Automatic Extraction of Road Networks from Aerial Images

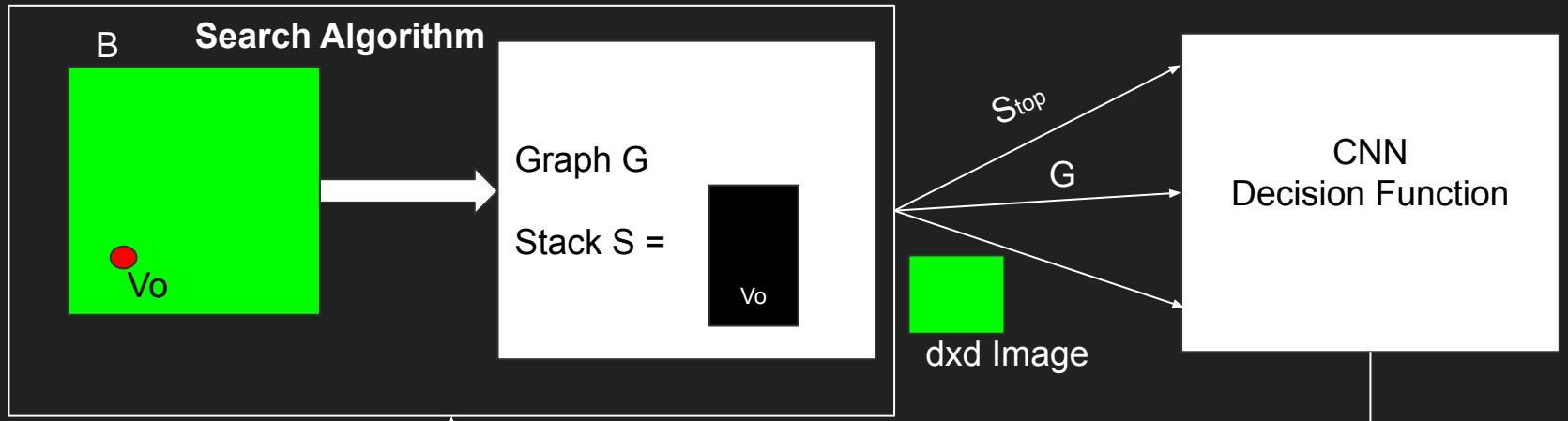
Critical Evaluation and Implementation of Proposed Approach

Presentation by - Rahul Bhojwani, Shriya Rai

Outline

1. Introduction
2. Proposed Approach
3. Data Acquisition and Processing
4. Tech Stack
5. Evaluation Metric
6. Model Pipeline
7. Result on baseline cities
8. Result on new cities
9. Observations, reasoning and proposed solutions
10. Takeaways
11. Extended - Results on other baseline cities

Proposed Approach



End-to-End Iterative Graph Construction algorithm

Walk distance D at an angle a

OR

Stop and pop S_{top} from S.

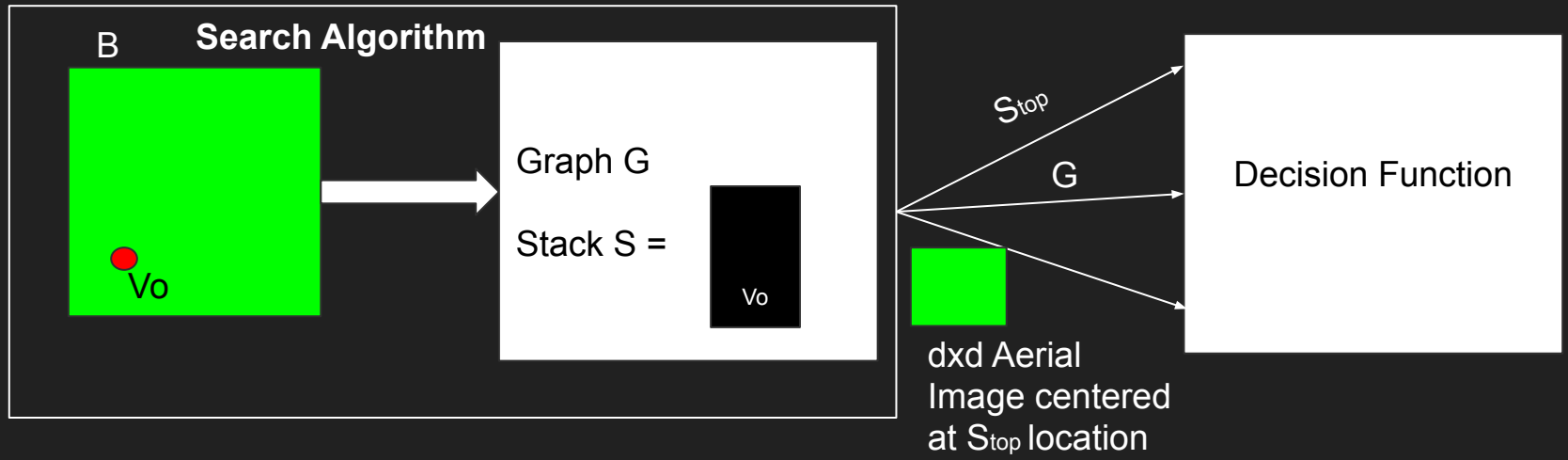
Introduction

- **Problem Statement**
  - Generating automatic road network model by utilizing aerial imagery and deep learning techniques

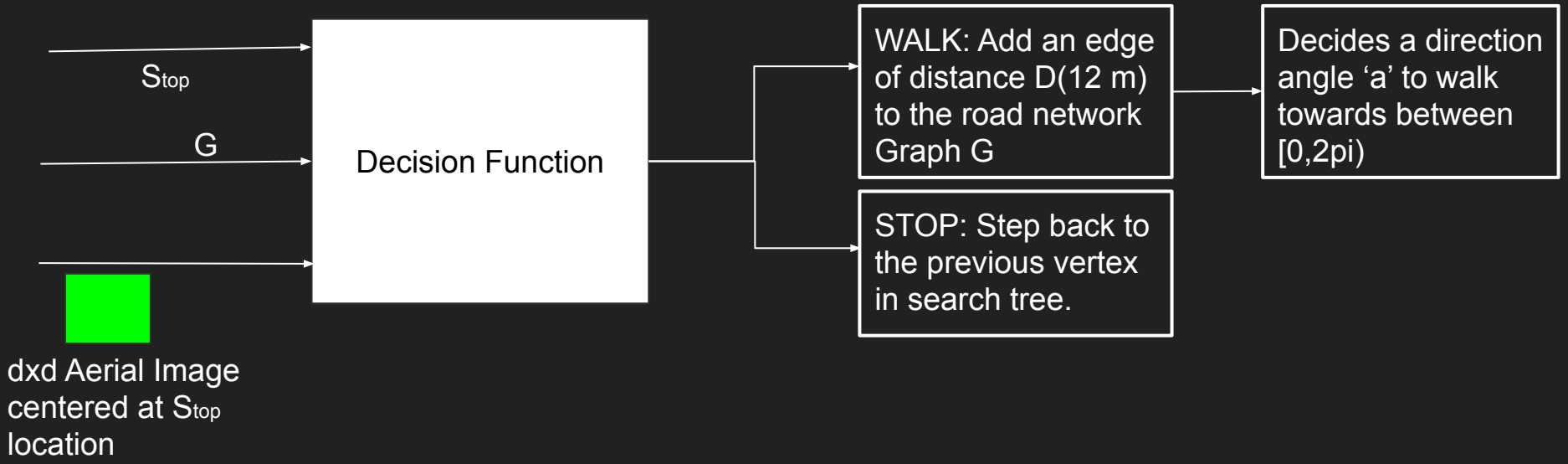
- **Dataset**
  - **Google Maps satellite images**
    - We utilized Google Static Maps API to download aerial imagery around 40 city centers.
    - The downloaded Aerial Imagery has a resolution of 60cm.
    - For each city, approx 24 sq km area is used.
  - **OpenStreetMap (OSM)**
    - We downloaded open source OSM data to extract ground truth for road networks.
    - Conversion of coordinate system of the road network so that the vertex spatial coordinate annotations correspond to pixels in the satellite images.

Proposed Approach

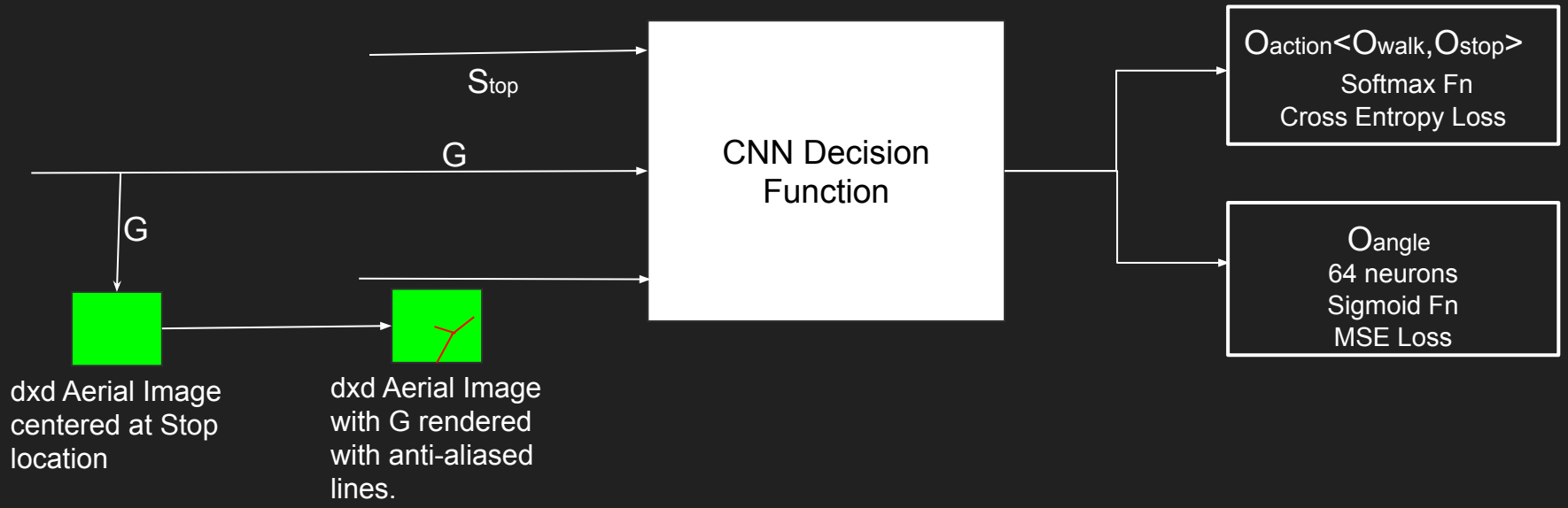
End-to-End Iterative Graph Construction algorithm



