**Computer Vision Capstone User Guide**

**Roboflow/data collection (Meg)**:

1. Gaining access to dataset on Roboflow:

* You need to create a Roboflow account
* Dr. Frommer or Dr. Bagchi Misra will be able to add you to the “USCGA” workspace
* Once you’re a member of the workspace, you can find the dataset under the “ORDA2023CAPSTONE” project.

1. Data Collection Process:
   * Cyber ARP team has a motion detection algorithm, running in real-time, that records whenever the camera detects motion.
   * Those recordings are then saved on the Server (see **Server** section of this document, section c.)
   * This motion detection algorithm is not the best at narrowing down the search for boats within the videos, ie we have to manually search all saved videos for boats because some of the videos were false positives.
     + A great future work for you guys would be utilizing another method or model to detect when there is a boat in the frame, for data collection. Potentially, using the model that we trained this semester that is on the server already)
   * Once you have the videos that you would like to add to the dataset, you can upload desired number of frames from that video to Roboflow by going to the “Upload” tab on the project’s home page.

Graphical user interface, application, Teams

Description automatically generated

1. Dataset features:

* There are a mixture of images taken from public datasets on Roboflow and data collected from the cameras on the Thames
* We ended up training on just the Thames data, which had the following class-breakdown: Commercial, 41.1%; Recreational, 34.4%; Sailing, 24.5%
  + As you collect more data, it is pertinent that you focus on balancing these percentages in order to produce better results.
* Roboflow allows you to download the dataset in the desired format for YoloV7, and other computer vision models. Example below:

Text

Description automatically generated

* An additional feature to Roboflow is when you download the dataset to use for the training process, you can choose which classes you want to include in the version you are downloading.
  + For example, you can exclude all images with the class label “apple”

1. Roboflow Pitfalls:

* You can only delete one image from the dataset at a time which is extremely time consuming when you wish to delete a large amount of images
  + This is why, even though we did not use the open-source data in our final dataset that we trained on, we did not delete those images from the Roboflow dataset.
* When you upload a video into Roboflow, be aware that when you’re asked to specify how many frames from the video you wish to include in your dataset, you usually get less frames than you want
* When uploading new data to the dataset, Roboflow assigns a certain percentage of the data to your train/test/validation subsets. However, this assignment is *not* randomized, and you get repetitive data within these subsets which efffets your models learning process and leads to a worse model
* It seemed that, as the dataset became larger, the Roboflow website became laggy and slow. This could be due to us not subscribing to the “Starter” or “Start

1. Potential Avenues for Data Collection:

* There are other open-source labeling tools such as CVAT, hasty.ai and Labelbox
* We did not look into these other programs due to Roboflow's flexibility with the format it downloads the dataset onto your computer, which offered ease when downloading the dataset in the correct format for YoloV7.
* If you are able to upgrade the workspace’s subscription in Roboflow, this might allow you to do more things with the program!

Other Labeling Link: https://dida.do/blog/the-best-labeling-tools-for-computer-vision

**Server (Elise)**:

1. VPN access:

* Ask cyber majors to get your computer a connection to the cybernet VPN
* You need to be connected to the cybernet VPN to get on the server

1. Connecting to the server

* Only one person can be actively connecting to the server at the same time
* Only one person can be connected to each username on the server
  + There is orda (where we have everything set up), orda2 and orda3 (we haven’t set up anything on either of these)
    - They all have the same password noted below
* Connect to the cybernet VPN
* Use the Remote Desktop app to connect to the computer 10.233.107.2
* Type in the username: orda and password: orcaisawesome on the blue screen
* Sign in to “orda” with the password: orcaisawesome on the purple screen
* This will bring you to the Linux server

1. Accessing videos recorded by server

* Videos saved in dvr\_lapsetup directory under videos
  + These videos are supposed to all have boats but the basic detection algorithm set up right now isn’t the best (this could be a great future work task for you guys!) so you have to watch each video to see if there is a boat in it
* The videos are saved by camera
  + Some of the videos are external (not from the Thames)

1. Accessing the yolo v7 model files

* Go to home > orda\_capstone > yolov7
* This is the yolov7 directory referred to in the majority of this document

