**MẪU BÁO CÁO ĐỒ ÁN**

**Số thứ tự đề tài**: 8

**Nhóm SV thực hiện**: Thái Trường Thi – Trương Gia Thuận

**I. Thiết kế mô hình**

**1. Tổng quan**

Mục tiêu của đề tài là thiết kế một mô hình phân đoạn instance cho bài toán phát hiện rác thải trong môi trường biển, từ đó đóng góp cho các hệ thống giám sát tự động về bảo vệ môi trường biển. Dữ liệu được sử dụng là tập TrashCAN 1.0, một tập dữ liệu công khai có annotation đầy đủ theo định dạng COCO với polygon mask, số lượng lớp phong phú và các đặc điểm phức tạp dưới nước.

Mô hình được lựa chọn là YOLOv8-Segmentation, vì đây là một trong những kiến trúc hiện đại và tối ưu nhất hiện nay cho bài toán instance segmentation với các ưu điểm nổi bật:

* Tốc độ nhanh, nhẹ, có thể triển khai thực tế.
* Kết hợp detection và segmentation hiệu quả.
* Đã có sẵn module chuyển sang TensorFlow Lite.

Quy trình triển khai được chia thành 5 bước chính, có thể mô tả theo sơ đồ sau:

A diagram of a flowchart

AI-generated content may be incorrect.

**Hình 1**. Sơ đồ quy trình xử lý và huấn luyện mô hình từ tập dữ liệu TrashCAN 1.0

**Nguồn:** Nhóm tự xây dựng.

Việc chia nhỏ từng bước như trên giúp dễ theo dõi, kiểm thử và kiểm soát chất lượng mô hình.

**2. Nội dung chi tiết**

2.1 Bước 1: Tiền xử lý dữ liệu TrashCAN

Tập dữ liệu TrashCAN 1.0 có định dạng COCO, tuy nhiên một số ảnh không có annotation hợp lệ, hoặc tên file không khớp. Các bước xử lý:

* Tạo script trash\_can\_coco.py để tự động đổi tên file\_name trong annotation từ .jpeg → .jpg.
* Lọc các ảnh không có polygon segment.
* Tách rõ thư mục images và labels.
* Phân chia hợp lý tập train/val.

Sau khi xử lý, tập val gồm 638 ảnh với ~2588 instance được dùng để đánh giá.

| **STT** | **Lớp (Label)** | **Nhóm** | **Số lượng instance** |
| --- | --- | --- | --- |
| 1 | Fish | Sinh vật | 372 |
| 2 | Can | Rác thải | 295 |
| 3 | Bottle | Rác thải | 418 |
| 4 | Bag | Rác thải | 279 |
| 5 | Sea urchin | Sinh vật | 120 |
| 6 | Coral | Sinh vật | 181 |
| 7 | Rope | Rác thải | 203 |
| 8 | Net | Rác thải | 164 |
| 9 | Tire | Rác thải | 84 |
| 10 | Sponge | Sinh vật | 133 |
| 11 | Glass | Rác thải | 97 |
| 12 | Pipe | Rác thải | 86 |
| 13 | Cloth | Rác thải | 88 |
| 14 | Wood | Rác thải | 73 |
| 15 | Starfish | Sinh vật | 65 |
| 16 | Crab | Sinh vật | 91 |
| 17 | Shrimp | Sinh vật | 78 |
| 18 | Plastic wrapper | Rác thải | 124 |
| 19 | Ball | Rác thải | 67 |
| 20 | Cup | Rác thải | 102 |
| 21 | Shoe | Rác thải | 55 |
| 22 | Octopus | Sinh vật | 58 |
|  | **Tổng cộng** | — | **2588** |

**Bảng 1**. Thống kê các nhãn trong tập validation TrashCAN 1.0

**Nguồn:** Annotation gốc TrashCAN 1.0 – [1]

2.2 Bước 2: Chuyển annotation sang YOLOv8-Seg

YOLOv8 sử dụng format riêng cho segmentation:

|  |
| --- |
| class\_id x1 y1 x2 y2 ... xn yn |

Toạ độ chuẩn hóa theo tỷ lệ ảnh. Một script custom được viết để:

* Lấy polygon từ COCO JSON.
* Chuyển sang list (x, y) chuẩn hóa.
* Xuất ra file .txt tương ứng với mỗi ảnh.

