

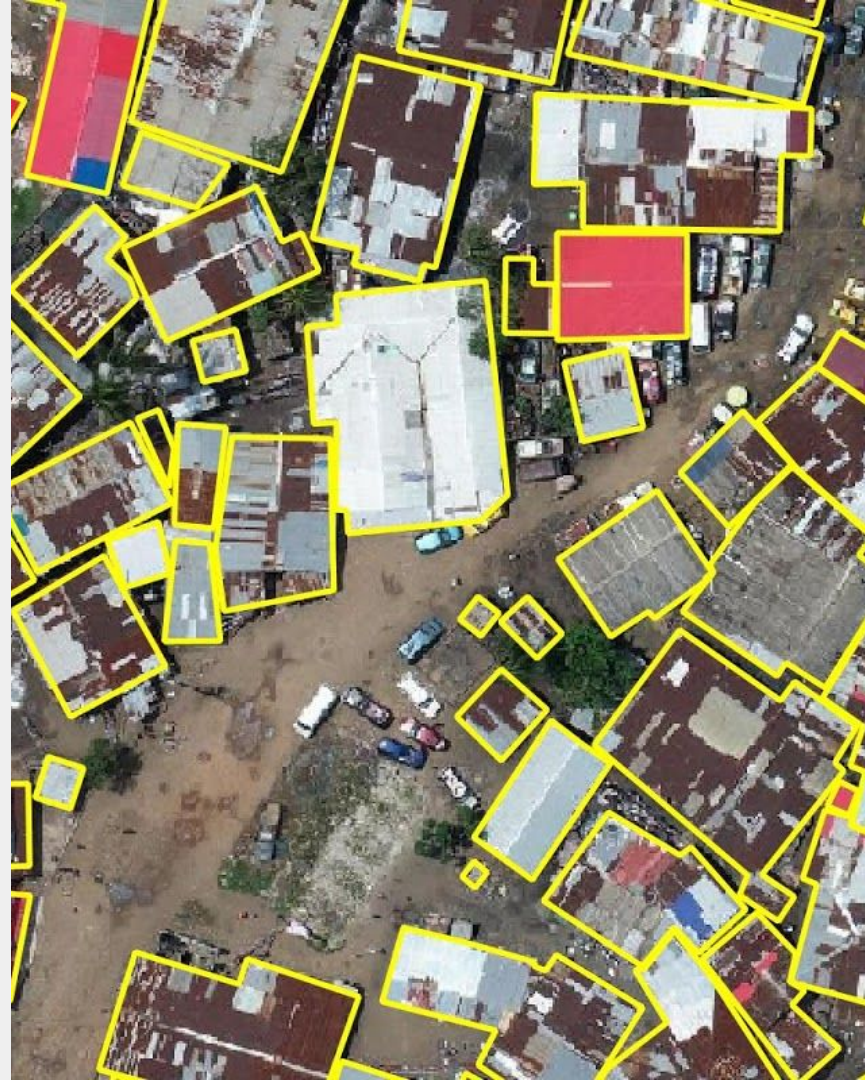
Open Cities AI Challenge: Segmenting Buildings for Disaster Resilience

