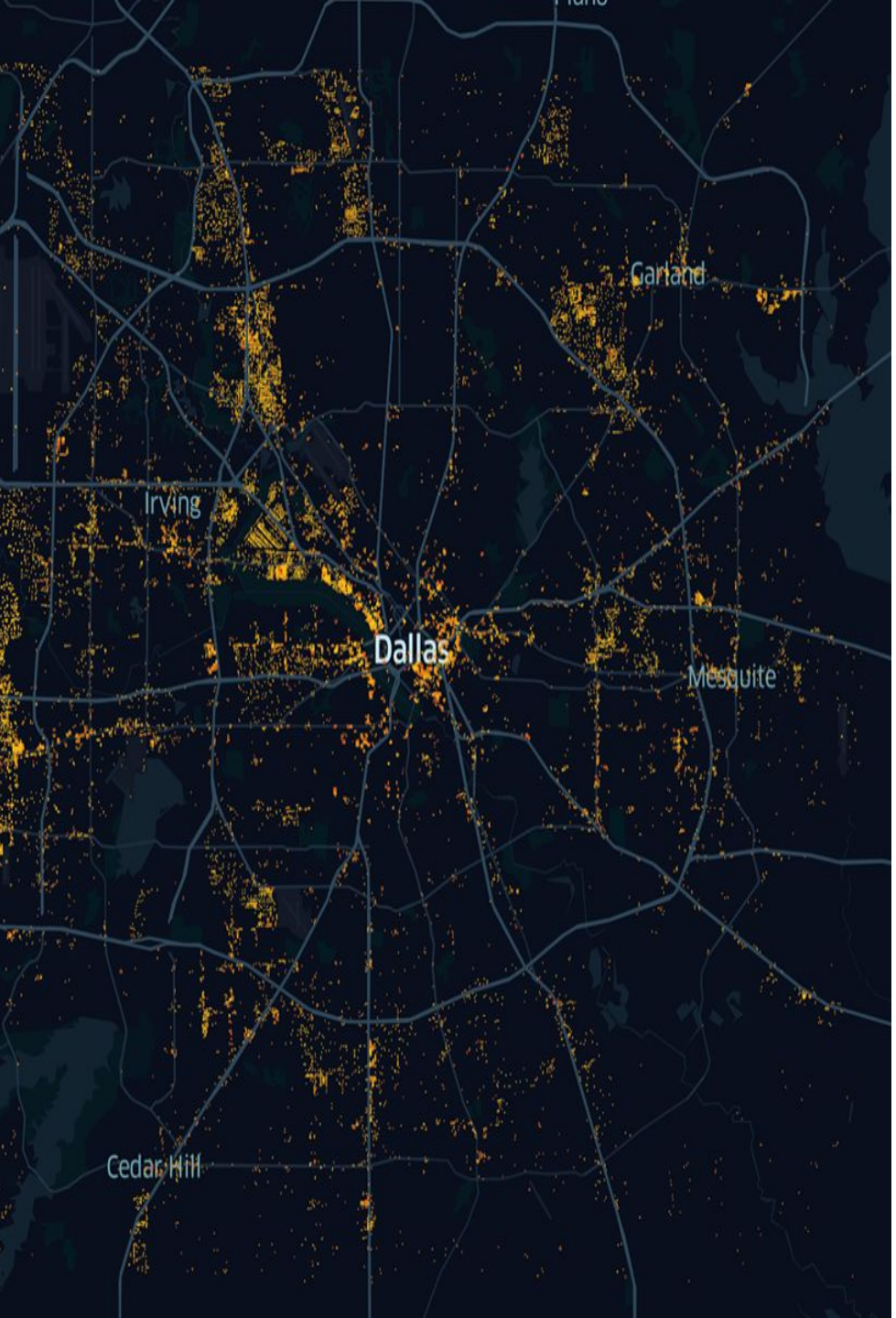


Truck Detection

SENIOR DESIGN 2020

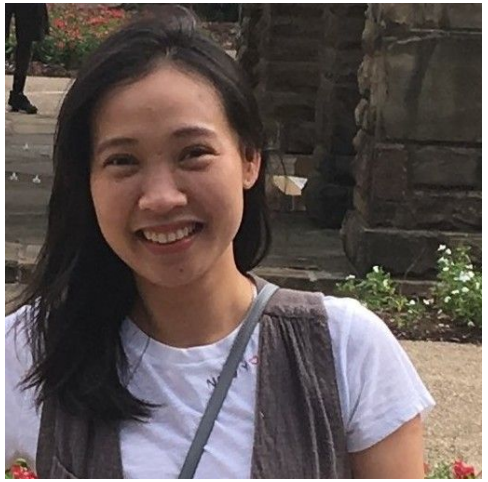
The Team

- Ben Ruelas, Team Lead
- Hy Dang, Tech Lead
- Minh Nguyen, Developer/Data Scientist
- Trang Dao, Developer/Data Scientist
- Dorian Dhamo, System Admin

Ben Ruelas

Hy Dang

Minh Nguyen



Trang Dao

Dorian Dhamo

The Outline

- Ben: Project Overview, Introduction to the Problem
- Dorian: Deep Learning Model, Technology Introduction
- Minh: Dataset and Programming Environment
- Trang: Project Demo and Technical Overview
- Hy: Project Improvements and Optimization

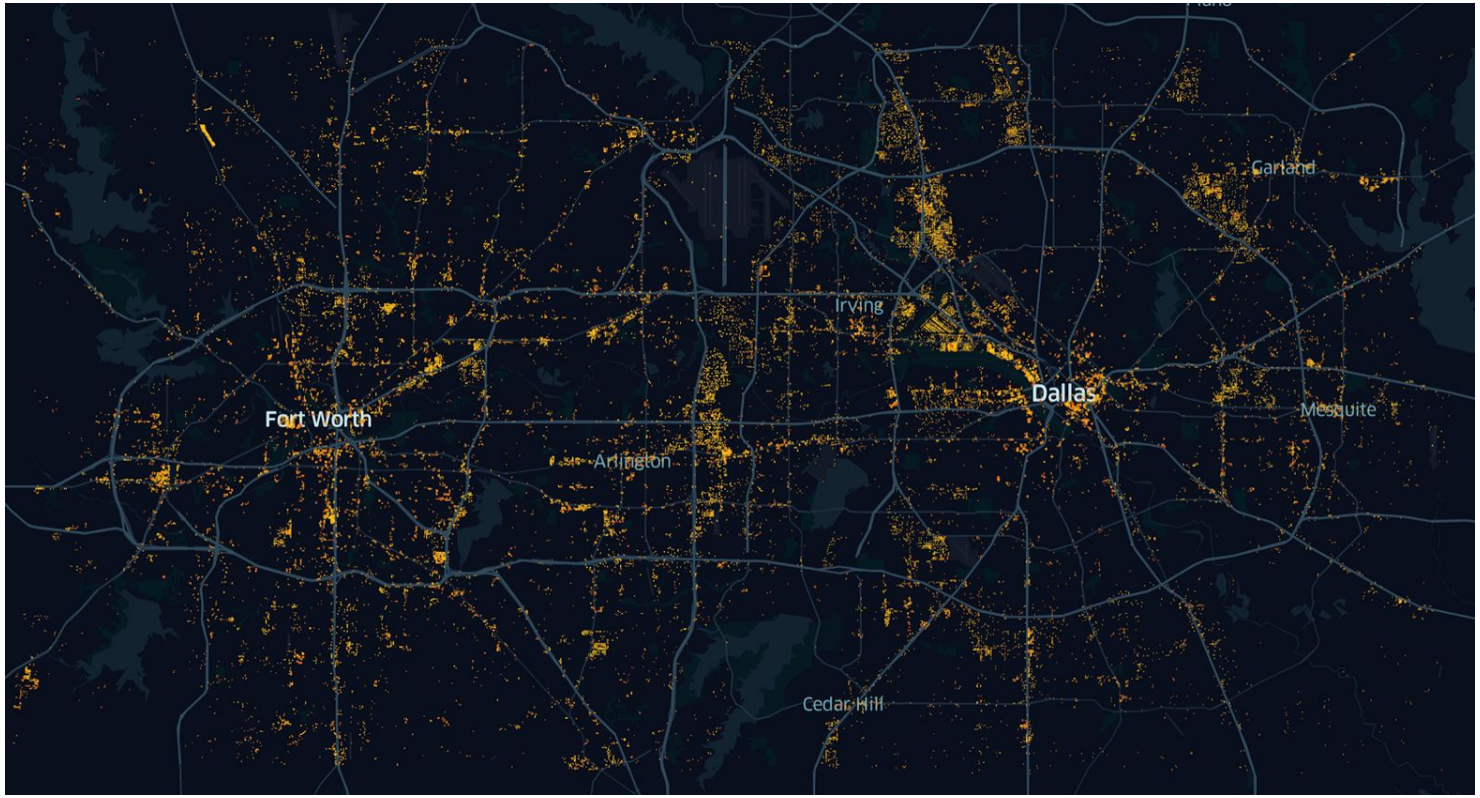
The Client

- Greg Adams, Director of Technology
- Real estate private equity fund
- Focus on acquiring class B industrial properties