Proposed Approach



Proposed Approach

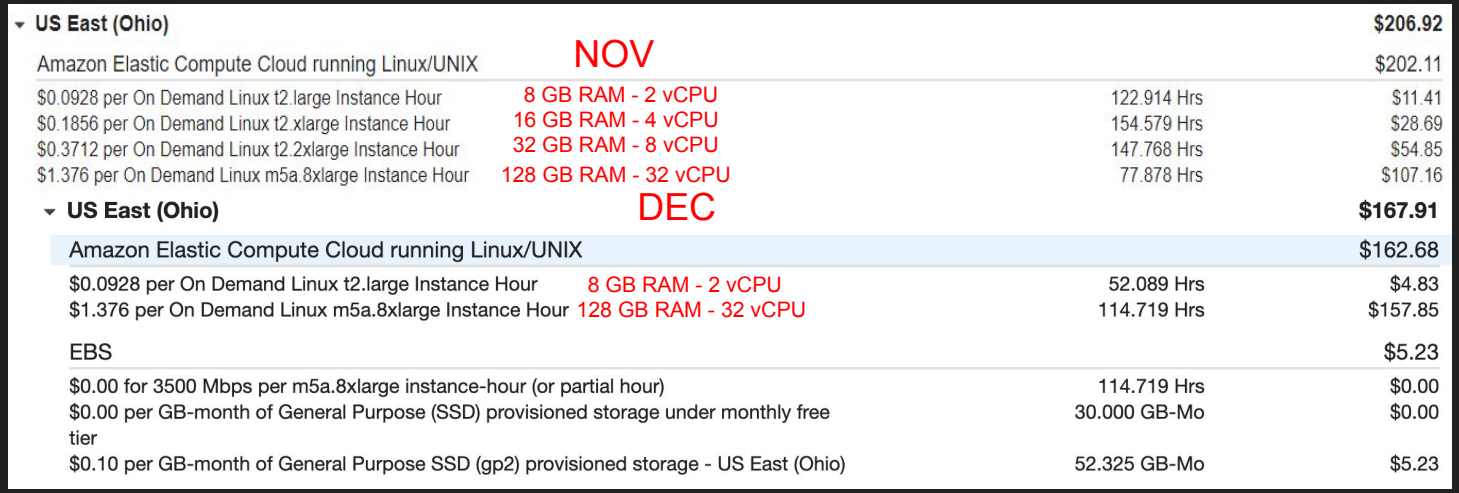


Data Acquisition and Processing

- **Step 1 : Acquiring satellite imagery data through google map API**
- **Step 2 : Acquiring and processing OSM data**
- **Step 3 : Generate starting locations**
- **Step 4 : Generate road masks**
- **Step 5 : Create test satellite imagery files**

Tech Stack - AWS Architecture

- AWS EC2 Instance:



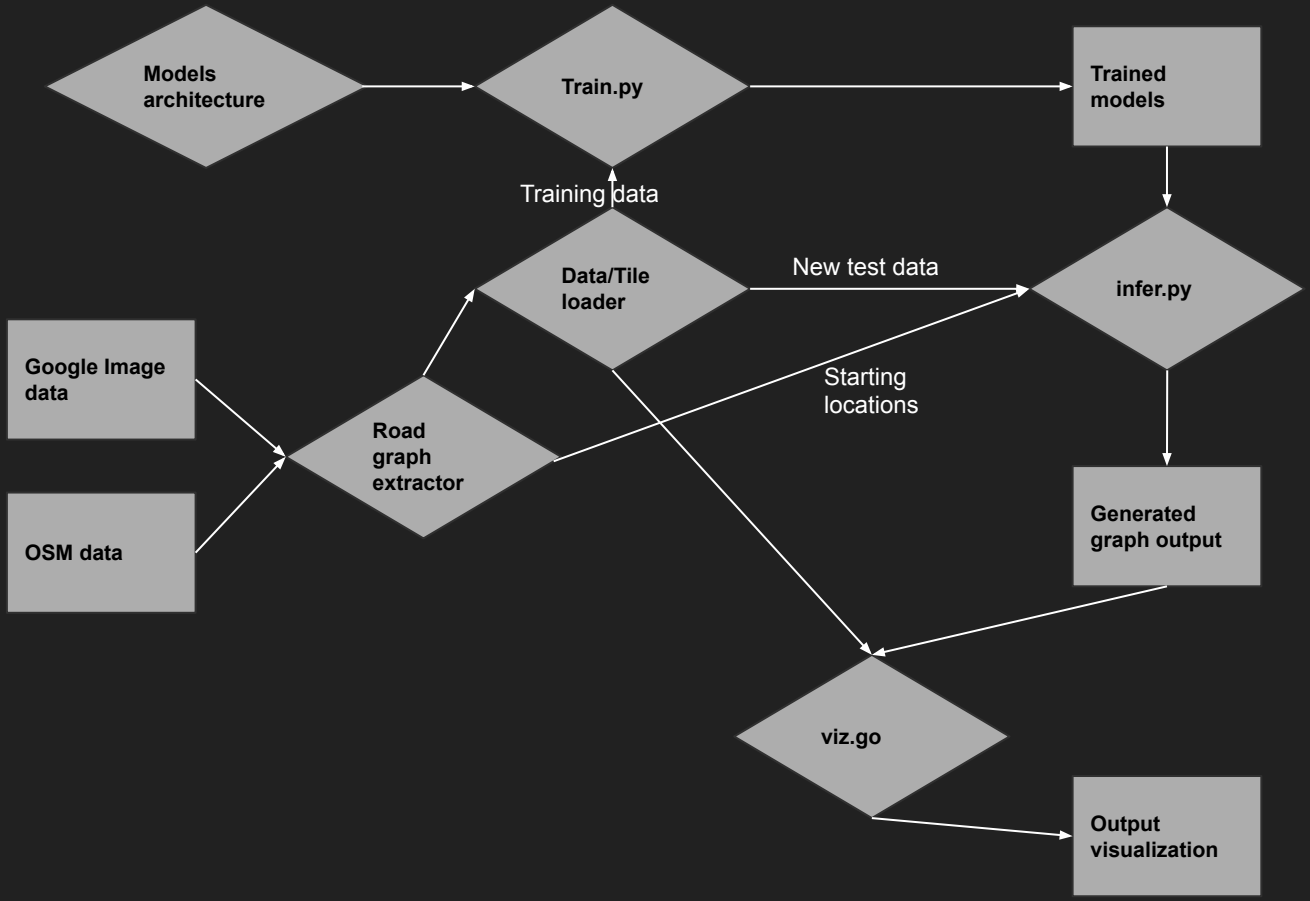
US East (Ohio)			**$206.92**
Amazon Elastic Compute Cloud running Linux/UNIX	NOV		$202.11
$0.0928 per On Demand Linux t2.large Instance Hour	8 GB RAM - 2 vCPU	122.914 Hrs	$11.41
$0.1856 per On Demand Linux t2.xlarge Instance Hour	16 GB RAM - 4 vCPU	154.579 Hrs	$28.69
$0.3712 per On Demand Linux t2.2xlarge Instance Hour	32 GB RAM - 8 vCPU	147.768 Hrs	$54.85
$1.376 per On Demand Linux m5a.8xlarge Instance Hour	128 GB RAM - 32 vCPU	77.878 Hrs	$107.16
US East (Ohio)	DEC		**$167.91**
Amazon Elastic Compute Cloud running Linux/UNIX			$162.68
$0.0928 per On Demand Linux t2.large Instance Hour	8 GB RAM - 2 vCPU	52.089 Hrs	$4.83
$1.376 per On Demand Linux m5a.8xlarge Instance Hour	128 GB RAM - 32 vCPU	114.719 Hrs	$157.85
EBS			$5.23
$0.00 for 3500 Mbps per m5a.8xlarge instance-hour (or partial hour)		114.719 Hrs	$0.00
$0.00 per GB-month of General Purpose (SSD) provisioned storage under monthly free tier		30.000 GB-Mo	$0.00
$0.10 per GB-month of General Purpose SSD (gp2) provisioned storage - US East (Ohio)		52.325 GB-Mo	$5.23

- gpt2 volume(Disk Space) - 125 GB
- Python 2 - anaconda2
- TensorFlow v1.14

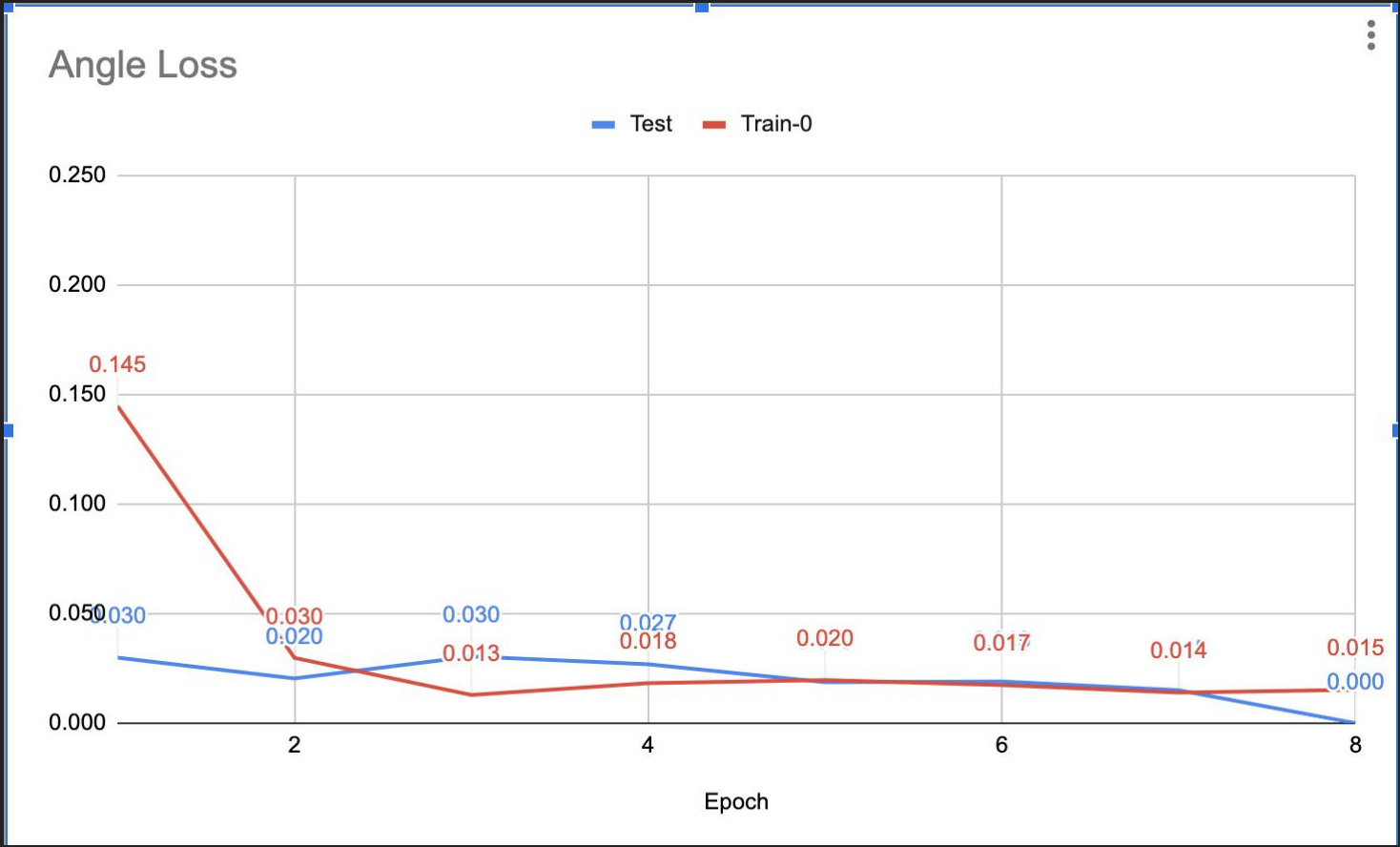
Evaluation Metric

- Measure utilised to judge performance of the model is **Junction Metric**
  - Compares the ground truth and inferred maps junction-by-junction, where a junction is any vertex with three or more edge
  - Gives a score that is representative of the inferred map’s practical usability
  - Is interpretable
  - Yields a precision-recall curve