DSC 672 CAPSTONE PROJECT

JOHN HOFF

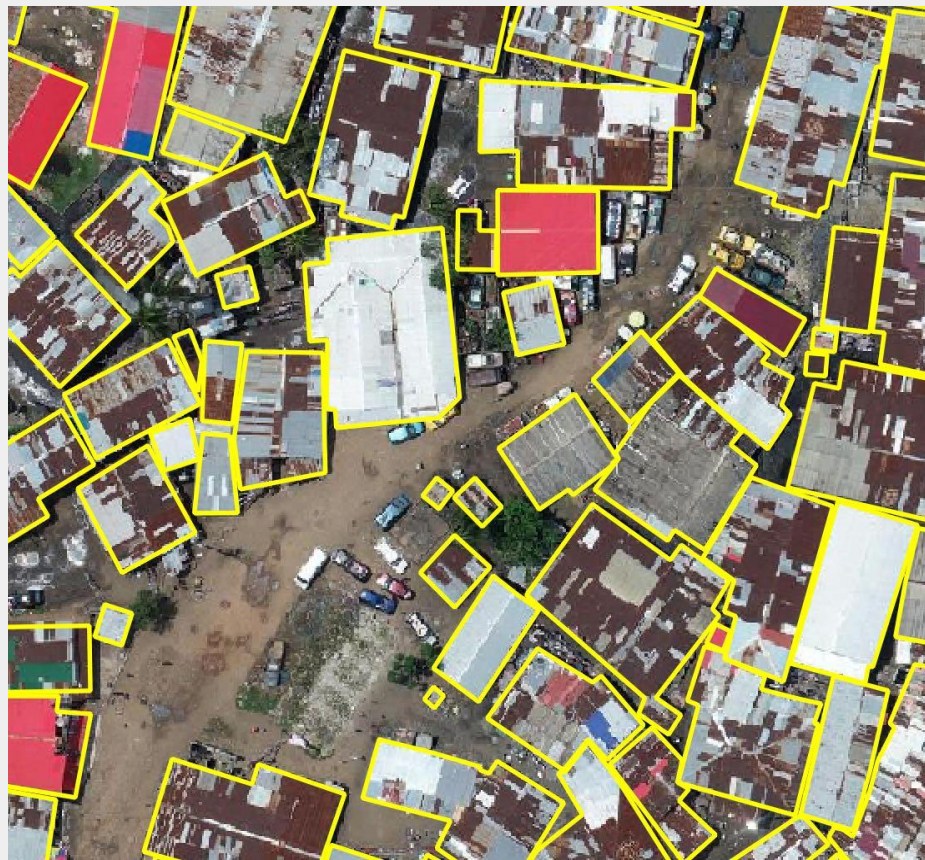
BRIAN PIRLOT

SEBASTIAN ZDAROWSKI



What is the project and why?

- The focus of this project will be to identify the footprint of buildings based on aerial images of African cities/regions on a pixel by pixel basis
- Data consists of drone images from 10 different cities/regions across Africa
- Goal is to develop a model that can perform proper classification on these images to identify if buildings are present within an image or not.
- Project is valuable primarily because of its applications to disaster risk management.
- Africa is expected to have one of the strongest population growth of any world region in the next 80 years. Due to this growth, it is increasingly important to be able to map cities/regions in case of disasters/humanitarian crisis that may occur within the region. [1]



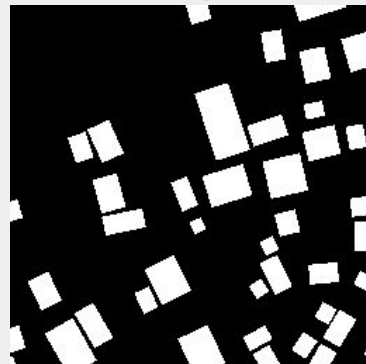
Project Approach

- Understanding the source data
- Cleaning and preparing the dataset
- Github/AWS setup
- Model approaches
- Scoring on model
- Mask RCNN
- Solaris
- Comparison of Mask RCNN & Solaris
- Conclusion/Future work



The Dataset

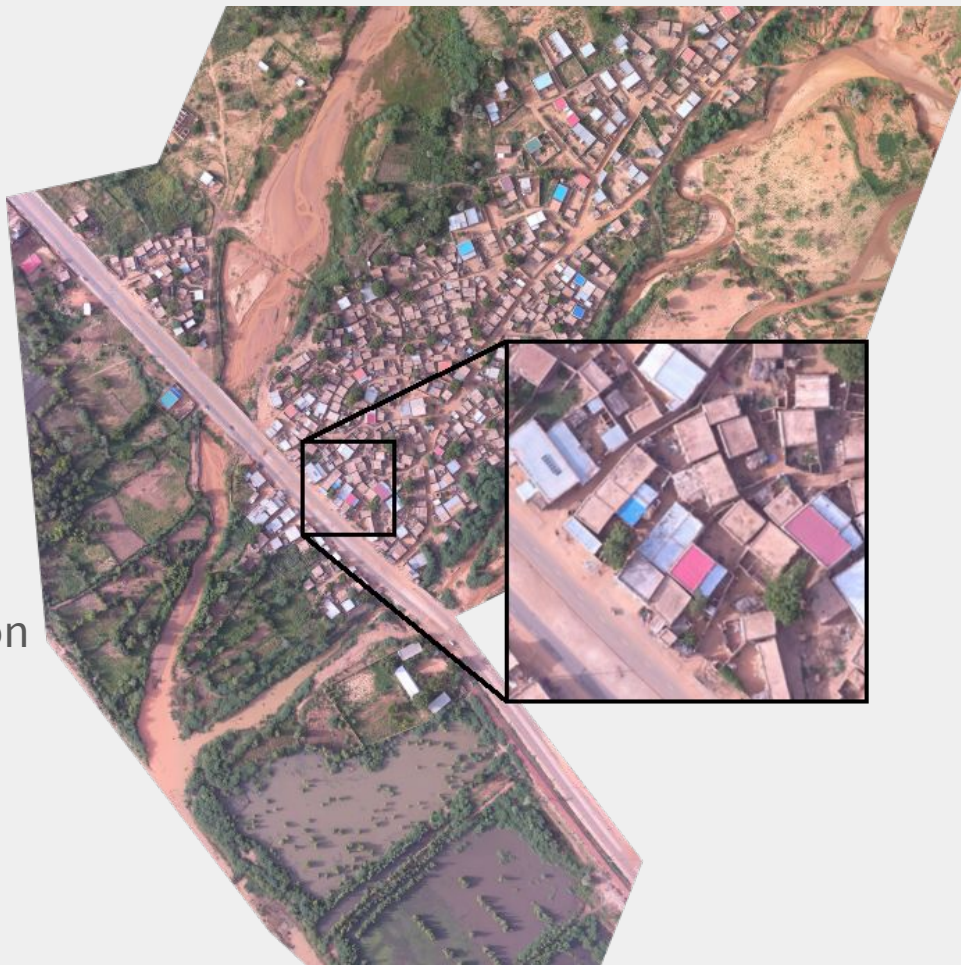
- Data is provided as part of an active competition on [drivendata.org](https://www.drivendata.org)
- Link:
<https://www.drivendata.org/competitions/60/building-segmentation-disaster-resilience/page/150/>
- Stored as STAC Files (Spatio Temporal Asset Catalogs)
 - Provides a way of querying geospatial imagery and labels
 - PySTAC: Python Library used for manipulating/working with STAC Objects
- Currently Split into Tier 1(32 GB), Tier 2 (40GB) and testing set (9 GB)



