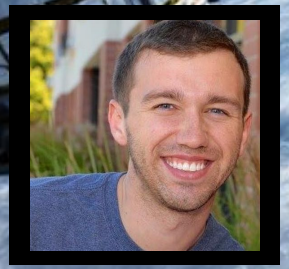
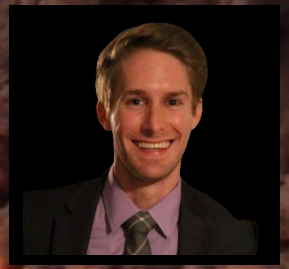
Natural Disaster Geospatial Damage Assessment

Team Disaster
DAEN 690 Project
Spring 2020
Sponsor: Accenture Federal Services

Team Members

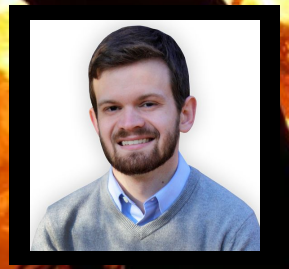


Billy Ermlick



Nick Newman
(Product Owner)

Devayani Pawar



Tyler Richardett
(Scrum Master)

Accenture Federal Services:
Marc Ruiz Bosch & Christian Conroy

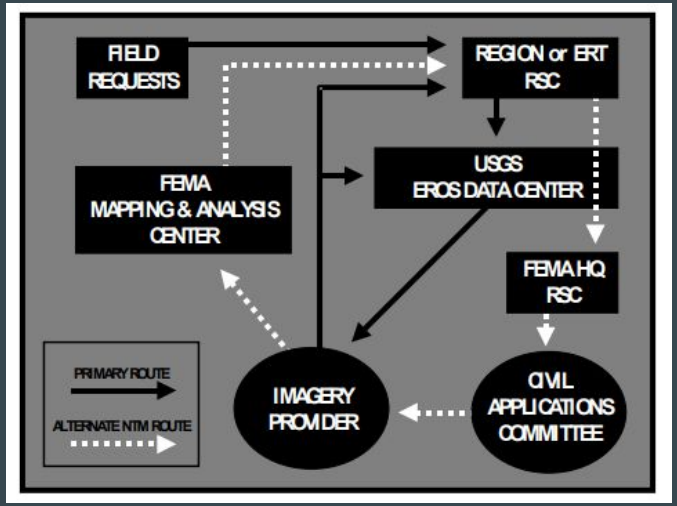
Problem Definition

- Relief efforts rely on damage assessments for identifying target areas, extent of damage, and required funding
- Typically a slow and dangerous “on-the-ground” assessment is performed
- Need a way to assess damage in a fast, accurate, and safe way

FEMA conventional damage assessments [1]

Problem Definition

- Satellite images provide fast information regarding damaged areas
- FEMA and others currently manually review images for building damage
- Provide assessments according to the United States National Grid
- Need to automate this review process



FEMA remote sensing procedure [2]

Project Definition

Objective

Create a proof-of-concept application which uses satellite images to automatically estimate building damage and visualize cost according to the United States National Grid coordinate system.

Agenda

1. Datasets
2. Financial Modeling
    - USNG Mapping
    - Financial Model & Visualization
3. Damage Estimate Modeling
    - U-net Architecture
    - Mask-RCNN
4. Application

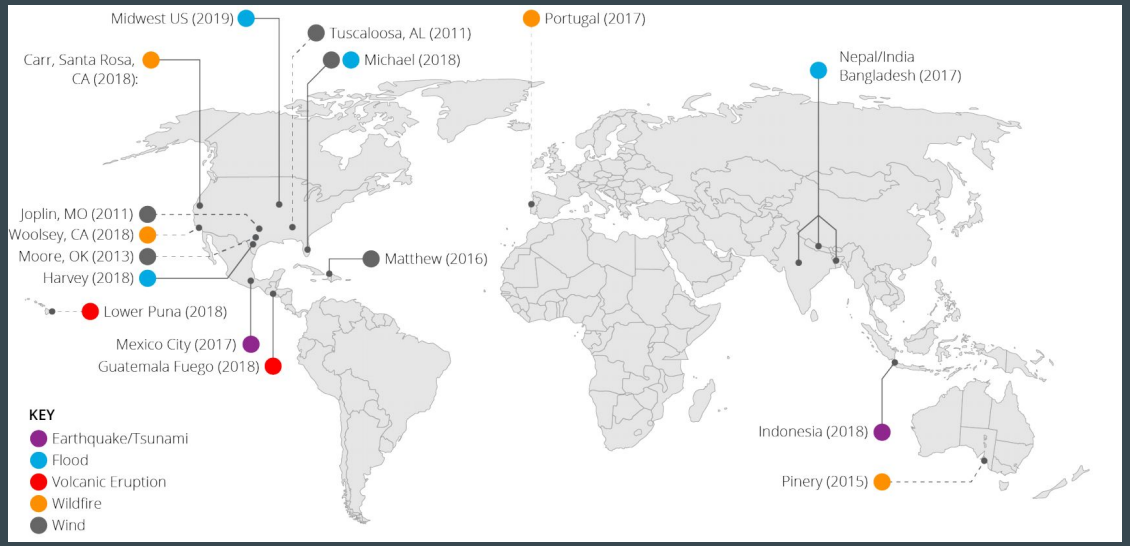
Datasets

Xview2

Xview2 Dataset

Overview

- Defense Innovation Unit & Maxar open competition ended January 2020
- Public dataset
- Images of 850k buildings before and after disasters from around the world from 2011-2018



Disaster events and locations [3]

Xview2 Dataset

Overview

- 6 types of disasters (hurricanes, wildfires, floods, earthquakes, etc.)
- Images of same location before/after disaster (nadir angles may vary)

Pre and post disaster image [3]

Xview2 Dataset

Annotations

- Unified labels for building locations and standardized damage level (0-4)
- Panel reviewed for consistency

Disaster Level	Structure Description
0 (No Damage)	Undisturbed. No sign of water, structural or shingle damage, or burn marks.
1 (Minor Damage)	Building partially burnt, water surrounding structure, volcanic flow nearby, roof elements missing, or visible cracks.
2 (Major Damage)	Partial wall or roof collapse, encroaching volcanic flow, or surrounded by water/mud.
3 (Destroyed)	Scorched, completely collapsed, partially/ completely covered with water/mud, or otherwise no longer present.

[3]

Annotated post-disaster image

Xview2 Dataset

Predictions & Evaluation

- Output two PNG image with pixel values representing
  - building locations (0/1)
  - damage level (0-4)
- Final f1 score is evaluated based on 30% location and 70% damage f1 values

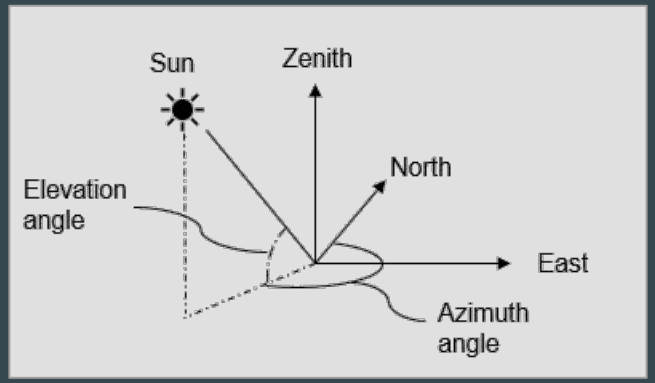
[3]

Annotated post-disaster image

Xview2 Dataset

Field Descriptions

Attribute	Type	Description
Image ID	STR	Unique ID identifying each image
Disaster ID	STR	Unique ID identifying each disaster
Image Name	STR	Pre/post and which disaster
Capture Date	Date	Date the image was captured
Height/Width	INT	All images are of size: (1024, 1024, 3)
Ground Sampling Distance	Float	Distance between pixel centers on the ground
Off_Nadir_Angle	Float	Image capture angle relative to ground plane
Pan_Resolution	Float	Resolution of single band image used for pan sharpening
Sun_Azimuth	Float	Angle of the sun from polar north
Sun_Elevation	Float	Angle of the sun from ground plane
Target Azimuth	Float	Angle of satellite from polar north
Provider_Asset_Type	STR	Satellite model image taken from
LAT/LON Polygon	Poly	Polygon lat/lon coordinates of each building
Damage-Classification	INT	Indication of level of damage for each building

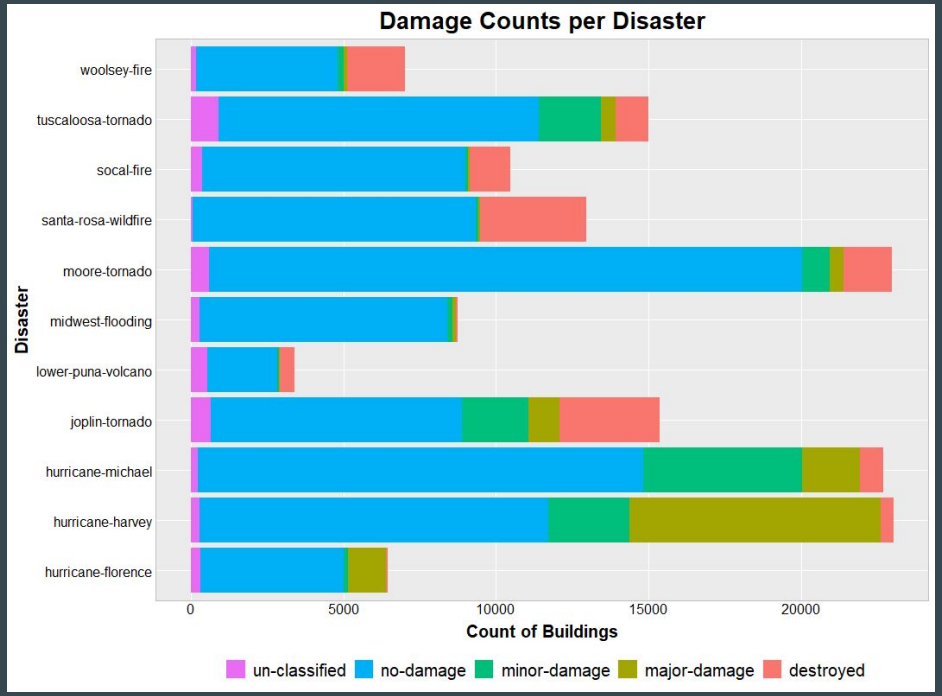


Sun azimuth and elevation angle [4]

Xview2 Dataset

Data Condition:

- Clean dataset
- No missing or incorrect values
- Buildings obsecured or newly built in an “after” image not counted
- Imbalanced by
  - disaster type
  - class - mainly un-damaged



Other Public Datasets

Zillow 2018 $/sqft:

- Dataset providing $/sqft for US zip codes
- Average cost used for missing zipcodes

US GeoID, County Tract, & Zipcode listings:

- Utilized to map building lat/lons to geoID -> countytract -> zipcode
- API & CSV files provided by US Census Bureau

NOAA Disaster Cost Evaluation Data:

- Ballpark cost of property damage for a few disasters by county

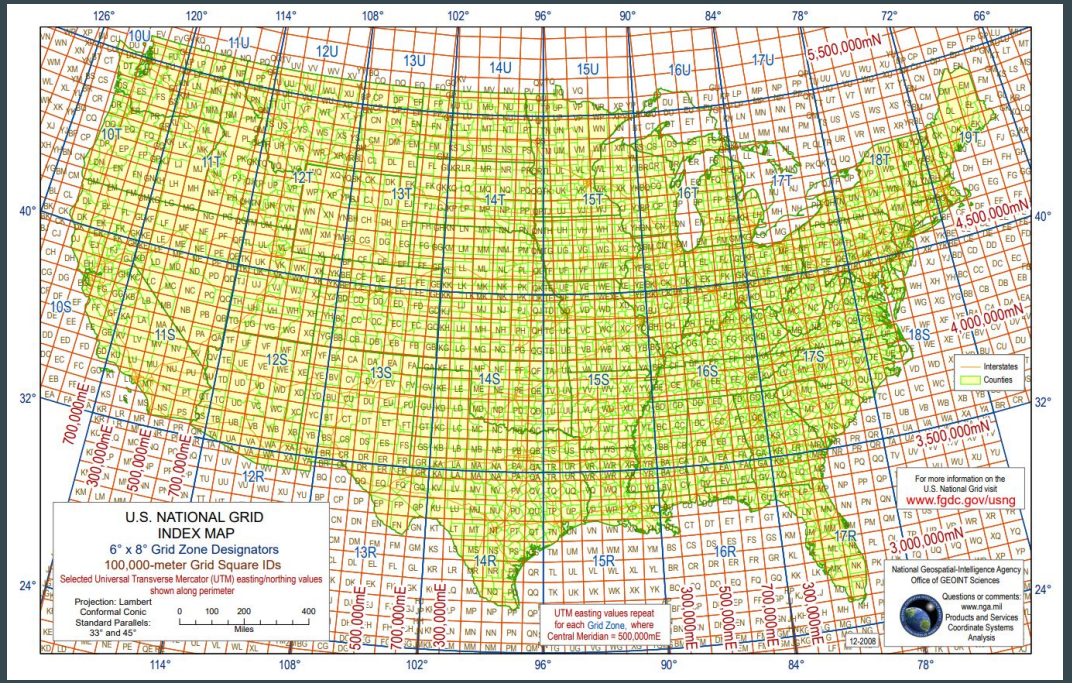
Financial Modeling

USNG Mapping Function

Financial Modeling - USNG Mapping

US National Grid Coordinates

- FEMA uses USNG coordinates for regional estimates
- Heierarchical rectuangular system which roughly follows lon/lat lines
  - 14 chars based on UTM and MGRS
- No available geocoding service for Python



[5]

