Post-Hurricane Structure Damage Assessment Leveraging Aerial Imagery and Convolutional Neural Networks

Allan Kapoor
Springboard School of Data
December 2021

The Need for Post-Disaster Damage Assessment

Major cause of damage/economic disruption in the United States:

- Hurricanes: $21.5 billion per event
- 2015-2020: 10+ >$1 billion storms per year

Damage assessment critical to response efforts but challenging:

- Lack of access
- Manual review is resource and time intensive

Problem Statement

In order to support effective natural disaster response and recovery, how can we leverage remote sensing imagery to quickly and efficiently produce accurate damage assessments after extreme storms?

Stakeholders: Federal Emergency Management Agency (FEMA), the Department of Defense (National Guard, etc.), state offices of emergency services, and county/municipal governments.

Data Source

University of Washington Disaster Data Science Lab:
https://ieee-dataport.org/open-access/detecting-damaged-buildings-post-hurricane-satellite-imagery-based-customized

- Aerial images of structures from Houston area after Hurricane Harvey (2017)
- 14,000 images tagged as with damage or no damage
    - Training set: 8,000 images
    - Validation set: 2,000 images
    - Test set: 2,000 images
- Labelling based on community crowdsourced ground-level data collection
- Equal number of images in each class

Challenges

"Damage" is not a single feature or object

Scattered materials or poor structure condition may be due to dilapidation, not recent damage

Buildings of same class may have very different structures

Data Exploration: Visual inspection

Color, RGB format

128x128 pixels

Observations:

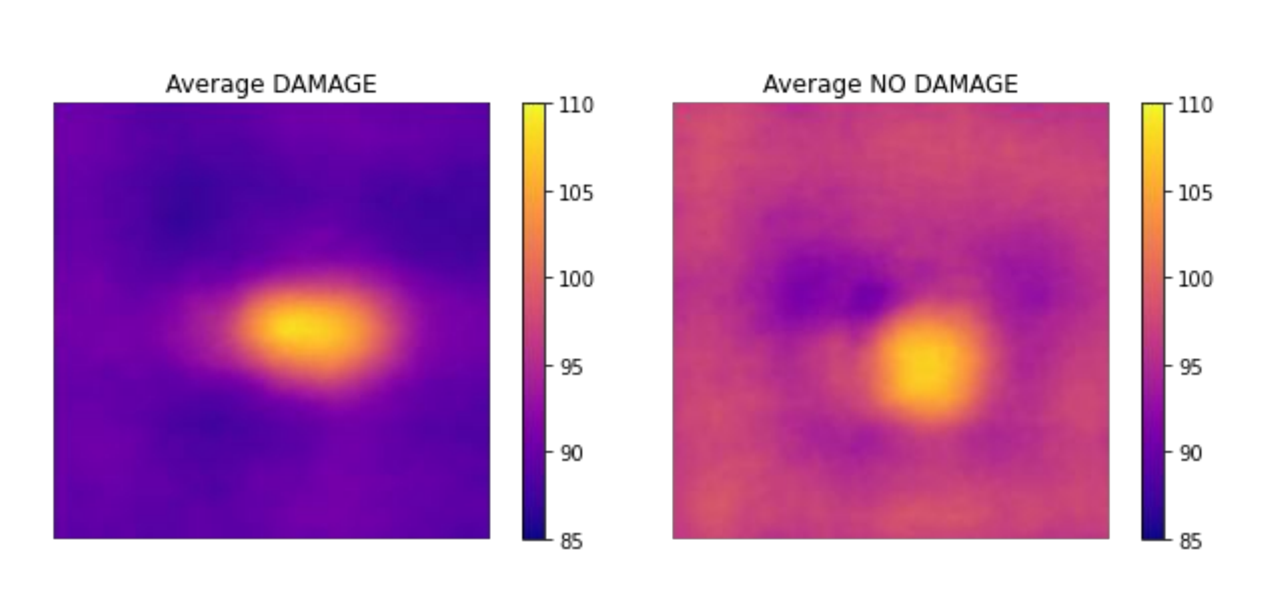
- Flooding: texture
- Debris
- Difficult task to human eye

