USER MANUAL
TRUCK DETECTION PROJECT
Version: 3.0

TRUCK DETECTION

FORT CAPITAL

Version	Description	Author
1.0	Initialize the project overview	Hy Dang
1.1	Add System Structure	Hy Dang
2.0	Add System Requirements	Hy Dang
3.0	Add Using Program	Hy Dang

1. GENERAL INFORMATION

a. Project Introduction

- The purpose of this document is to collect, analyze, and define the business requirements, i.e. high-level needs, desired ultimate business outcomes, and features of the Truck Detection.
- It focuses on the capabilities needed by the stakeholders and the target users, and why these needs exist in the first place. The details of how the Truck Detection fulfills these needs are detailed in the use-case and supplementary specifications.

b. Project Perspective

- This product will be completely self-sustained until a time Fort Capital is able to merge with other developing technologies.

2.System Structure.

a. Types of Users

There will be only one type of user which is Fort Capital’s analysts. The user has the access to all the features of the codes referred to later in the report.

b. System Organization

All the features are available for the users including:

- Getting the satellite images

- Training the model
- Predicting the images

3. Using the system

a. Requirements

- Users have to install all requirements before using the program. All the requirements are stored in “requirements.txt” and it is easily installed by the command “pip3 install -r requirements.txt”.
- All the required packaged are:

```
torch == 1.7.1
torchvision == 0.8.2
keplergl
mercantile
pillow
git+https://github.com/facebookresearch/detectron2.git
fvcore
keplergl
```

b. Initialize the Parameters

- Users have to initialize parameters to run the program. They can do it by adjusting the “config.yml” file.

```
dataset:
  raw_link: "https://github.com/trangdao909/TruckDetective/raw/main/TruckDataset_TD.zip"
  num_workers: 2
  train_path: "truck_train"
  test_path: "truck_test"

model:
  model_path: "COCO-InstanceSegmentation/mask_rcnn_R_50_FPN_3x.yaml"
  ims_per_batch: 2
  base_lr: 0.00025
  max_iter: 2000
  num_classes: 1
  output_dir: "models"
  score_thresh_test: 0.5
  output_csv: res/test_result_amazon_warehouse_2.csv

data_getter:
  top_left_position: [32.830252, -97.326590]
  bottom_right_position: [32.822020, -97.313497]
  zoom_scale: 21
  output_img_dir: "dataset/predict_img/dorian_predict/"
  batch_img_size: 18
  saved_location: "dataset/saved_location_2_dorian_test2.txt"
```

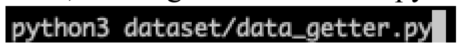
- There are three main key factors in the “config.yml” file:
  - **Dataset:**

        - The dataset dictionary contains parameters about the labeled training dataset (raw_link).
        - The name for the training part (train_path).
        - The name for testing part (test_path)
    - **Model**
        - Configuration for Mask_rcnn path (model_path).
        - Number of image per batch (ims_per_batch)
        - The base learning rate for the model (base_lr)
        - Maximum of iterations for training (max_iter)
        - The number of classes for classifying. In this case, since we only classify the truck. num_classes = 1
        - Output Result of the model after training. (output_dir)
        - Check the threshold for the truck (confidence level for the prediction) (score_thresh_test)
        - Output CSV file path. (output_csv)
    - **Data_getter**
        - Top_left_position: The latitude/longtitude coordinate for the top left portion of the location
        - Bottom_right_position: The latitude/longitude coordinates for the bottom right portion of the location
        - Zoom_scale: The zoom scale for satellite images. Please check MapBox API Documentation for detail
        - Batch_img_size: Number of images you want to download once at a time (saving storage)
        - Saved_location: Storing downloaded location in mercantile coordinates

c. Getting the Satellite Images

If you just want to get the satellite images. You can initialize the bottom_right_position and top_left_position in the config.yml file.