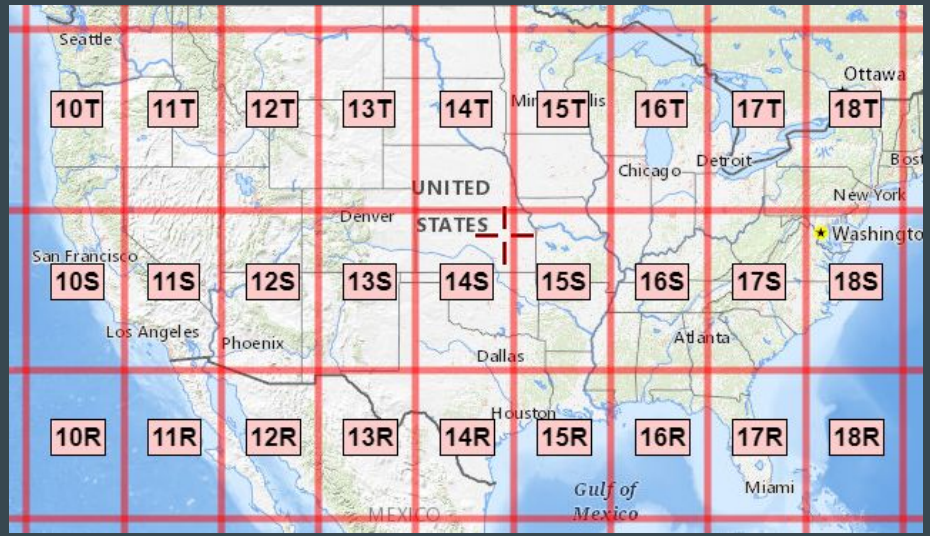
USNG coordiantes

Financial Modeling - USNG Mapping

US National Grid Coordinates

- 15S .....precision level 6° × 8°
- 15S WV .....precision level 100 km
- 15S WV 8 6 .....precision level 10 km
- 15S WV 83 68 .....precision level 1 km
- 15S WV 832 683.....precision level 100m
- 15S WV 8320 6830.....precision level 10m
- 15S WV 83200 68300 .....precision level 1m

UTM zone group main easting northing



[6]

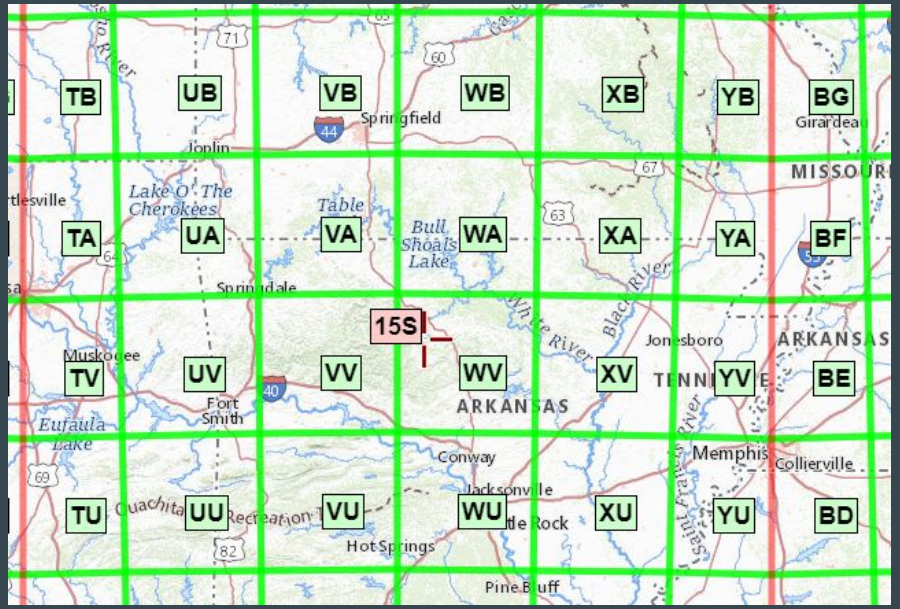
USNG coordiantes

Financial Modeling - USNG Mapping

US National Grid Coordinates

- 15S .....precision level 6° × 8°
- 15S WV .....precision level 100 km
- 15S WV 8 6 .....precision level 10 km
- 15S WV 83 68 .....precision level 1 km
- 15S WV 832 683.....precision level 100m
- 15S WV 8320 6830.....precision level 10m
- 15S WV 83200 68300 .....precision level 1m

UTM zone main group easting northing



[6]

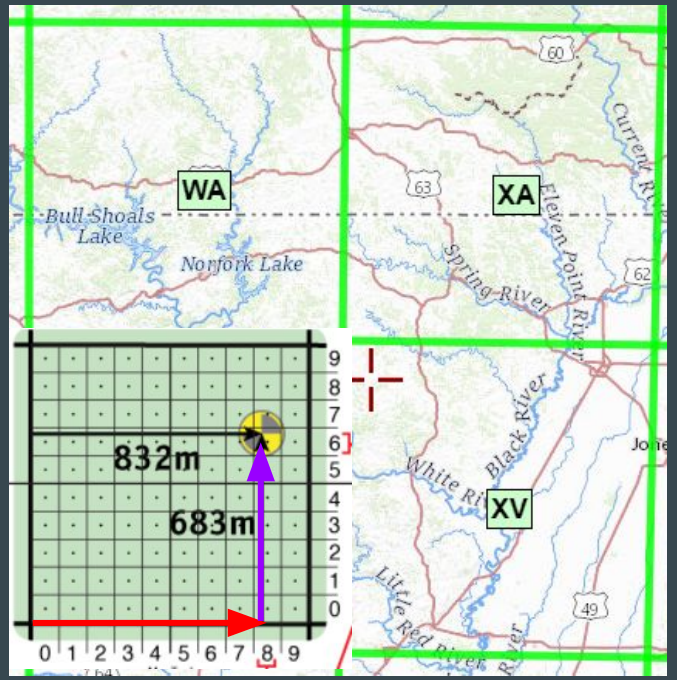
USNG coordiantes

Financial Modeling - USNG Mapping

US National Grid Coordinates

- 15S .....precision level 6° × 8°
- 15S WV .....precision level 100 km
- 15S WV 8 6 .....precision level 10 km
- 15S WV 83 68 .....precision level 1 km
- 15S WV 832 683.....precision level 100m
- 15S WV 8320 6830.....precision level 10m
- 15S WV 83200 68300 .....precision level 1m

GZD main group easting northing

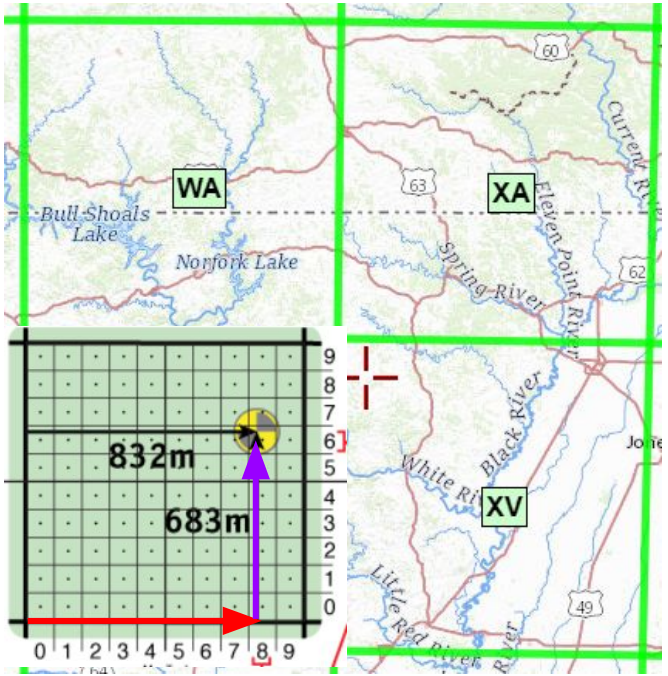


USNG coordiantes

[6]

US National Grid Coordinates

- 15S .....precision level 6° × 8°
- 15S WV .....precision level 100 km
- 15S WV 8 6 .....precision level 10 km
- 15S WV 83 68 .....precision level 1 km
- 15S WV 832 683.....precision level 100m
- 15S WV 8320 6830.....precision level 10m
- 15S WV 83200 68300
  GZD main easting northing
  group



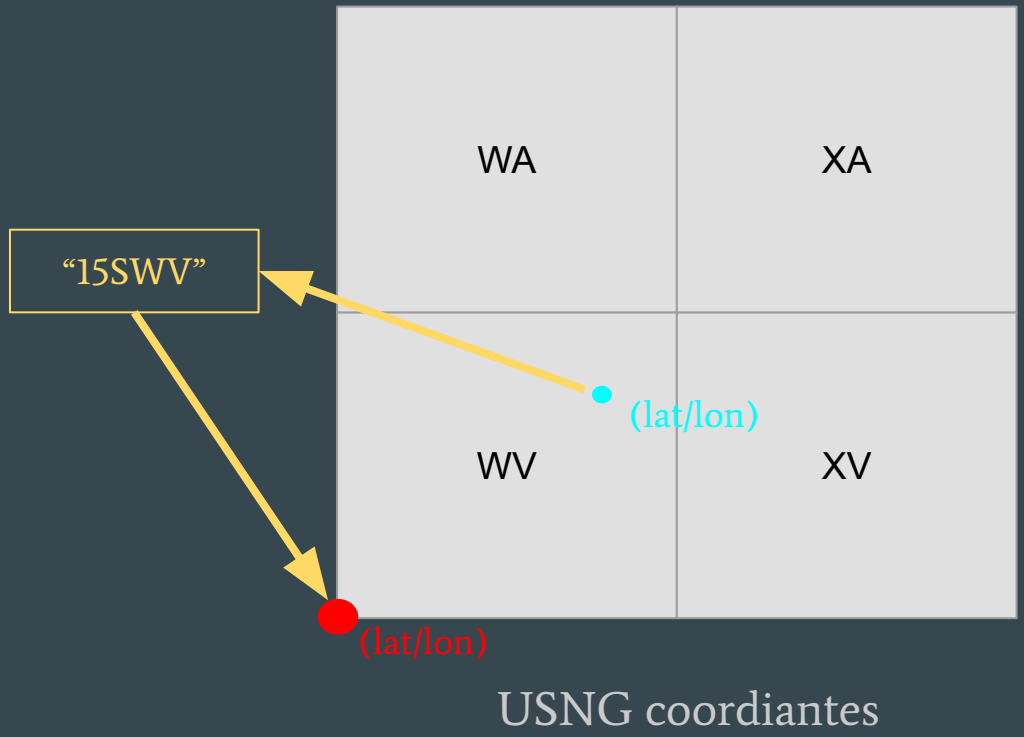
USNG coordiantes

[6]

Financial Modeling - USNG Mapping

USNG Mapping Function

- Python package mgrs to convert lat/lon to cell and vice versa
- Backwards gives cell SW corner lat/lon
- Need polygons for each grid of any precision...

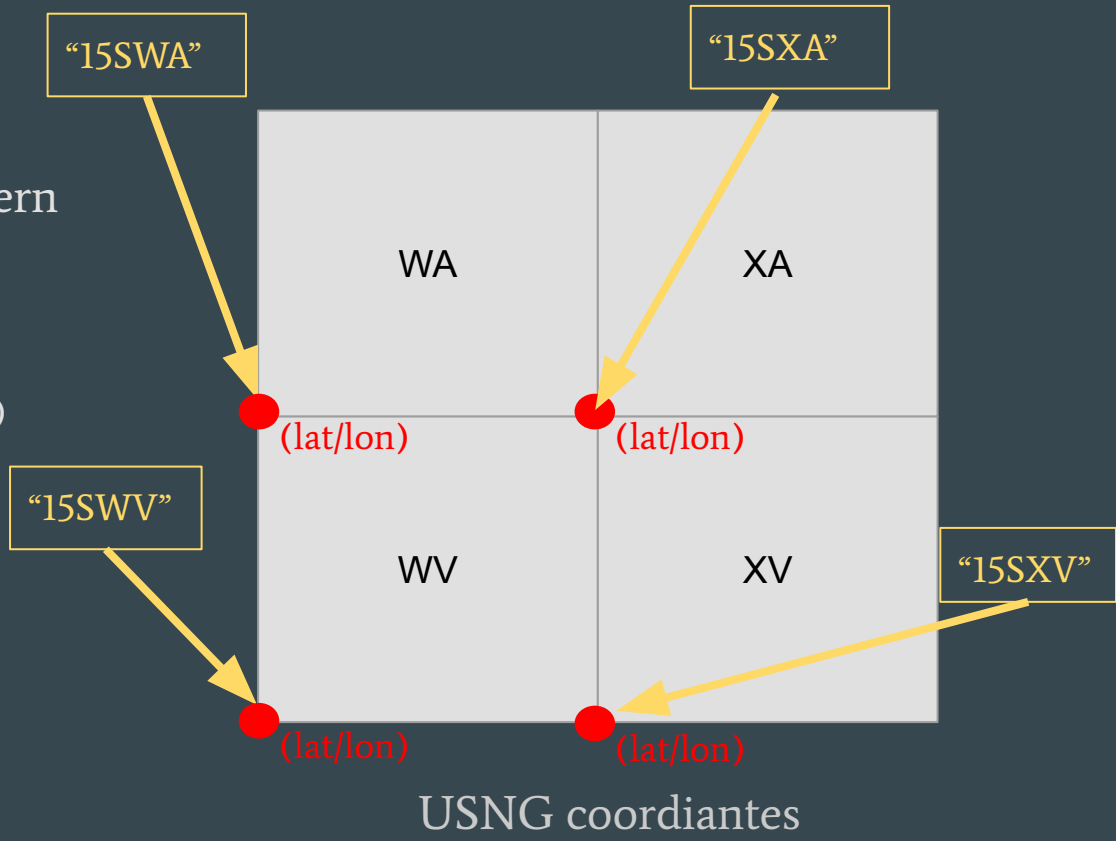


USNG coordiantes

Financial Modeling - USNG Mapping

USNG Mapping Function

- Main groups have a general pattern
  - first letter A-Z (without I and O) west-east
  - second letter A-V (without I and O) south-north
- Use 4 adjacent cells to get full polygon