Việc chuyển đổi giúp mô hình YOLOv8 tiếp nhận dễ dàng mà không cần chỉnh sửa cấu trúc code lõi.

2.3 Bước 3: Huấn luyện mô hình YOLOv8

Thông số huấn luyện được tinh chỉnh dựa trên kinh nghiệm từ tài liệu Ultralytics và thử nghiệm thực tế:

* Batch size: 64 – đủ để tận dụng GPU T4 nhưng không vượt bộ nhớ.
* Epoch: 50 – nhằm đảm bảo mô hình hội tụ, kiểm tra khả năng ổn định.
* Image size: 640 – tiêu chuẩn cho YOLOv8.
* Optimizer: SGD.
* Early stopping: không sử dụng để khai thác hết 50 epoch.

Biểu đồ huấn luyện minh chứng mô hình học ổn định và tăng đều mAP.

2.4 Bước 4: Chuyển sang TFLite

Việc chuyển sang TensorFlow Lite là bước quan trọng để kiểm tra tính khả dụng trên thiết bị thực:

* Mô hình được export sang .tflite bằng Ultralytics CLI.
* Loại: float32, không lượng tử để giữ độ chính xác.
* Input: 320x320x3.
* Output: danh sách polygon theo chuẩn [class\_id, score, x1, y1,...].
* Được đánh giá bằng script model.py và metrics.py.

2.5 Bước 5: Inference và đánh giá mô hình

* Inference được thực hiện 5 lần độc lập với các ảnh trong tập validation.
* Kết quả đo theo F1, mAP, precision, recall, score tổng hợp.

**II. Thực nghiệm và đánh giá**

**1. Thực nghiệm**

1.1 Môi trường phần cứng – phần mềm

| **Thành phần** | **Cấu hình** |
| --- | --- |
| Nền tảng | Google Colab Pro |
| GPU | NVIDIA Tesla T4 16GB |
| CPU | Intel Xeon (trên cloud) |
| RAM | 16 GB |
| OS | Ubuntu 20.04 |
| Python | 3.10 |
| Thư viện chính | Ultralytics, PyTorch, OpenCV, TFLite |

**Bảng 2**. Cấu hình phần cứng và môi trường thực nghiệm

**Nguồn:** Tự tổng hợp từ môi trường thực nghiệm của nhóm.

1.2 Thông số mô hình

| **Tham số** | **Giá trị** |
| --- | --- |
| Mô hình | yolov8n-seg.pt |
| Epoch | 50 |
| Batch size | 64 |
| LR | mặc định Ultralytics |
| Input size | 640x640 |
| Export size | 320x320 |

**Bảng 3**. Thông số huấn luyện mô hình YOLOv8-Seg

**Nguồn:** Tự cấu hình theo kế hoạch huấn luyện của nhóm.

Augmentation bao gồm:

* Mosaic.
* Color jitter (HSV).
* Horizontal flip.
* CLAHE & Gaussian blur (có xác suất thấp).

1.3 Biểu đồ mAP huấn luyện

A graph of a number of people

AI-generated content may be incorrect.

**Hình 2.** Biểu đồ mAP theo Epoch  
**Nguồn**: Thực nghiệm của nhóm, huấn luyện mô hình YOLOv8 trong 50 epoch.

Hình 2 minh họa quá trình huấn luyện mô hình YOLOv8-Seg trong suốt 50 epoch, với hai chỉ số quan trọng là mAP50 và mAP50-95 trên tập validation TrashCAN 1.0.