Extracting Training Data

Creating Map Tiles

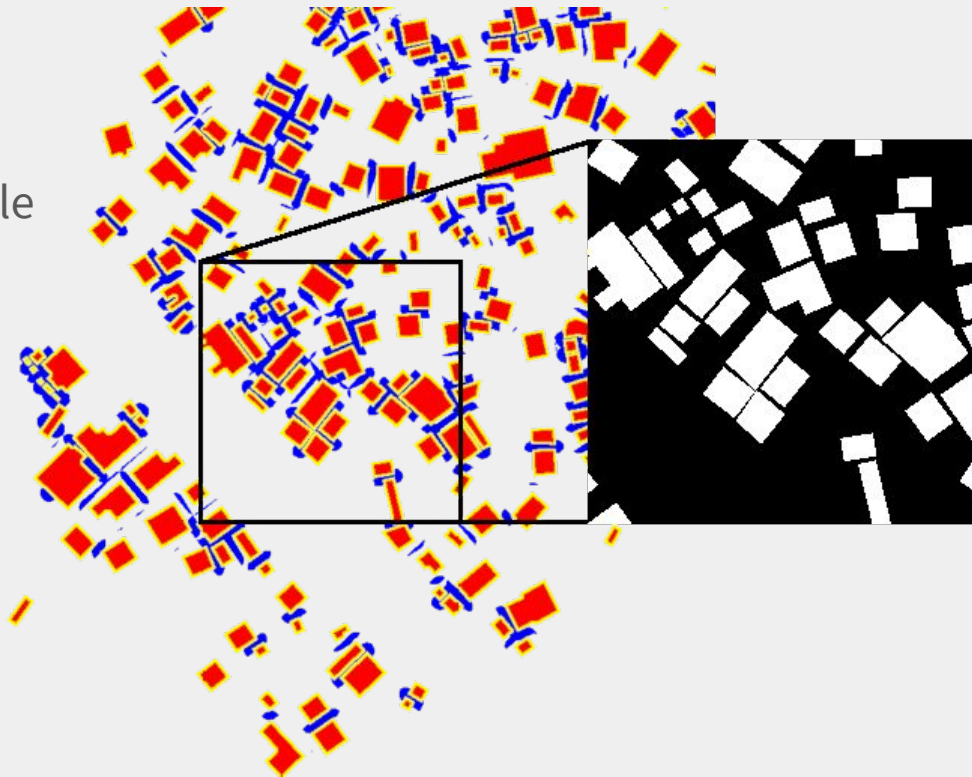
- Fixed tile size and variable zoom factor
 - 256x256 tiles
 - Zoom level 19 and 20
- Tiles without map data skipped
- 100-4500 tiles per map depending on zoom level
- 67 maps



Extracting Label Data

Creating Label Masks

- Clip all polygons to corresponding tile extract from GeoTiff
- Keep only building footprint
- Convert into binary image
 - True/White: Building
 - False/Black: Not Building



Training Data

For our contest submission we are using only Tier 1 data due to the higher quality of labels

- Tier 1
 - Zoom level 19 and 20
 - 40,000 random images
- 95/5 Training Testing Split
 - 5% used only for threshold tuning
 - Contest hold-out used for final scoring

	Zoom 19	Zoom 20
Maps	28	28
Tiles	8,160	31,840
Map Pixels	499,052,300	2,012,537,520
Building Pixels	43,803,468	173,684,448
Building Percentage	8.8%	8.6%