USNG coordiantes

Financial Modeling - USNG Mapping

USNG Mapping Function

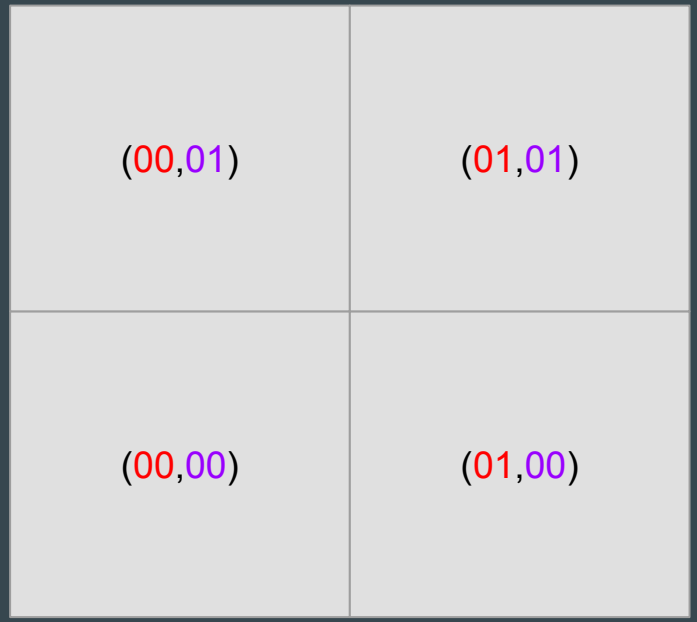
- Number are easier:
  - Add 1 to easting/northing

Ex: 15SWV 83200 50000

-> SE - 15SWV 83201 50000

-> NW - 15SWV 83200 50001

-> NE - 15SWV 83201 50001



USNG coordiantes

Financial Modeling - USNG Mapping

USNG Mapping Function

- Few edge cases:
- Ex: 15SWV 83200 99999

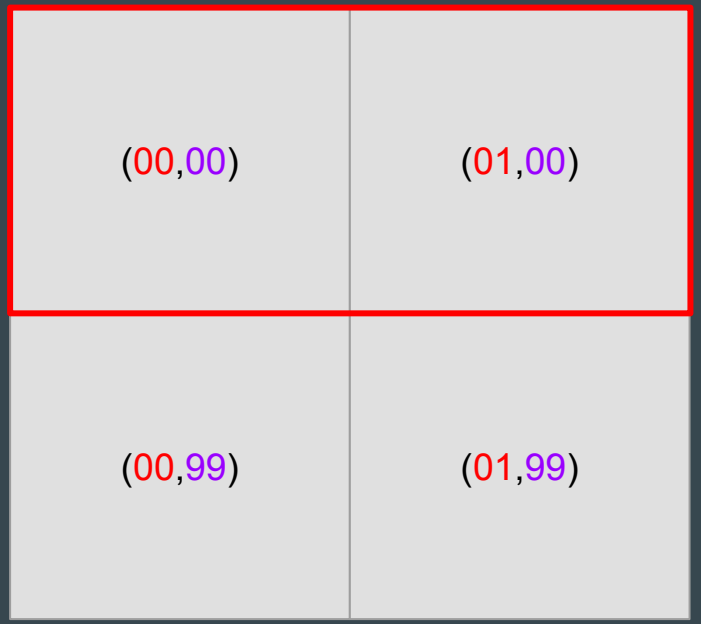
-> SE - 15SWV 83201 99999 (normal)

-> NW - 15SWA 83200 00000 (edge)

- UTM zone changes:

-> main group easting break pattern every 6° lon

-> some main groups only within certain latitudes



USNG coordiantes

Financial Modeling

Xview2 USNG Visualization

Financial Modeling - Xview2 Mapping

Financial Model

- Mapped zip-code for all xview2 data based on buildings lat/lon centroids
- Pulled Zillow data for a price/sqft for each zipcode from 2018 and averaged missing data
- Validated model against county level data obtained from NOAA for a few trusted counties (same order of magnitude accuracy)
- Proof of concept

Financial Cost Model:

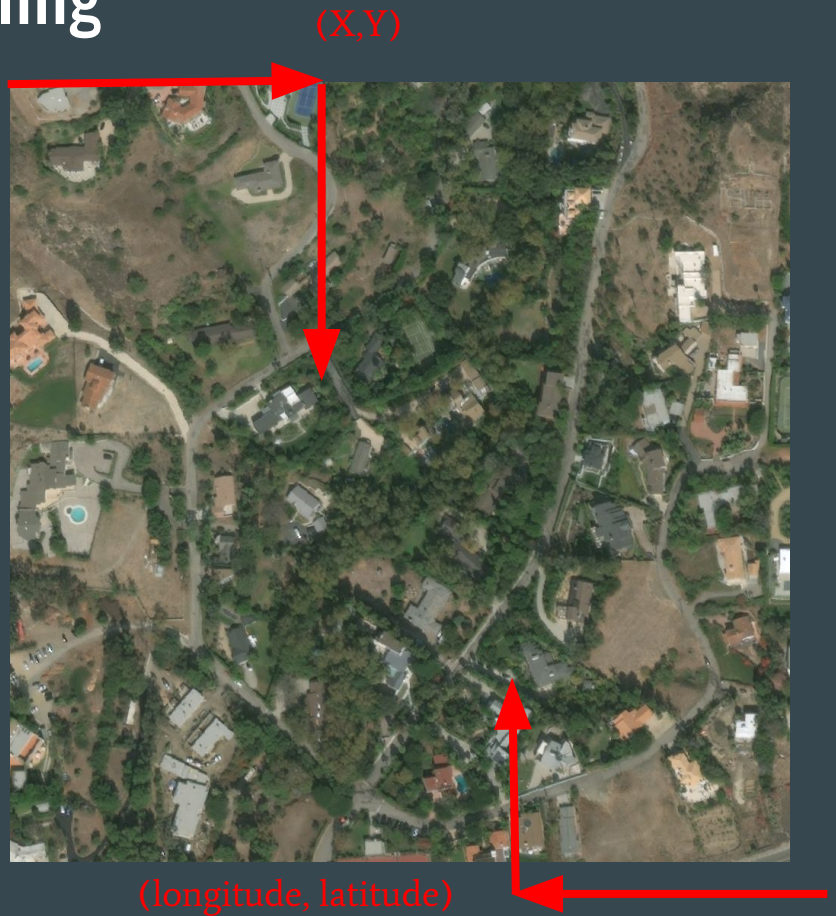
$/sqft/zipcode * sqft * 2floors * damage level factor

damage level factor	
0	no damage
0.5	minor
0.8	major
1	destroyed

Solution Models - Financial Modeling

Xview-2 Image Lat/Lon

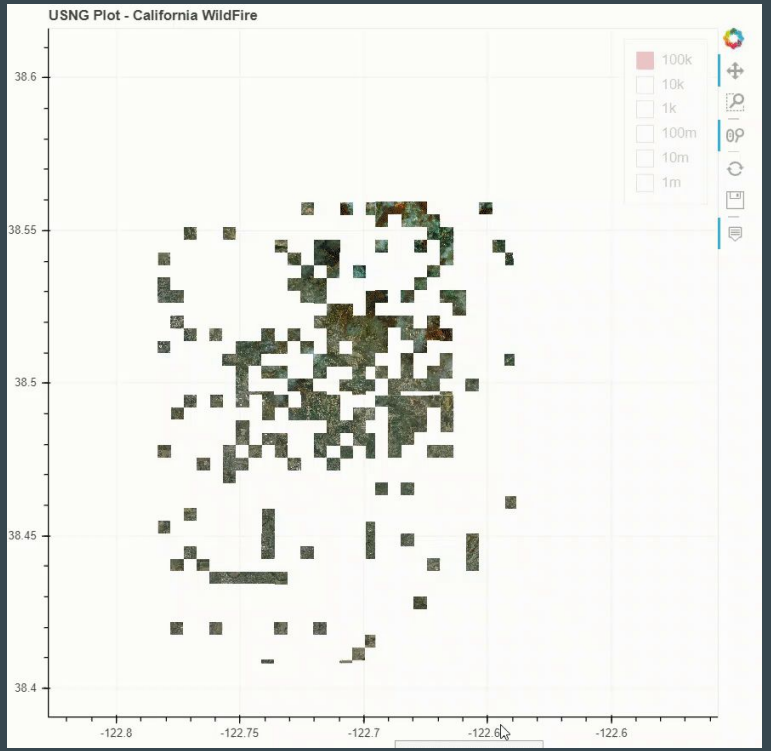
- Xview2 data provides only lon/lat for each building - not image as a whole
- lat/lon position is needed for conversion from model -> lat/lon -> USNG coords
- All xview2 (x,y) are w.r.t. NW corner
- Used building lat/lons to get
    - Δlatitude/pixel
    - Δlongitude/pixel
    - corner lon/lat



Financial Modeling - Xview2 Mapping

USNG Large Scale Visualization

- https://ermlickw.github.io/
- Uses centroids, areas, and damage labels of each xview2 building
- Financial cost summed over each UNSG grid over all precisions
- Methods used in application



Financial cost estimation using USNG coordinates

Xview2 Modeling

Connected UNets

Solution Models - Final Report

Keras Model Pipeline

- Image masks are created
  - Use polygons from json files to draw building outlines in an array
  - Result is a (1024, 1024) shaped array where each pixel is a class
  - 0: No Building, 1: No Damage, 2: Minor Damage, 3: Major Damage, 4: Destroyed
- Keras data loader is created
  - At each iteration, pre image, post image, and mask are loaded from the training data
  - Prevents all images from being loaded into memory at once
- Model is trained on these images

Solution Models - Final Report

Keras Model Pipeline continued

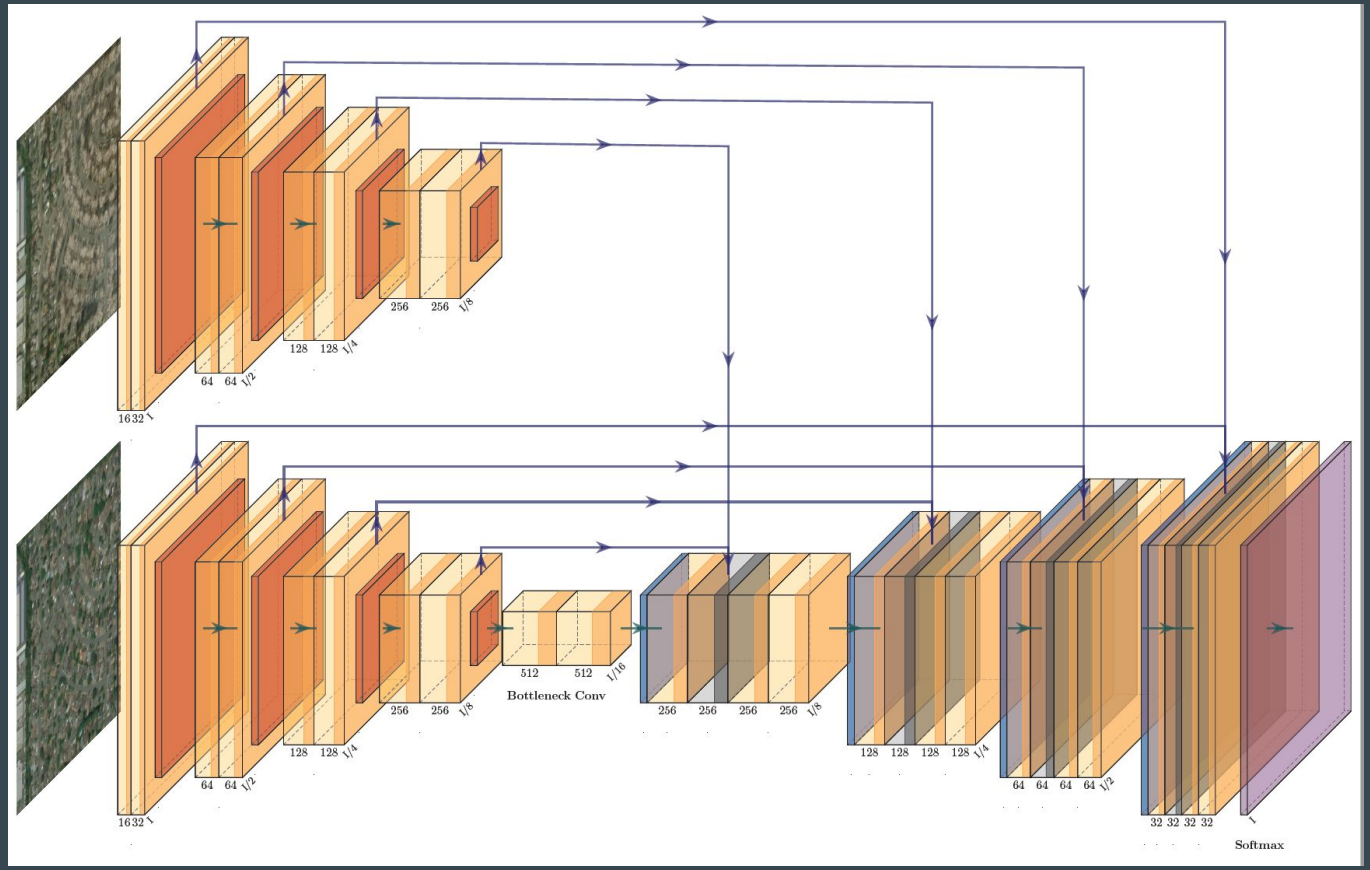
- The model is validated
  - At each epoch, a random subset of the training data was used for validation
  - Due to the limited training data, being able to use all for training was key
- Best model weights are saved
- Inference is performed
  - Iterate over test images and return pixel-by-pixel predictions for each class
  - Output is a softmax function and each array is shape (1024, 1024, num_classes)
  - Take the argmax over the classes to reshape to (1024, 1024)
  - This is fed into the metrics script, and F1 scores are returned