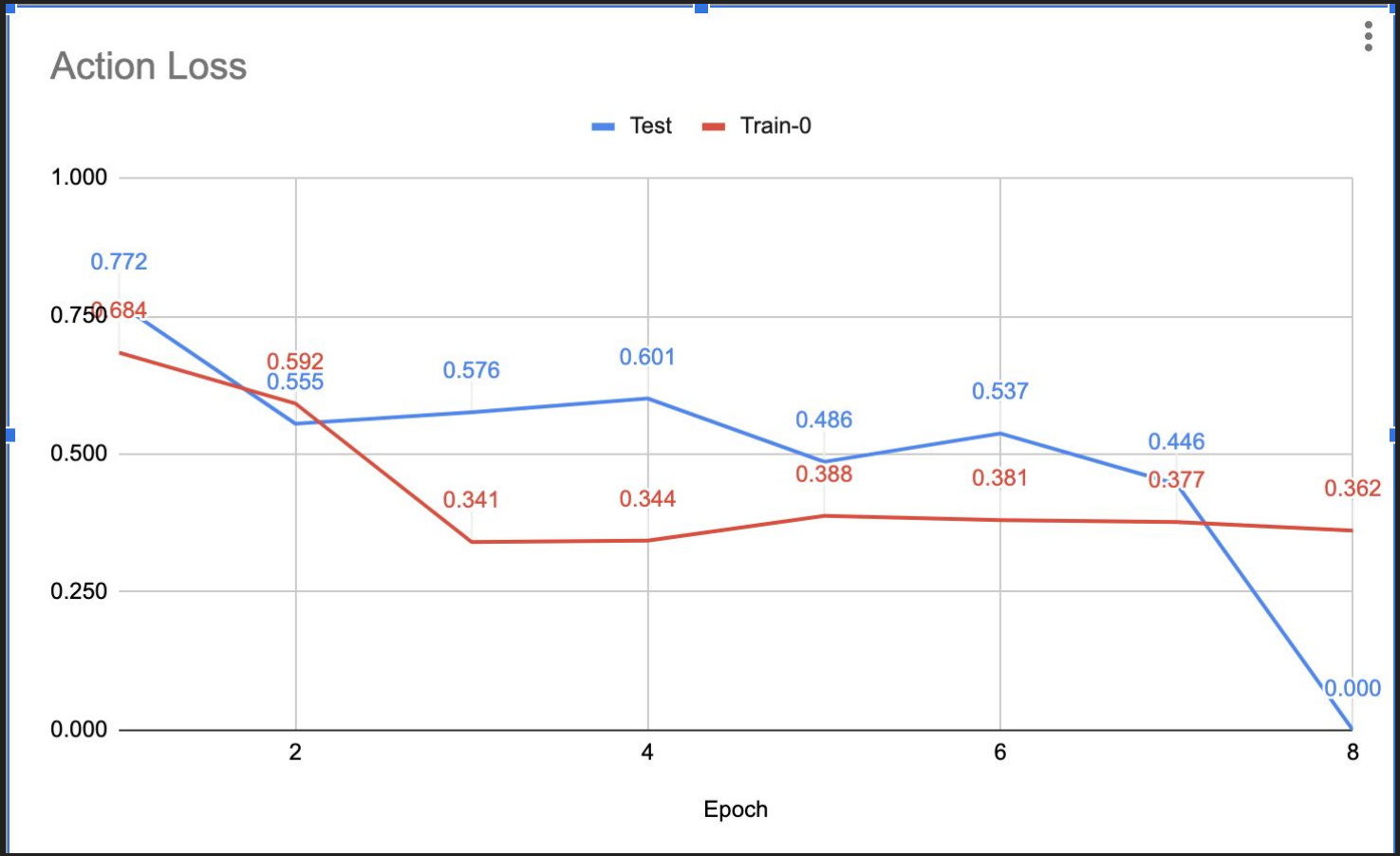
Model Pipeline



Training example visualisation



Training example visualisation



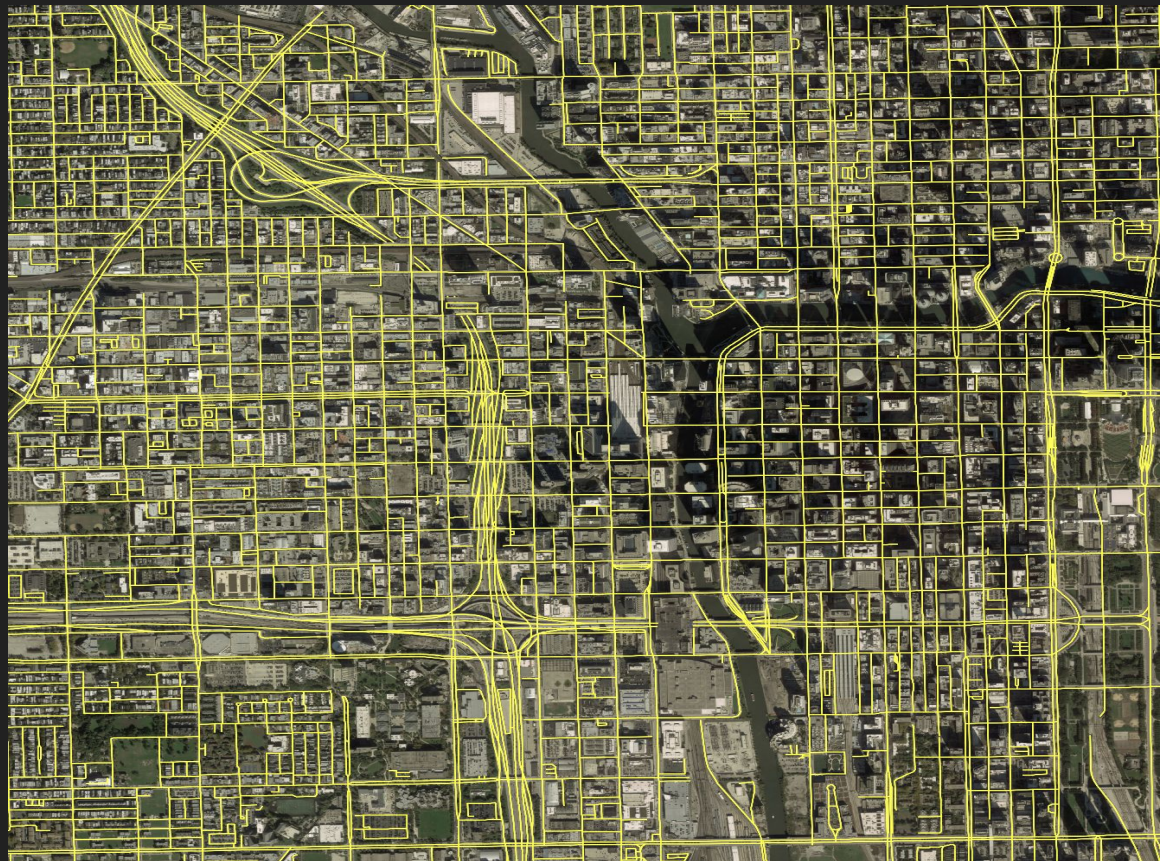
Results

Our implementation on Baseline Cities for Comparative Evaluation

1. Amsterdam
2. Chicago
3. Denver
4. Los Angeles
5. Montreal
6. Paris
7. Pittsburgh
8. Salt Lake City
9. Tokyo
10. Toronto

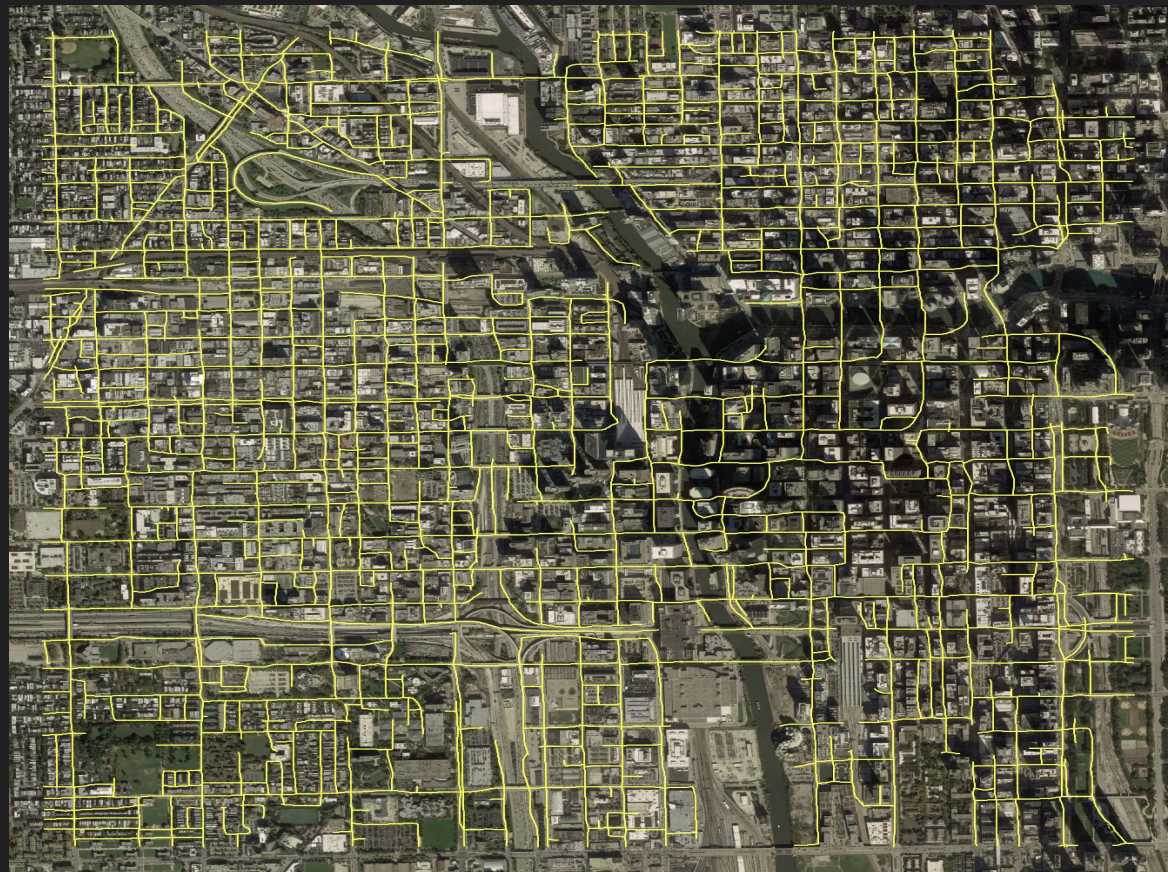
Result for Chicago

a) True Road Network for Chicago



Result for Chicago

b) Predicted Road Network for Chicago



Results

Testing the model on new cities

To get better understanding of the usability of the model, we explored it on critical cities of interest.