Then, running the command: “python3 data_getter.py”



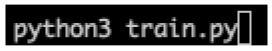
```
python3 dataset/data_getter.py
```

Then the images are saved in the “output_img_dir” folder in config.yml.

d. Training the Model

If you want to train the model. You can initialize all the parameters in the “model” part in the config.yml file.

Then running the command:



```
python3 train.py
```

The outputs of the models are saved into the file “output_dir” folder in config.yml

e. Predicting the Images

If you want to predict the model. You can initialize all the parameters in the "model" part in the config.yml file.

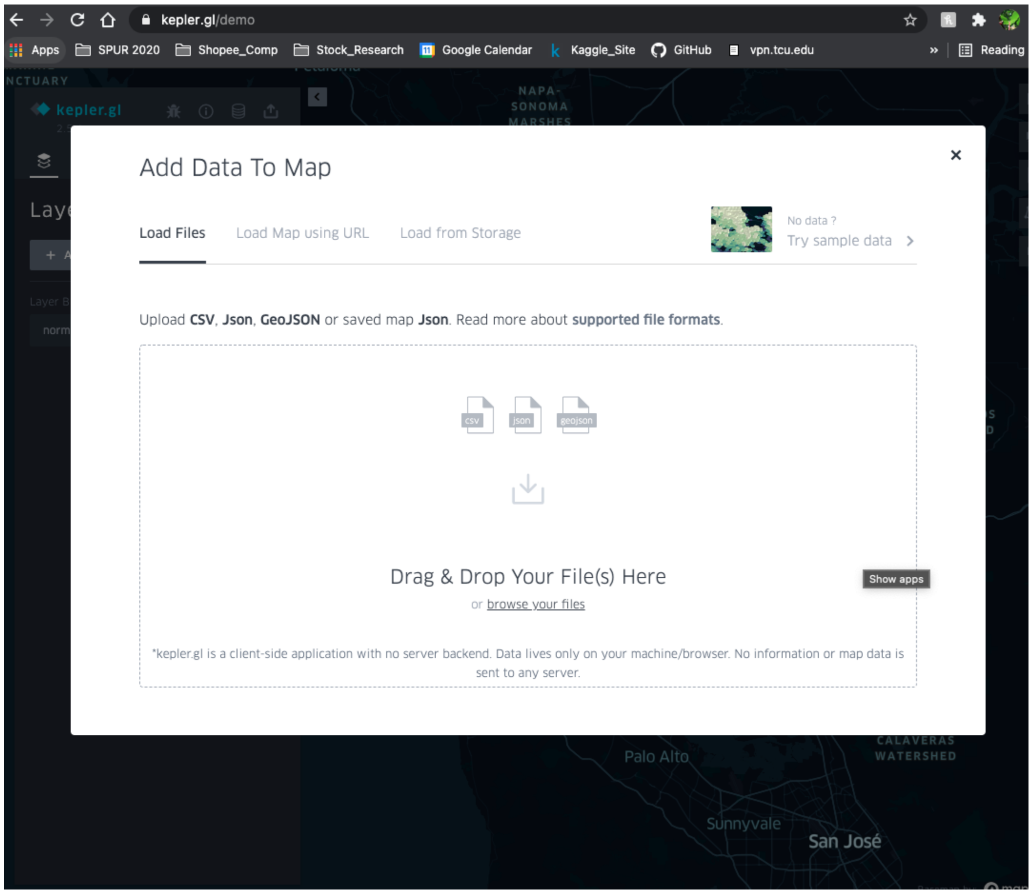
Then running the command:

```
python3 predict.py
```

The outputs of the prediction are saved into the file "output_csv" in config.yml

f. Checking/Visualizing the result

Users can check the result by putting csv result file into kepler.gl. Please check kepler.gl documentation for detail.



Upload **CSV**, **Json**, **GeoJSON** or saved map **Json**. Read more about **supported file formats**.

*kepler.gl is a client-side application with no server backend. Data lives only on your machine/browser. No information or map data is sent to any server.