Solution Models - Final Report

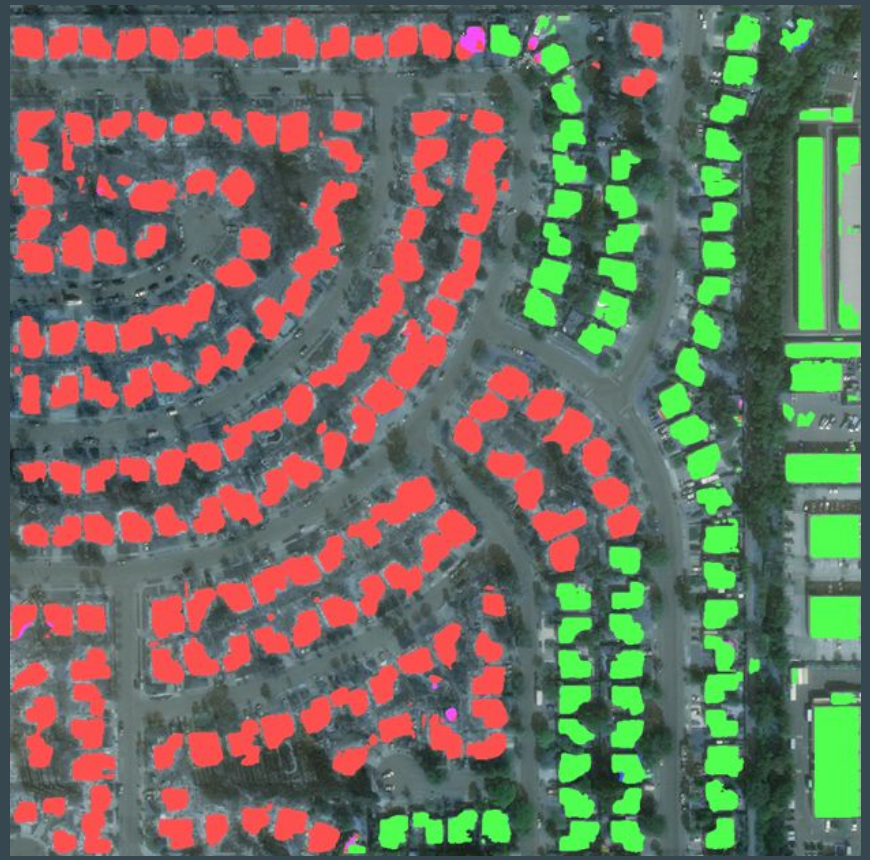
U-Net Model

- U-Net models have had success in image segmentation problems
- Created a dual-input U-Net model using Keras and TensorFlow
  - Needed a way to use both pre and post disaster images
  - Images are encoded differently, but share the same weights
  - The image differences are taken into account during the decoder stage
- Used a combination of Weighted Cross Entropy and Dice Loss
  - RMSProp loss function
  - Adjusted class weights and learning rate throughout training
  - augmentations were not effective

Solution Models - Final Report



Santa Rosa Wildfire - Post Disaster



Xview2 Modeling

Mask RCNN

Analytics & Algorithms -Localization&Damage Assessment

- Here we have used open-source library that implements Mask RCNN on top of TensorFlow/Keras.

- It is a simple model that combines two extremely powerful existing models together - Faster R-CNN and semantic segmentation. Two Main components of this model are RPN and RoIAlign Layer

    1. Region Proposal Network (RPN)is able to look at the feature map and get a set of bounding boxes that would essentially have an object of relevance by using selective search
    2. RoIAlign Layer helps with pixel-level segmentation and uses bilinear interpolation to avoid mis alignments

Analytics & Algorithms- Damage Assessment

- Mask RCNN approach for Damage Detection in keras.

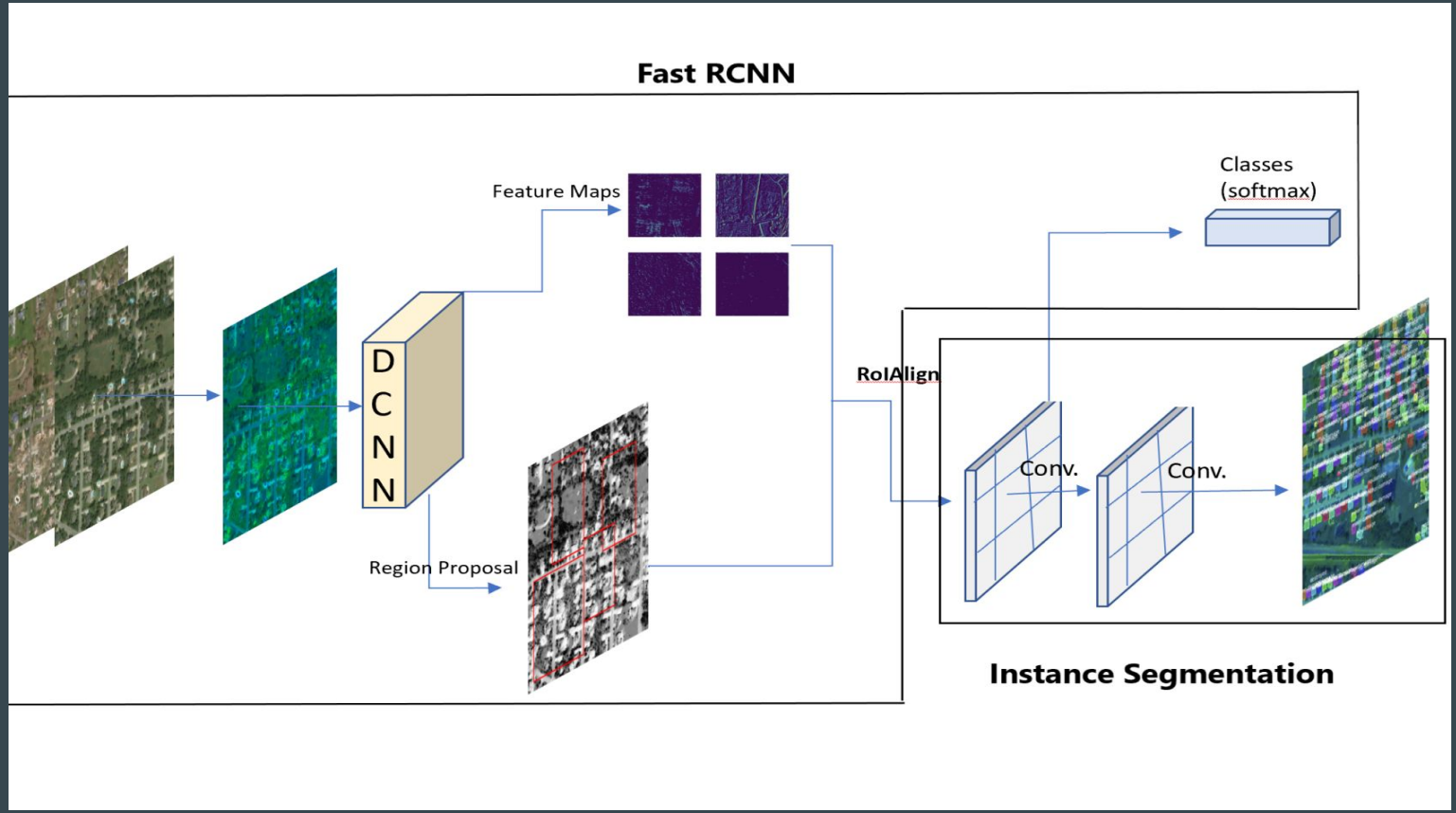
1. Combine pre - post images to get 6-channel input image.
2. Use Pretrained weights from the localization model.
3. Update Classes to - 0 : No-damage 1: Minor-damage 2: Major-damage 3: Destroyed and BG

   Supercategory: Building

3 Architecture remains same with **Hyper - Parameter Tuning**.

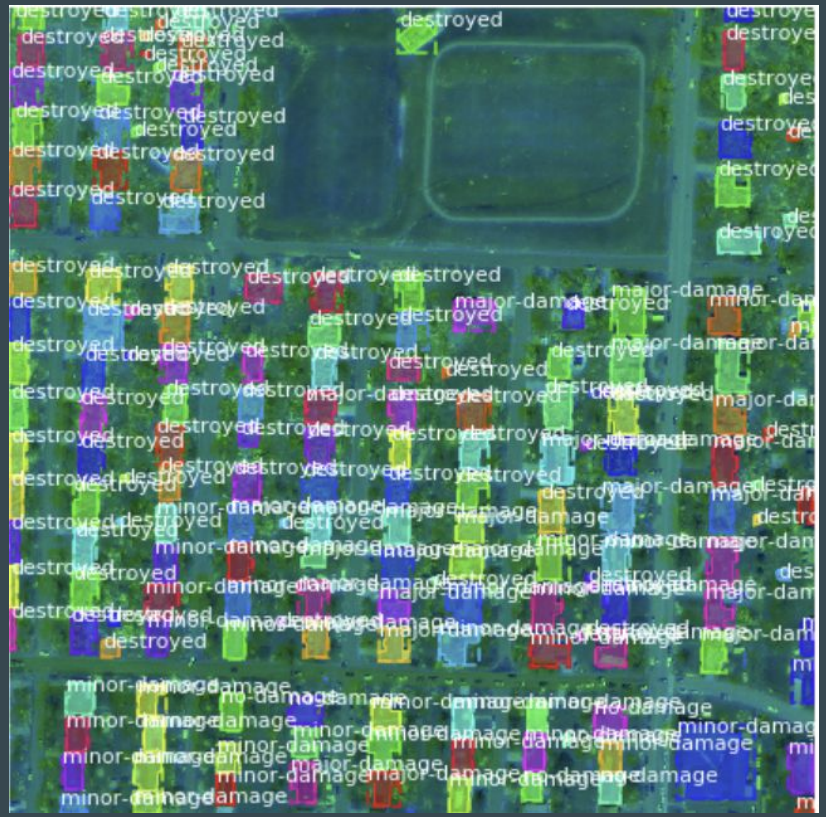
- Learning rate
- RPN anchor scale : This adjusts pixel level for detection of Region proposals.
- Detection threshold: This adjusts the detection threshold to give room to false positives when the objects detected are much smaller
- non-maximum suppression : It groups highly overlapped detection into one class to avoid duplication

Analytics & Algorithms -Damage Assessment

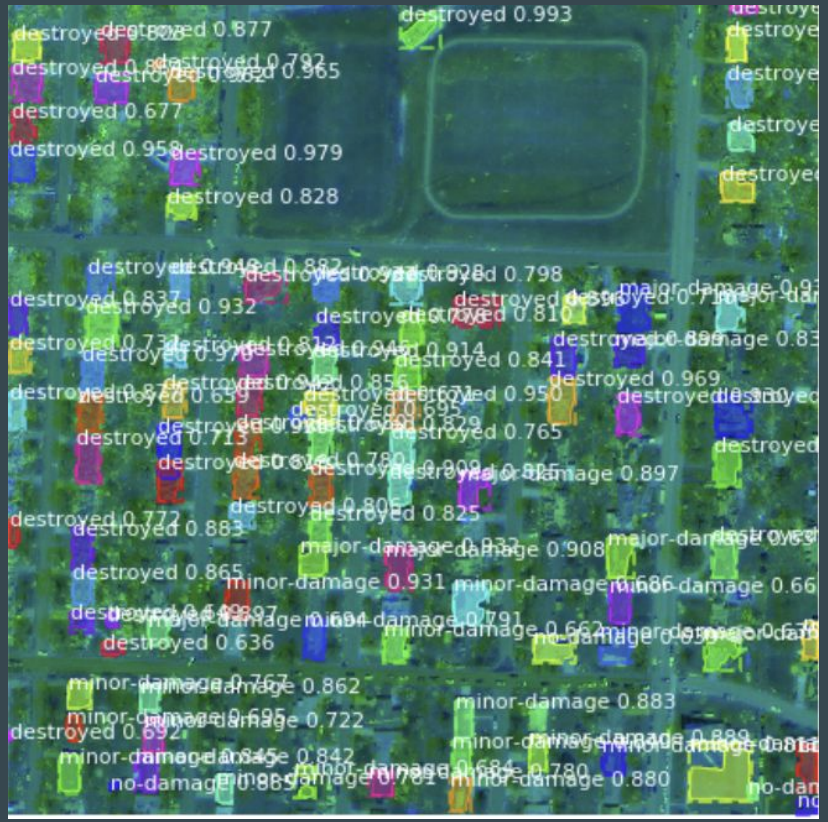


```
mAP @ IoU=50::  0.4570493460264878 3
F1:  0.588235294117647 1
```

Ground Truth



After Detection

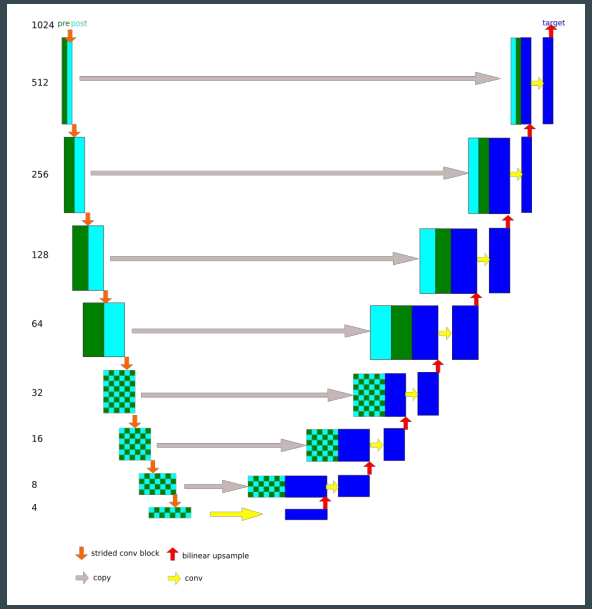


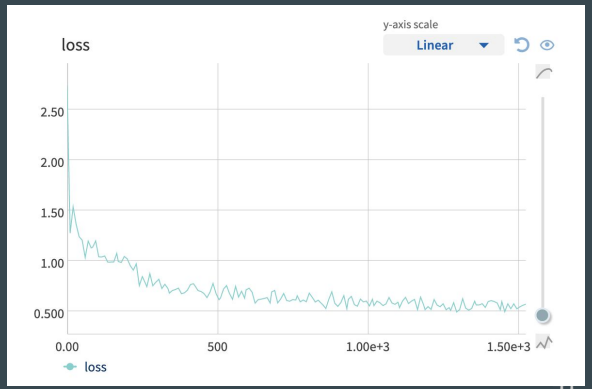
Application

Application

Modified U-Net with Paired Image Inputs

- To establish an MVP — and use inference outputs to validate the financial modeling approach — we applied two, relatively simple, open source U-Nets with near-top-50 weighted score (0.68)
  - 6–8 GPUs leveraged, both models trained overnight
  - Modified U-Nets with EfficientNet encoder blocks
- ~5 seconds for inference per image
  - Damage classification is done at the pixel level (as opposed to the building level) with the localization applied as a “mask”



