# DEVELOPER MANUAL
# TRUCK DETECTION PROJECT
# Version: 2.0

**TRUCK DETECTION**

**FORT CAPITAL**

| Version | Description | Author |
|---|---|---|
| 1.0 | Initialize the Project Developer Manual | Hy Dang |
| 2.0 | Revision of the Developer Manual | 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 Fort Capital can merge with other developing technologies.

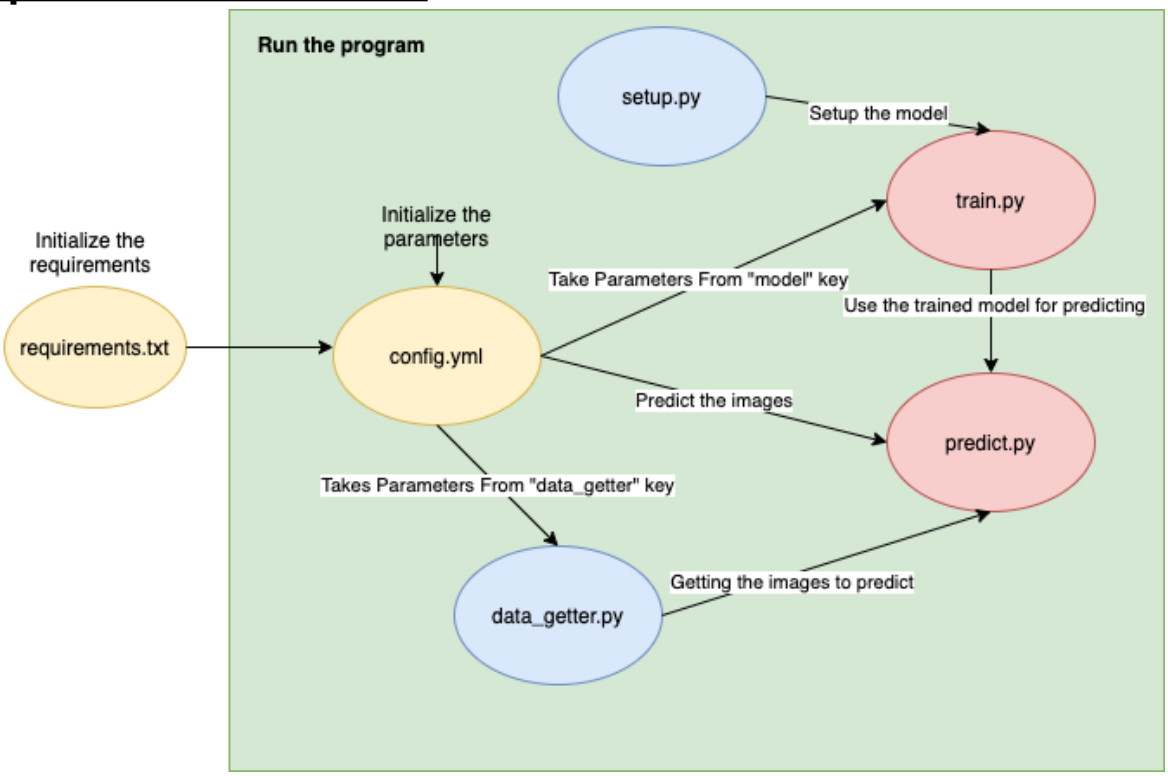
# 2.System Structure.

## a. System Organization

All the features are available for the users, including:

- Getting the satellite images
- Training the model
- Predicting the images

## b. **Components Interactions**




# 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:

| **Packages** | **Documentation** |
|---|---|
| python 3.8 | https://www.python.org/ |
| torch==1.7.1 | https://pytorch.org/ |
| torchvision==0.8.2 | https://pytorch.org/ |
| mercantile | https://pypi.org/project/mercantile/ |
| pillow | https://pypi.org/project/Pillow/ |
| detectron2 | https://github.com/facebookresearch/detectron2 |

| | |
|---|---|
| fvcore | https://pypi.org/project/fvcore/ |

## b. Initialize the Parameters

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

```yaml
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 the 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

This feature allows the users to get the images from MapBox API. To get the pictures from MapBox API, we need to initialize the top left and bottom right for the region. Then, we can run two loops to search horizontally and vertically for images. Then we can get the images by running script.

```
requests.get('https://api.mapbox.com/v4/mapbox.satellite/'
```

Please check MapBox API documentation for more details

## d. Training the Model

To train the model and to develop the model, the developer can check the file train.py. This file implements detectron2 as the Deep Learning model. Thus, it needs a setup, and it is in setup.py. In train.py, we implement DefaultTrainer. Before training the datasets, we need to convert the dataset into COCO format.

## e. Predicting the Images

To predict the model, the developer can check the file predict.py. The developer should check CUDA (GPU available) to implement the code faster. Moreover, the file implements batching prediction, which is predicting multiple images as a file. Please check detectron2 documentation for more detail.

Predict.py inherits both the data_getter file and config.yml file. The process is getting the images by implementing data_getter.py and using predict.py to predict the image. Moreover, it is implementing the storage solution, predicting parts of regions, then deleting those to get the new images. It helps to solve the storage problem.

# 4. API Documentation

Please check API Documentation of setup, predict and train in our main deliverables page.