Tegan Shiver

EEL6990 – Project 3

December 2, 2021

Vehicle Detection from Aerial Footage Using YOLOv5 Models

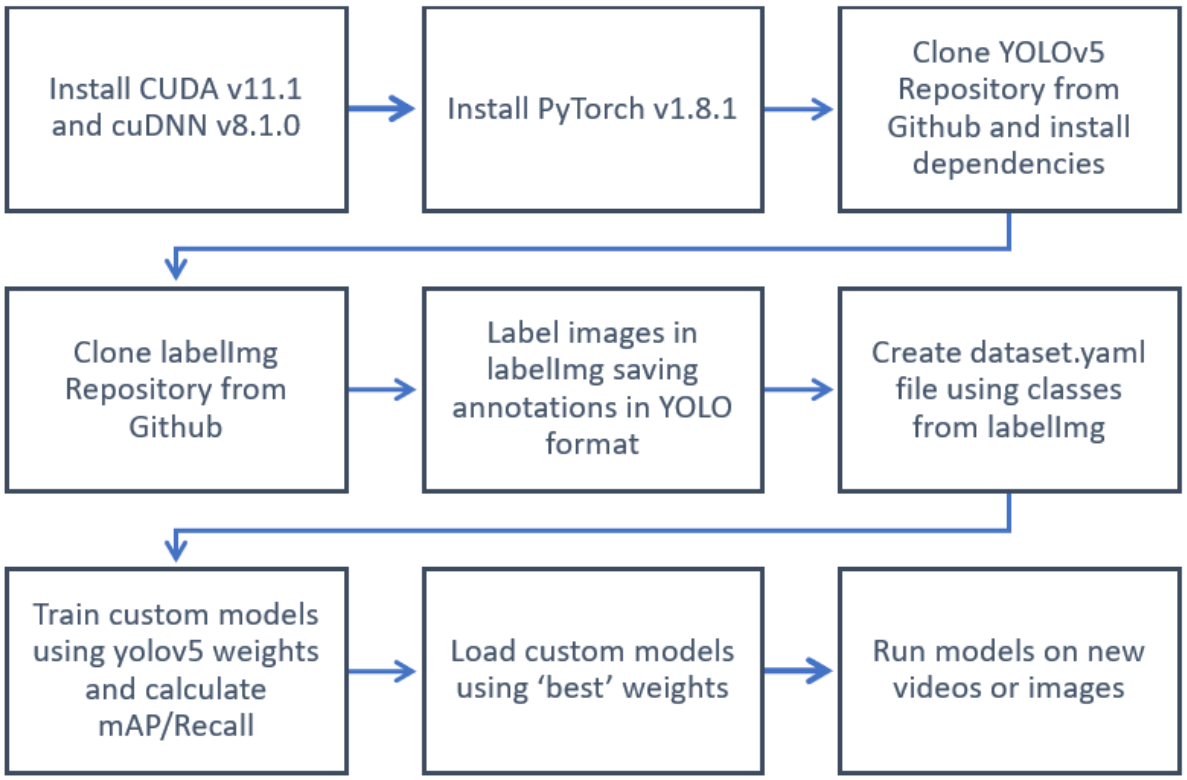
Description

YOLO (*You Only Look Once*) is an algorithm that employs convolutional neural networks (CNN) to detect objects in real-time. This algorithm is popular because of its speed and accuracy. YOLOv5 is the latest version of YOLO that utilizes the PyTorch framework and is pretrained on the COCO dataset. There are four different models to choose from for custom training – yolov5s (small), yolov5m (medium), yolov5l (large), and yolov5x (extra-large).

For this project, the yolov5s, yolov5m, and yolov5l models were trained using image snips from drone footage of UWF's campus after hurricane Sally. These images were hand labeled in labelImg using the labels – 'car', 'truck', and 'suv'.