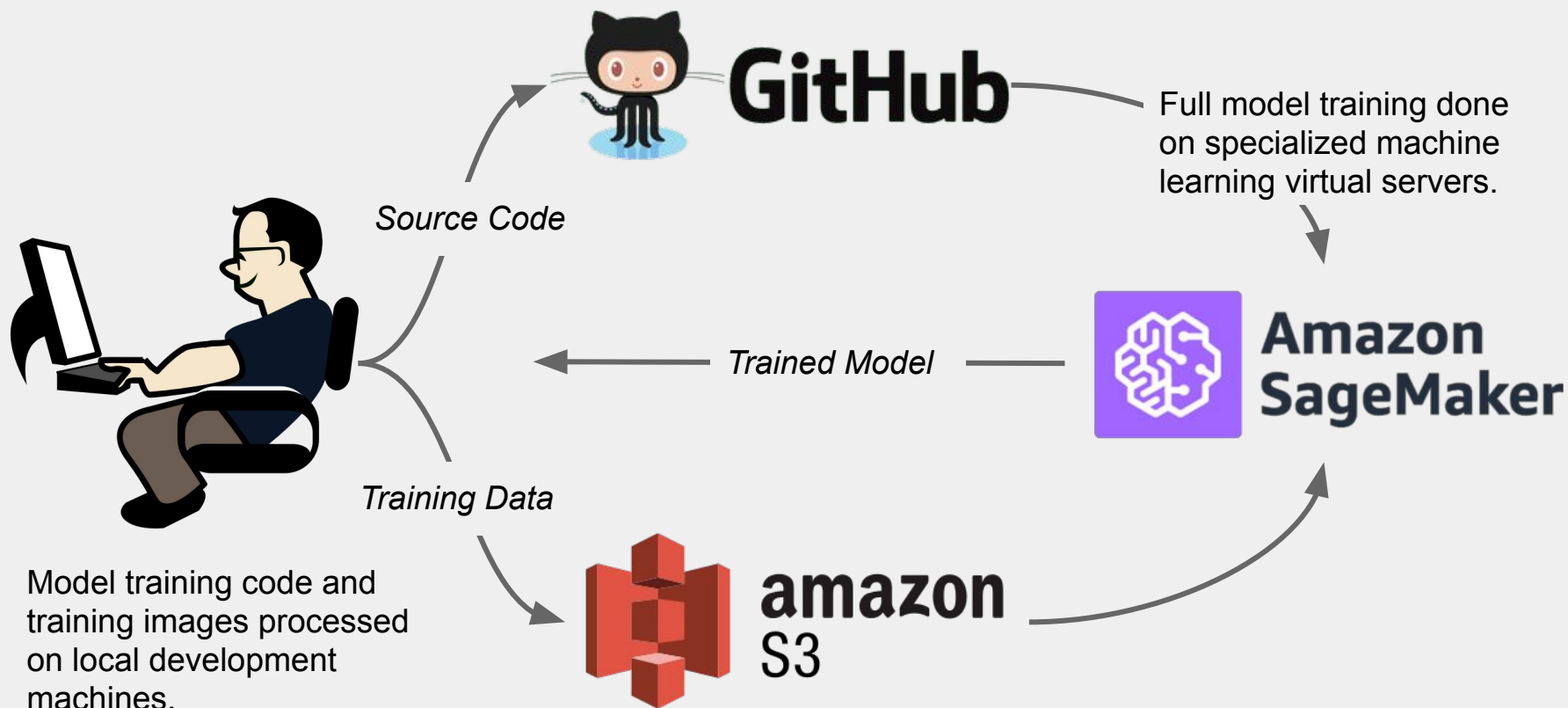
We include a more traditional 80/20 training/testing split using data from both Tier 1 and Tier 2 in our paper.

Testing Data

- Used to generate contest submission
- 11,481 tiles not present in the training data sets
- Georeferences removed
- Labels withheld from contest participants
- Multiple resolutions in each file
- Appear to have non-uniform zoom levels
- 9 GB



Our Workflow for Collaboration and Training



Cloud Computing Expenses


	Week 4	Week 5	Week 6	Week 7	Week 8	Week 9	Week 10	Week 11	Total
SageMaker			\$8.89	\$22.74	\$29.43	\$15.11	\$48.88		\$125.05
S3	\$0.80	\$3.50	\$1.24	\$2.95	\$0.76	\$3.92	\$0.31		\$13.48
Paperspace					\$22.06				\$22.06
	\$0.80	\$3.50	\$10.13	\$25.69	\$52.25	\$19.03	\$49.19	\$0.00	\$160.59

This project could not have been completed without these cloud resources. Only a single team member had hardware capable of training our models in reasonable time.

Model Approaches

- Two model approaches proposed on solving problem at hand
 - Both are deep neural networks that aim to solve instance segmentation problems
- Mask Region Convolutional Neural Network (Mask RCNN)
 - Region based convolutional neural network that can be utilized for object detection
 - Benefit of creating a pixel - level mask of where each instance is located
 - Extension of existing model: Faster RCNN
- Solaris
 - Built upon SpaceNets Tool suite called SpaceNet Utilities
 - Provides a way of preprocessing and modeling overhead images
 - Allows for custom deep learning or pre-trained models

Mask RCNN

The logo for Solaris features the word "Solaris" in a large, black, sans-serif font. The letter "o" is replaced by a stylized blue globe with a grid pattern and a bright blue glow. A black orbital ring with a small black dot at its center encircles the globe.