1. Nairobi - Developing city
2. Port au Prince - Disaster Prone
3. Delhi - Developing city
4. Chandigarh - Well planned

Results for Nairobi

a) True Road Network for Nairobi

Results for Nairobi

b) Predicted Road Network for Nairobi

Results for Port Au Prince

a) True Road Network for Port Au Prince

Results for Port Au Prince

b) Predicted Road Network for Port Au Prince

Results for New Delhi

a) True Road Network for New Delhi

Results for New Delhi

b) Predicted Road Network for New Delhi

Results for Chandigarh

a) True Road Network for Chandigarh

Results for Chandigarh

b) Predicted Road Network for Chandigarh

Observations and proposed solutions

- Observation-1
  - Method does not perform well in detecting highways

a) Denver true cropped image

b) Denver predicted crop image

Observation-1

a) Chicago true cropped image

b) Chicago predicted crop image

Reasoning and proposed solution

- The highways are either:
  - Case 1: A road dividing into multiple roads(lanes) and running parallel to each other.
  - Case 2: A highway whose starting and ending points are beyond the scope of the image.
- Solution for case 1:
  - Due to the underlying concept of the graph based expansion, the network proceeds for a long time when an highway is met, leading up to the problem of not being able to find parallel edges.
  - This can be solved by storing the high confidence score matches separately and tracing them again.
- Solution to case 2:
  - This one is the most prevailing case and it can be solved by finding a starting point on a highway.
  - To achieve that, we can use the segmentation output to detect wider roads[highways] and use a starting location from there.

Observation-2

- Method does not perform well in case of disconnected parts in images

a) True Road Network for Pittsburgh

a) True Road Network for Pittsburgh

b) Predicted Road Network for Pittsburgh

Observation-2

- Method does not perform well in case of disconnected parts in images

b) Predicted Road Network for Pittsburgh

Reasoning and proposed solution

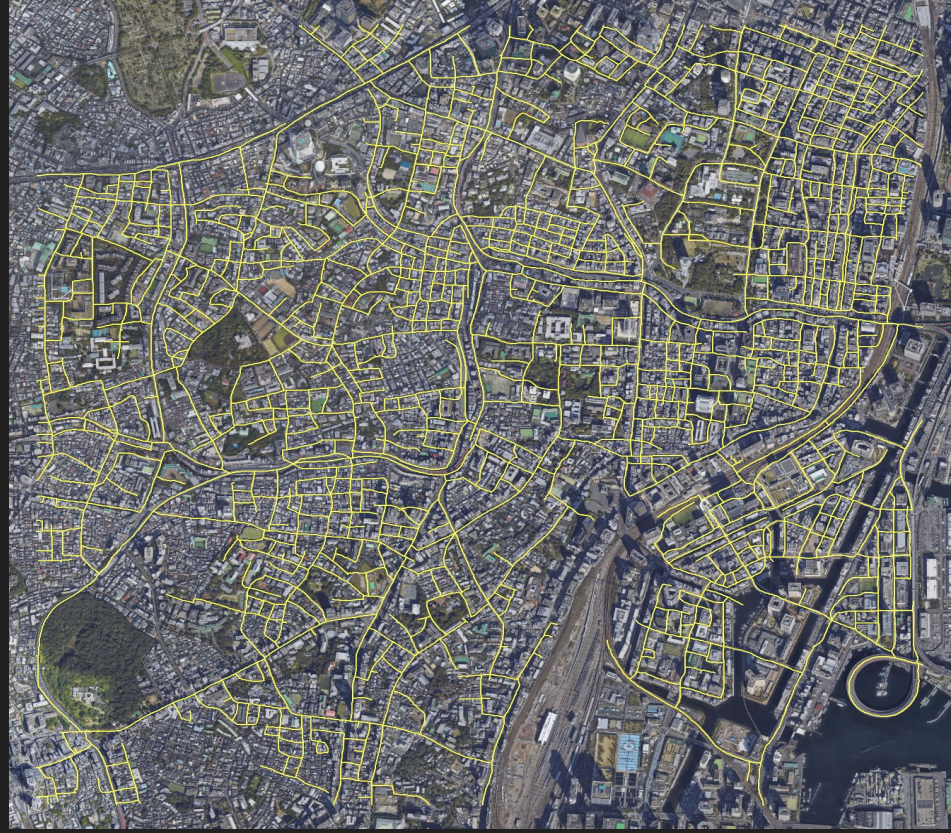
- This is because the graph based model cannot explore the disconnected components.
- Solution:
  - Multiple starting locations can be used to help the model find roads in all disjoint components.
  - One optimal way of doing that is by finding the starting location in the area of the image unexplored after each complete pass.