Flow Chart

Process using YOLOv5:



Tegan Shiver

EEL6990 – Project 3

December 2, 2021

Install CUDA v11.1 and cuDNN v8.1.0
Install versions of CUDA and cuDNN that are compatible with PyTorch.

Install PyTorch v1.8.1
YOLOv5 requires PyTorch >= 1.7

Clone YOLOv5 Repository from Github and install dependencies
Clone the YOLOv5 repository from 'https://github.com/ultralytics/yolov5' and install all dependencies using 'cd yolov5 & pip install -r requirements.txt'

Clone labelImg Repository from Github
Clone labelImg repository from 'https://github.com/tzutalin/labelImg'.

Label images in labelImg saving annotations in YOLO format
Open labelImg and point the directory to image folder. Change annotation format to YOLO before saving labels, otherwise YOLOv5 will not work. 15 images were annotated for this project.

Create dataset.yaml file using classes from labelImg
LabelImg populates a 'classes.txt' file based on selected YOLO model (yolov5s). To train a custom model a YAML file must be created containing these class labels. Create a file names dataset.yaml in the same location YOLOv5 was cloned.

Train custom model using yolov5 weights and calculate mAP/ Recall
Train each model using selected weights in the Anaconda Prompt – 'python train.py --img 320 --batch 16 --epochs 500 --data dataset.yaml --weights yolov5s.pt --workers 2'. For the medium and large YOLOv5 weights, use yolov5m.pt and yolov5l.pt, in place of yolov5s.pt. Calculate mAP and Recall from generated confusion matrix.
The mean average precision is the proportion of detections that were correct,

$$\frac{TP}{TP + FP}$$

The average recall is the proportion of the actual objects that were captured,

$$\frac{TP}{TP + FN}$$

Load custom model using 'best' weights
model = torch.hub.load('ultralytics/yolov5', 'custom', path='
yolov5/runs/train/exp5/weights/best.pt', force_reload=True)

Run model on new video or images
Import media using OpenCV – run model (See code).

Tegan Shiver

EEL6990 – Project 3

December 2, 2021

Results and Analysis

Below are the Confusion Matrices, Training Batch images, and Validation Batch images with Predictions generated by the custom trained models described above.

YOLOv5s (Small)

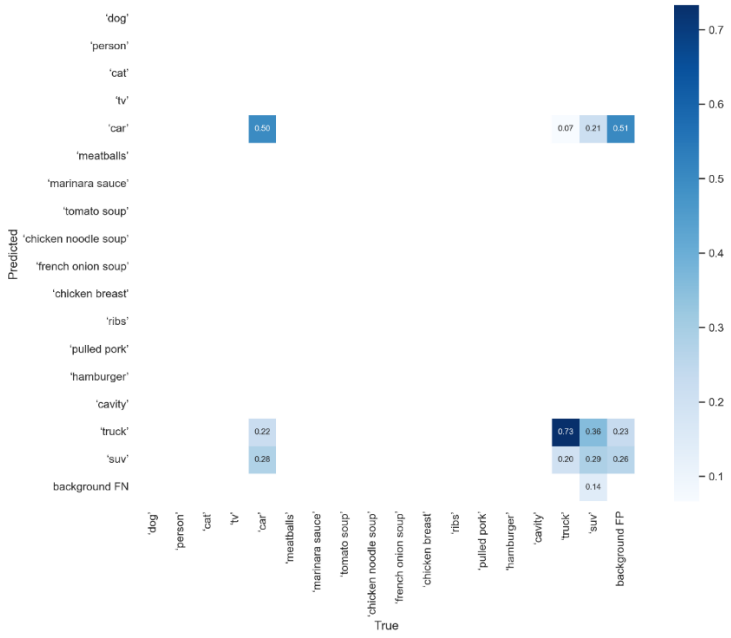


Figure 1: yolov5s Confusion Matrix

Figure 2: yolov5s Batch Images – Training (left) and Validation with Predictions (Right)

Tegan Shiver
EEL6990 – Project 3
December 2, 2021

YOLOv5m (Medium)