Data Exploration: Trends by Class

Mean value for each pixel by class

Structure apparent in both

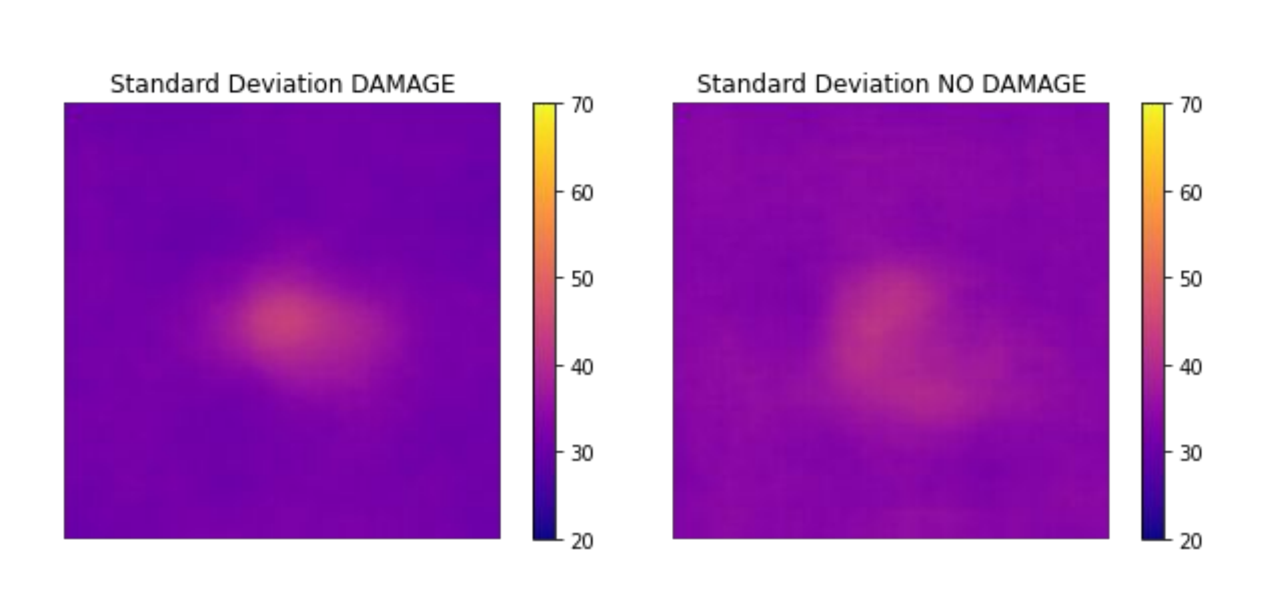
Lower value pixels surrounding structure in average image with damage than without



Data Exploration: Trends by Class

Standard deviation for each pixel by class

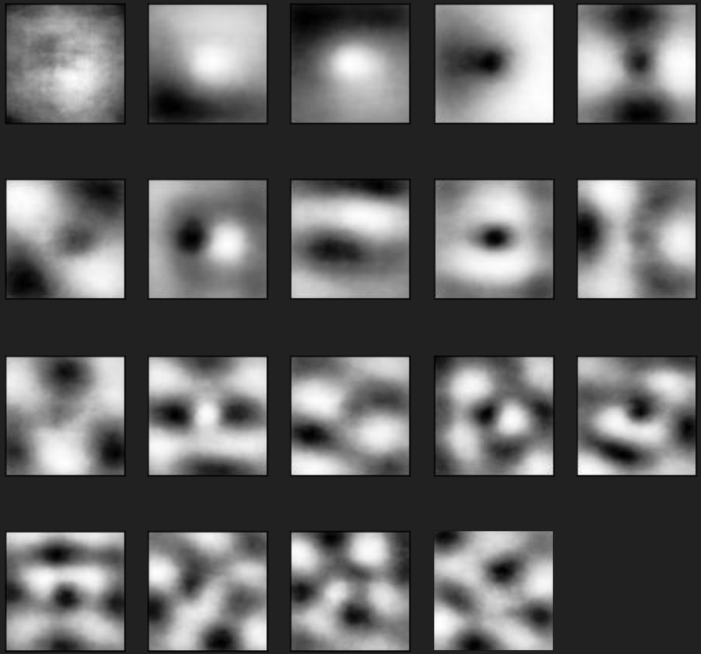
Variation around structure greater for no damage - because ground is visible above flood water?