FORT CAPITAL

The Problem

- Inaccurate county data
- Poor property classifications don’t reflect purpose of properties
- Lack of insight

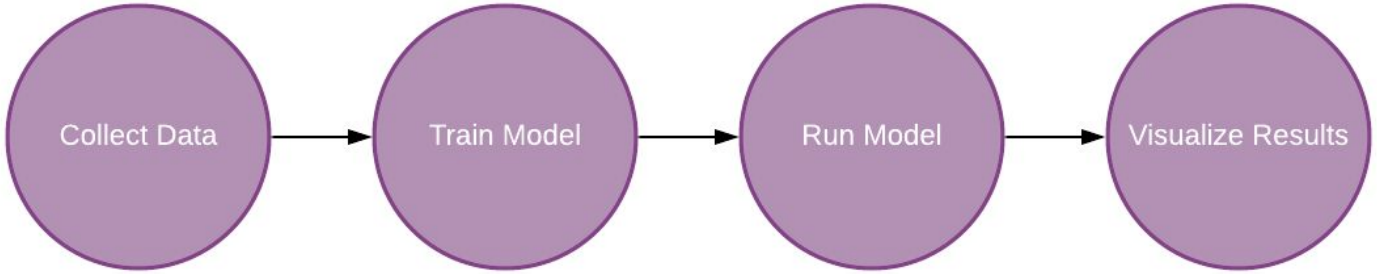


The Goal

- Detect 18-Wheeler trucks using image segmentation
- Help detect how property is being used
- Reclassify property data
- Gain geographic insight

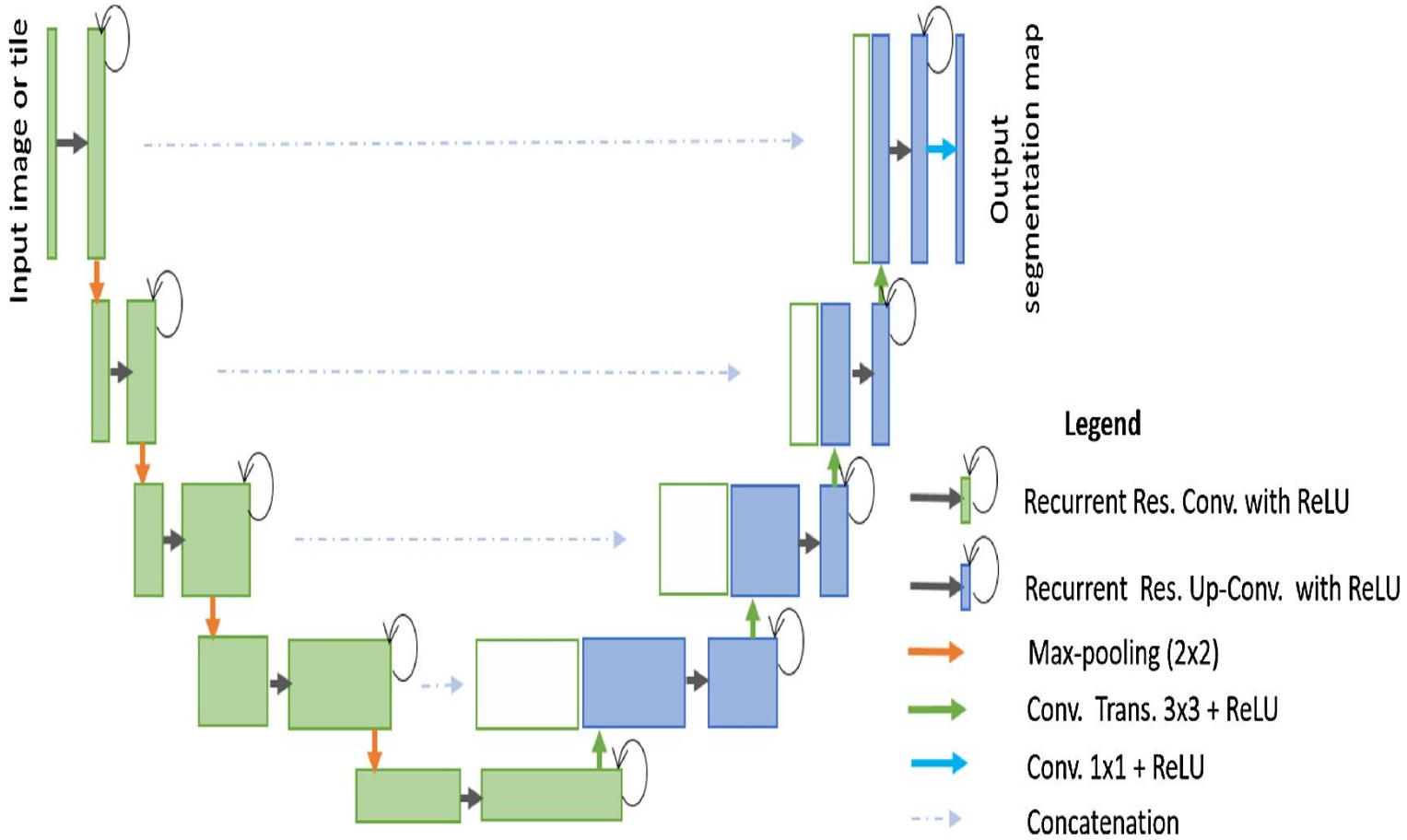
The Process

- Collect thousands of satellite images
- Train a deep learning model to detect the trucks
- Run the model on our dataset
- Visualize the results