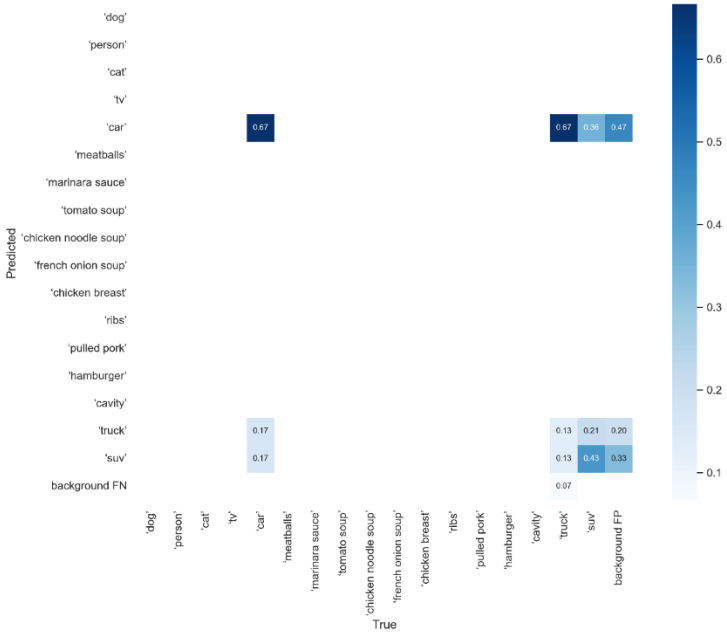


Figure 3: yolov5m Confusion Matrix

Figure 4: yolov5m Batch Images – Training (left) and Validation with Predictions (Right)

Tegan Shiver

EEL6990 – Project 3

December 2, 2021

YOLOv5l (Large)

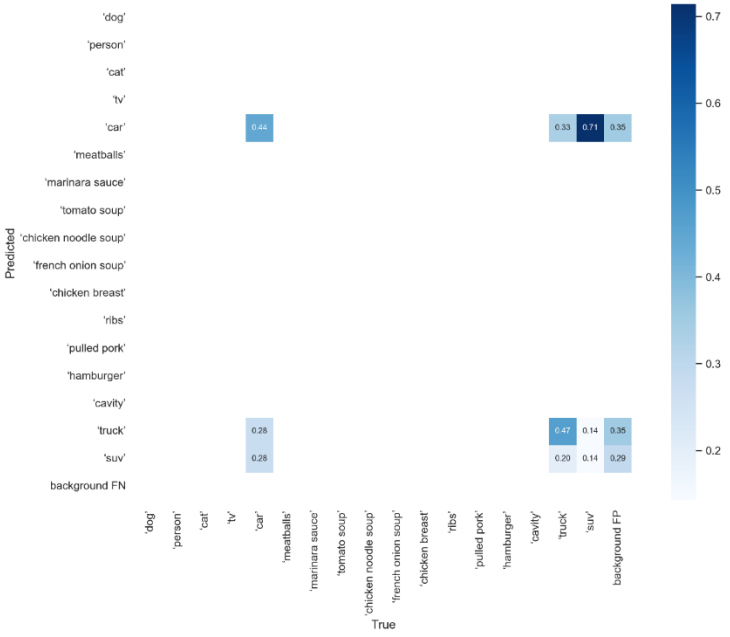


Figure 5: yolov5l Confusion Matrix

Figure 6: yolov5l Batch Images – Training (left) and Validation with Predictions (Right)

The Confusion Matrices were utilized to calculate the mean average precision (mAP) and average recalls of each model.

Model	Mean Average Precision (mAP)	Average Recall
yolov5s	60%	92%
yolov5m	55%	95%
yolov5l	51%	100%

Figure 7: mAP and Average Recall Table

As the models get larger, the mAP decreases (less vehicles are correctly classified), while the average recall increases (more vehicles are identified in total regardless of precision).