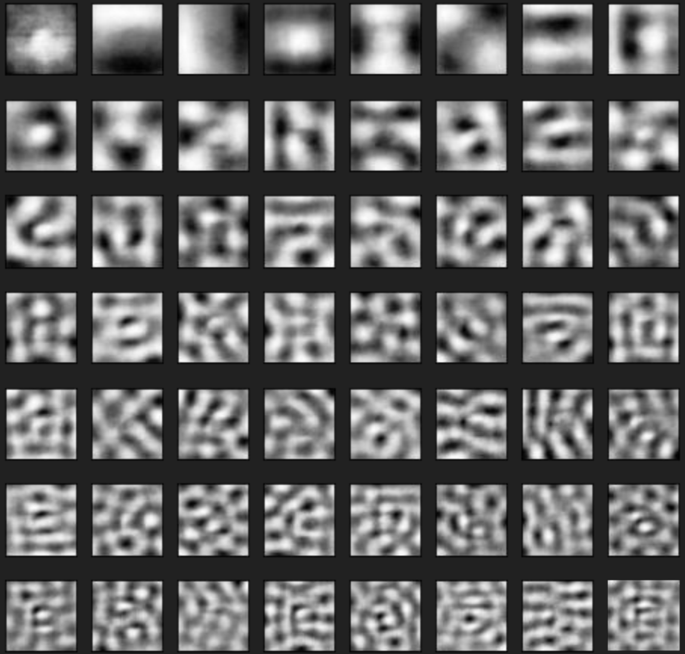
Data Exploration: PCA/Eigenimages

PCA explaining 70% of variation

With damage: 19 principle components



With damage: 56 principle components

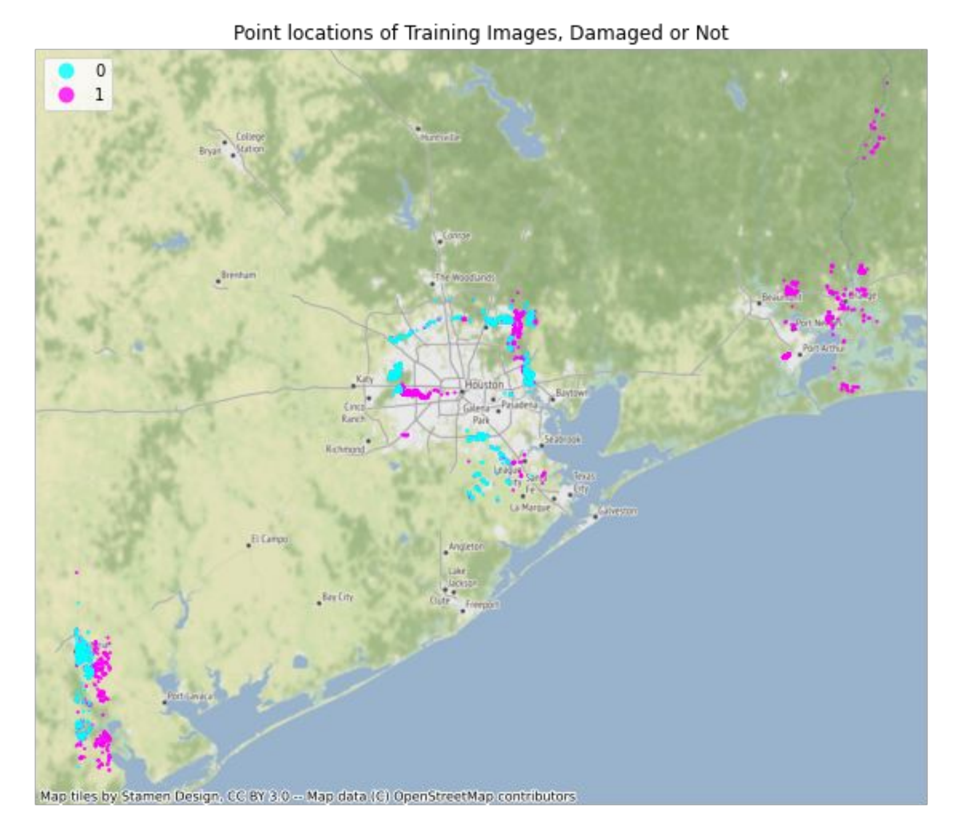


Data Exploration: Geographic Distribution

Geographic distribution of training data by class
(0=damage, 1=no damage)

3 distinct areas

Spatial concentrations of damaged and not damaged structures - may result in training on irrelevant features?