Solaris

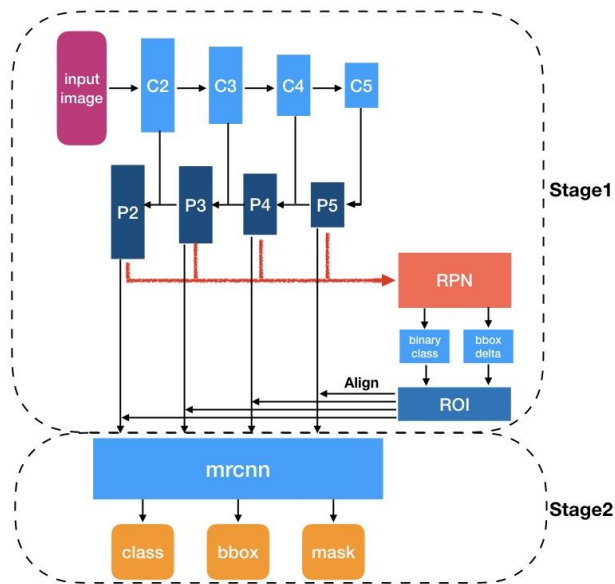
Scoring: Jaccard Index

- Jaccard index is the accuracy metric being used for the DrivenData.org competition
- It is calculated as the intersection divided by the union
 - Where 'A' is the true value and B is the predicted value
 - 'A' and 'B' in this case are the sets of pixels that make up the actual and prediction masks
- A higher score is better

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

Mask R-CNN: (Mask Region-based Convolutional Neural Networks)

- Mask RCNN is a deep neural network aims to solve instance segmentation problems
 - Extends Faster-CNN / by having an additional branch for predicting segmentation masks on each region (RoI)
- There are two stages of Mask RCNN
 - First, it generates proposals about the regions where there might be an object based on the input image.
 - Then it predicts the class of the object based on the image, refines the bounding box and then generates a mask on the pixel level
 - Built upon Region Proposal Network



Mask RCNN: Setup

- Initial model setup
 - Model Template used from Github Repo Matterport
 - Link: https://github.com/matterport/Mask_RCNN
- Setup of Virtual Environment
 - Tensorflow: 1.14.0
 - Keras 2.2.4
- Mask RCNN Building Config
 - # of classes: 1 background + 1 building
 - Images Min & Max Parameter: 256 (Save time on training)
 - Mean Pixel: Configured on Training set
 - RPN Anchor Scales:
 - (32,64,128,256,512)
 - (16,32,64,128,256)

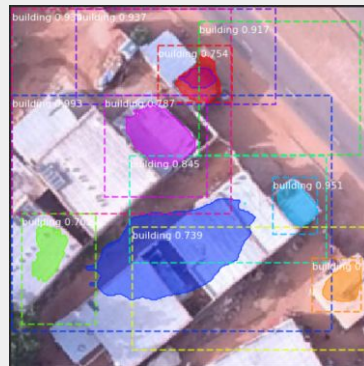


Mask RCNN: Training Load

- Loading of Training set into Mask RCNN
 - Building Masks were set as 1 object at first
 - Mask Splitting allowed buildings to be recognized as individual objects
 - This allows us to identify individual buildings instead of one large object
- Training Files:
 - Tier 1 Images (*Train set 2*)
 - Zoom level 19/20
- Training Parameters
 - Head: 40 Epochs, 4th layer: 20 Epochs, All layers: 20 Epochs
 - Learning Rate
 - For Head layer and 4th Layer: 0.001
 - For all layers: .0001



Mask RCNN: Training Results



- Multiple Training Results Displayed
- Training times varied if ran on local machine or on GPU (S3)
- RPN Anchors
 - (32,64,126,256,512)
 - (16,32,64,126,256)

