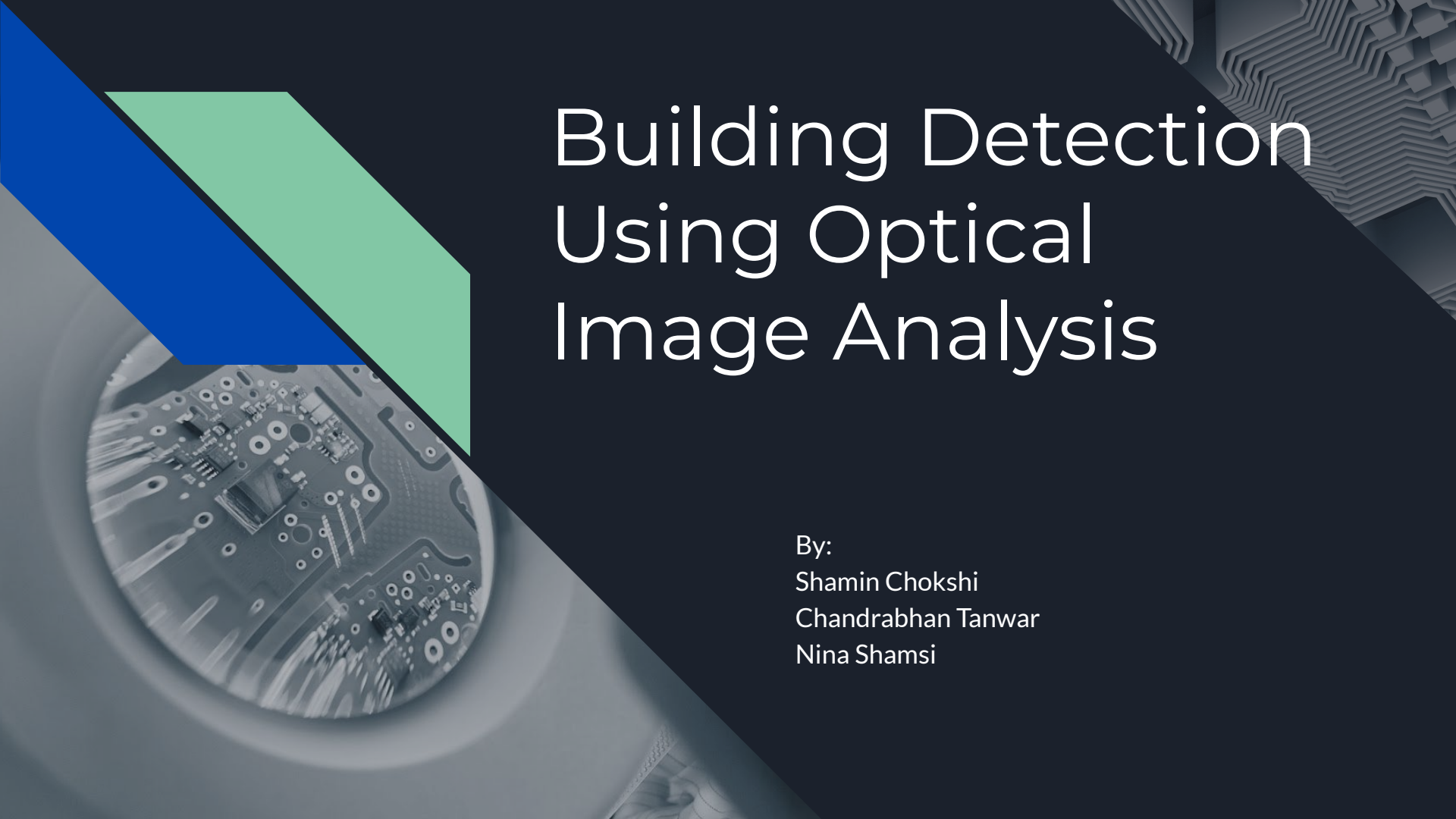


Building Detection Using Optical Image Analysis

By:
Shamin Chokshi
Chandrabhan Tanwar
Nina Shamsi

Working Of the Project

Objective: obtain masks with building segmentation predictions from satellite data.