The Model

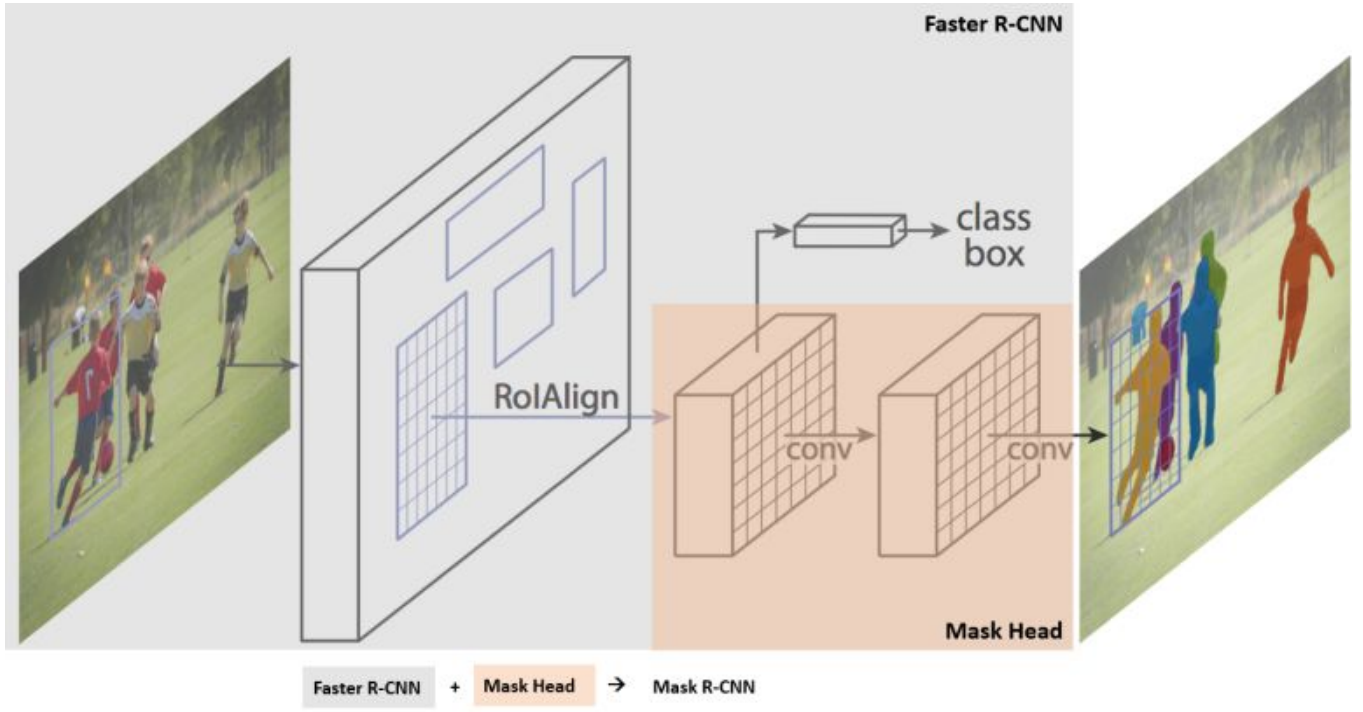
- Started with U-NET Model
- Biomedical image segmentation
- Transitioned to Mask R-CNN



U-NET Model

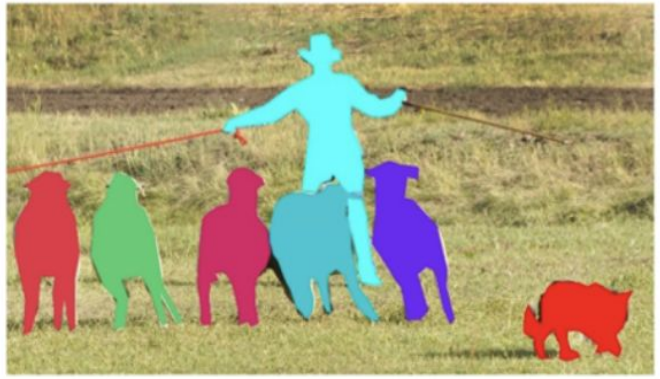
The Model (part 2)

- Work to improve prediction accuracy
- Prevent overfitting
- Speed and storage optimizations



Mask R-CNN

(c) Semantic segmentation



(d) Instance segmentation

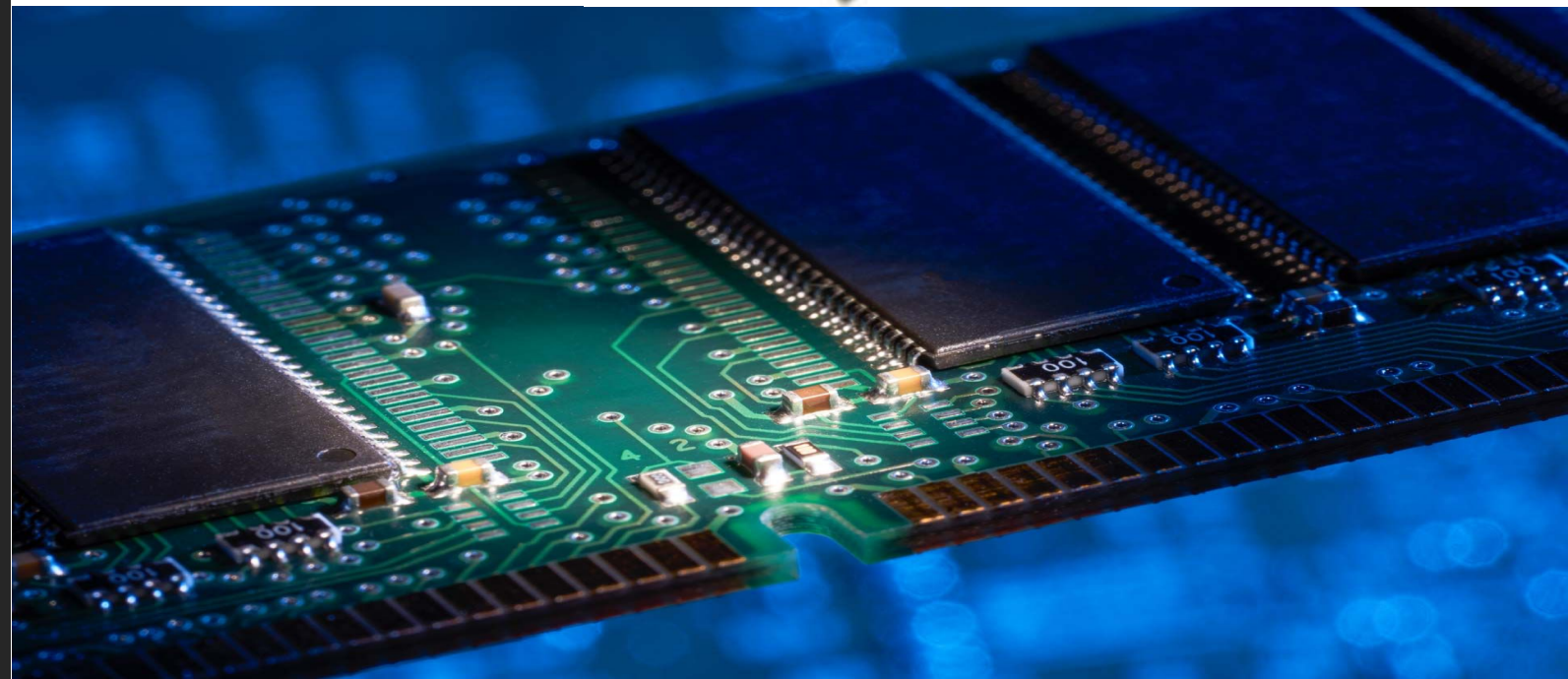
The Environment

Google Cloud

- 16 GB RAM
- 15 GB GPU
- 100 GB Storage
- Environment

Set Up

- Debugging

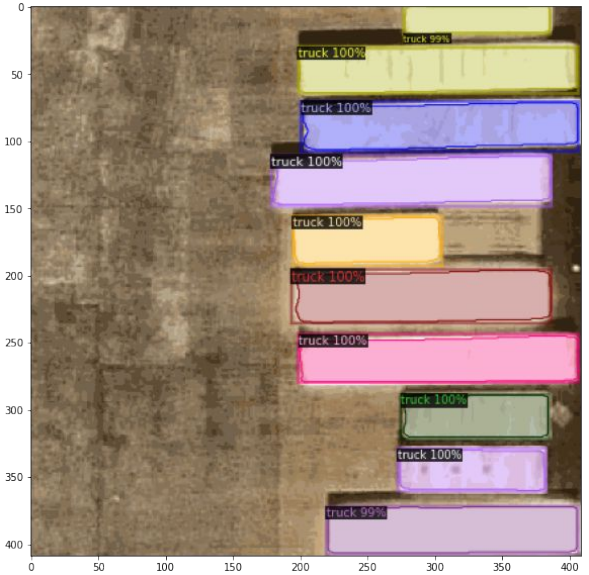


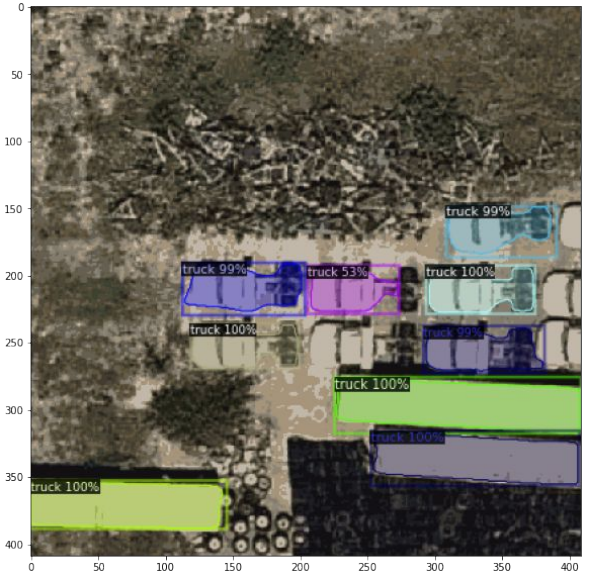
Mapbox API

- Mapbox is a location data platform that powers the maps and location service used in many popular app
- The Mapbox APIs allow us to access Mapbox tools and services