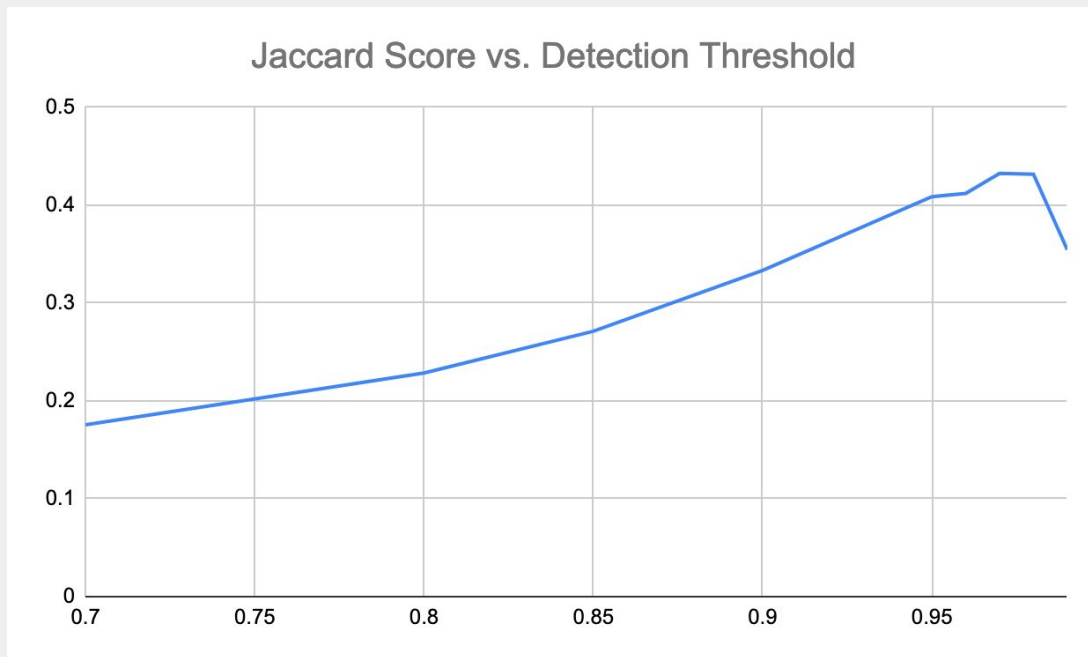


Mask RCNN: Region Proposal Network(RPN) Box Threshold



- By increasing the RPN threshold, we are able to filter out noise from prediction results
 - Noise can defined as trees, rocks, streets, cars, etc..

Mask RCNN: Jaccard Score Vs RPN Detection Threshold



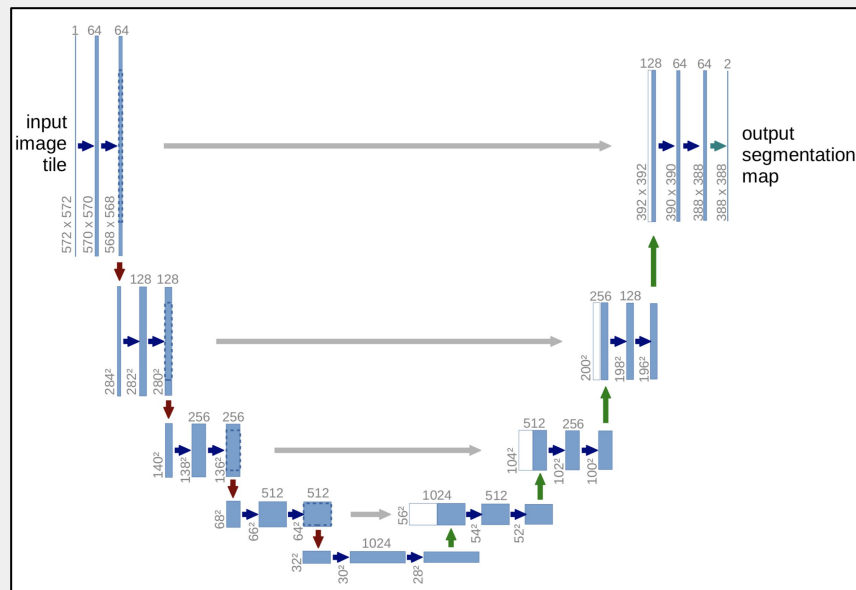
Mask RCNN: Results

- Choice of a Region Proposal prediction threshold allows us to filter out noise
- Predictions were made on a 5% holdout from the training set
- Global Jaccard scores were calculated using a variety of RPN thresholds at intervals of 0.05 to 0.01 to evaluate the optimal value
- The score peaked using 0.97

RPN Detection Threshold	Global Jaccard Score
0.70	0.175601
0.80	0.228504
0.85	0.271116
0.90	0.333414
0.95	0.40871
0.96	0.411988
0.97	0.43256
0.98	0.431722
0.99	0.354536

Solaris Model: U-Net

- Solaris uses an architecture based on U-Net proposed by Ronneberger et al (2015)
- The network is thought of as "U-shaped" because it contracts and then expands
 - The contracting side is like a typical CNN, but without fully connected layers for prediction
 - The expanding side uses the same layers as the contracting side, but the convolutional layers are replaced with transpose convolutional layers
 - The output is a set of pixel-wise predictions
- Our implementation uses a popular image classification architecture called VGG16 for both sides of the network



Solaris: Training

- Training was split between a local GPU and a cloud GPU
- Total training time for the project was well over 100 hours
- Hyperparameter tuning was difficult due to training times, so most attempts used the following:
 - Batch Size: 8 for local GPU, 24 for cloud GPU
 - Learning Rate: 0.0001
 - Epochs: 60
 - Augmentations: Pixels Normalized, 50% Random Horizontal Flip, 50% Random 90 Degree Rotate