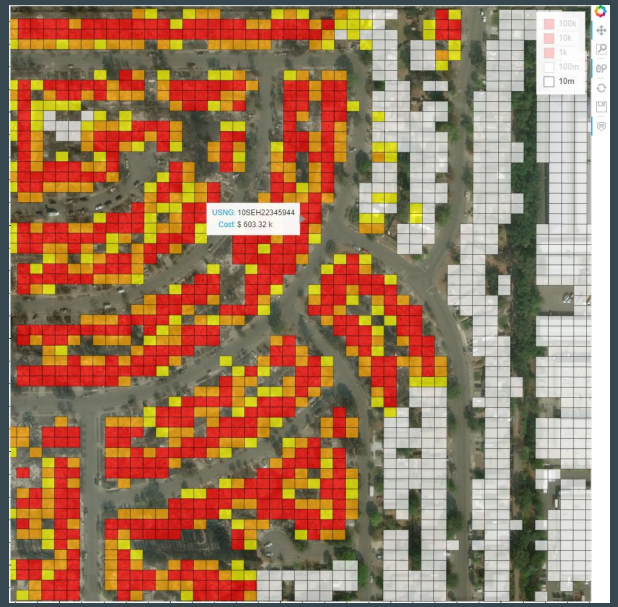


Application

Overview

- User inputs
  - before and after image
  - lat/lon of NW and SE corners

Model



Application

Workflow

inputs

- Before and after image
- NE and SW corner lat/lons

methods (should be ~15 seconds)

- run building localization and damage assessment and output pixel map (5~sec)
- convert building masks to polygons
- divide polygons into smaller polygons (to ensure all USNG grids are represented)
- find centroids of polygons in lat/lon
- find MGRS grids coords for each centroid lat/lon using our in-house function
- convert MGRS grid coords to image coordiantes
- sum pixel values within each MGRS grid multiplied by sqft/pixel and $/sqft based on zip code

outputs

- display image with USNG overlaid with aggregate cost

Application

Methods

- User inputs
  - before and after image
  - lat/lon of NW and SE corners

Damage Classification Model

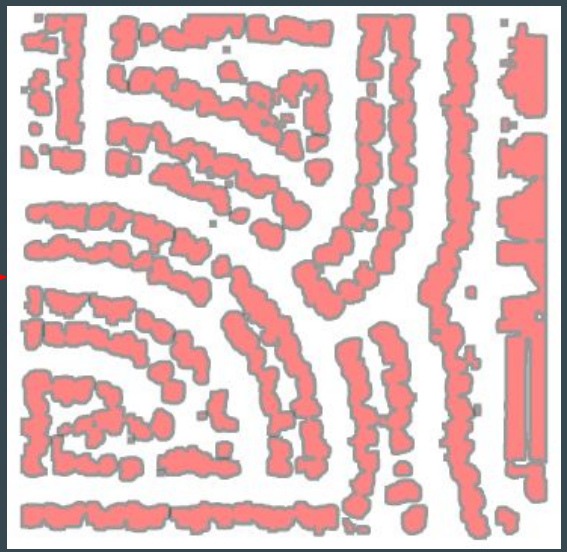
Application

Polygon Conversion and Image lat/lon Transform

- Determine building polygons using contour function
- Convert pixel coordiantes to lat/lon
  - Δlat/pix and Δlon/pix

translate
&
scale

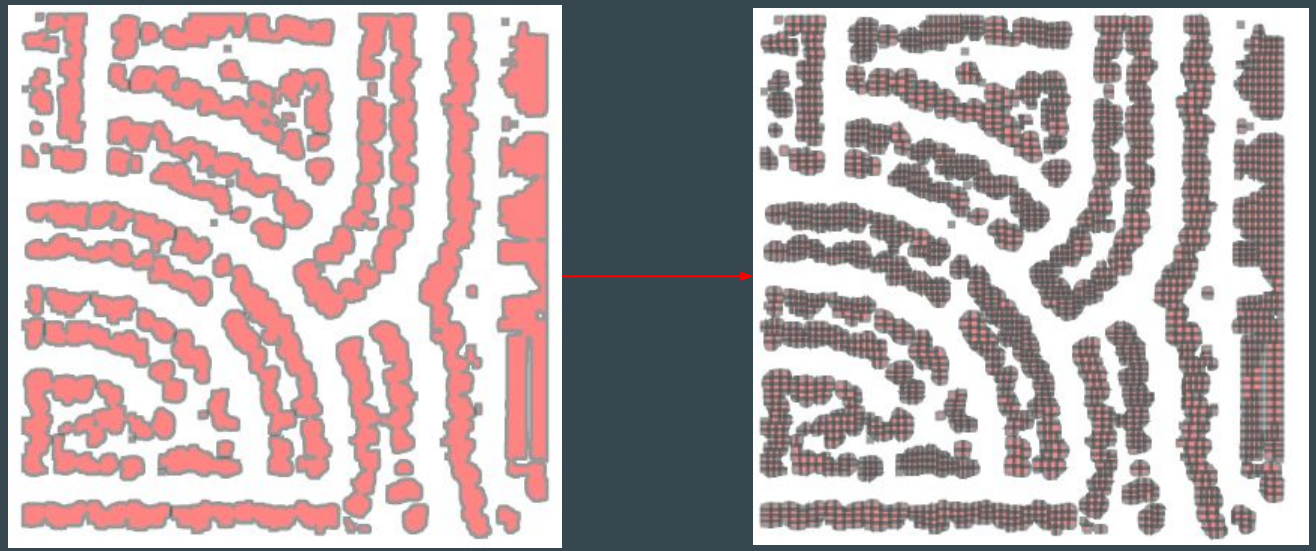
Damage Classification Model



Application

Polygon Division

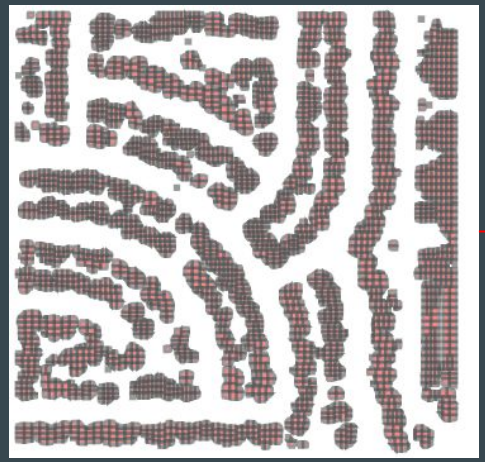
- Divide large polygons (model incorrectly combines buildings)
- Determine centroid of each polygon (USNG all accounted for)

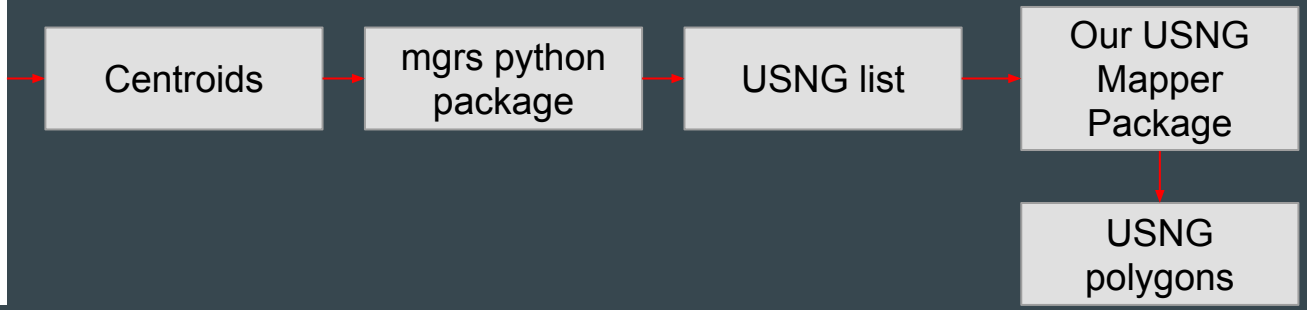


Application

Get USNG polygons

- Determine list of unique USNG grids from centroids
- Get polygons of USNG grids using our in house function (all precisions)

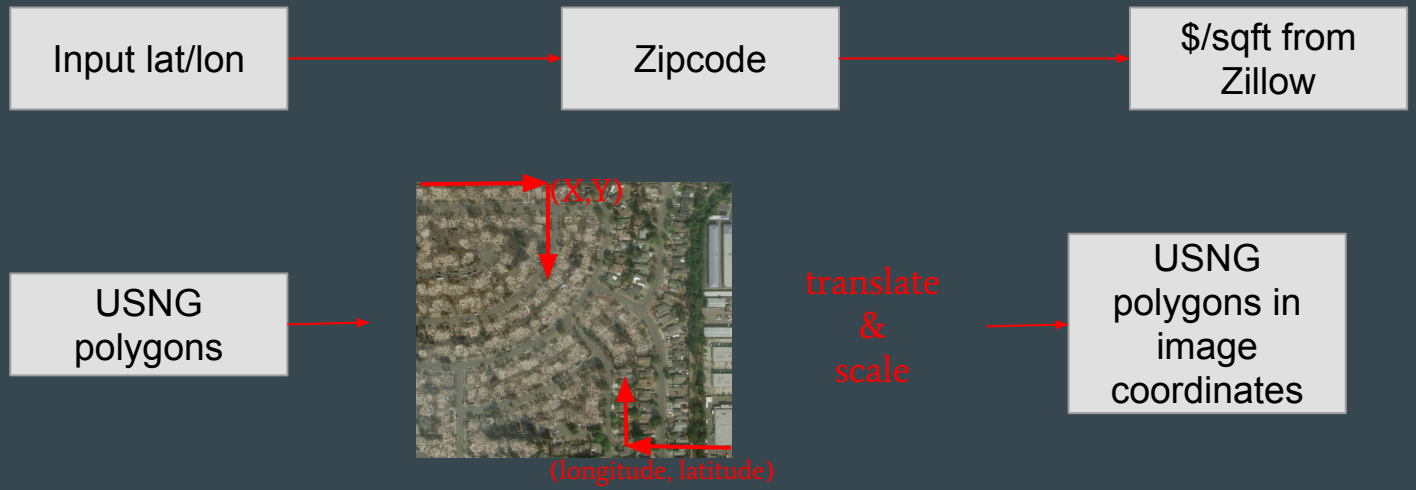




Application

Get cost for each USNG

- Get zipcode from lat/lon and $/sqft from Zillow data
- Backconvert USNG lat/lons to image (x,y) coordinates

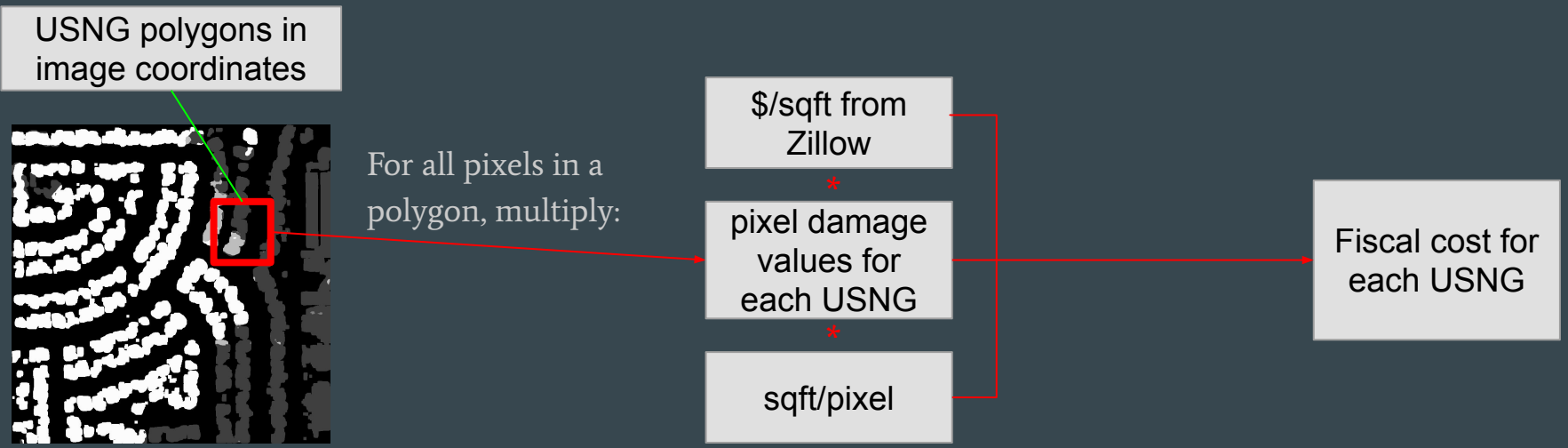


Application

Get cost for each USNG

- Find sqft per pixel - area of image / resolution
- Change pixel values to financial model damage levels

damage level factor	
0	no damage
0.5	minor
0.8	major
1	destroyed



Application

Financial Model -- Flask Application Pipeline

- Around 5 seconds for pixel level building localization & damage classificaiton
- Less than 15 seconds for financial analysis
- Accurate financial estimates to 10 meter precision (1m is possible but computationally expensive and GSD is >1m)

Contibutions

- ✔ USNG Mapping Function
- ✔ Connected U-Net Model
- ✔ Mask R-CNN Model
- ✔ Proof of Concept Application http://test-env.eba-6qua3x4j.us-east-1.elasticbeanstalk.com/

Lessons Learned

- Large images and dense networks are not a good combination
  - The computation power needed to run models was significant
  - Multiprocessing on GPUs didn't always offer significant speed gains
  - Used vectorization where possible, to improve performance
- The U-Net struggled on classifying minor and major damage
  - There is not a clear delineation between the classes
  - The classes are significantly imbalanced
  - Oversampled minor and major damage
  - Gave these classes more weight with the loss function
- Mask RCNN is not a ideal model for satellite images due to the sheer detailed objects in the images
  - It is still a good model for experimenting with dataset and with minor tweaks in the architecture can get better at detecting satellite image objects due to its simplicity.