The Dataset

- Achieve from Mapbox API
- Use longitude and latitude
- The area of DFW



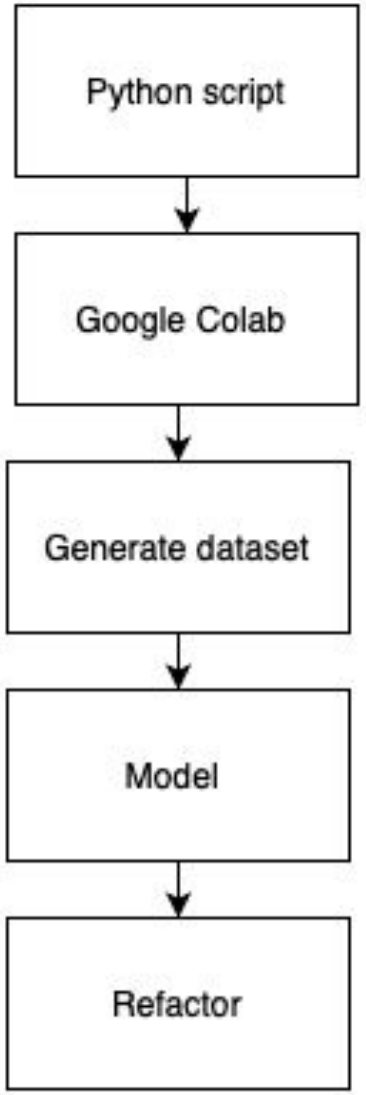


Google Colab

- Colab notebooks are Jupyter notebooks that run in the cloud
- It allows us to edit the notebook together

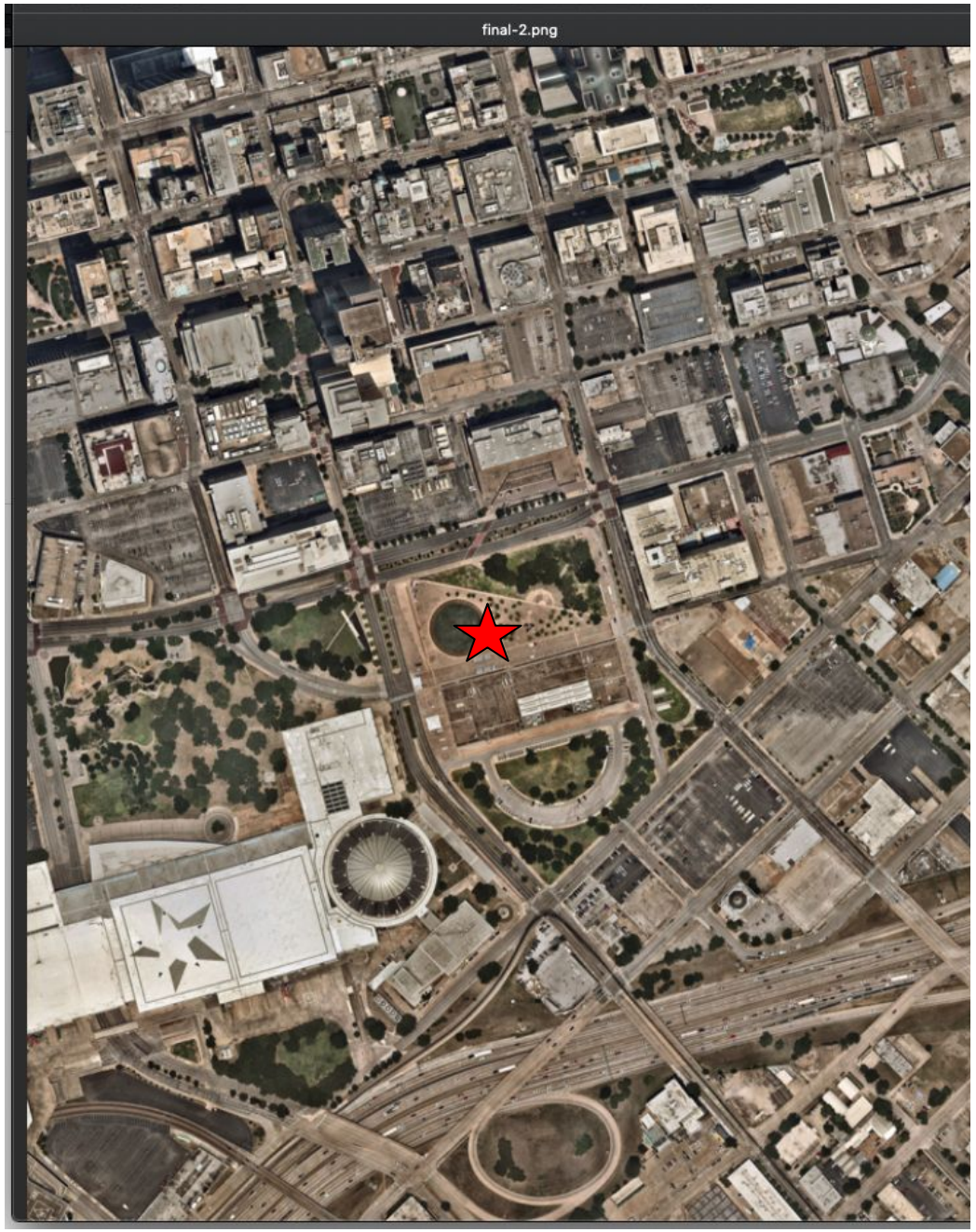
Google Colab For Data Generalization

- Script to achieve data from Mapbox API and store in the the cloud storage
- Visualize and validate the process of data getter