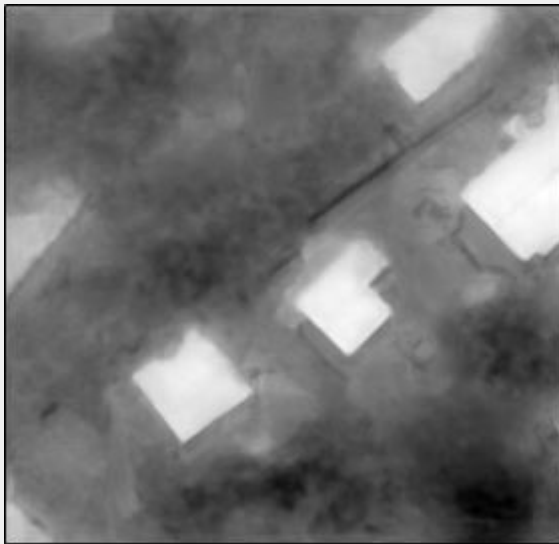
Solaris: Predictions

- The raw output from the network provides a probability for each pixel, with higher values more likely to be buildings
 - The competition requires a binary classification (building vs not a building), so a threshold must be applied to the probabilities

Input Image



Raw Output



Binary Classification



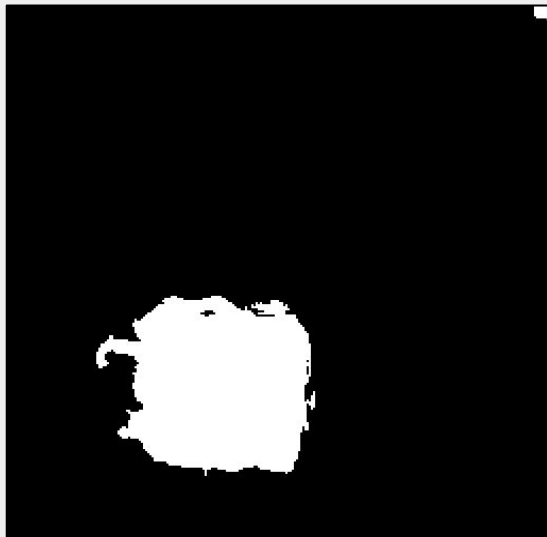
Solaris: Douglas-Peucker Algorithm

- The Douglas-Peucker algorithm provides a way reducing the number point points to define a shape
 - Most buildings can be defined with a limited number of points, so this process can produce more likely shapes when predictions are irregular

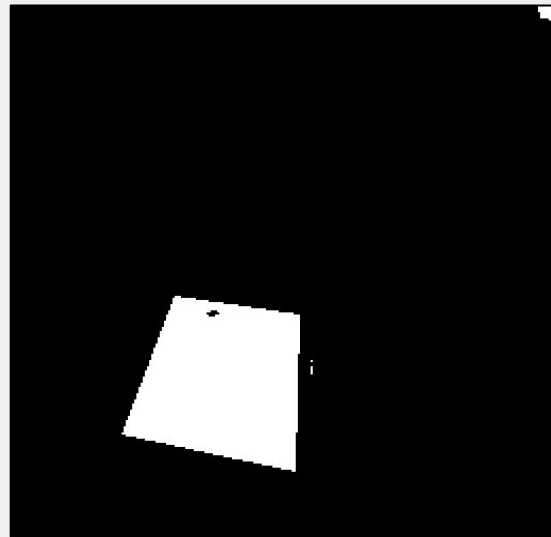
Input Image



Prediction Before Algorithm



Prediction After Algorithm



Solaris: Minimum Polygon Size

- Another helpful strategy for generate more likely predictions is only accept shapes over a minimum size
 - Most small predictions are noise without the presence of a building
 - This strategy works well when combined with the Douglas-Peucker algorithm