How did we do it:

1) Train a Dynamic U-Net in PyTorch and evaluate a part of the data (Python),
2) Perform image processing and and image transformation (Scala),
3) Analyse the prediction output against expected scores (Scala)

Why do it this way:

Large model trained on web-scale data preclude the need for training specialized models because these large models are very good generalizers. For example, this is how models for NLP like GPT or for image segmentation like Segment Anything work.

So the goal is to have a flexible processing and inference analysis pipeline.

Dataset Used

1. The SpaceNet6 data was used.
2. It consists of satellite imagery over an area of Rotterdam as captured by MAXAR's Worldview-3 satellite.

AOI	Area (km^2)	# Images	# Building Footprint Labels
Rotterdam	120	3401	48000

Imagery features:

	PAN	RGBNIR	PS-RGB	PS-RGBNIR	SAR-Intensity
GSD (m)	0.5	2.0	0.5	0.5	0.5
Chip size (px)	900 x 900	450 x 450	900 x 900	900 x 900	900 x 900

https://earth.esa.int/eogateway/missions/worldview-3

https://torchgeo.readthedocs.io/en/stable/api/datasets.html

Image Ingestion

We ingest (900x900) RGB .TIF images and convert them into an appropriate type for out Image processing pipeline.

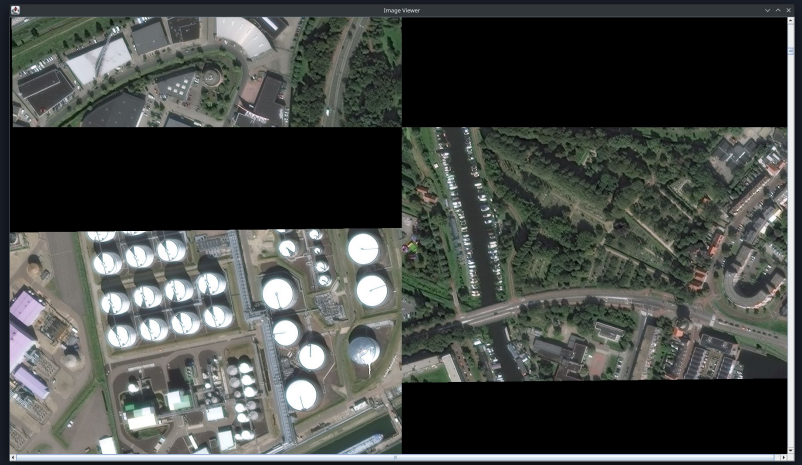


Image processing

The image ingestor reads images lazily and forwards it to the image processing code which then applies various functions onto it to for better accuracy and result

Image Processing and Transform:

- Images are converted to grayscale
- Grayscale images are deblurred using a gaussian kernel
- Images are further sharpened to increase edge detection capability
- Finally the images are resized and tiled into appropriate sized

