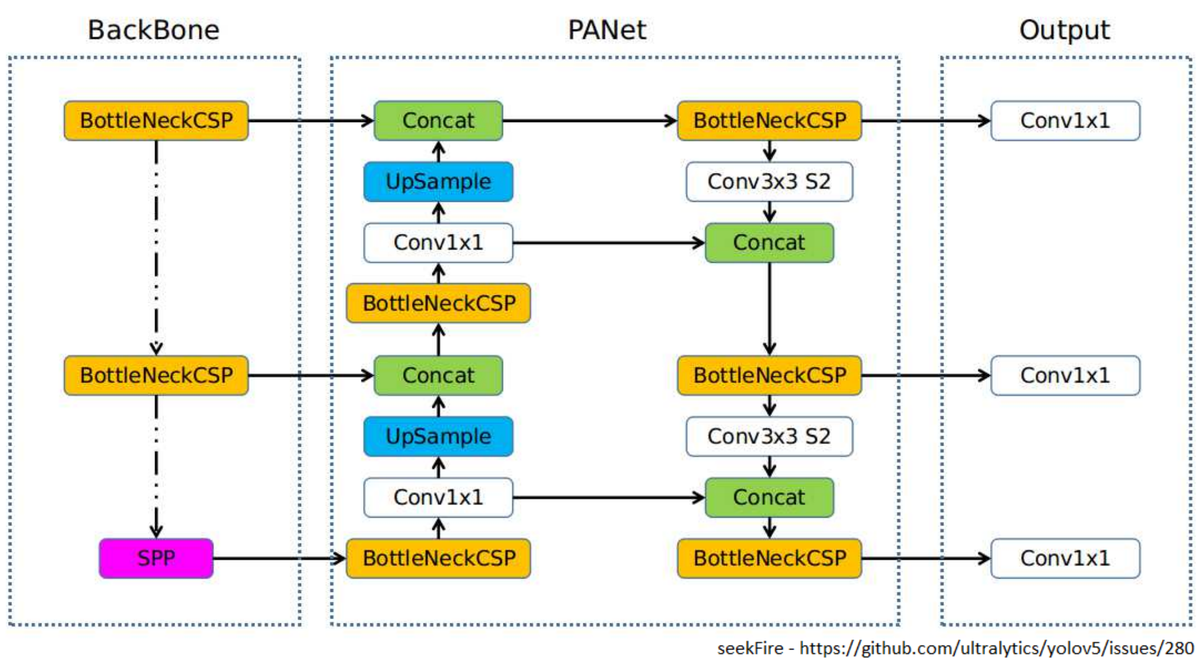
YOLOv5 Documentation

Design

YOLOv5 is an Object Detection Algorithm based machine learning platform. When processing images it Divides each Image into a grid where Each Cell is responsible for detecting objects within itself. This allows for faster response times because each portion of the image is processed only once.
Multiple neural network model configurations are available "out of the box" to meet the needs of different applications. Due to the small objects we are trying to detect we began by using the YOLOv5x6 model configuration. We have found that the number of neural nodes it provides provides us with better results during training than the other models. At its worst it will process 50 images a second.



Overview of YOLOv5 Model Structure

Processes

A Cross Stage Partial Network which is the Backbone used to connect layers in convolutional neural networks with the goal of reducing bottlenecks in the computational process. Much like the YOLOR model we were using earlier, the YOLOv5 CSP model encourages reusing

features within the network and alleviates constraints and losses with a deep neural network.
For the Model Neck a PANet Feature Pyramid Network is used to further extract features and output predictions while keeping the spatial information found in lower layers from being lost in the upper layers of the network.
The Model Head is the final process which outputs the locations of detected objects, the final predictions and scores, and generates bounding boxes. This process has relatively stayed the same since YOLOv3

Benefits

Utilizes PyTorch machine learning framework vs the Darknet framework used in YOLOv3. This allows the code to be easier to debug, faster code optimizations, and many APIs that extend the base framework. Training optimization using SGD or ADAM which lessens the computational load when calculating the gradient of different learning rates in our model

Dataset

We began training utilizing a previous semester's pre annotated dataset. This allowed us to refine our training techniques while waiting for the WEB APP group to provide a new set of images.
The Dataset contained 4,511 images with 4,555 separate annotated objects.
Based on our initial results with a previously collected dataset and observations, we provided the WEB APP group with modified Google Street View Image collection parameters which focused on areas within an image where trash was mostly detected and increased the object's size within the full image. Due to the large number of images that did not contain any litter and the small size of many of the objects, we used an augmentation technique called tiling. This method involves splitting an image into multiple smaller images of equal sizes. We then filtered out all but 10% of images containing no litter objects.