Observation-3

- Method shows poor results at image borders

a) True Road Network for Tokyo

a) True Road Network for Tokyo



b) Predicted Road Network for Tokyo

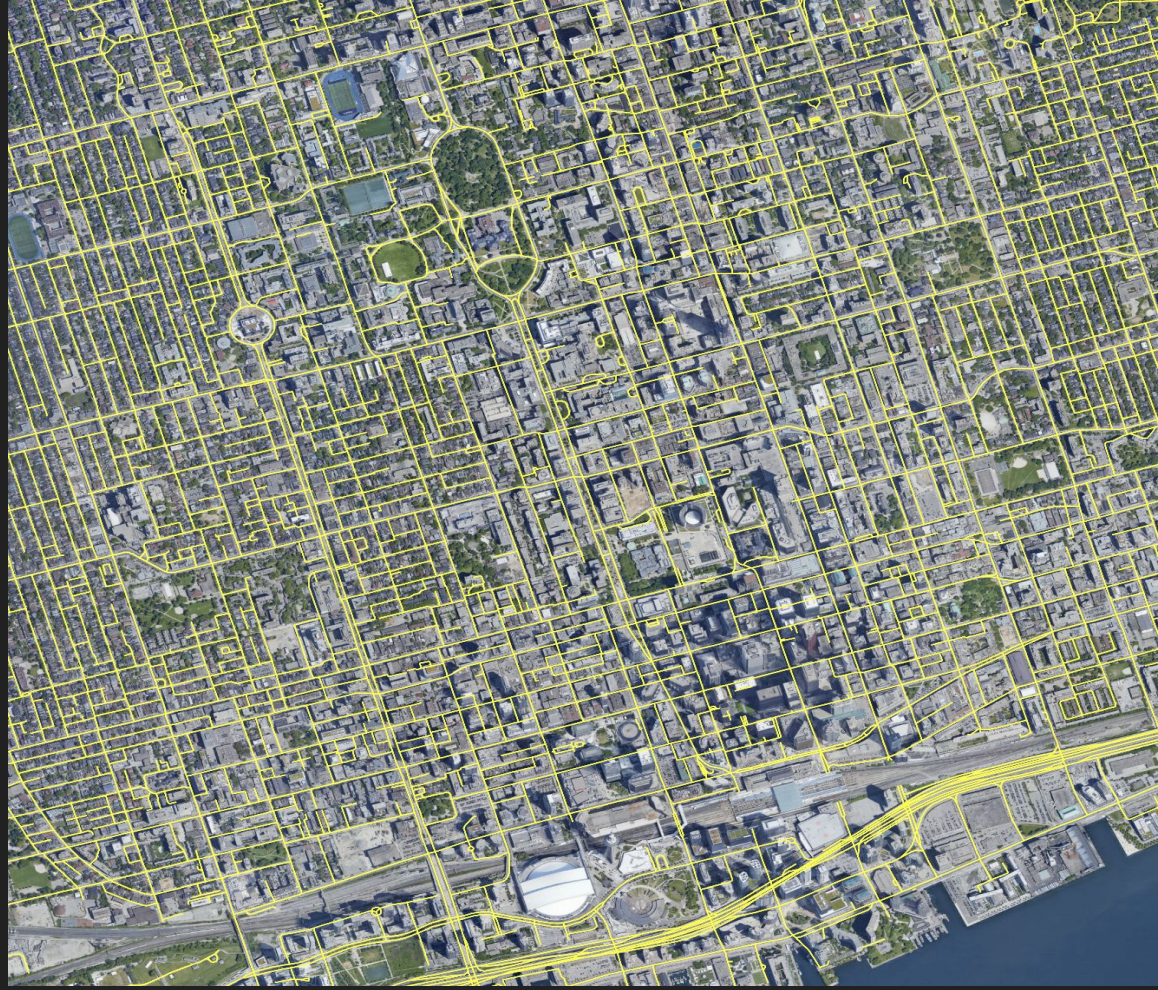
Observation-3

- Method does not perform predictions at image borders

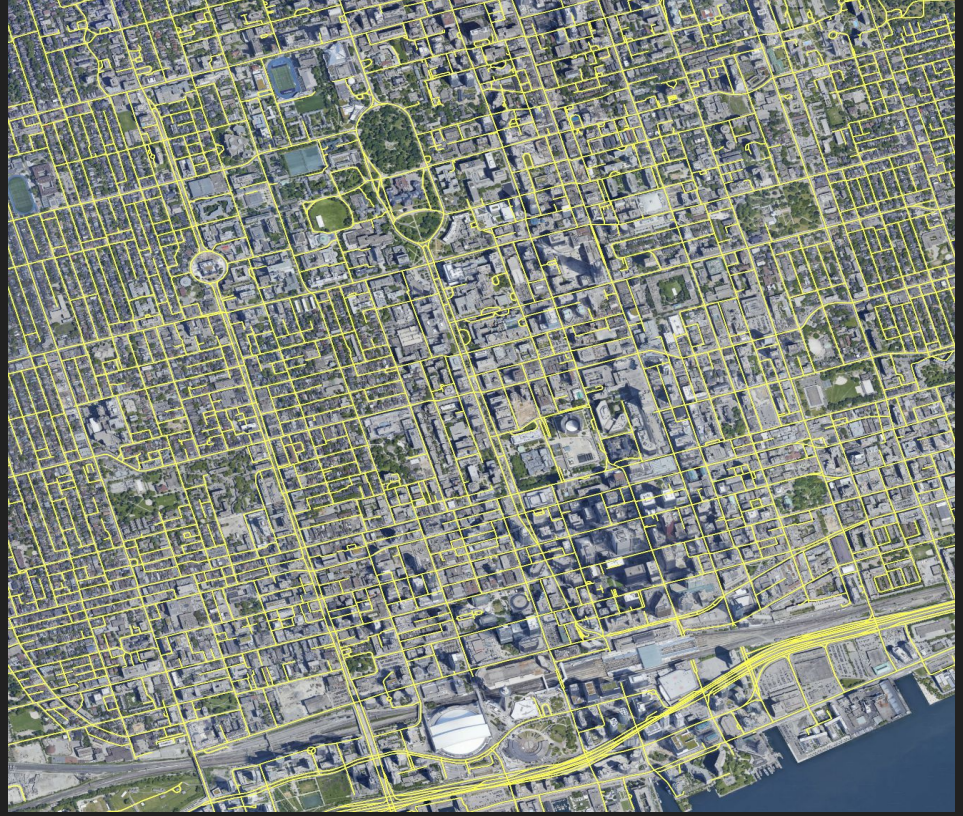
b) Predicted Road Network for Tokyo

Observation-3

- Method does not perform predictions at image borders



a) True Road Network for Toronto



a) True Road Network for Toronto

b) Predicted Road Network for Toronto

Observation-3

- Method does not perform predictions at image borders

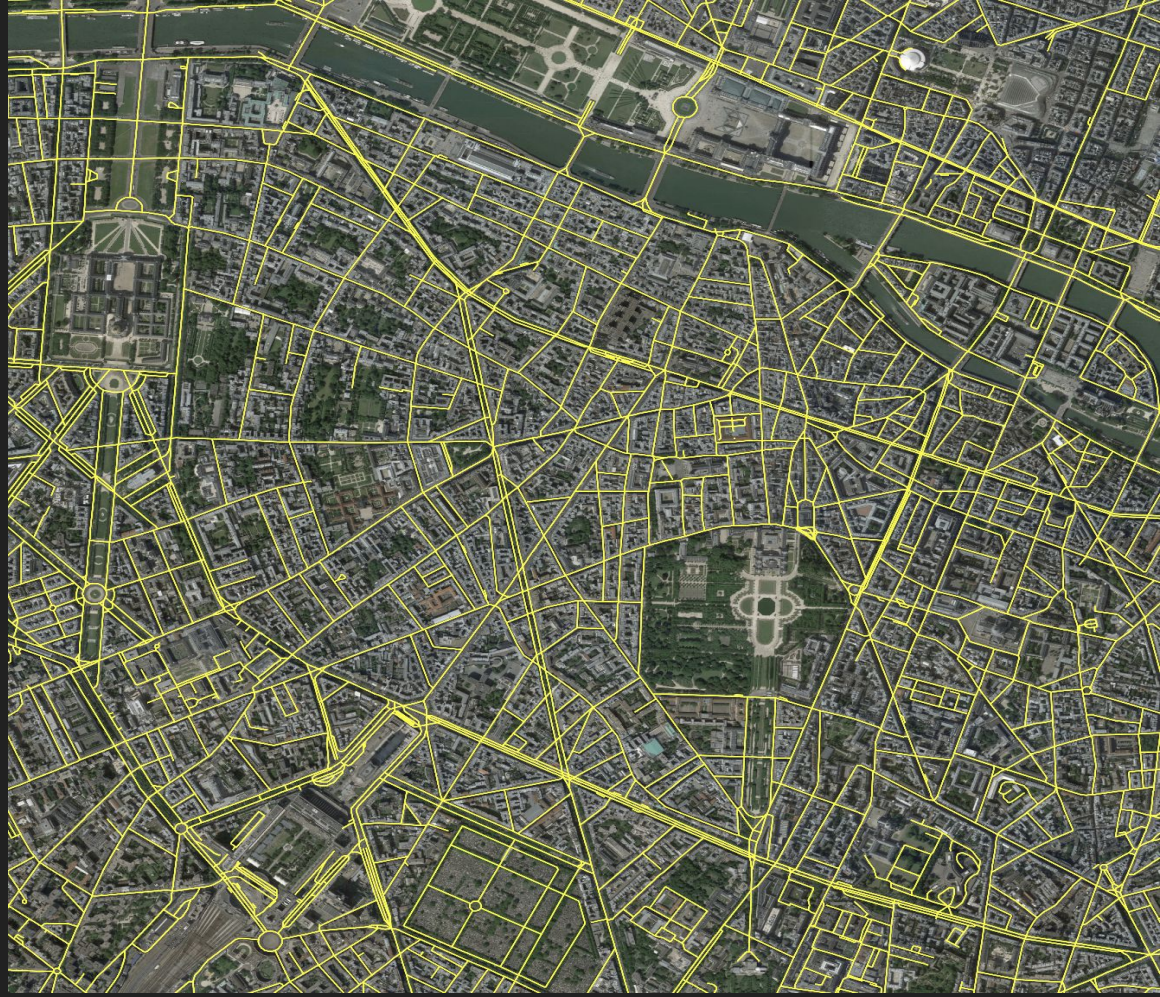
b) Predicted Road Network for Toronto

Reasoning and proposed solutions

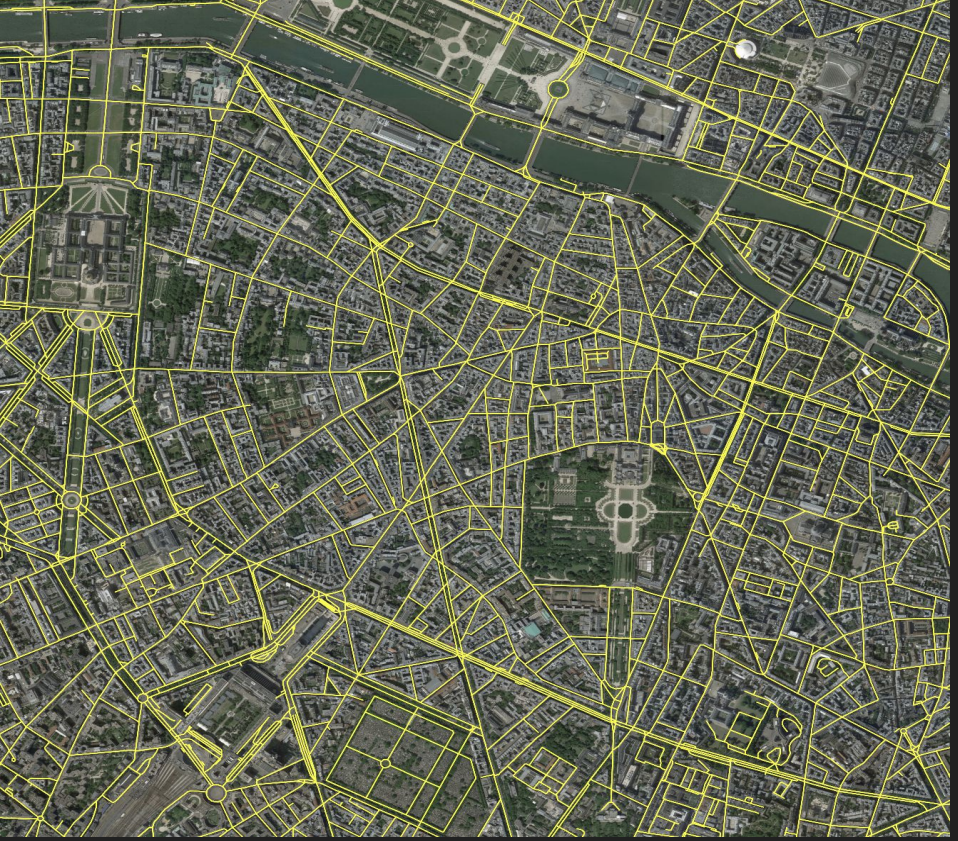
- The current CNN architecture is unable to detect the road-network at image borders because it’s using 0 padding at image borders.
- This makes the pattern different(not continuous) for the border kernels to learn from it.
- Solution:
  - One of the novel methods to solve this is by filling in the values with mirroring the image at the borders.
  - And then filter the results beyond the border part.

Observation-4

- Method does not perform predictions at circular loops



a) True Road Network for Paris



a) True Road Network for Paris

b) Predicted Road Network for Paris

Observation-4

- Method does not perform predictions at circular loops

b) Predicted Road Network for Paris

Reasoning and proposed solutions

- The algorithm is very susceptible to the distance hyperparameter, D.
- Taking an edge of distance D throws it out of the circle and that’s the reason we notice it not working better on the tighter circular roads.
- Solution:
  - This can be solved by tweaking the hyperparameter, D.
  - The problem is that smaller the D, the slower the model will explore/converge.

Takeaways

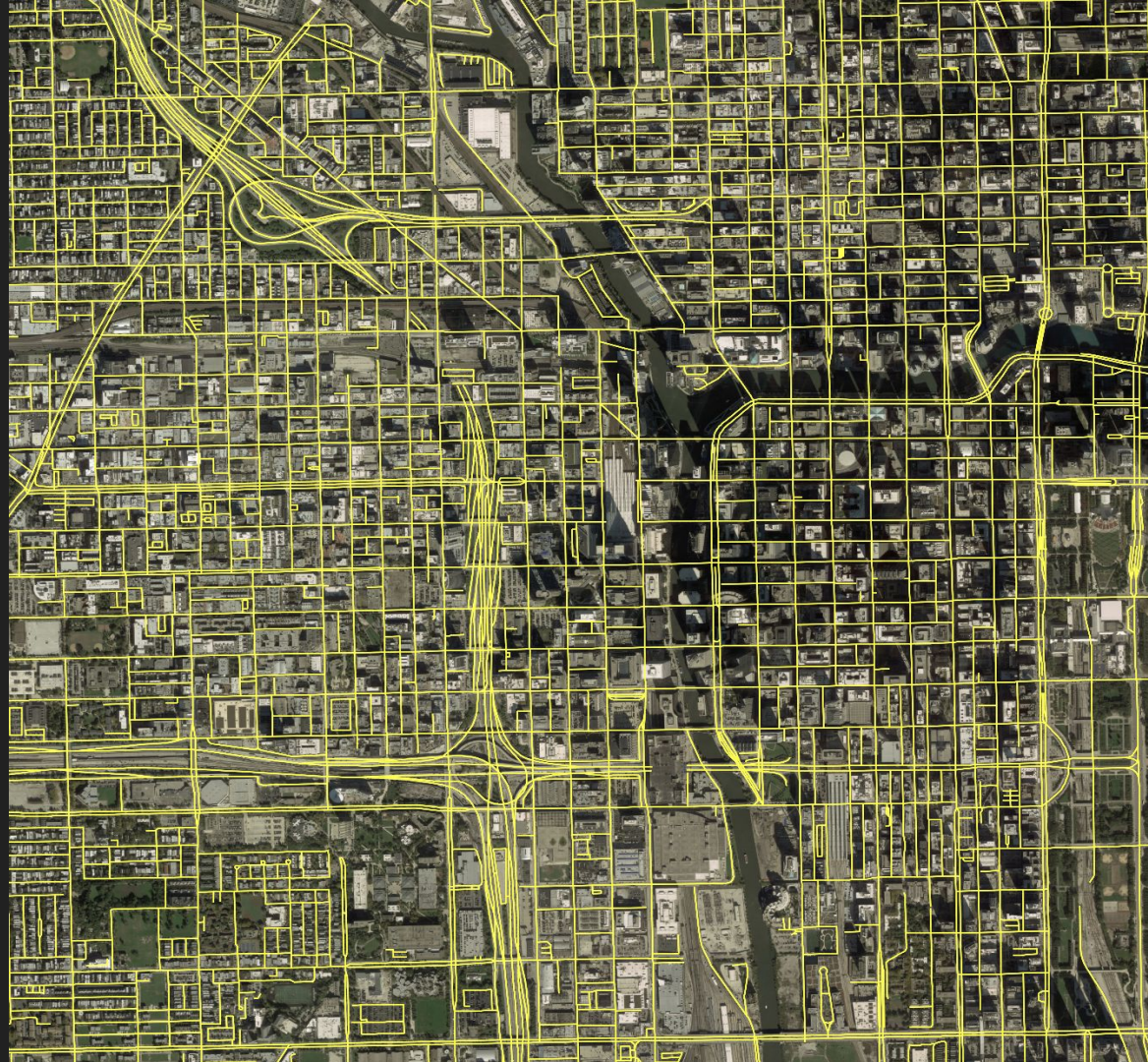
- Computational resources have been a major bottleneck for utilising this massive dataset and we gained a deeper understanding of managing the AWS tech stack.
- Google Maps Static API is very robust and very handy to access images for a required bounding box.
- The data usage and processing of OSM and Imagery to construct graph.
- The Iterative end to end approach of generating graph directly from images is very novel and can be used in many other domains/problems.
- The idea of dynamic training and rendering the graph into an image before input to CNN also broadens our perspective to expand such techniques in other domains.
- For open-source purposes, stating the hardware requirements and software versions makes the project more robust and streamlined.