Image processing pipeline

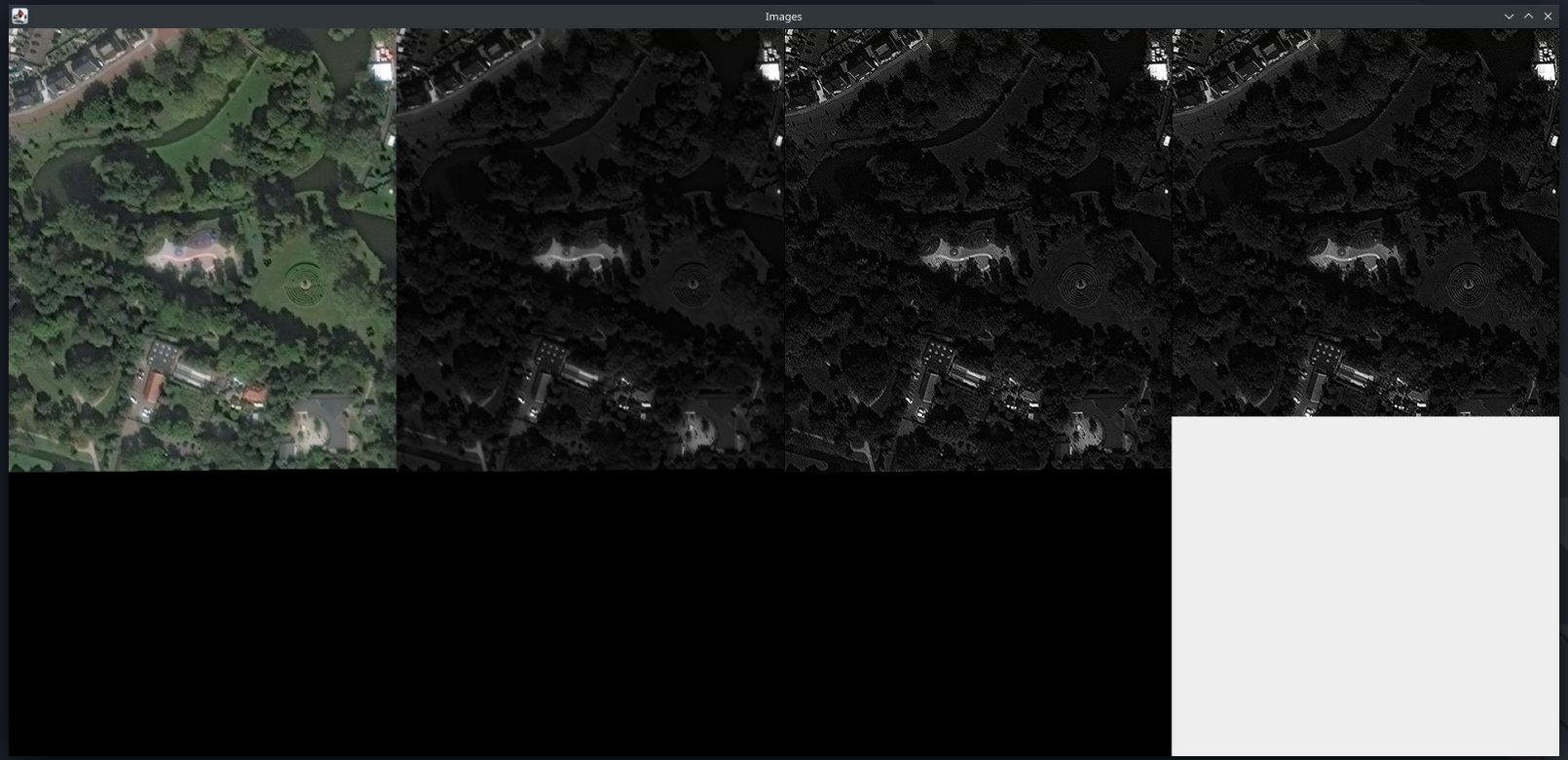


Image Manipulation

There are various built in function in the program for manipulating and testing ingested and prediction images. These include:

Analysis:

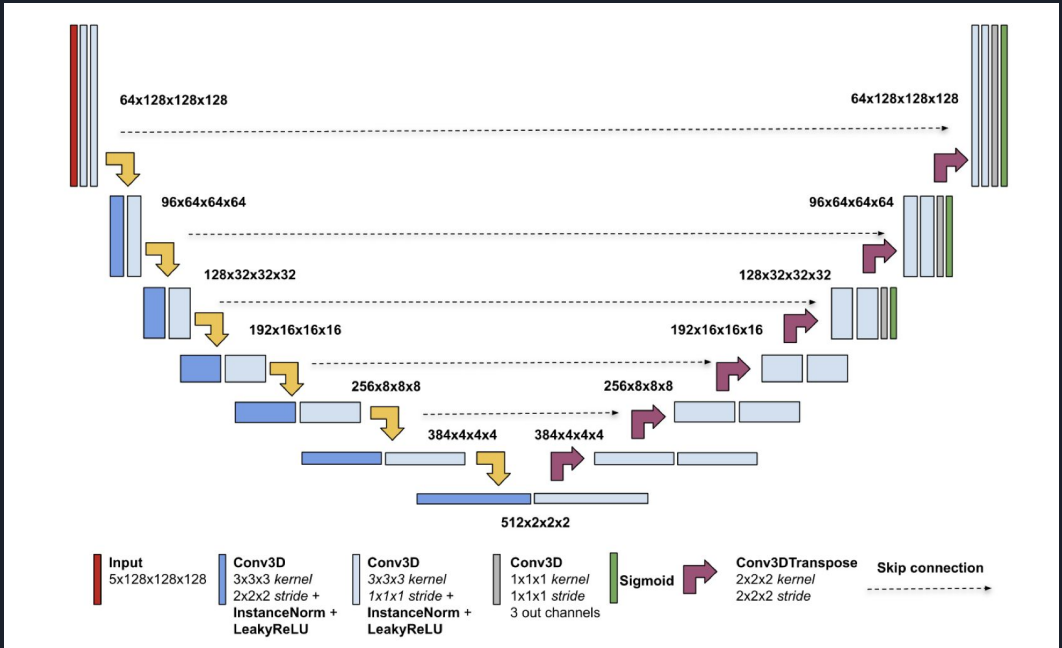
1. Image <-> Array conversion
2. Array <-> Set conversion
3. Format and size checks
4. Dice and IOU score calculation