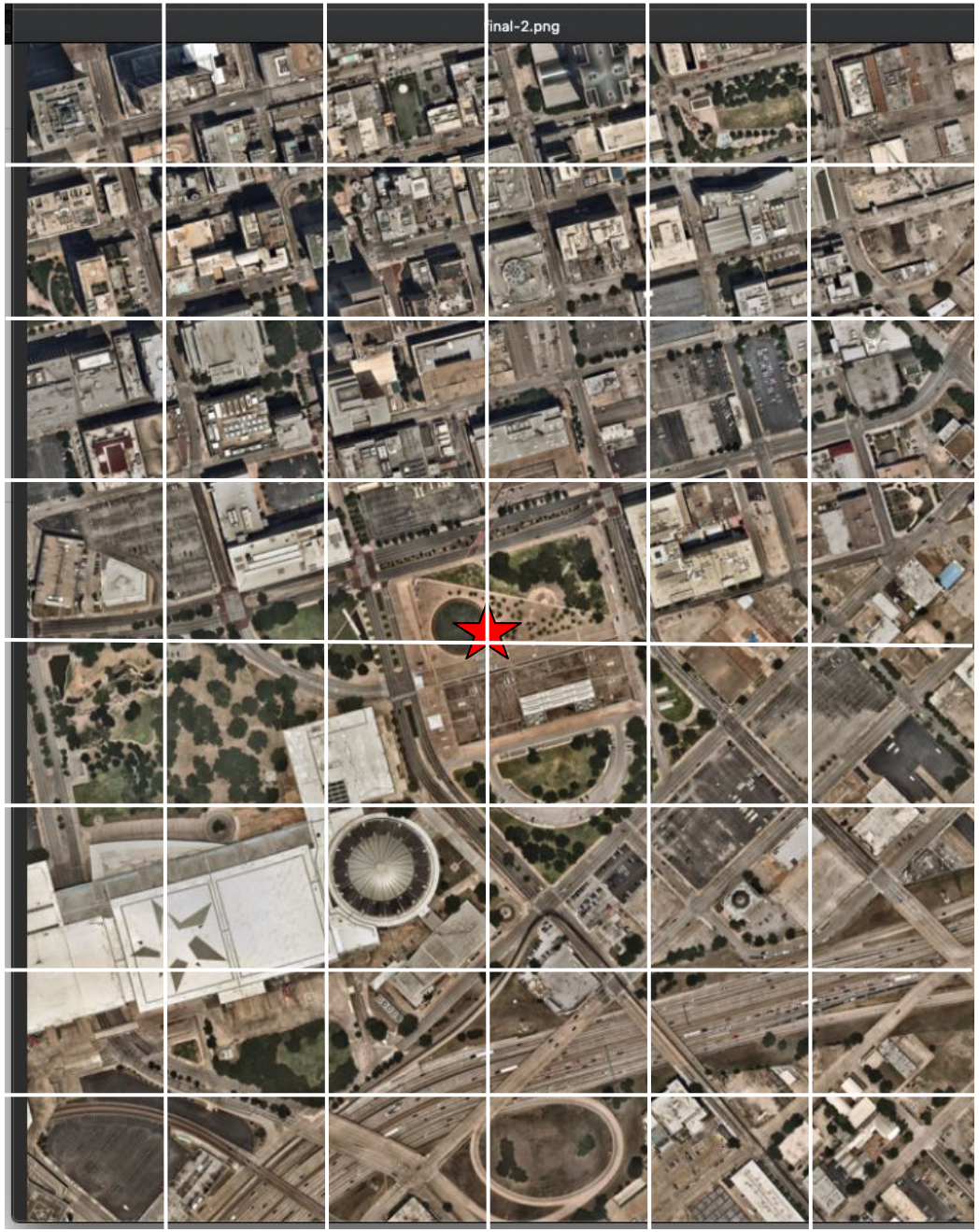
Google Colab For Data Generalization

- Visualize and validate the process of data getter



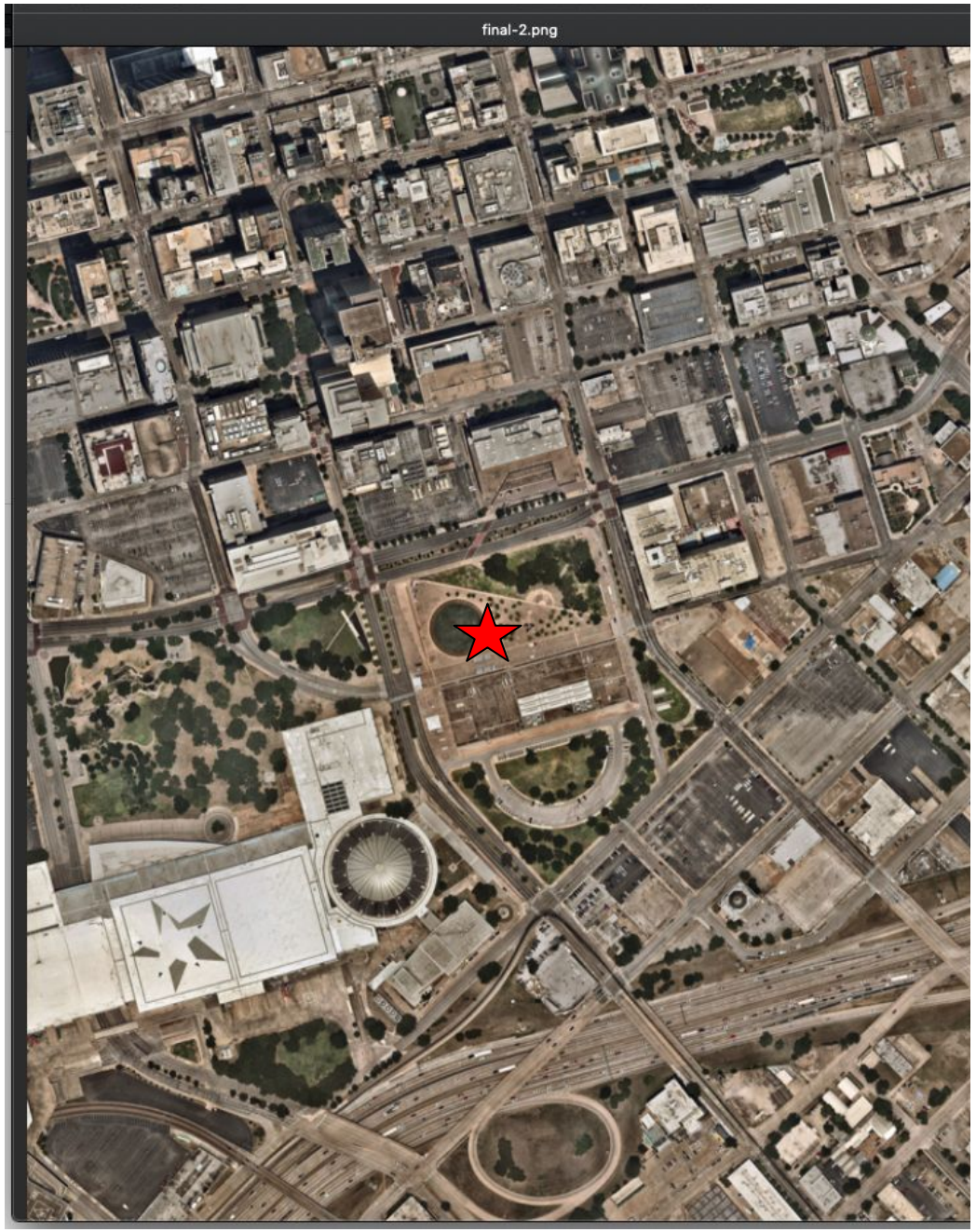
Google Colab For Data Generalization

- Visualize and validate the process of data getter



Google Colab For Data Generalization

- Visualize and validate the process of data getter



Google Colab Demo

Step 1

- Setup Detectron2
- Get labeled data

Step 2

- Register dataset
- Coco format

Step 3

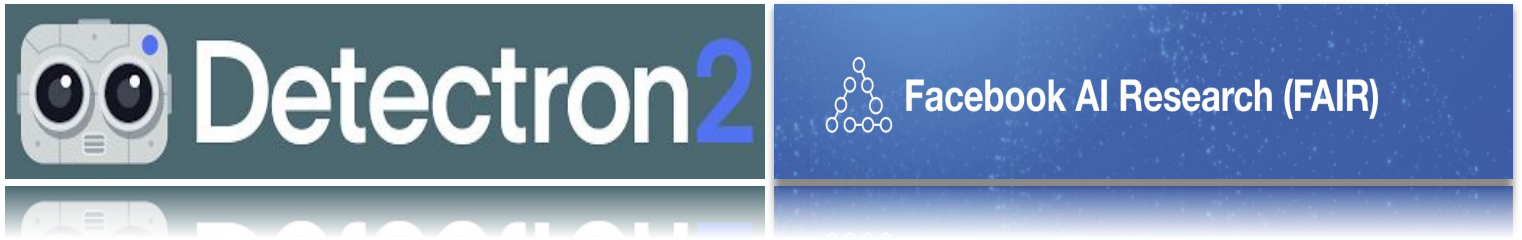
- Train model
- Save model and config

Step 4

- Test model on validation set
- Evaluate model

Step 5

- Apply model on larger dataset
- Save result to .csv file

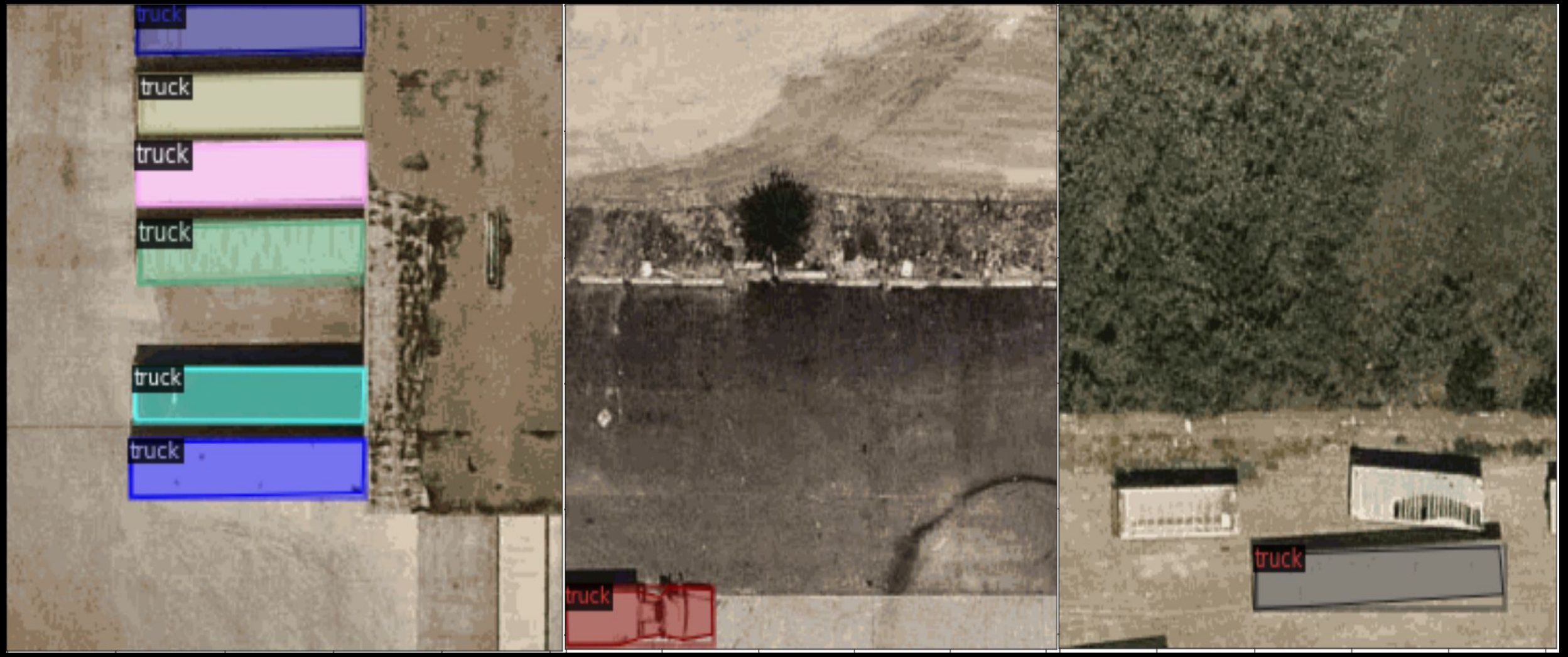


Detectron2 includes high-quality implementations of state-of-the-art object detection algorithms, including Mask R-CNN model.

From Detectron2,
import zoo_model

Import dataset
with trucks labeled
from github

Visualize dataset



Step 1

- Setup Detectron2
- Get labeled data

Step 2

- Register dataset
- Coco format

Step 3

- Train model
- Save model and config

Step 4

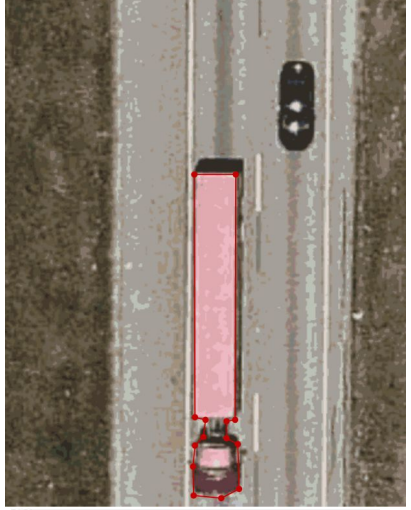
- Test model on validation set
- Evaluate model

Step 5

- Apply model on larger dataset
- Save result to .csv file

COCO is large scale images with Common Objects in Context (**COCO**) for object detection, segmentation, and captioning dataset

COCO format defines exactly how annotations (bounding boxes, object classes, etc) and image metadata (like height, width, image sources, etc)




```
{
  "version": "4.5.6",
  "flags": {},
  "shapes": [
    {
      "label": "truck",
      "points": [
        [
          283.0652173913043,
          237.58695652173913
        ],