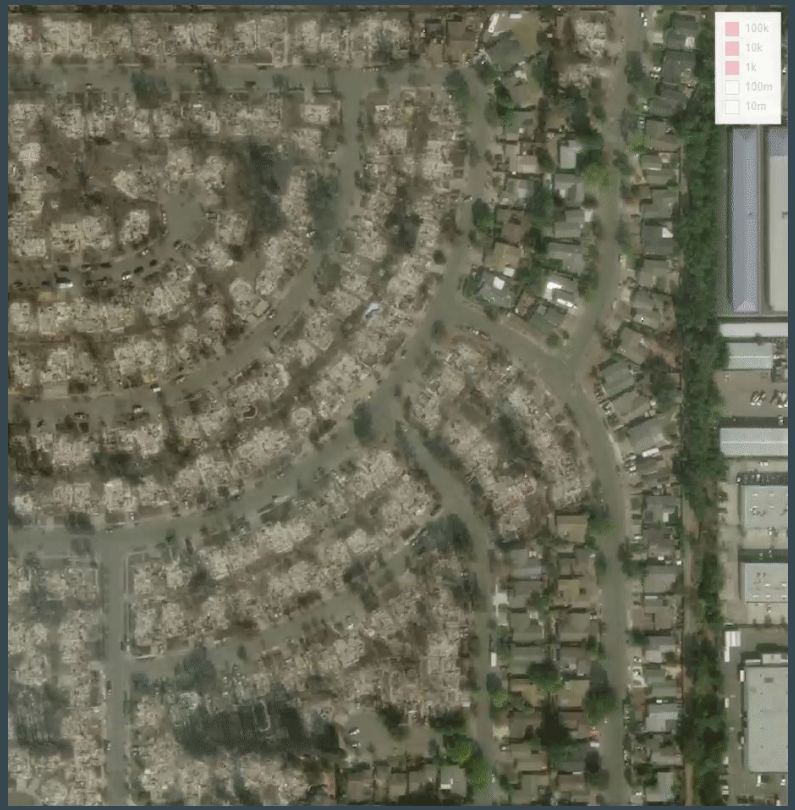
Example Flooding Image

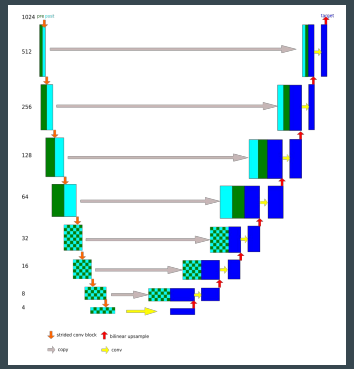
- Some of the flooded buildings are classified as major damage and some are classified as minor damage
  - Green = No Damage
  - Blue = Minor
  - Purple = Major
- Hard for humans to tell the difference

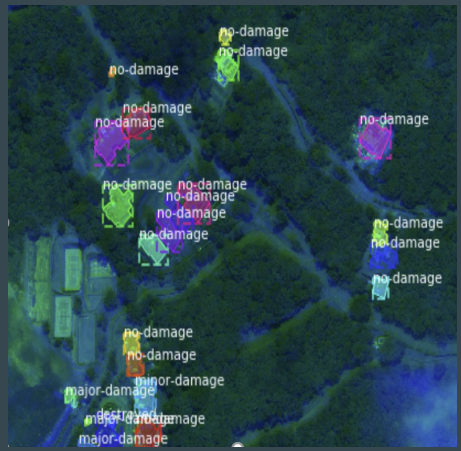
Future Work

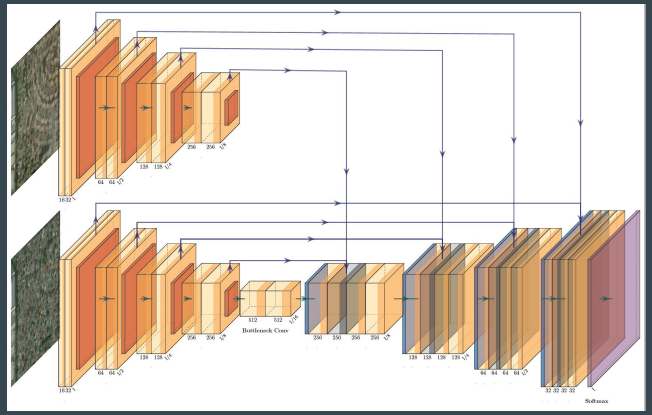
- More accurate pixel-based financial model if claims data can be obtained
- Flooding around buildings was difficult to classify
  - Hard to see on the images for humans
  - Model had the most trouble classifying these instances
- Experiment with different U-Net encoder blocks
  - VGG, Inception, ResNet, etc.
- Use Quadratic Weighted Kappa as a loss function
  - The model blends together some predicitons (i.e. no damage and major)
  - Places higher penalty on classes that are further away from the actual
- Improve Mask RCNN further by combining pre post images during segmentation phase and implement weighted cross entropy loss.

Questions


100k
10k
1k
100m
10m


1024
512
256
128
64
32
16
8
4


no-damage
no-damage
no-damage
no-damage
no-damage
no-damage
no-damage
no-damage
no-damage
no-damage
no-damage
no-damage
no-damage
no-damage
no-damage
no-damage
minor-damage
major-damage
major-damage



References

1) https://www.fema.gov/news-release/2017/09/07/fact-sheet-what-expect-when-you-register-fema-disaster-assistance
2) https://www.fema.gov/pdf/library/remotes.pdf
3) https://xview2.org/ & https://arxiv.org/pdf/1911.09296.pdf
4) https://keisan.casio.com/exec/system/1224682277
5) https://legallandconverter.com/p50.html
6) https://mappingsupport.com/p2/coordinates-usng-gissurfer-maps.html

APPENDIX SLIDES

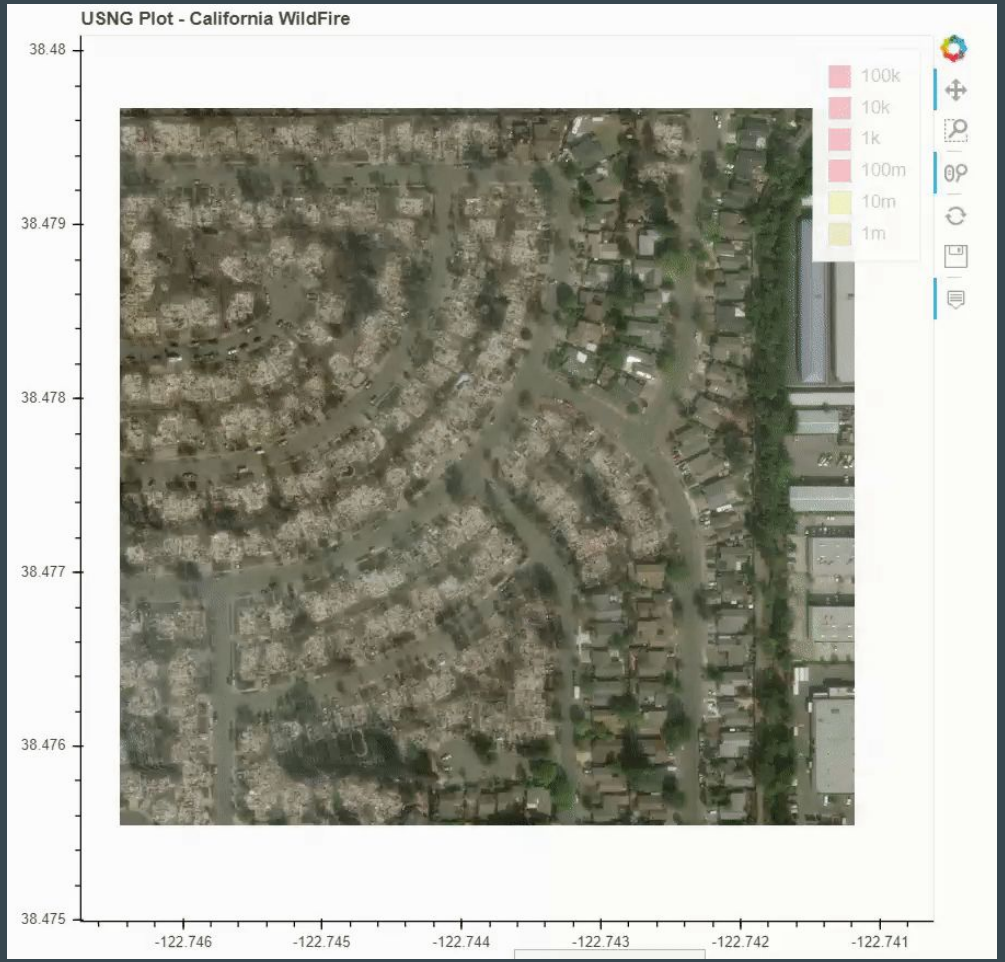
APPENDIX SLIDES

APPENDIX SLIDES

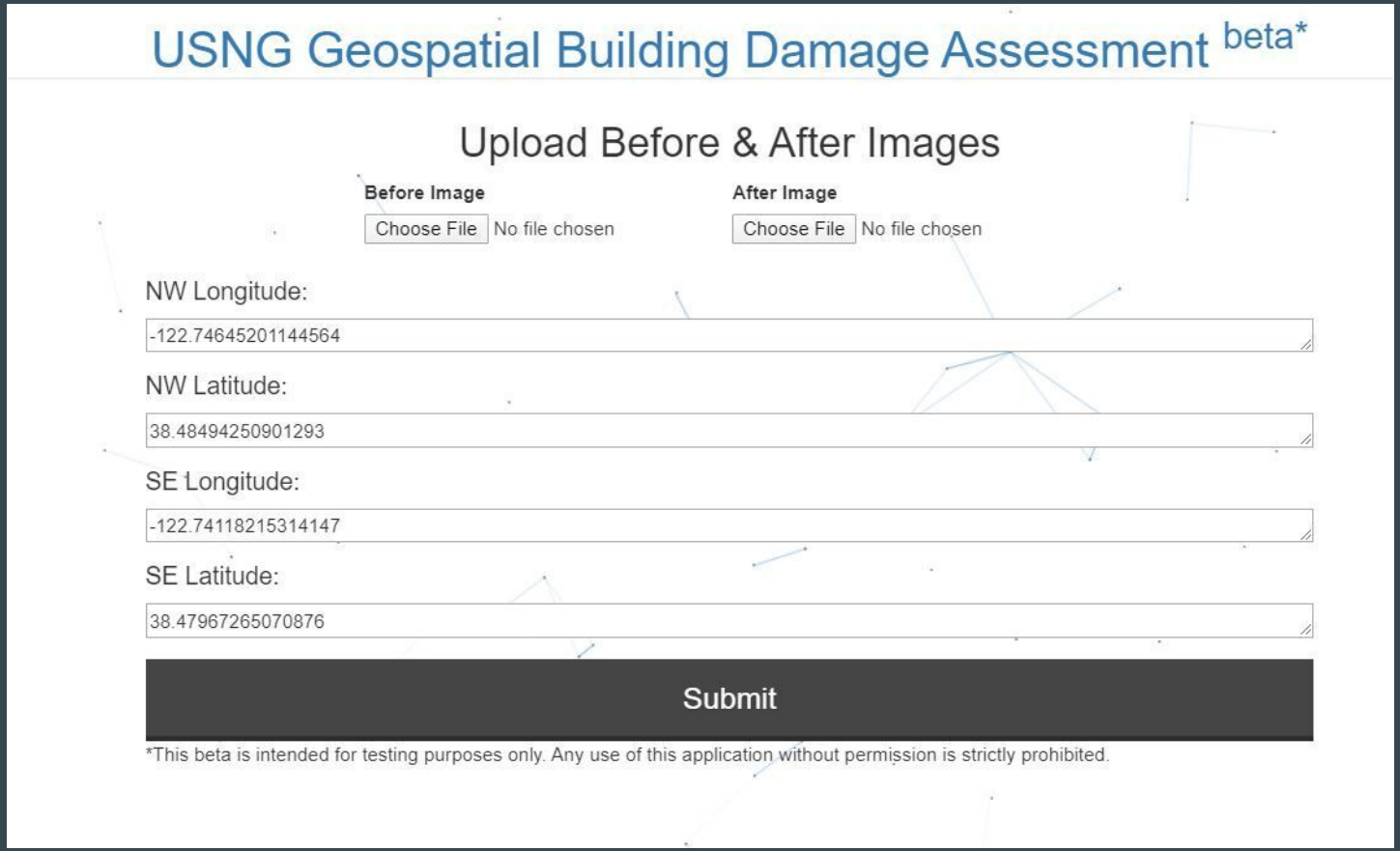
Application

Financial Model -- Flask Application Pipeline

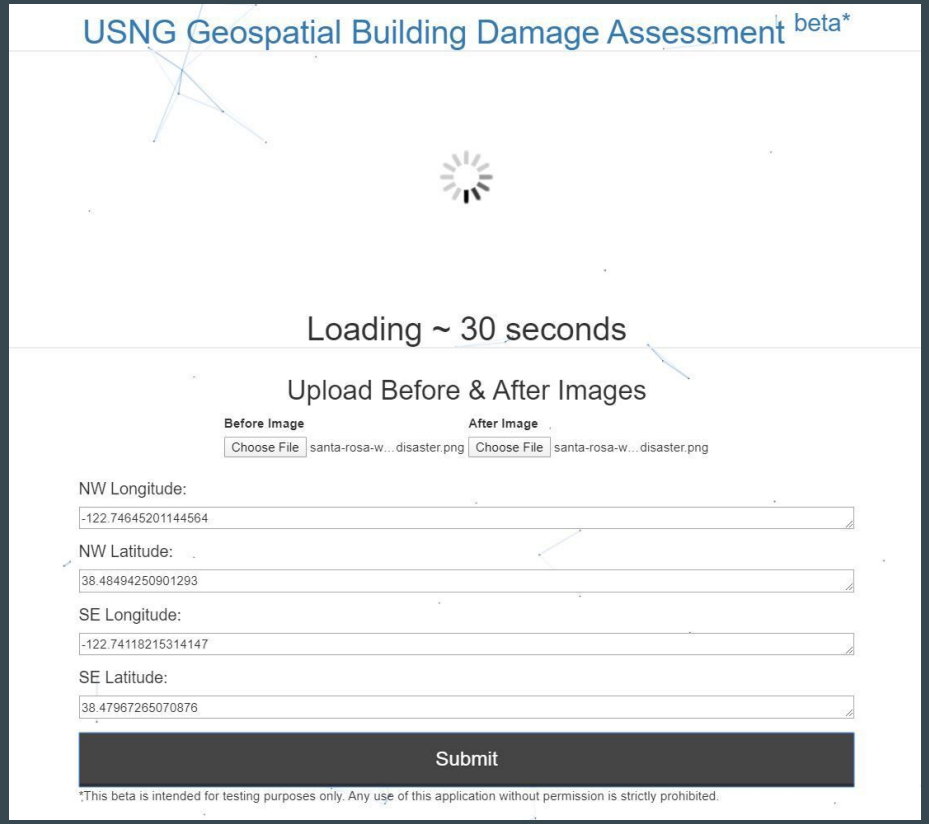
- Around 5 seconds for pixel level building localization & damage classificaiton
- ~7 seconds for financial analysis
- Accurate financial estimates to 10 meter precision (1m is possible but computationally expensive and GSD is >1m)
- http://test-env.eba-6qua3x4j.us-east-1.elasticbeanstalk.com/