Remaining results

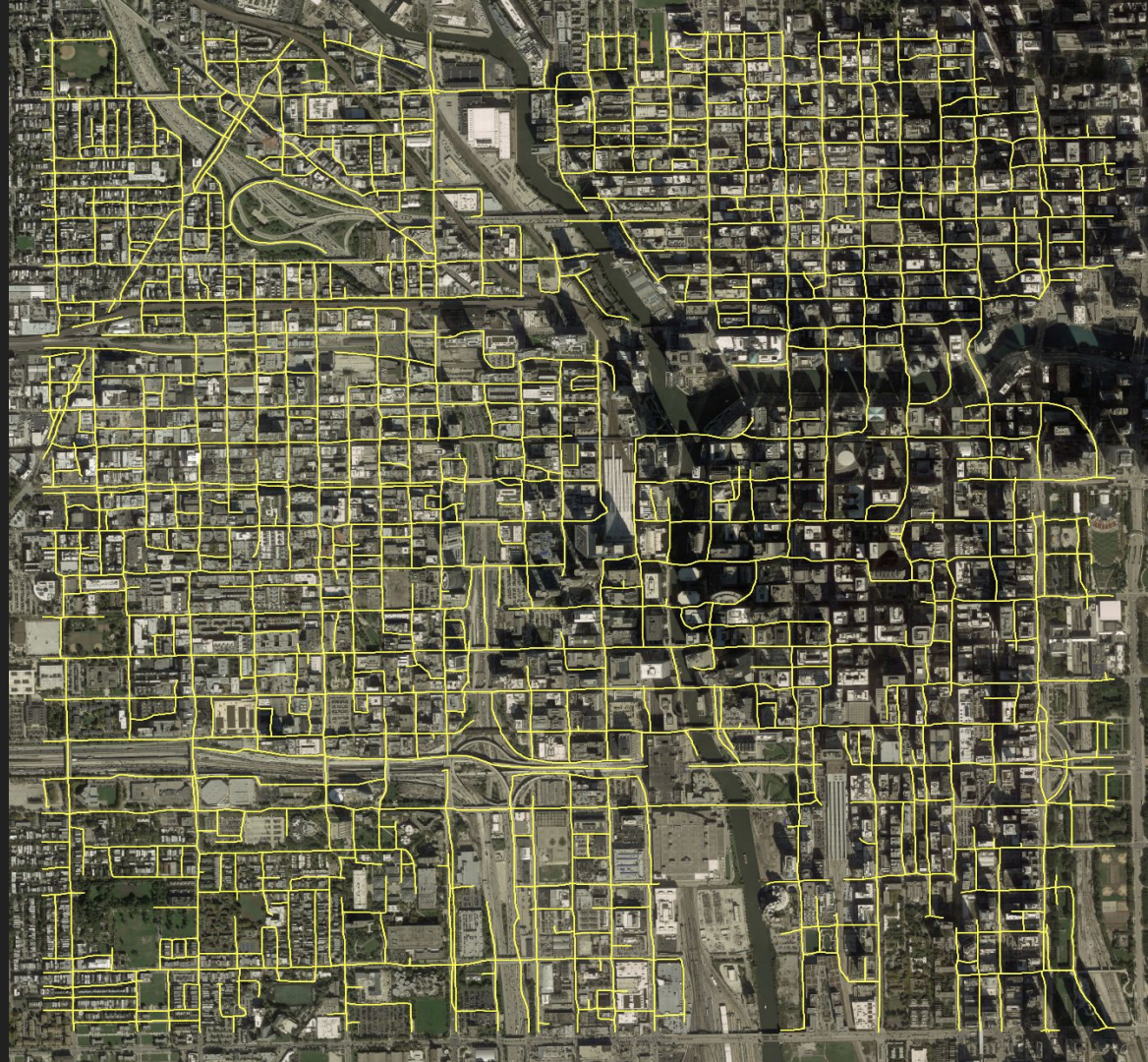
a) True Road Network for Amsterdam

b) Predicted Road Network for Amsterdam

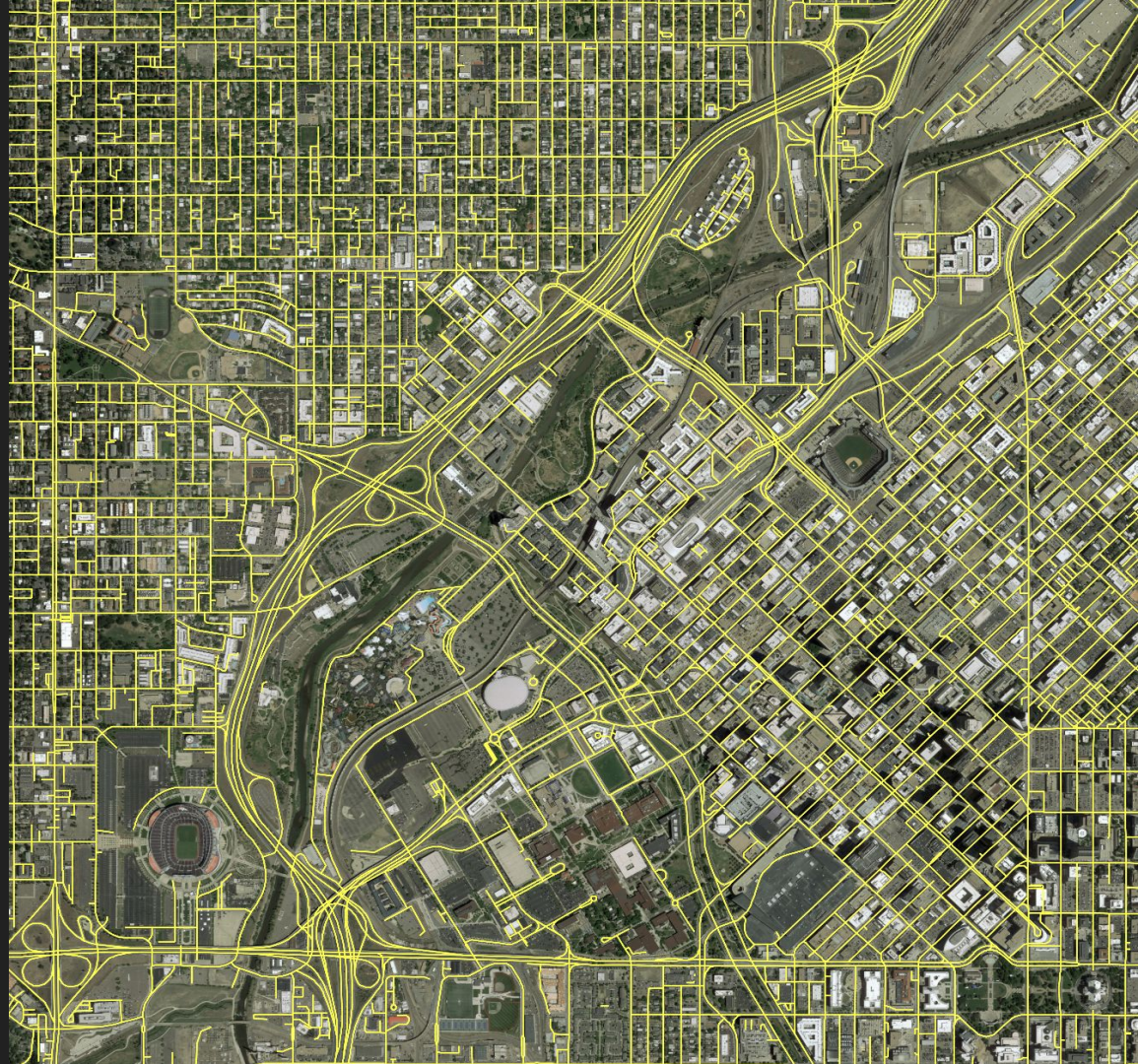
a) True Road Network for Chicago



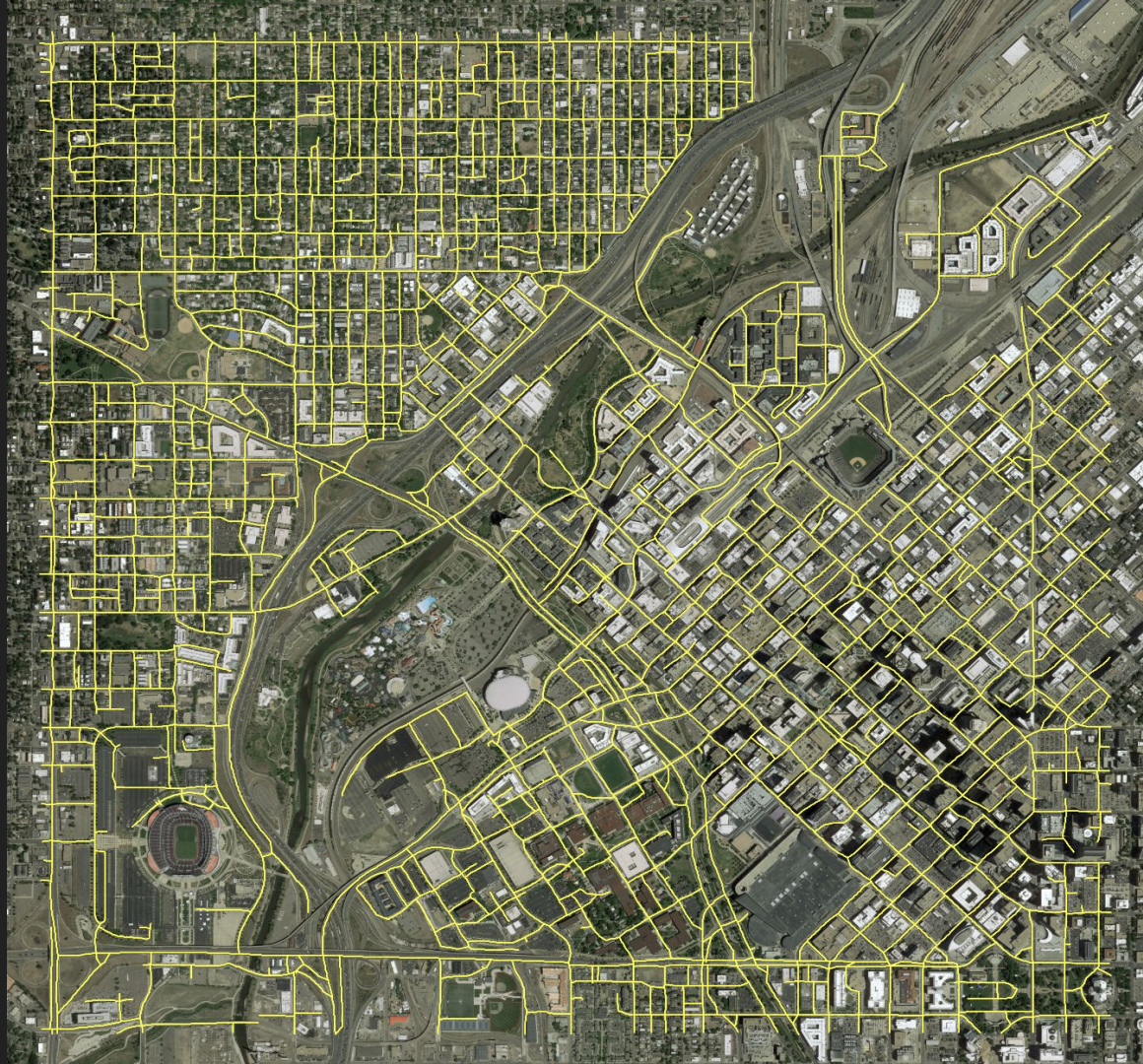
b) Predicted Road Network for Chicago



a) True Road Network for Denver



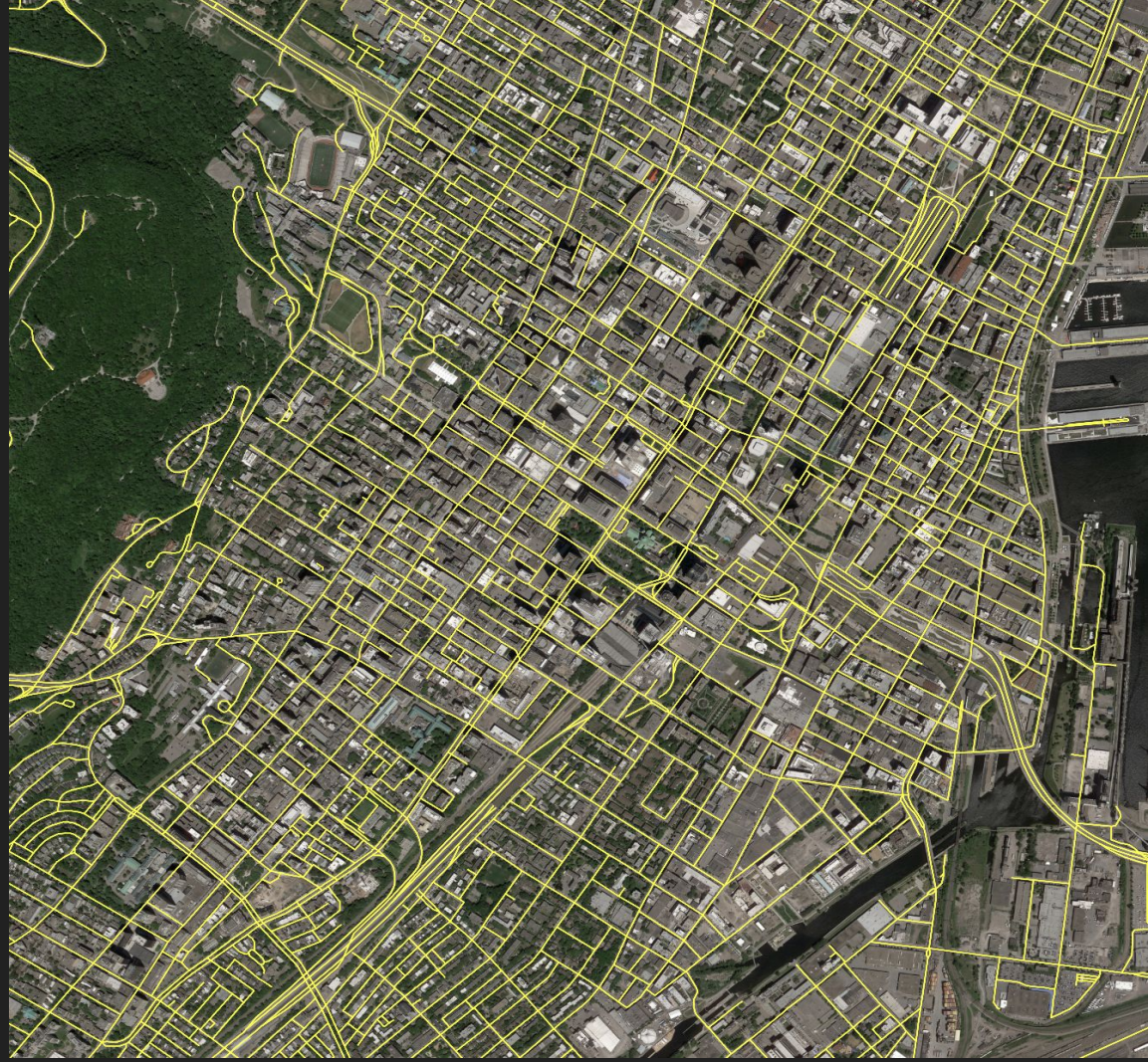