        [
          317.30434782608694,
          237.58695652173913
        ],
        [
          316.7608695652174,
          440.30434782608694
        ],
        [
          309.695652173913,
          441.39130434782606
        ],
        [
          309.695652173913,
          455.52173913043475
        ],
        [
          318.9347826086956,
          460.4130434782609
        ],
        [
          320.02173913043475,
          497.36956521739125
        ],
        [
          305.3478260869565,
          504.9782608695652
```


Detectron2

Step 1

- Setup Detectron2
- Get labeled data

Step 2

- Register dataset
- Coco format

Step 3

- Train model
- Save model and config

Step 4

- Test model on validation set
- Evaluate model

Step 5

- Apply model on larger dataset
- Save result to .csv file

Train model

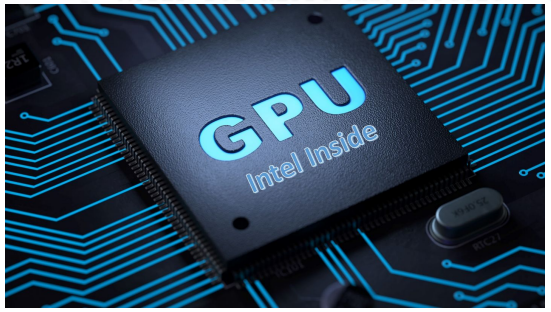
Now, let's fine-tune a pretrained FasterRCNN instance segmentation model on the truck data-set.

```
[ ]  from detectron2.engine import DefaultTrainer
     from detectron2.config import get_cfg
     import os

     cfg = get_cfg()
     cfg.merge_from_file(model_zoo.get_config_file("COCO-InstanceSegmentation/mask_rcnn_R_50_FPN_3x.yaml"))
     cfg.DATASETS.TRAIN = ("truck_train",)
     cfg.DATASETS.TEST = ()
     cfg.DATALOADER.NUM_WORKERS = 2
     cfg.MODEL.WEIGHTS = model_zoo.get_checkpoint_url("COCO-InstanceSegmentation/mask_rcnn_R_50_FPN_3x.yaml")
     cfg.SOLVER.IMS_PER_BATCH = 2
     cfg.SOLVER.BASE_LR = 0.00025
     cfg.SOLVER.MAX_ITER = 2000
     cfg.MODEL.ROI_HEADS.NUM_CLASSES = 1

     os.makedirs(cfg.OUTPUT_DIR, exist_ok=True)
     trainer = DefaultTrainer(cfg)
     trainer.resume_or_load(resume=False)
     trainer.train()
```