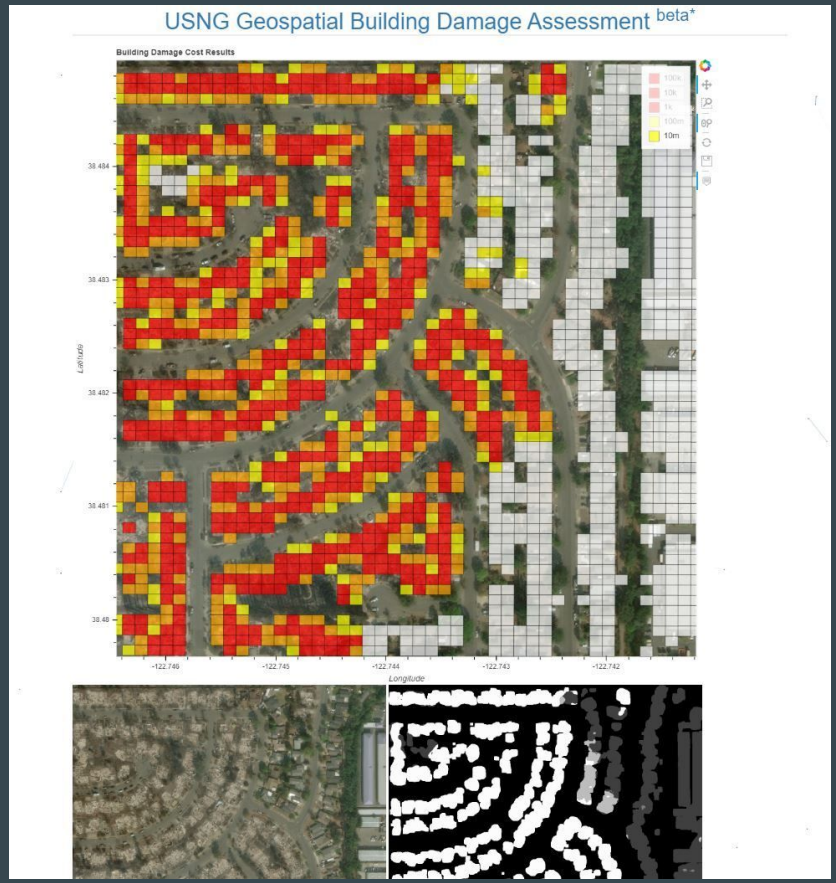
Application



Application



Application



Solution Models - Final Report

Keras Model Pipeline

- Image masks are created
  - Use polygons from json files to draw building outlines in an array
  - Result is a (1024, 1024) shaped array where each pixel is a class
  - 0: No Building, 1: No Damage, 2: Minor Damage, 3: Major Damage, 4: Destroyed
- Keras data loader is created
  - At each iteration, pre image, post image, and mask are loaded from the training data
  - Prevents all images from being loaded into memory at once
- Model is trained on these images

Solution Models - Final Report

Keras Model Pipeline continued

- The model is validated
  - At each epoch, a random subset of the training data was used for validation
  - Due to the limited training data, being able to use all for training was key
- Best model weights are saved
- Inference is performed
  - Iterate over test images and return pixel-by-pixel predictions for each class
  - Output is a softmax function and each array is shape (1024, 1024, num_classes)
  - Take the argmax over the classes to reshape to (1024, 1024z0
  - This is fed into the metrics script, and F1 scores are returned

Solution Models - Final Report

Baseline Model

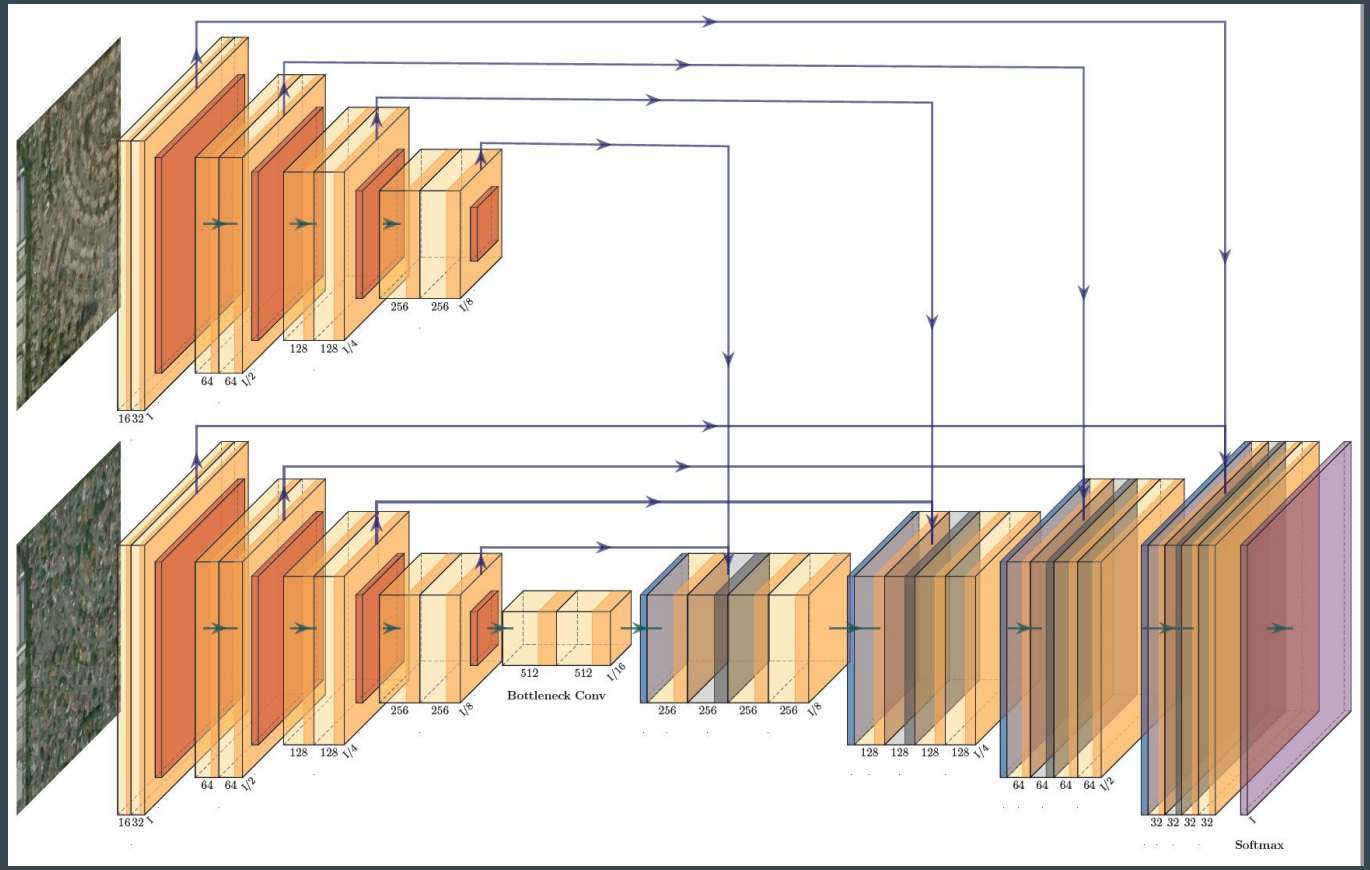
- Used a baseline model provided with the competition
  - Localization: spacenet building segmentation model
  - Classification: simple CNN architecture
- Results were effective for localization, but poor for classification
  - Method was also too slow to implement in an application
- Needed a model that simulataneously performs classification and prediction

Solution Models - Final Report

U-Net Model

- U-Net models have had success in image segmentation problems
- Created a dual-input U-Net model
  - Needed a way to use both pre and post disaster images
  - Images are encoded differently, but share the same weights
  - The image differences are taken into account during the decoder stage
- Used a combination of Weighted Cross Entropy and Dice Loss
  - RMSProp loss function
  - Adjusted class weights and learning rate throughout training
  - augmentations were not effective

Solution Models - Final Report



Lessons Learned

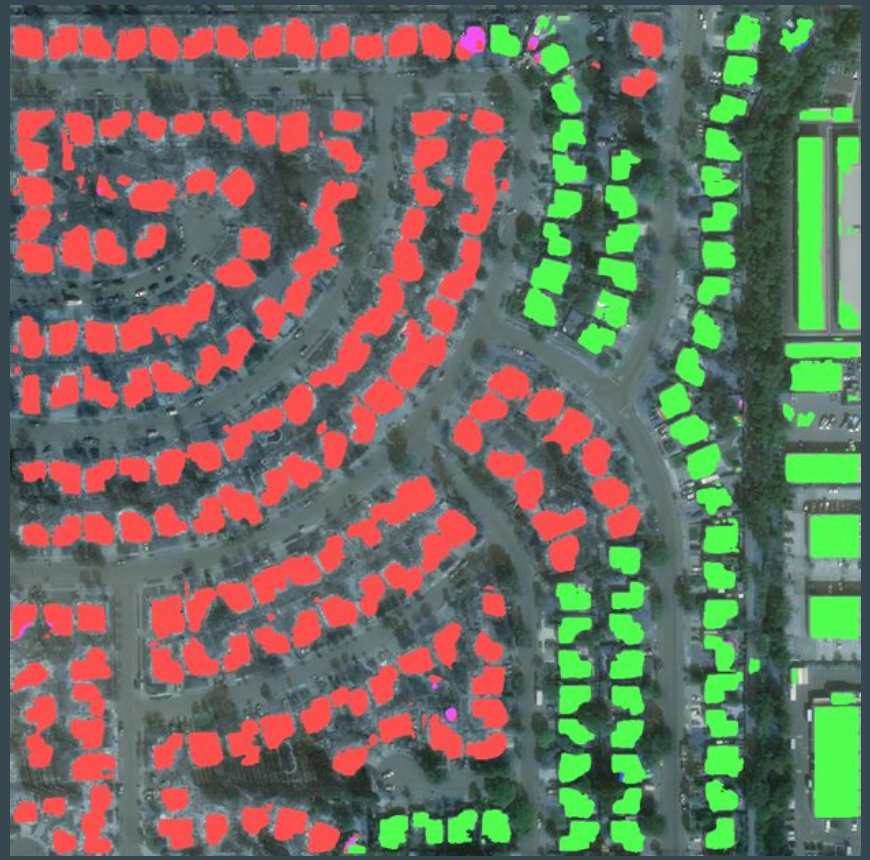
- Large images and dense networks are not a good combination
  - The computation power needed to run models was significant
  - Multiprocessing on GPUs didn't always offer significant speed gains
  - Used vectorization where possible, to improve performance
- The model struggled on classifying minor and major damage
  - There is not a clear delineation between the classes
  - The classes are significantly imbalanaced
  - Oversampled minor and major damage
  - Gave these classes more weight with the loss function

Disaster Level	Structure Description
0 (No Damage)	Undisturbed. No sign of water, structural or shingle damage, or burn marks.
1 (Minor Damage)	Building partially burnt, water surrounding structure, volcanic flow nearby, roof elements missing, or visible cracks.
2 (Major Damage)	Partial wall or roof collapse, encroaching volcanic flow, or surrounded by water/mud.
3 (Destroyed)	Scorched, completely collapsed, partially/ completely covered with water/mud, or otherwise no longer present.

Future Work

- Flooding around buildings was difficult to classify
  - Hard to see on the images for humans
  - Model had the most trouble classifying these instances
- Experiment with different U-Net encoder blocks
  - VGG, Inception, ResNet, etc.
- Use Quadratic Weighted Kappa as a loss function
  - The model blends together some predicitons (i.e. no damage and major)
  - Places higher penalty on classes that are further away from the actual

Santa Rosa Wildfire - Post Disaster



Solution Models - Final Report

Problem areas that were solved

- Large images and dense networks are not a good combination
  - The computation power needed to run models was significant
  - Multiprocessing on GPUs didn't always offer significant speed gains
  - Used vectorization where possible, to improve performance
- The model struggled on classifying minor and major damage
  - There is not a clear delineation between the classes
  - The classes are significantly imbalanaced
  - Oversampled minor and major damage
  - Gave these classes more weight with the loss function

Score	Label	Visual Description of the Structure
0	No damage	Undisturbed. No sign of water, structural damage, shingle damage, or burn marks.
1	Minor damage	Building partially burnt, water surrounding the structure, volcanic flow nearby, roof elements missing, or visible cracks.
2	Major damage	Partial wall or roof collapse, encroaching volcanic flow, or the structure is surrounded by water or mud.
3	Destroyed	Structure is scorched, completely collapsed, partially or completely covered with water or mud, or no longer present.