Modeling Approach

Convolutional neural network

TensorFlow (Keras API)

Performance metric: accuracy on validation set

Google Cloud VM: 8 vCPUs (30 GB RAM) + 1 NVIDIA T4 GPU (16 GB)

Iterations:

- Baseline Model
- Add max pooling, drop layers
- Transfer learning

Modeling: Baseline

3 convolution layers, 3 dense layers

- Batch normalization
- ReLU activation
- Adam optimizer

Accuracy (validation set):

- **0.94650** (no image augmentation)
- **0.95650** (with image augmentation)

Layer	Output Shape	# of Params
Rescaling	(128, 128, 3)	0
Convolution (filters=32, kernel_size=5, strides=2)	(64, 64, 32)	2,432
Batch Normalization	(64, 64, 32)	128
Activation (ReLU)	(64, 64, 32)	0
Convolution (filters=32, kernel_size=3, strides=1)	(64, 64, 64)	18,496
Batch Normalization	(64, 64, 64)	256
Activation (ReLU)	(64, 64, 64)	0
Convolution (filters=32, kernel_size=3, strides=1)	(64, 64, 64)	36,928
Batch Normalization	(64, 64, 64)	256
Activation (ReLU)	(64, 64, 64)	0
Flattening	262,144	0
Dense (512 nodes, ReLU activation)	512	134,218,240
Dense (256 nodes, ReLU activation)	256	131,328
Dense (124 nodes, ReLU activation)	124	32,896
Dense (2 nodes, Softmax activation)	2	258

Modeling: Improving Performance

Reducing overfitting:

Max Pooling layers (reduce spatial sensitivity of convolution filters)

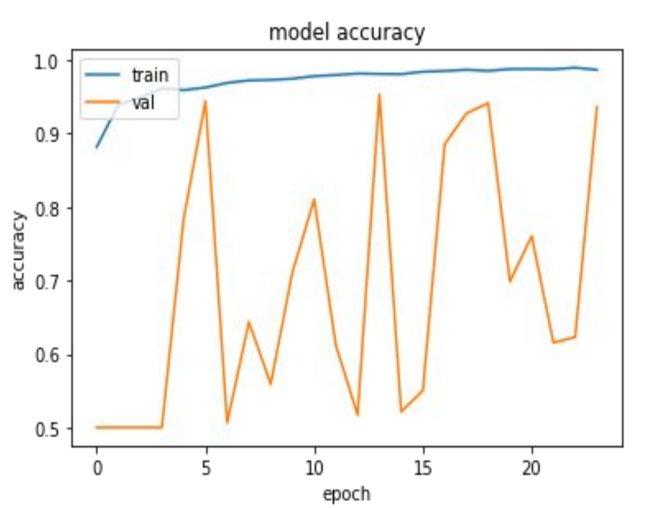
Drop out layers (reduce overfitting to noise)

Modeling: Improving Stability/Convergence

- Reduction in kernel size and stride for the first convolution layer
- Reduction in number of filters in first convolution layer
- Reduction in number of nodes in each dense layer by 50%
- Reduction in initial learning rate for the Adam optimizer from the default (0.001) to 0.0001

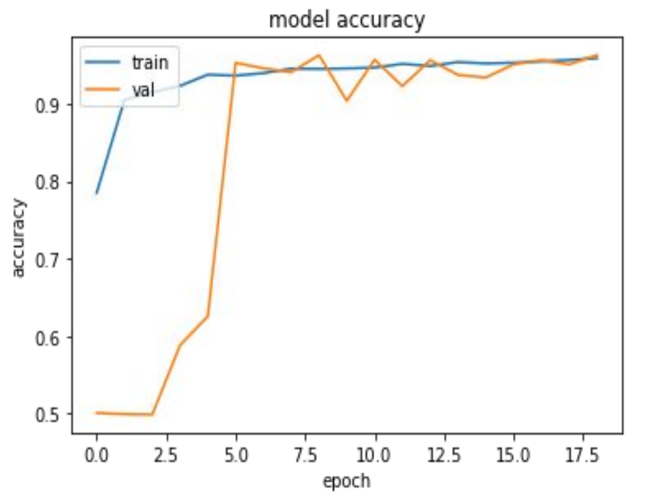
Not Converging:

(before updates)



Converging:

(after updates)