1. Accessing pre-saved videos we have put into Roboflow

* Go to home > orda\_videos

1. Accessing jupyter notebooks through anaconda navigator

* This is also weird because of Linux

Text

Description automatically generated

1. Roboflow will not break up the data into train/test/validation randomly or in your specified percentages, so we used splittingFolders.ipynb to do this (located in orda\_capstone/yolov7)

* Make sure you make the necessary folders within the directory before you run or there will be an error (this is outlined in comments in the jupyter file)
  + Follow the format that the train, test, and valid folders are in
* We also found it helpful to create copies of each final train, test, valid folders containing the labels and images and store them in a different directory because then if they got messed up for any reason you wouldn’t have to make all new copies

1. Count the number of instances of each class with the CountClasses.ipynb stored in orda\_capstone/yolov7

* You will need to change the word “train/test/valid” to count accordingly

1. Accessing the virtual environment to run yolov7:

* You need to be in the virtual environment to run evolve, training, and testing
* Virtual environment called yolov7training
* First command brings you to the base directory
* Second activates the virtual environment
* Third brings you to the directory where everything for yolov7 is stored
* The fourth line is where you would type commands to run evolve, training, or testing

Text

Description automatically generated

1. Run hyperparameter evolution on server:

* Set the number of evolutions/epochs you want to do in the train.py file in line 667 (300 is recommended but that could take a week, so we did 100 and it took about four days)
  + Click “save” in the top right corner after you edit a .py file in Linux
* This is the command to run hyperparameter evolution
  + It is the same as training but you put -- evolve on the end
* The epoch value here will be how many training evolutions happen per evolve epoch
  + If you have 80 training epochs and 100 evolve epochs there will be 800 combined which is why this takes multiple days to run (it can take up to a week to evolve if you evolve every hyperparameter)
* To make a hyperparameter not evolve, in the train.py file in lines 622-652 put a 0 in the first index or a 1 to make it evolve, then CLICK SAVE



* + Here you can see fliplr will not evolve and mosaic will
* It will tell you where the evolved hyperparameter values are saved and the outputted graph will be saved in the yolov7 directory
* After you do this, you need to train the algorithm with the evolved hyperparameter weights

1. Run training on server:

* This is the command for running training







* call the train.py
* Specify which weights you want to use
  + These will either be one of the default weights or your custom/evolved weights
* Specify your data
* Specify image size (we usually did 320 so the Nano could handle it and we compared it to the optimal 1056 size with default values)
* Specify the cfg (either full or tiny)
* Hpy – the hyperparameter file
* Specify the number of training epochs
* Specify batch size

1. Run testing on server:

* This is the command for running testing (all one line)





* Call the test.py
* Specify which weights you want to use
  + These will either be one of the default weights or your custom/evolved weights
* Specify your data
* Task is test
* Specify image size (we usually did 320 so the Nano could handle it and we compared it to the optimal 1056 size with default values)
* Specify your desired name
* Specify batch size

1. Viewing the results:
   1. W and B

* Create your own account at <https://wandb.ai/home>
  + When you start training it will give you a link to send the results to your WandB account (change this to your account from our account)
    - I believe this can also be changed through the yolov7>utils>wandb\_logging code
* W and B allows you to see the outputted graphs being generated through each training iteration
  + This allows you to gauge how many epochs you should run (see where the graphs level out)

* 1. The results will also be stored under the runs folder found in yolov7>runs
* Check the rough key of what each of the output graphs are/ what they will ideally look like

1. Putting model on the Jetson Nano

* We set up yolov7 on the nano (see appendix in paper)
* We sent cyber majors our optimized weight file and they implemented it

1. Deploy the model

* This was mainly done by the cyber majors on the nano after we sent them the optimized weights
* Run the detect.py (all of this is one line)



* Specify your desired weights
* Source is what the detect.py will be running on, in this case it’s an example video but the cyber majors will run in on the live camera feed
* Img is the input image size

Yolov7 Paper: <https://github.com/WongKinYiu/yolov7/blob/main/paper/yolov7.pdf>

Yolov7 GitHub: <https://github.com/WongKinYiu/yolov7>

Hyperparameter evolution GitHub: <https://github.com/ultralytics/yolov5/issues/607>