YOLOv5 Documentation

Original Image

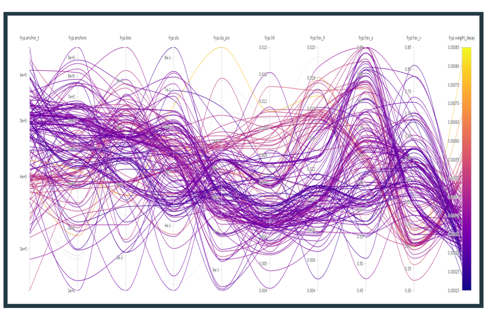
Tiled Image

Training

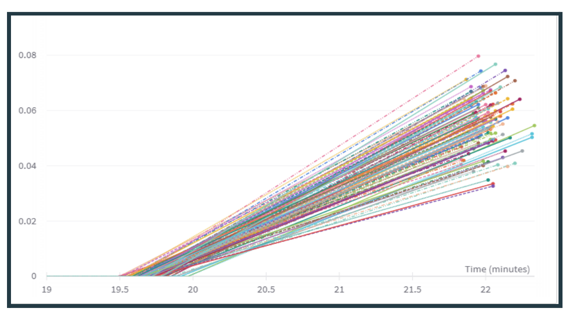
Based on the knowledge we gained through hundreds of iterations and training we were able to increase the accuracy of our model.

Hyperparameter Evolution Training

Using Hyperparameter Evolution training which over the course of 300 generations each with 20 epochs we were able to find the best hyperparameters to use for our training. Evolution training modifies 25 different training parameters to gain insight on which mix provides the best results.



Parameter Evolutions

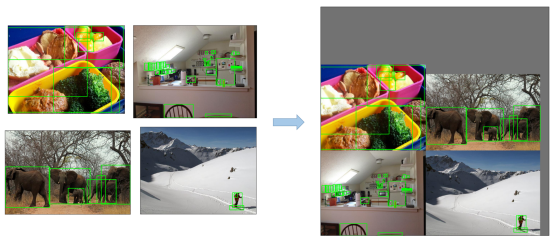


Results

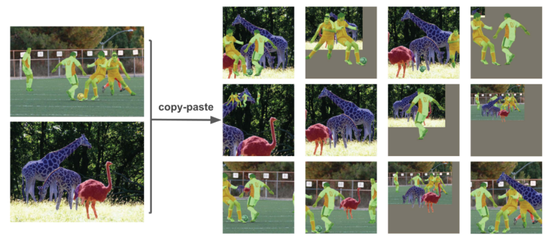
YOLOv5 Documentation

Data Augmentations

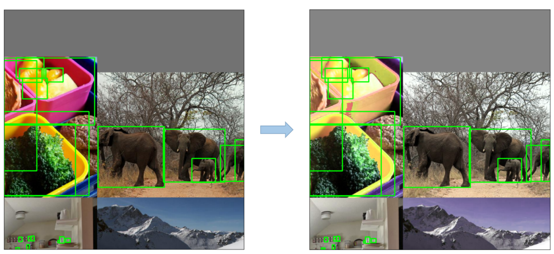
We utilized data augmentations during testing to modify aspects of an entire image or just the bounding box to increase our model's resilience and effectively increase the image count within our dataset. Augmentations such as flipping an image, changing its perspective, scaling, and image hue, contrast and brightness. The probability of these occurring is located in the hyperparameter file used to train.



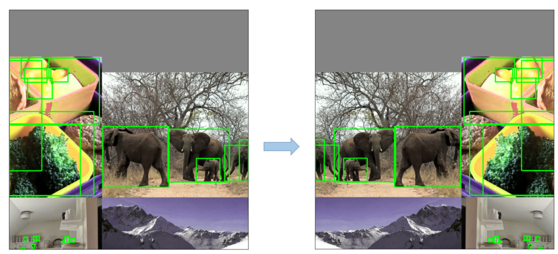
Mosaic



Copy-Paste



Random HSV (Hue, Saturation, Value)



Random Horizontal Flip

Modified Bounding Box Acceptance

We modified the YOLOv5 code to reject tiny bounding boxes that encompassed objects that could not be verified as littler. The features of these objects are often found in other parts of the Google Street Images due to their resolution constraints. We reject any bounding box with an edge less than 7 pixels. This is accomplished in the box_canidates() function in the augmentations.py file located in the data folder. The box edges are defined by wh_thr.

```python
def box_candidates(box1, box2, wh_thr=7, ar_thr=100, area_thr=0.1, eps=1e-16):  # box1(4,n), box2(4,n)
    # Compute candidate boxes: box1 before augment, box2 after augment, wh_thr (pixels), aspect_ratio_thr, area_ratio
    w1, h1 = box1[2] - box1[0], box1[3] - box1[1]
    w2, h2 = box2[2] - box2[0], box2[3] - box2[1]
    ar = np.maximum(w2 / (h2 + eps), h2 / (w2 + eps))  # aspect ratio
    return (w2 > wh_thr) & (h2 > wh_thr) & (w2 * h2 / (w1 * h1 + eps) > area_thr) & (ar < ar_thr)  # candidates
```

Refer to https://github.com/ultralytics/yolov5 for more information. Images attributable to Ultralytics unless otherwise noted.