b) Predicted Road Network for Denver



a) True Road Network for LA

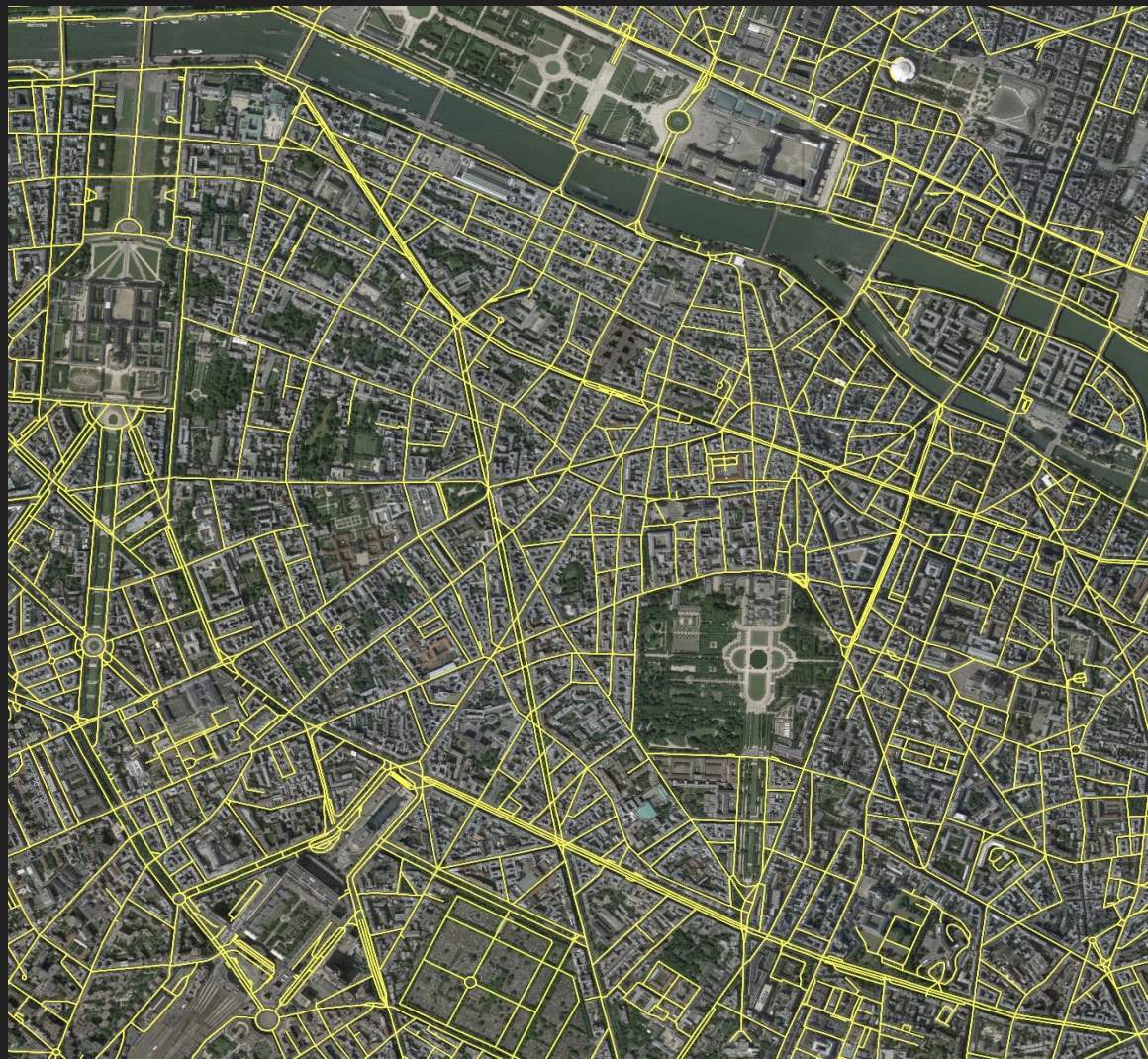
b) Predicted Road Network for LA

a) True Road Network for Montreal



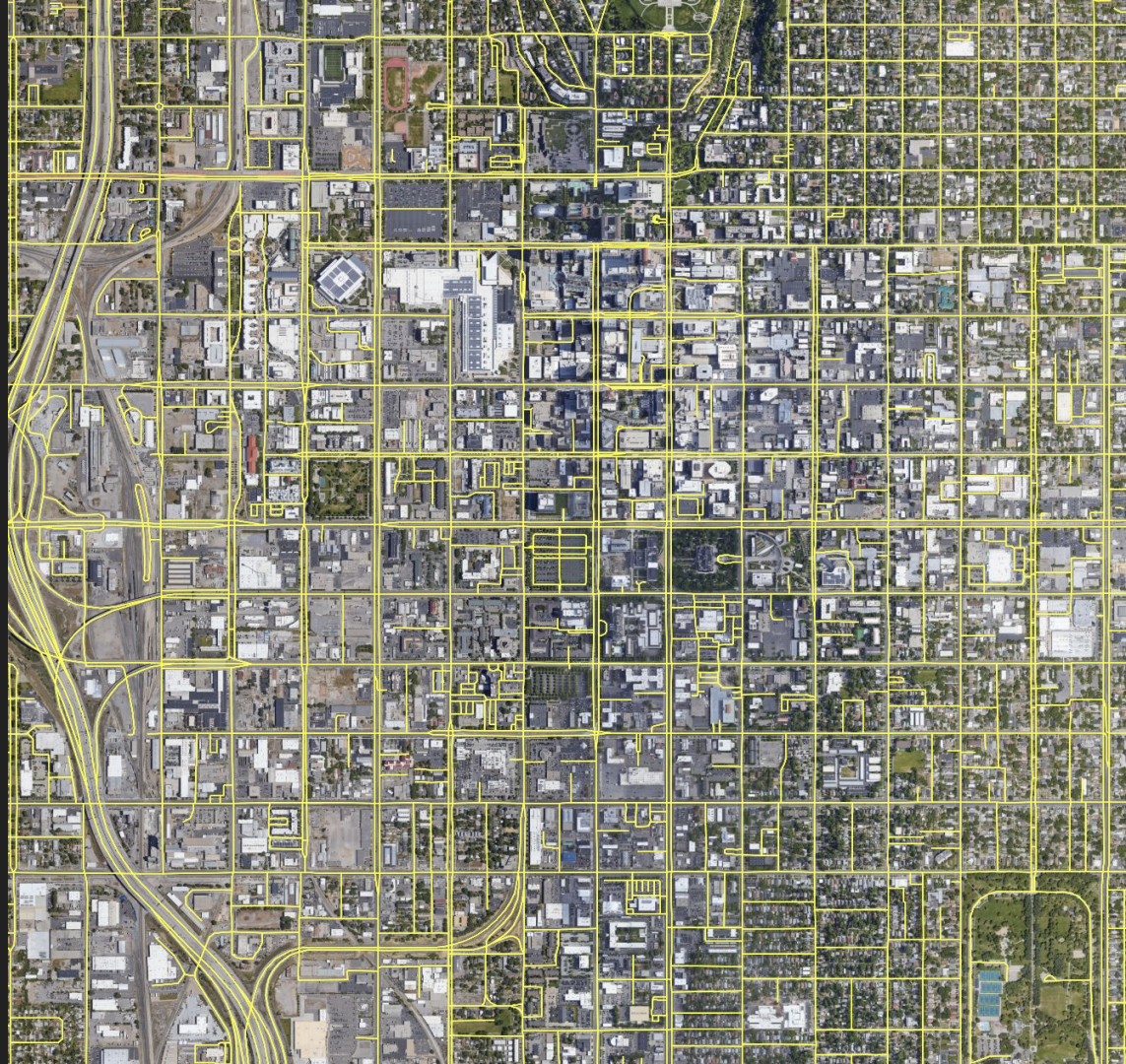
b) Predicted Road Network for Montreal

a) True Road Network for Paris

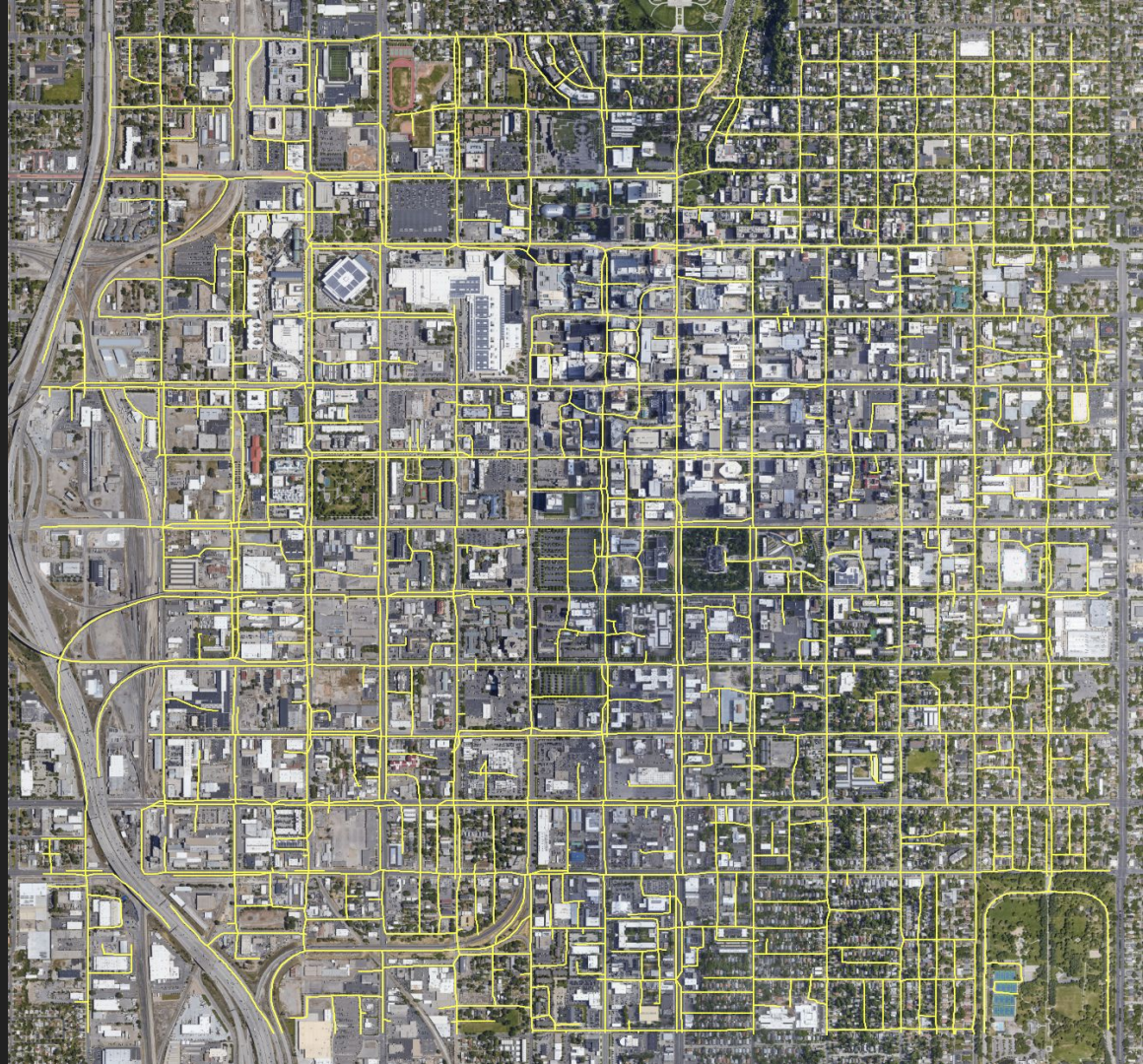


b) Predicted Road Network for Paris