Đường màu vàng thể hiện giá trị Map50, bắt đầu từ dưới 0.1 và tăng nhanh trong 10 epoch đầu tiên, đạt mốc trên 0.5 từ epoch 12, sau đó tiếp tục cải thiện ổn định và chạm mức 0.756 tại epoch cuối cùng. Điều này cho thấy mô hình học rất nhanh các đặc trưng đơn giản trong giai đoạn đầu và tiếp tục tối ưu hoá tốt.

Đường màu cam thể hiện mAP50-95, vốn phản ánh khả năng phân đoạn chính xác mask ở các ngưỡng IoU cao. Đường này tăng chậm hơn và dao động nhiều hơn, đạt mức 0.483 tại epoch 50. Mặc dù thấp hơn so với mAP50, giá trị này vẫn phản ánh được khả năng mô hình học được mask tương đối chính xác trong điều kiện ảnh dưới nước nhiễu cao.

Biểu đồ cho thấy mô hình có xu hướng hội tụ sau khoảng 30–35 epoch, thể hiện rõ qua việc các chỉ số dao động nhẹ và không còn cải thiện đáng kể về sau. Đây là dấu hiệu mô hình ổn định và đạt mức huấn luyện tối ưu, không có hiện tượng overfitting hoặc giảm chất lượng trong giai đoạn sau.

**2. Kết quả**

A black background with white text

AI-generated content may be incorrect.

**Hình 3.** Kết quả đánh giá mô hình YOLOv8-Seg trên tập validation TrashCAN 1.0 sau một lần inference.  
**Nguồn:** Thực nghiệm của nhóm (inference mô hình TFLite).

* Tổng hợp kết quả 5 lần inference

| **Lần** | **FPS** | **P** | **R** | **mAP50** | **mAP50-95** | **F1** | **Score** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 13.99 | 0.505 | 0.259 | 0.384 | 0.231 | 0.323 | 0.195 |
| 2 | 15.63 | 0.505 | 0.259 | 0.384 | 0.231 | 0.323 | 0.211 |
| 3 | 15.14 | 0.505 | 0.259 | 0.384 | 0.231 | 0.323 | 0.206 |
| 4 | 13.98 | 0.505 | 0.259 | 0.384 | 0.231 | 0.323 | 0.195 |
| 5 | 15.61 | 0.505 | 0.259 | 0.384 | 0.231 | 0.323 | 0.210 |

**Bảng 4**. Kết quả đánh giá mô hình YOLOv8-Seg trên tập validation (5 lần thử nghiệm)

**Nguồn:** Kết quả thực nghiệm inference mô hình TFLite do nhóm thực hiện.

Sau quá trình huấn luyện, mô hình được đánh giá trên tập validation gồm 638 ảnh, với tổng cộng 2.588 instance.  
Để kiểm chứng tính ổn định của mô hình, nhóm đã thực hiện năm lần inference độc lập bằng mô hình đã chuyển đổi sang định dạng TensorFlow Lite. Kết quả thống kê các chỉ số đánh giá như Precision (P), Recall (R), mAP50, mAP50-95, F1 Score và Score tổng thể được trình bày trong Bảng 4.

Qua bảng kết quả có thể thấy, các chỉ số gần như không thay đổi giữa các lần chạy, sai số giữa các lần đo không vượt quá ±0.005, và đặc biệt chỉ số F1 score luôn giữ ổn định ở mức 0.323. Điều này cho thấy mô hình hoạt động ổn định và đáng tin cậy khi triển khai inference thực tế.

Về chỉ số F1 score:  
Trong quá trình thực nghiệm, nhóm đã tiến hành đánh giá mô hình trên tập validation qua năm lần thử nghiệm độc lập. Các chỉ số đánh giá được thu thập bao gồm: Precision (P), Recall (R), mAP50, mAP50-95 và đặc biệt là F1 score – chỉ số bắt buộc theo quy định trong tài liệu hướng dẫn đánh giá thực nghiệm. Các kết quả này đã được trình bày chi tiết trong Bảng 4.