45 minutes



Detectron2

Step 1

- Setup Detectron2
- Get labeled data

Step 2

- Register dataset
- Coco format

Step 3

- Train model
- Save model and config

Step 4

- Test model on validation set
- Evaluate model

Step 5

- Apply model on larger dataset
- Save result to .csv file

Load model and config

Use “truck_test” dataset

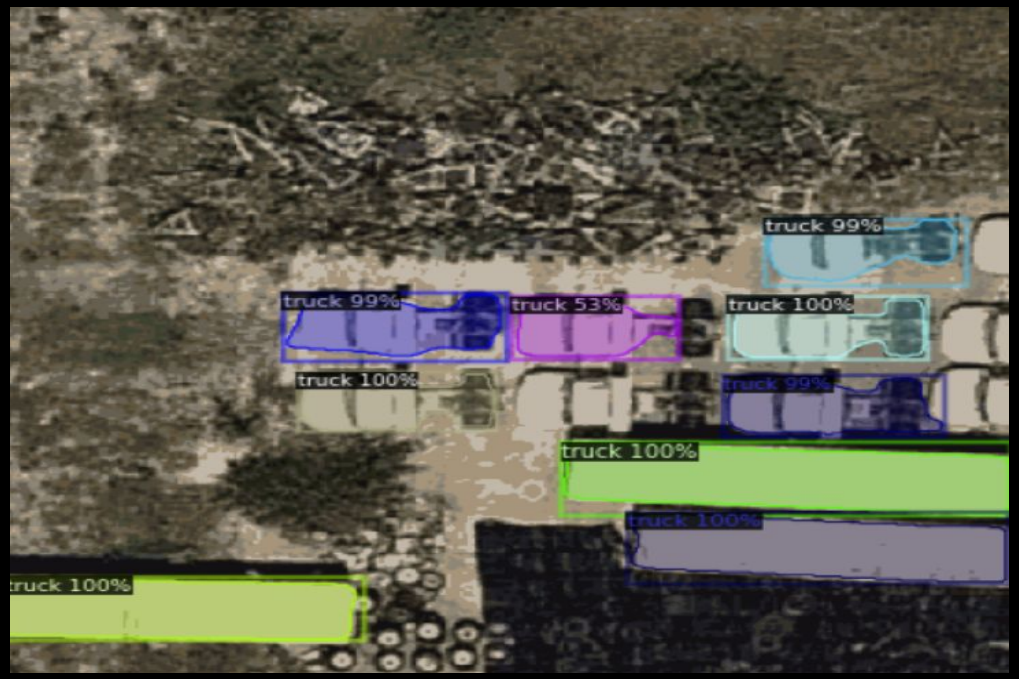
Predict trucks on the dataset

Average Precision

AP
69.512

Detectron2

Visualize on Validation Set



Step 1

- Setup Detectron2
- Get labeled data

Step 2

- Register dataset
- Coco format

Step 3

- Train model
- Save model and config

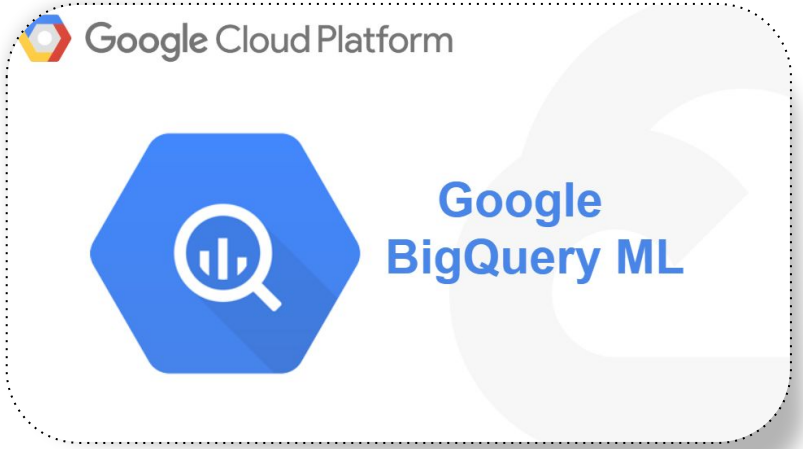
Step 4

- Test model on validation set
- Evaluate model

Step 5

- Apply model on larger dataset
- Save result to .csv file

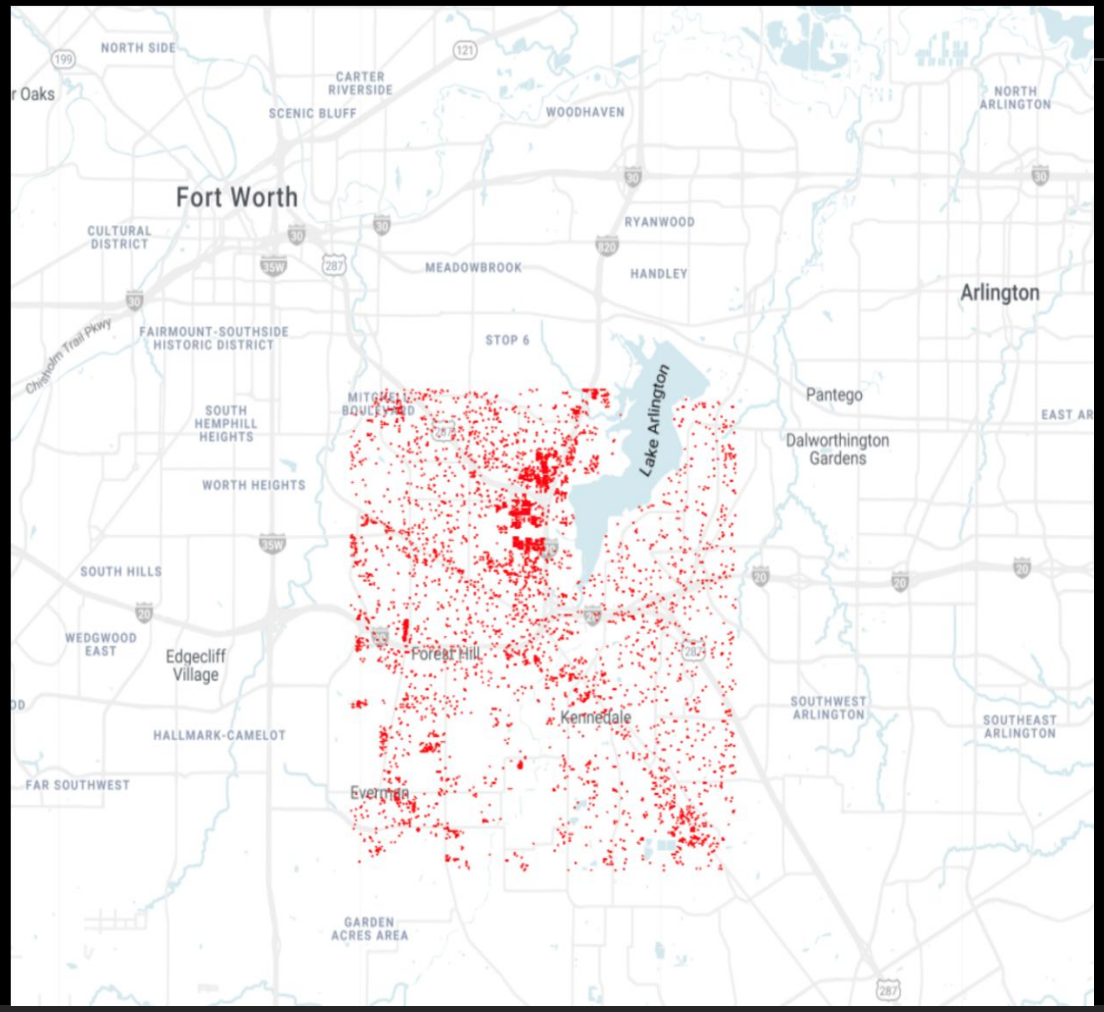
115,000 images



Detectron2

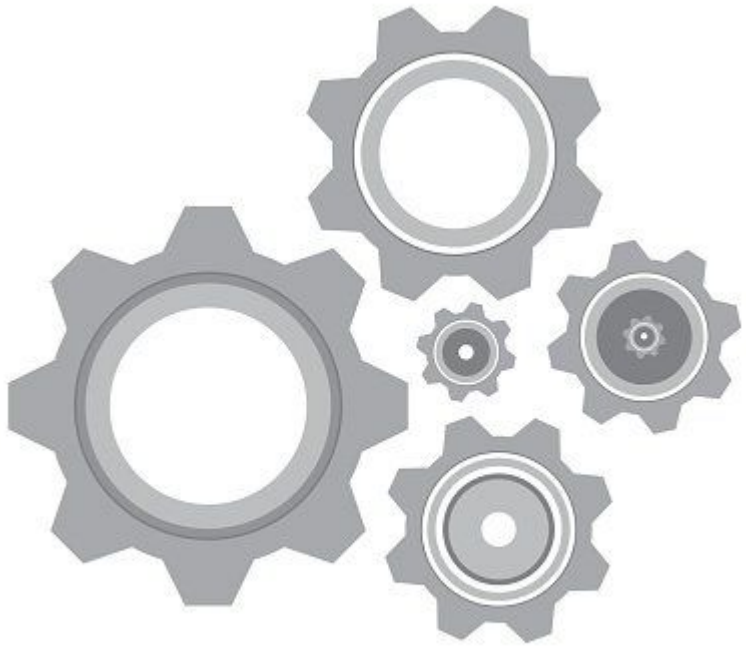
The Result

Detect at least 1 truck	4,316 locations
Detect at least 5 trucks	264 locations
Detect at least 10 trucks	42 locations