a) True Road Network for Salt lake City



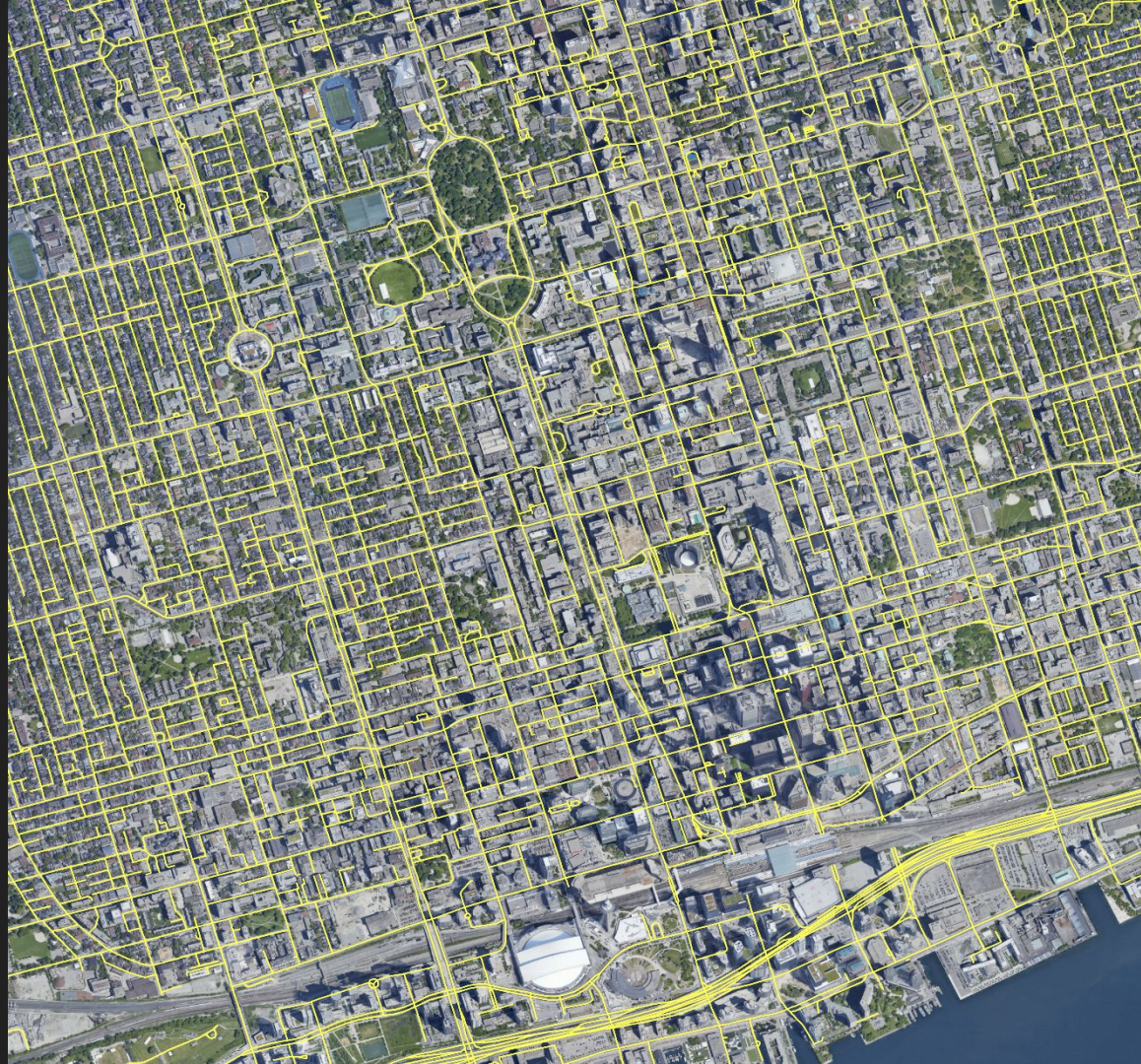
b) Predicted Road Network for Salt lake City



a) True Road Network for Tokyo

b) Predicted Road Network for Tokyo

a) True Road Network for Toronto



b) Predicted Road Network for Toronto

a) True Road Network for Pittsburgh

b) Predicted Road Network for Pittsburgh

Thanks!

Questions!!!