MAIS 202 – Preprocessing and First Steps

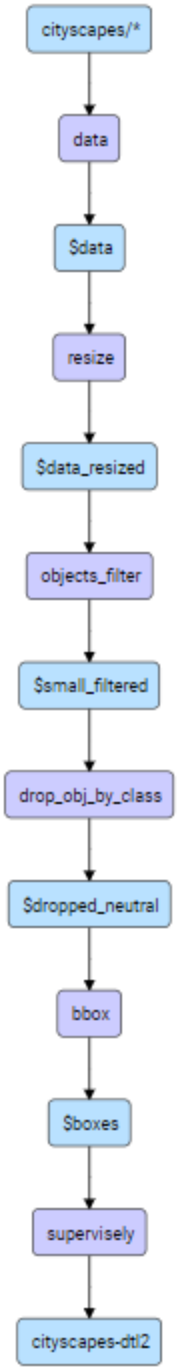
1. Problem statement: The idea of this project is to perform object detection on the Cityscapes dataset. A user would ideally input a photo of outdoor urban streets/scenery and the application would be able to detect certain elements inside the photo.

2. Datasets: The data I am working with is the Cityscapes dataset, specifically two files: *gtFine* and *leftImg8bit*. The former contains annotations in JSON format while the latter contains the images. Both contain train, val and test sets then divided into about 10 directories representing different cities. Additionally, each photo is 2048x1024 and each annotation is very finely annotated with 30 classes that are all polygons.

3. Data Pre-processing: I imported the datasets into Supervisely website, which is an app that allows you to pre-process annotations. Essentially, I wrote configurations in JSON format using their data transformation language which allowed me to do various operations. Below are the pre-processing operations. Please see the following directory for DTL: *workspace\training_demo*

   Preprocessing_datasets.JSON

```
C:.
|   Data Selection Proposal.pdf
|   Data Preprocessing Update.pdf
|   README.md
|   tree.txt
|
\---workspace
    \---training_demo
        |   JSON_to_CSV.py
        |   Preprocessing_datasets.JSON
        |   README.md
        |
        +---annotations
        |   +---test
        |   |       frankfurt[].json
        |   |       test_labels.csv
        |   |
        |   \---train
        |           aachen[].json
        |           bochum[].json
        |           bremen[].json
        |           train_labels.csv
        |
        +---exported-models
        +---images
        |   +---test
        |   |       frankfurt[].png
        |   |       |
        |   \---train
        |           aachen[].png
        |           bochum[].png
        |           bremen[].png
        |
        \---pre-trained-models
```



Class mapping: this involved clustering into 5 groups: vehicle, neutral, person, road, building

Resize: All images were resized to 224x224

Object filtering: All objects of classes road, person, building and vehicle that were less than 0.2% of the image area were filtered out as neutral.

Dropped class: neutral class was dropped, leaving person, road, building vehicle

Box bounding: Since all objects were polygons represented by more than 4 coordinates, they were transformed into rectangles with 4 coordinates (xmin, xmax, ymin, ymax)

New dataset saved.

4. Preprocessing result: 853 224x224 images with rectangular objects from 4 classes: person, building, road, vehicle.

5. Machine learning model: I have decided to use TensorFlow Object Detection API, which is an open source framework with models pre-trained on COCO, KITTI, and Open Images Dataset. I will use this to then train my model further.

6. Steps done:

- Create virtual anaconda environment with TensorFlow, TensorFlow Object detection API, Protobuf with many other packages.

- The annotations were all in JSON format and to the tensorflow models generally take as input XML then convert to CSV then to tfrecord. With the help of stackoverflow, I was able to customize an algorithm that created one CSV with the exact annotation information and format that I needed. Please see *workspace\training_demo\ JSON_to_CSV.py*

7. Decisions to make:

- What type of CNN model I would like to use, keeping in mind I need to balance speed and performance. Also, I currently am only using an integrated GPU, and tried using the "Faster R-CNN" model which was taking way too long to train solely running on CPU.

Credits:

Tensorflow Object Detection API
https://github.com/tensorflow/models/

Tutorials:
https://github.com/EdjeElectronics/TensorFlow-Object-Detection-API-Tutorial-Train-Multiple-Objects-Windows-10
https://tensorflow-object-detection-api-tutorial.readthedocs.io/en/latest/index.html

Protobuf:
https://github.com/protocolbuffers/protobuf

StackOverflow
https://stackoverflow.com/questions/51023367/how-to-make-tfrecords-from-json-annotated-images