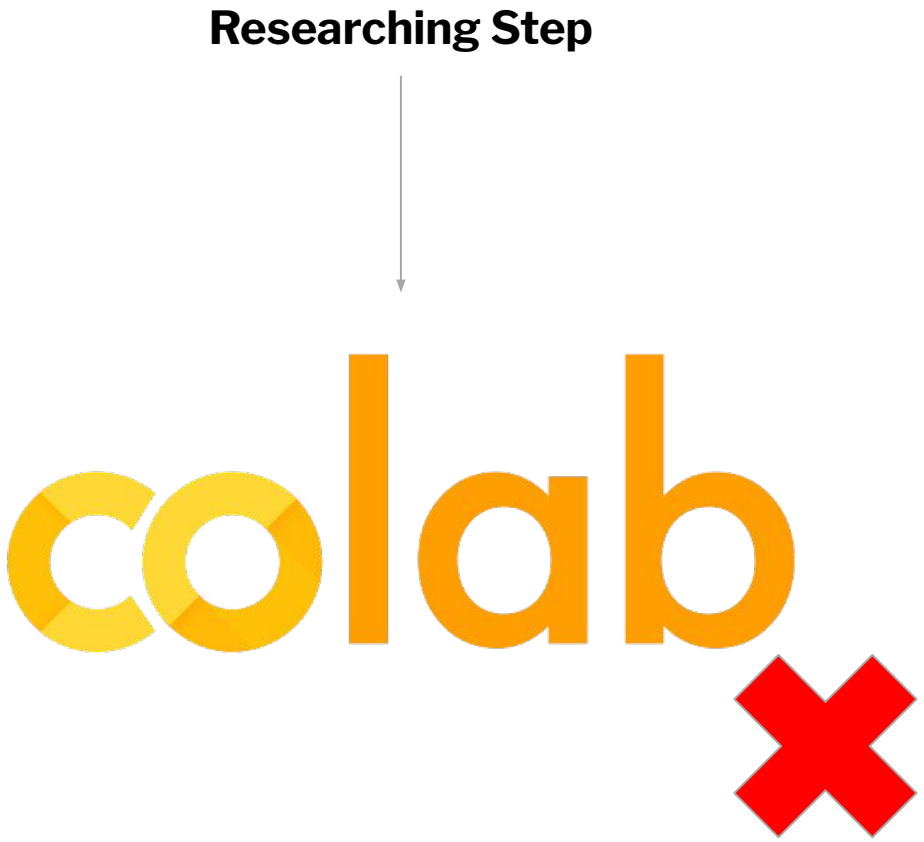
After Researching

1. Refactoring Step
2. Solving Raising Problems



Refactoring Step

1. Why we need refactoring step?

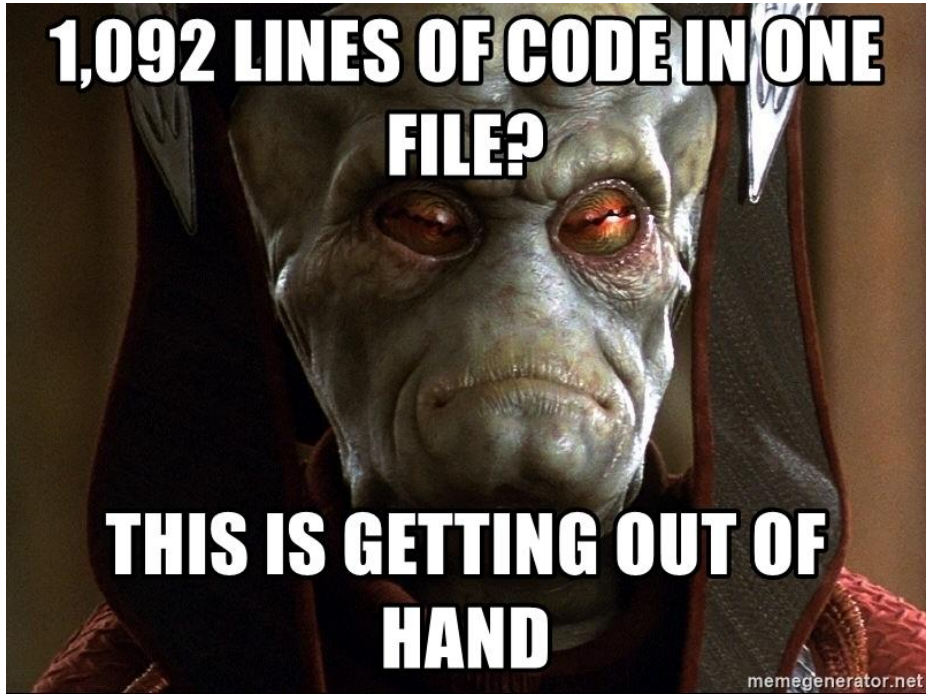


Problem with Google Colab

- Hard to modify parameter

```
cfg = get_cfg()
cfg.merge_from_file(model_zoo.get_config_file("COCO-InstanceSegmentation/mask_rcnn_R_50_FPN_3
cfg.DATASETS.TRAIN = ("truck_train",)
cfg.DATASETS.TEST = ()
cfg.DATALOADER.NUM_WORKERS = 2
cfg.MODEL.WEIGHTS = model_zoo.get_checkpoint_url("COCO-InstanceSegmentation/mask_rcnn_R_50_FP
cfg.SOLVER.IMS_PER_BATCH = 2
cfg.SOLVER.BASE_LR = 0.00025
cfg.SOLVER.MAX_ITER = 2000
cfg.MODEL.ROI_HEADS.NUM_CLASSES = 1
```

- Hard to maintain



Refactoring Step

- The solution for Google Colab
- Convert notebook file to a complete Python program.

Testing

- **Unit Testing:**
  - Pass test cases for each components
  - Each component behave as expected
- **Regression Testing:**
  - Pass replicate bugs after fixing code and integrating
- **Integration Testing:**
  - Successful integration among all components
- **Model Testing:**
  - The result accuracy and error are in acceptable margin of error

Storage Problem

- 100 GB Storage
- 

Approach To Storage Problem

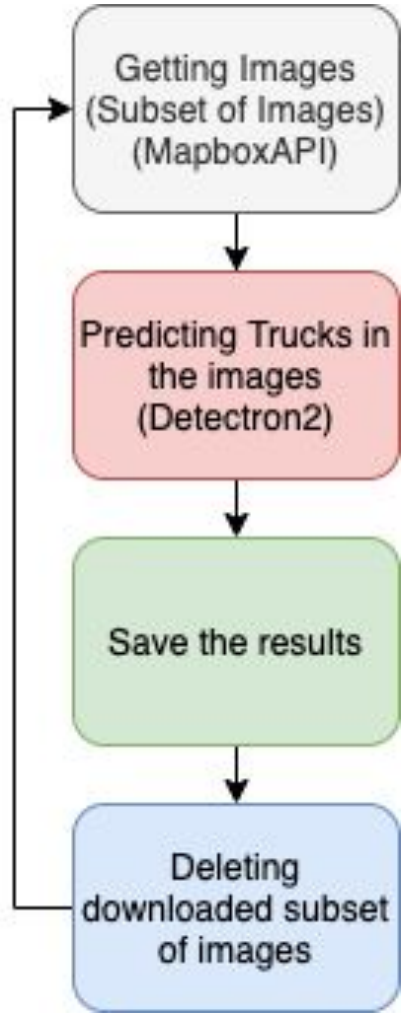
- Necessary information in result file.

```
sub_df['numb_truck'] = numb_truck
sub_df['lat'] = lats
sub_df['lon'] = lons
```


It is not necessary to store all images

“Divided Work” Solution

Workflow:



The Final Product

- Accurate Data model
- Scalable code
- Optimized for storage

The Retrospective

- Teamwork
- Working with a client
- Planning
- Communication
- New technology

Questions?