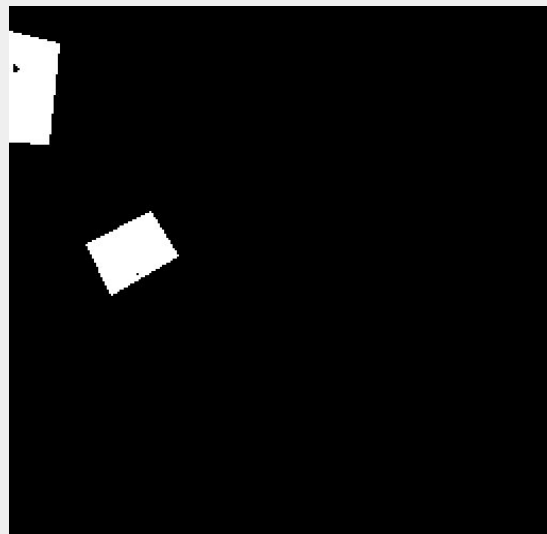
Input Image



Without Min Size or Simplified



With Min Size and Simplified



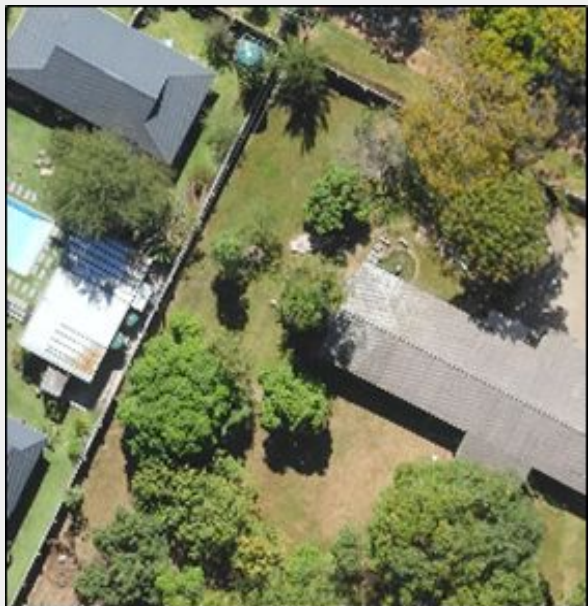
Solaris: Jaccard Results by Threshold

- Choice of a binary prediction threshold is crucial to model performance
- The score peaked using 0.80

Binary Threshold	Global Jaccard
0.60	0.1621
0.65	0.1722
0.70	0.1821
0.75	0.1904
0.80	0.1950
0.85	0.1947
0.90	0.1893
0.95	0.1803

Solaris: Challenges - Greenery

- Images with large amounts of green (grass and trees) predicted poorly
- This is likely due to the Tier 1 training samples containing relatively little green



Solaris: Challenges - False Positives on Blank Tiles

- There are a large amount of false positives for input images without buildings
- It is not clear what is causing this, but it could be related to the green issue
- The false positives tend to be too large to be removed with a minimum shape size
 - The simplification algorithm is also ineffective for this issue



Comparison

Mask RCNN

- Advantages
 - Mask R-CNN is easy to generalize to other tasks
 - Region Proposal Predictions allow each building to be identified individually
 - Multiple configuration settings allow for hyperparameter tuning
- Disadvantages
 - Poor documentation
 - Training times: affects how many times you can “tune”
 - Each epoch of Mask RCNN training only sees 10% of the dataset
- Score
 - Final score 0.43256 at .97 RPN threshold

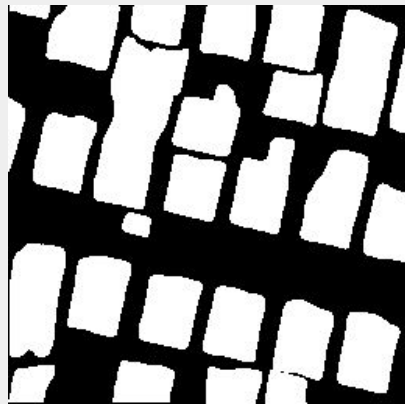
Solaris

- Advantages
 - With proper batch size, training was stable without any crashing
 - Each epoch saw a larger portion of the data
 - Post-processing was easy to integrate into the workflow
- Disadvantages
 - Pixel-wise predictions created more irregular shapes than region-based predictions
 - False positives near edge of tiles were common
 - New library with limited documentation
- Scores
 - Final score 0.3369 at 0.7 binary threshold

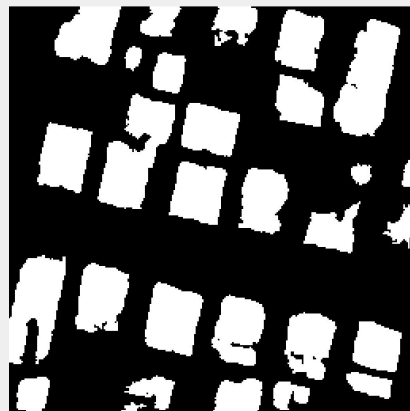
Contest Submission Results

Model	Submission Score	Rank (Out of 1,082)
Mask RCNN	0.4555	95
Solaris	0.3369	103

Mask RCNN



Solaris



Conclusion/Future Work

- Are we able to segment buildings using Computer Vision systems?
 - Yes, but in order to obtain a higher score, more resources are required.
 - While not perfect semantic segmentation across all test images, we do have results that do show most buildings on the test set.
- Could this be a reliable strategy for mapping out areas?
 - Yes because manually extracting reliable population data from photos can be painfully slow and expensive, our methods allows it to be done automatically with a well trained model.
 - A well trained model could be applied to segment and identify building footprints within hours
- Future Work:
 - Further tuning of Douglas-Peucker simplification algorithm and minimum polygon size
 - A bigger budget/time so that we are able to run more epochs with each model
 - Usefulness of Tier 2 images: Manually mapping mask images with VGG Image Annotator (VIA)