In Project 2, yolov5s was trained on the same data using 500 epochs, which resulted in a mAP of 65% and average Recall of 96%. For the sake of processing time, the models in the table above (Figure 7) were trained using 250 epochs for Project 3.

The mAP here is overall similar to the pre-trained Haar Cascade Classifier used in Project 1 (62%), but on average YOLOv5 was able to identify 70% more of the vehicles present (Cascade recall = 26%). This difference was not surprising since the YOLO models were trained using custom images/labels from the drone footage.

Tegan Shiver

EEL6990 – Project 3

December 2, 2021

Code

Python/PyCharm/Anaconda Prompt used

```
# install CUDA 11.1 & cuDNN v8.1.0
# https://developer.nvidia.com/cuda-11.1.0-download-archive
# https://developer.nvidia.com/rdp/cudnn-archive

# install PyTorch in Anaconda Prompt
# pip install torch==1.8.2+cu111 torchvision==0.9.2+cu111 torchaudio===0.8.2
-f https://download.pytorch.org/whl/lts/1.8/torch_lts.html

# Clone YOLOv5 from Github
# git clone https://github.com/ultralytics/yolov5

# install YOLOv5 dependencies
# cd yolov5 & pip install -r requirements.txt

# Clone labelImg from Github
# git clone https://github.com/tzutalin/labelImg

# Open labelImg in Anaconda Prompt

# Label Images - make sure to save annotations in YOLO format

# Create dataset.yaml file using classes file generated from labelImg

# Train custom models using Anaconda Prompt using yolov5 weights (yolov5s.pt,
yolov5m.pt, yolov5l.pt)
# python train.py --img 320 --batch 16 --epochs 250 --data dataset.yaml --
weights yolov5s.pt --workers 1

import torch
import numpy as np
import cv2

# use custom model
model = torch.hub.load('ultralytics/yolov5', 'custom',
path='C:/Users/TVS/anaconda3/CV/yolov5/runs/train/yolov5s'

'/weights/best.pt', force_reload=True)

cap = cv2.VideoCapture('Videos/Parking Services 2 1080p.avi')
while cap.isOpened():
    ret, frame = cap.read()
    # Make detections
    results = model(frame)
    cv2.imshow('YOLO', np.squeeze(results.render())
    if cv2.waitKey(10) & 0xFF == ord('q'):
        break
cap.release()
cv2.destroyAllWindows()
```

Tegan Shiver

EEL6990 – Project 3

December 2, 2021

YAML Code – dataset.yaml

```yaml
path: ../data  # dataset root dir
train: images
val: images

# Classes
nc: 17  # number of classes
names: [ 'dog', 'person', 'cat', 'tv', 'car', 'meatballs', 'marinara sauce', 'tomato soup', 'chicken noodle soup', 'french onion soup', 'chicken breast', 'ribs', 'pulled pork', 'hamburger', 'cavity', 'truck', 'suv' ]  # class names
```

References

- Lowande, R., Clevenger, A., Mahyari, A., Sevil, H.E., "Analysis of Post-Disaster Damage Detection using Aerial Footage from UWF Campus after Hurricane Sally". *International Conference on Image Processing, Computer Vision, & Pattern Recognition (IPCV'21)*, Las Vegas, USA, 26-29 July 2021.

- YOLO Algorithm Background
  https://www.section.io/engineering-education/introduction-to-yolo-algorithm-for-object-detection/

- YOLOv5 Tutorial
  https://www.youtube.com/watch?v=tFNJGim3FXw

- Install CUDA v11.1
  https://developer.nvidia.com/cuda-11.1.0-download-archive

- Install cuDNN v8.1.0
  https://developer.nvidia.com/rdp/cudnn-archive

- Install PyTorch v1.8.1
  https://pytorch.org/get-started/locally/

- Clone YOLOv5 from Github
  https://github.com/ultralytics/yolov5

- Clone labelImg from Github
  https://github.com/tzutalin/labelImg