GREAT BARRIER REEF: INVASIVE STARFISH DETECTOR

Nick Wagner and David Unger

Institute of Signal Processing and System Theory, University of Stuttgart, Germany

1. Problem Statement

Abstract

The Great Barrier Reef in Australia is known world-wide for its diverse animal and coral species. Recently, the overpopulation of the coral-eating crown-of-thorns starfish starts threatening the existance of many corals. To allow divers to efficiently remove these star fishes from the corals, an object detection algorithm gets used.
The algorithm is based on the ”YOLO”–framework and takes video sequences as input, detects the starfishes and draws bounding boxes around them.

Dataset

- ~23000 images from 3 video sequences
- bounding box labels from csv
- binary problem

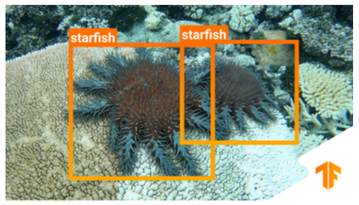

Figure 1: Example bounding boxes [1]

2. Architecture

The basis for the following customized architecture is the YOLO (”You Only Look Once”) paper. The most significant changes are the removal of a dense layer, the addition of a dropout layer and dimension changes.

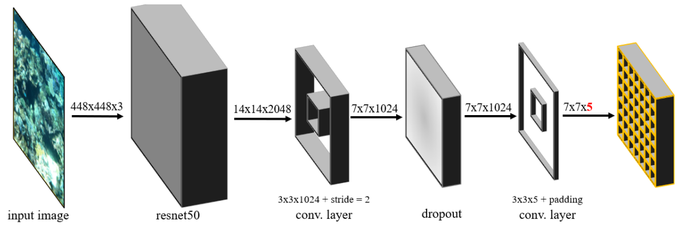

Figure 2: customized YOLO architecture

- **input layer**: image size of 448x448 with three color channels (R,G,B)
- **resnet50**: transfer model that is pretrained on ImageNet data
- **conv1**: reduction of channels and grid size to 7x7 due to stride
- **dropout**: generalization technique to prevent overfitting
- **conv2**: final reduction of channels to 5 with padding

3. Network Output

The final output of the network is a 7x7 grid. Each grid cell predicts one bounding box, which results in 49 possibly predicted bounding boxes.

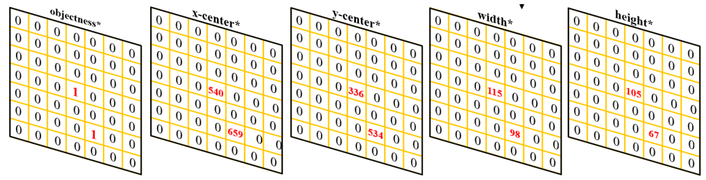

Figure 3: customized YOLO architecture

- **objectness**: confidence of the prediction
- **x-center**: x-coordinate in relation to its cell center
- **y-center**: y-coordinate in relation to its cell center
- **width**: box-width measured in cell widths
- **height**: box-height measured in cell heights

* The values are shown in pixel for understanding purposes. In reality, smaller values relative to the cell size are easier for the model to predict. Hence, the implementation is done that way.

4. Input Pipeline

Compared to a classification network, a detection network requires a more complicated input pipeline. This mainly comes from the labels which are dependent on the image content and scale, rotation, crop and flip variant.Therefore, the input pipeline consists of the following steps:

- CSV loading
- Bounding box text to grid conversion
- Image reading and resizing
- Image augmentation

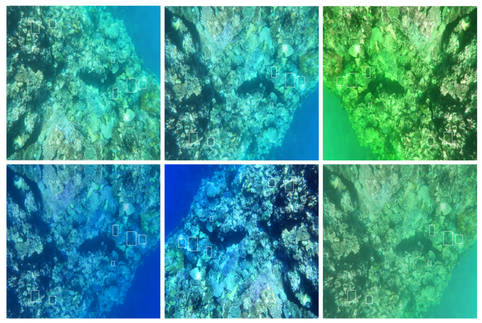

Figure 4: input image augmentated differently

5. Loss Function

To ensure that the network learns the bounding boxes as represented by the ground truth labels, using a standard loss function is not sufficient. Therefore, the loss function from the YOLO paper is used and adapted that it fits the described problem setup.
The four parts of our loss function are the following and summed up during training

- Objectness loss: $l_{obj} = \lambda_{obj} \sum_{i=0}^{(S-1)^2} \mathbb{1}_i^{obj} (1 - \hat{p}_i)^2$
- No object loss: $l_{noobj} = \lambda_{noobj} \sum_{i=0}^{(S-1)^2} \mathbb{1}_i^{noobj} (0 - \hat{p}_i)^2$
- BBox center loss: $l_{center} = \lambda_{center} \sum_{i=0}^{(S-1)^2} \mathbb{1}_i^{obj} [(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2]$
- BBox size loss: $l_{size} = \lambda_{size} \sum_{i=0}^{(S-1)^2} \mathbb{1}_i^{obj} [(\sqrt{w_i} - \sqrt{\hat{w}_i})^2 + (\sqrt{h_i} - \sqrt{\hat{h}_i})^2]$

The loss term weightings λ are determined in a way, that all loss terms are in a similar range and therefore are optimized in a similar strength.

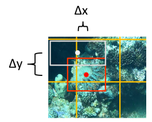

Figure 6: A figure

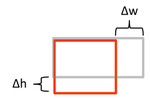

Figure 7: Another figure

6. Challenge

fff
fff
ffff
ffff

Figure 8: Starfish at downscaled images with poor quality

As an alternative dataset, we looked into the detection of traffic cones. The almost same network that failed to generalize on the starfish dataset was able to master the traffic cone dataset quickly. The results are shown in the follwoing:

Deep Learning Lab 2022, February 8, Stuttgart, Germany