Solution Models - Final Report

Areas to focus on - if given more time

- Flooding around buildings was difficult to classify
  - Hard to see on the images for humans
  - Inconsistency in classification, minor or major
  - Model had the most trouble classifying these instances
- Experiment with different U-Net encoder blocks
  - VGG, Inception, ResNet, etc.
- Use Quadratic Weighted Kappa as a loss function
  - The model blends together some predicitons (i.e. no damage and major)
  - Places higher penalty on classes that are further away from the actual

Solution Models - Updates

Financial Model

- Will need to integrate model output into financial model
  - identify buildings as polygons
  - squarefootage
  - centroids
  - lat/lon conversion
  - MGRS grids cost summation - could be building level summation or pixel level

Solution Models - Updates

Financial Data

- validated results with county level data

updated model is more accurate (same order of magnitude as approximations)

$/sqft/zipcode * sqft * 2floors * damage level factor

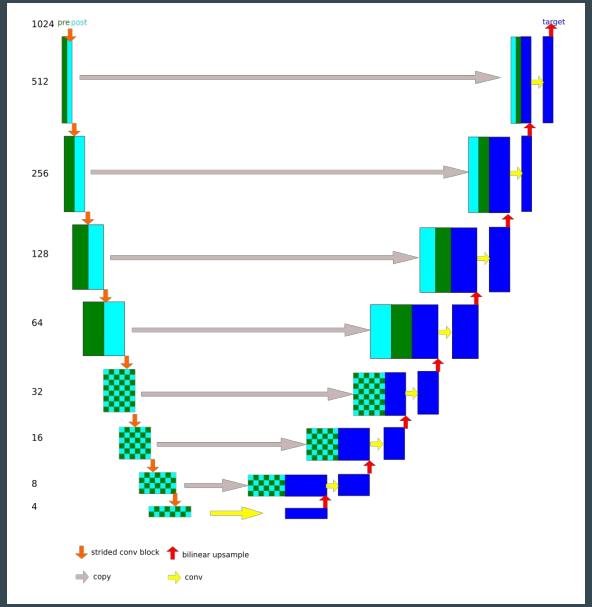
where damage level factor

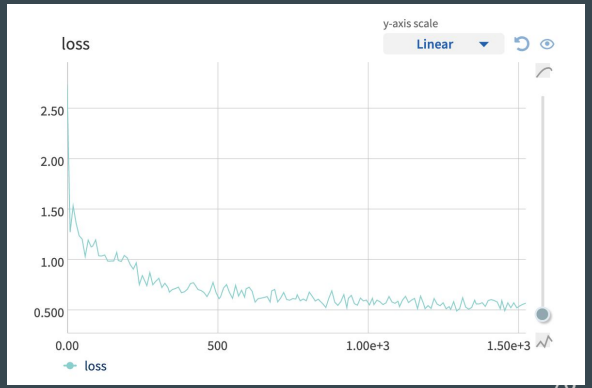
= 0 for no damage, 0.5 minor, 0.8 major, 1 destroyed

Solution Models - Updates

Modified U-Net with Paired Image Inputs

- To establish an MVP — and use inference outputs to validate the financial modeling approach — we applied two, relatively simple, open source U-Nets with near-top-50 weighted score (0.68)
  - 6–8 GPUs leveraged, both models trained overnight
  - Modified U-Nets with EfficientNet encoder blocks
- ~5 seconds for inference per image
  - Damage classification is done at the pixel level (as opposed to the building level) with the localization applied as a “mask”





Solution Models - Updates

Financial Data - next steps

- Will compare these values with our obtained county level values and see how accurate they are and what needs to be adjusted

Solution Models - Updates

Xview-2 lat/lon mapping

- Xview2 data provides lon/lat for each building but not image as a whole
- This lat/lon position is needed for mapping and also for conversion from model -> lat/lon -> USNG coords
- We found an algorithm to determine lat/lon of each xview2 image for mapping
- We will use this for final flask app as well

Solution Models - Updates

Financial Data

- Mapped zip-code for all xview2 data
- Pulled zillow data for a price/sqft for each zipcode

financial cost model is currently

$/sqft/zipcode * sqft * damage level factor

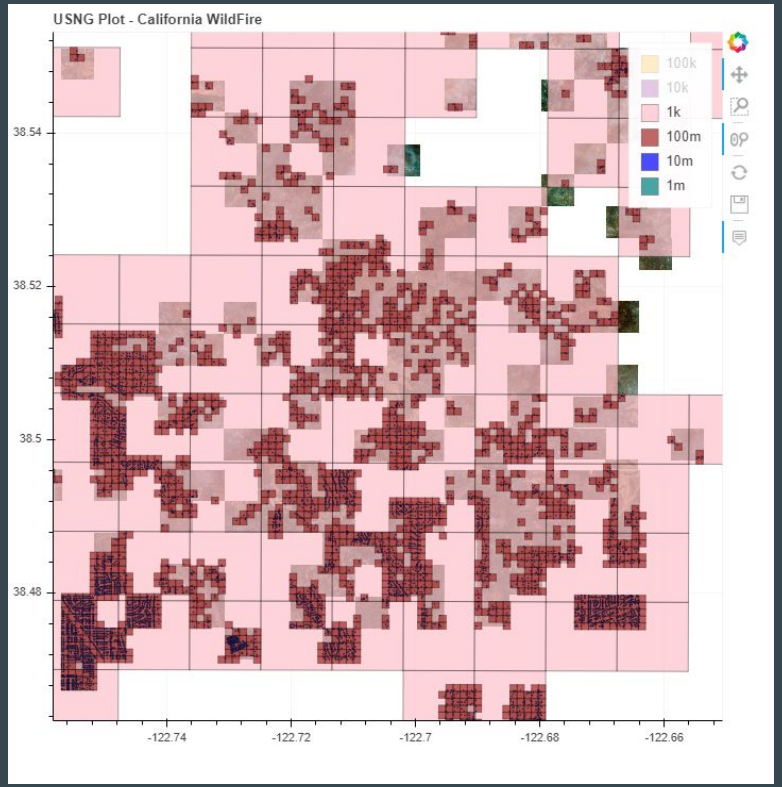
where damage level factor

= 0 for no damage, 0.25 minor, 0.75 major, 1 destroyed

Solution Models - Updates

US National Grid Coordinates Viz

- Added xview2 images to Bokeh plot
- Color scaled each precision layer by financial cost
- Precision layers are hideable

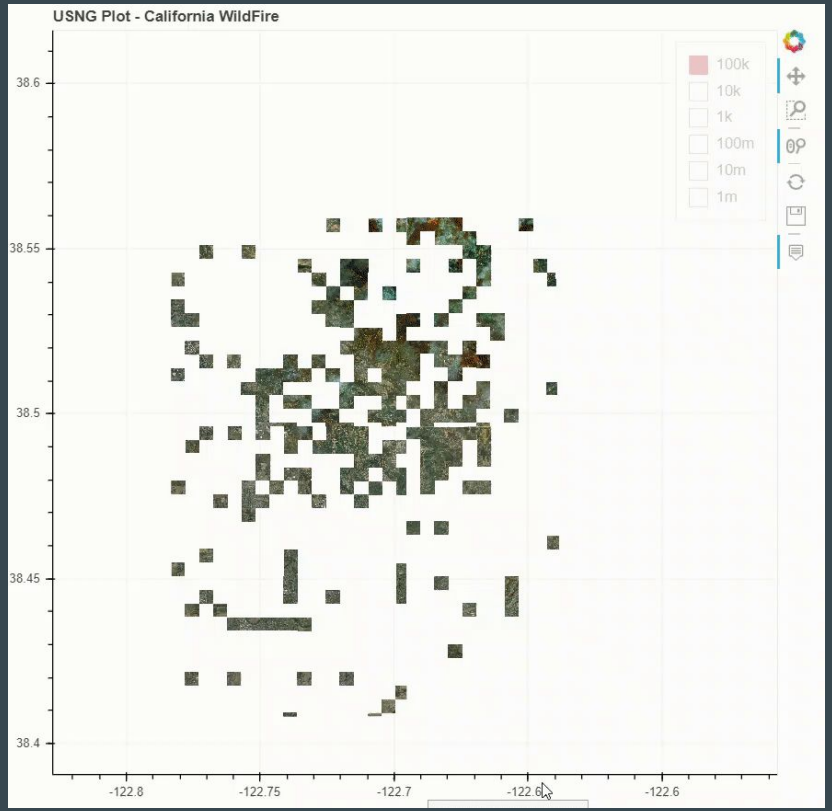


Financial cost estimation using USNG coordinates

Solution Models - Updates

US National Grid Coordinates Viz - next steps

- Will need these processes to display images received in app
- https://ermlickw.github.io/

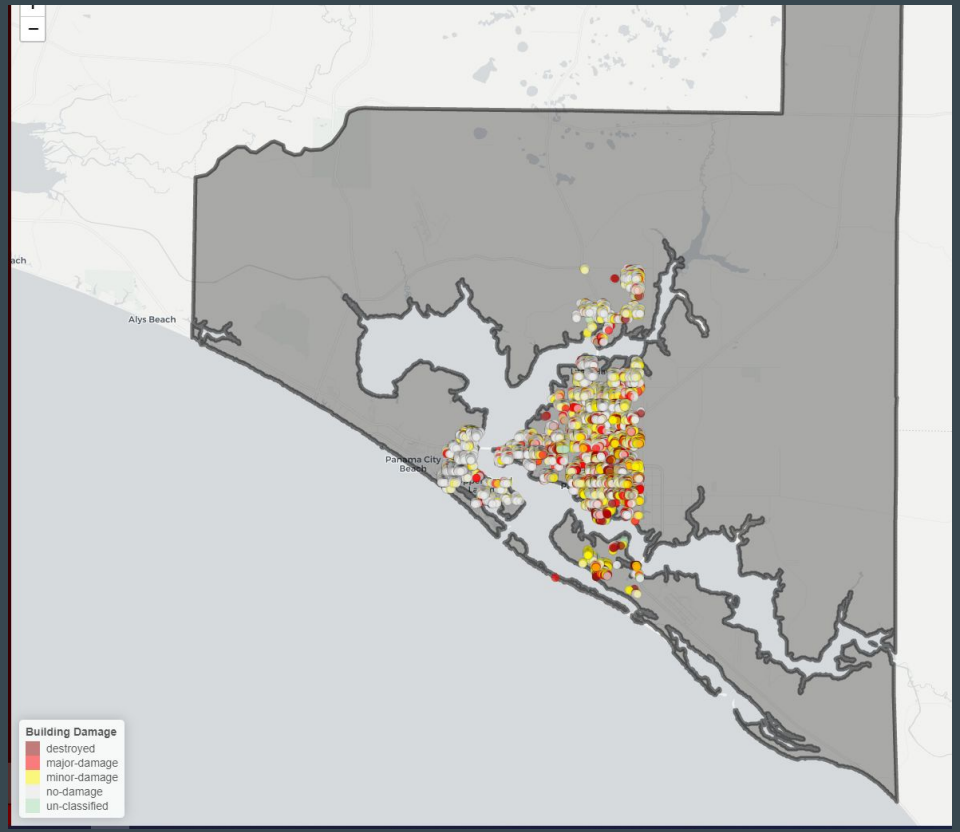


Financial cost estimation using USNG coordinates

Analytics & Algorithms -Updates

County level visualizations

- Noticed damage area is much smaller than county area
- Will result in poor estimations
- Some disasters in Xview2 (moore, tuscaloosa, joplin tornadoes and Hawaii eruption) may have captured all affected buildings
- Focus on these for financial modeling checking only



Hurricane Michael Bay County Xview2

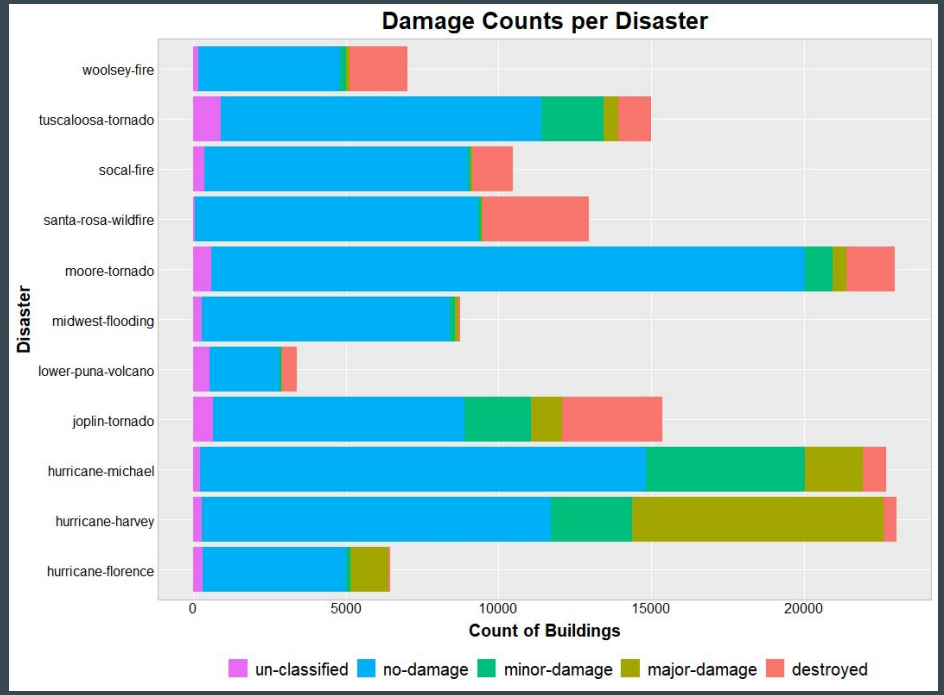
DAEN 690 Schedule

Sprint	Milestone Goals		Presentation
Problem Definition and Project Plans	• Problem (decision) defined • Understanding of complexity • Potential data source identified • Potential analytics identified	• Project schedule defined • Participant roles assigned • Risks identified and mitigation plan	Mid-Sprint 28 May; Full Sprint 4 Jun
Data Sets	• Data located and accessed • Initial processing underway	• Risks identified and mitigated	Mid-sprint 11 Jun; Full Sprint 18 Jun
Analytics/algorithms	• Algorithms defined and coded • Initial applications completed	• Risks identified and mitigated	Mid-sprint 25 Jun & 2 Jul; Full Sprint 9 Jul
Visualizations	• Visualization concepts defined • Visualization implemented	• Risks identified and mitigated	Mid-sprint 16 Jul; Full Sprint 23 Jul
Final Presentations	• Project components completed • Project components integrated	• Project supports final decision • Presentation made	Mid-Sprint 30 July; Full Sprint 6 Aug

Analytics & Algorithms -Updates

Dataset imbalance & resampling

- Imbalanced classes will need to be upsampled or downsampled for each disaster type
- Critical for training



Hurricane Michael Bay County and Xview2 Image Data