không thể phân loại được object này thuộc class nào. Có 1 object mà bounding box không bao hết được (gói xà phòng) và bị phân loại sai class (dù trong ảnh ở trên object này được phân loại đúng).

Ta sẽ đến với trường hợp khó và gần với thực tế hơn:



Model không thể nhận diện được từng object có trong hình khi khoảng cách giữa các object khá nhỏ và chúng còn nằm chồng lên nhau, chỉ có một bounding box duy nhất bao quanh tất cả object.

Thử nghiệm model với video:

<https://drive.google.com/drive/u/1/folders/1uOql0PIPLXcmjyjhJI27CZpNl0ZJEgQLQ>

Nhận xét: Model không thể nhận diện được khi các object ở quá gần nhau, có thể là do lỗi ở quá trình thu thập dữ liệu vì bộ data của nhóm chỉ là các bức ảnh chụp riêng cho từng loại rác, hầu hết các ảnh chỉ có một object và không có các trường hợp khó như nhiều object nằm gần nhau hoặc chồng lên nhau. Trong bài nghiên cứu “Development of a method of detection and classification of waste objects on a conveyor for a robotic sorting system” tuy trong bộ data để train của họ cũng có các ảnh thuộc trường hợp khó như trên nhưng khi test model cũng không thể nhận diện hết được các object trong các trường hợp khó, tuy vậy model của họ vẫn nhận diện tốt hơn model của nhóm dù cho có mAP thấp hơn.

V. Ứng dụng và hướng phát triển

Để có thể ứng dụng trong thực tế thì việc đầu tiên là phải xây dựng lại bộ, tăng số lượng và sự đa dạng cho bộ data, đây sẽ là một quá trình khá dài. Sau đó, ta sẽ cải tiến model để có thể nhận diện real-time với tốc độ nhanh và chính xác để phù hợp với công việc trong thực tế. Một việc cũng rất khó nữa là tích hợp model này vào hệ thống nhúng có camera để có thể tự động hóa việc phân loại rác, tuy nhiên hệ thống này cũng sẽ tốn nhiều chi phí để chế tạo (hoặc mua nếu có). Vậy nên sẽ khá khó để ứng dụng model này vào thực tế trong một tương lai gần, nhưng với sự phát triển nhanh chóng của công nghệ cũng như nhu cầu của con người ngày càng tăng cao thì việc phát triển model như thế này là hoàn toàn khả thi.



*** Các nguồn tham khảo:**

1. Vũ Hữu Tiệp (2018). “Bài 33: Các phương pháp đánh giá một hệ thống phân lớp”,
<https://machinelearningcoban.com/2017/08/31/evaluation/>
2. Cường Hoàng (2020). “Series YOLOv4: #1 Train model trên Google Colab – Object detection.”,
<https://devai.info/2020/12/15/huong-dan-training-object-detection-voi-yolov4-su-dung-google-colab/>
3. Cường Hoàng (2020). “Series YOLOv4: #3 Đánh giá model bằng mAP -Object detection”,
<https://devai.info/2020/12/17/tim-hieu-mapmean-average-precision-danh-gia-mo-hinh-object-detection-su-dung-yolov4/>
4. Nguyễn Chiến Thắng (2019). “[YOLO Series] #1 – Sử dụng Yolo để nhận dạng đối tượng trong ảnh”,
<https://www.miai.vn/2019/08/05/yolo-series-1-su-dung-yolo-de-nhan-dang-doi-tuong-trong-anh/>
5. Github darknet của tác giả AlexeyAB:
<https://github.com/AlexeyAB/darknet>
6. Github yolov4-opencv-python của tác giả Asadullah-Dal17:
<https://github.com/Asadullah-Dal17/yolov4-opencv-python>

*** Nguồn các bài nghiên cứu:**

1. Qingqiang Chen, Qianghua Xiong (2020). ”Garbage Classification Detection Based on Improved YOLOV4”,
https://www.researchgate.net/publication/348456533_Garbage_Classification_Detection_Based_on_Improved_YOLOV4
2. A V Seredkin, M P Tokarev, I A Plohih, O A Gobyzov and D M Markovich (2019). ”Development of a method of detection and classification of waste objects on a conveyor for a robotic sorting system”,<https://iopscience.iop.org/article/10.1088/1742-6596/1359/1/012127>