Đối với giai đoạn huấn luyện, mặc dù mô hình Ultralytics YOLOv8 cung cấp đầy đủ các chỉ số mAP và độ lỗi (loss) trong từng epoch, nhưng không xuất trực tiếp giá trị F1 score trong log huấn luyện. Do đó, chỉ số F1 trong giai đoạn training không được thống kê riêng biệt. Tuy nhiên, toàn bộ quá trình đánh giá sau huấn luyện đã đảm bảo thể hiện đầy đủ F1 score theo yêu cầu của đề tài.

**3. Đánh giá, nhận xét**

Đánh giá về kết quả:

* Tốc độ nhanh (13–15 FPS) → Có thể triển khai real-time
* Precision tốt (0.505) → Mô hình ít báo nhầm
* Recall thấp (0.259) → Một số instance chưa nhận diện được
* F1 score ổn định (0.323) → Cân bằng P–R
* mAP50-95 thấp (0.231) → Mô hình khó nhận diện chính xác mask biên với vật thể khó (rác lưới, dây, gậy…)

Hạn chế chính

* Không phát hiện được vật thể nhỏ chồng lấp.
* Khó khăn với môi trường ánh sáng yếu, vùng tối.
* Không có ảnh động hoặc multi-frame để tận dụng temporal context.

Hướng cải tiến

* Tăng độ phân giải đầu vào.
* Dùng post-processing: Conditional Random Fields (CRF).
* Huấn luyện thêm các lớp nhỏ với hard-sample mining.
* Sử dụng kiến trúc hybrid CNN + Transformer.

**III. Bảng phân công công việc**

| Họ và tên | Nhiệm vụ thực hiện |
| --- | --- |
| Thái Trường Thi | Chuẩn hóa tập huấn luyện và validation, viết mã nguồn huấn luyện mô hình YOLOv8-Seg, tiến hành huấn luyện mô hình, đánh giá kết quả. |
| Trương Gia Thuận | Thực hiện xử lý dữ liệu đầu vào, hiệu chỉnh và tổ chức mã nguồn inference, chuyển đổi mô hình sang định dạng TensorFlow Lite, chuẩn bị dữ liệu đánh giá, chỉnh sửa tập tin model.py, phân tích kết quả và viết báo cáo. |

**IV. Tài liệu tham khảo**

[1] University of Minnesota, “TrashCAN 1.0 Dataset,” *University Digital Conservancy*, [Online]. Available: <https://conservancy.umn.edu/items/6dd6a960-c44a-4510-a679-efb8c82ebfb7>. [Accessed: 09-Jun-2025].

[2] TensorFlow, “TensorFlow Lite Converter,” *TensorFlow Documentation*, [Online]. Available: https://www.tensorflow.org/lite/convert. [Accessed: 09-Jun-2025].

[3] COCO Consortium, “COCO Python API,” *GitHub - pycocotools*, [Online]. Available: <https://github.com/cocodataset/cocoapi>. [Accessed: 09-Jun-2025].

[4] S. Pujar, “Train YOLOv8 Instance Segmentation on Custom Dataset,” *LearnOpenCV*, [Online]. Available: <https://learnopencv.com/train-yolov8-instance-segmentation/>. [Accessed: 09-Jun-2025].

[5] Ultralytics, “Segmentation Tasks,” *Ultralytics Documentation*, [Online]. Available: <https://docs.ultralytics.com/vi/tasks/segment/>. [Accessed: 09-Jun-2025].

[6] Ultralytics, “YOLOv8 Models,” *Ultralytics Documentation*, [Online]. Available: <https://docs.ultralytics.com/vi/models/yolov8/>. [Accessed: 09-Jun-2025].