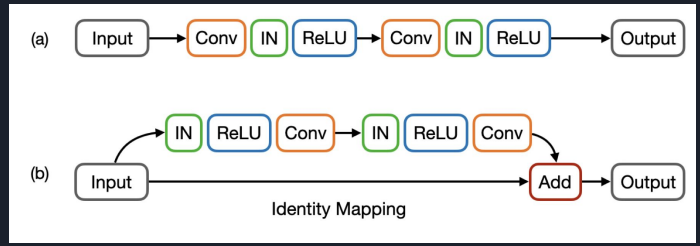
Processing:

1. Image and kernel padder
2. Deblurrer with Gaussian kernel noise variance
3. Boundary polygon generation
4. Tiler with greyscale and sharpening

Model for Building Detection

Dynamic U-Net (MONAI)





https://arxiv.org/pdf/2110.03352.pdf

Prediction Quality Analysis

Dice

$$\frac{2|X \cap Y|}{|X| + |Y|}$$

Dice Coefficient

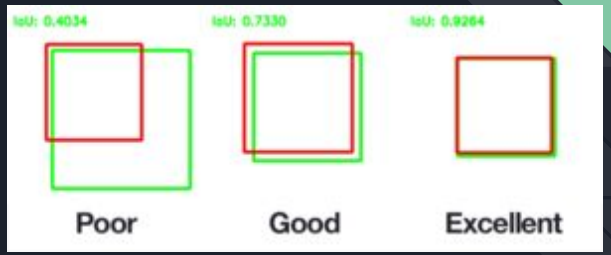
Jaccard

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|} = \frac{|A \cap B|}{|A| + |B| - |A \cap B|}.$$

IoU

Dice score is a measure of true positives, but it also penalizes for the false positives.

The main difference is in the denominator, where Dice is only looking at the total number of true positives, so Dice also penalizes true positives that were not found.



https://en.wikipedia.org/wiki/Jaccard_index

Where can this be used and future work

This application can be used for many purposes where post-training processing and analysis of geospatial data and predictions is required for inference

Military Use for surveillance in areas

Police use for detecting criminal activity in buildings

Can be used by the government for sites that suit construction

Acceptance Criteria

We got permission to forgo the UI.

Ingestion and processing used lazy loading and was simulated with writing to binary. Ingestion was within spec, processing took <4 and <2min (~0.5 GB small : ~10GB big)

Th model training Dice loss for the validation set was 0.49, while the Dice score was 0.73.

IoU was not used because it is not differentiable.

The Dice and IoU scores are 0.7 or greater for test image.

	Metric	Passing Criteria
Frontend	Response time	< 1ms
Backend	Data ingest	< 3 min (big) < 1 min (not big)
Data Processing	Imaging tiling	< 3 min (big) < 1 min (not big)
Model Training	IOU score Dice loss	>= 70 <= 0.5
Model Inference	Dice/IOU score	>=70