Modeling: Refined Model Architecture

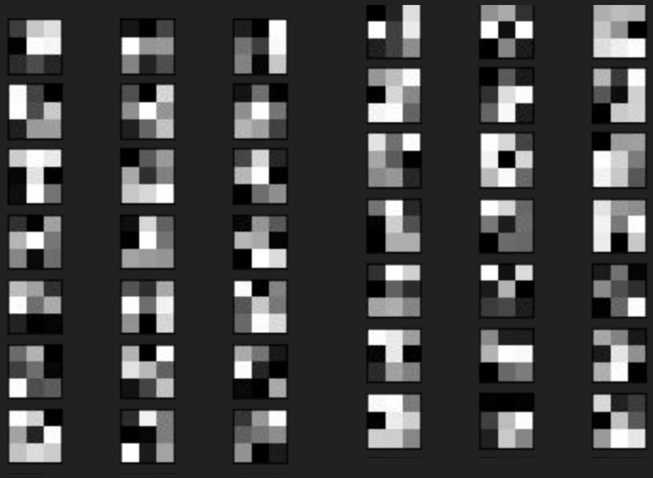
Validation accuracy: **0.9735**

Layer	Output Shape	# of Params
Rescaling	(128, 128, 3)	0
Convolution (filters=32, kernel_size=3, strides=1)	(128, 128, 32)	896
Max Pooling (pool size=2, strides=2)	(64, 64, 32)	0
Batch Normalization	(64, 64, 32)	128
Activation (ReLU)	(64, 64, 32)	0
Convolution (filters=32, kernel_size=3, strides=1)	(32, 32, 64)	18,496
Max Pooling (pool size=2, strides=2)	(16, 16, 64)	0
Batch Normalization	(16, 16, 64)	256
Activation (ReLU)	(16, 16, 64)	0
Convolution (filters=32, kernel_size=3, strides=1)	(8, 8, 64)	36,928
Max Pooling (pool size=2, strides=2)	(4, 4, 64)	0
Batch Normalization	(4, 4, 64)	256
Activation (ReLU)	(4, 4, 64)	0
Flattening	1024	0
Dense (512 nodes, ReLU activation)	512	524,800
Dropout (rate=0.3)	512	0
Dense (256 nodes, ReLU activation)	256	131,328
Dropout (rate=0.2)	256	0
Dense (128 nodes, ReLU activation)	128	32,896
Dropout (rate=0.1)	128	0
Dense (2 nodes, Softmax activation)	2	258

Modeling: Convolution Filters

Smaller kernel size (3) performed better than medium (5) or large (10)

Kernel size=3 (best performance)



Kernel size=10 (more visible shapes, but poor performance

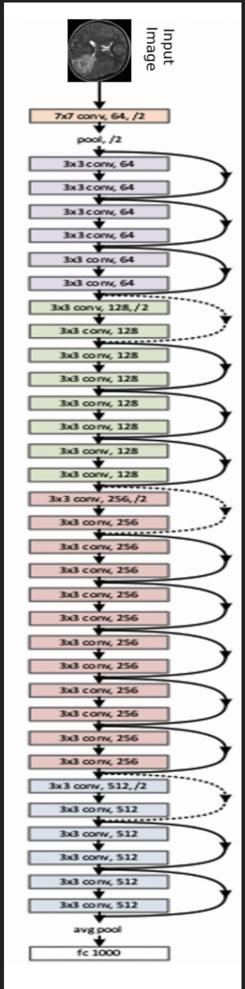
Modeling: Deep Network via Transfer Learning

Resnet50

- Balance of performance/computational efficiency
- Weights trained on ImageNet
- Used for xView2 competition baseline

Incorporation into model:

- Convolution/max pooling
- Resnet model architecture (frozen weights)
- Dense layers



Comparing Models

Deep network achieved highest accuracy, but very slow

More basic model only misclassified 6 more images but classifies much more efficiently

Model	Validation Accuracy
Baseline (no image augmentation)	0.9365
Baseline	0.9440
With Max Pooling (kernel=5) & Dropout Layers	0.9500
With Max Pooling (kernel=10) & Dropout Layers	0.9230
With Max Pooling (kernel=3) & Dropout Layers (dense layers with 50% less nodes)	**0.9735**
Transfer Learning (with Max Pooling kernel=5)	0.9765
Transfer Learning (with Max Pooling kernel=3)	0.9735

Selected Model: Performance on Test Data

Accuracy: 0.9775

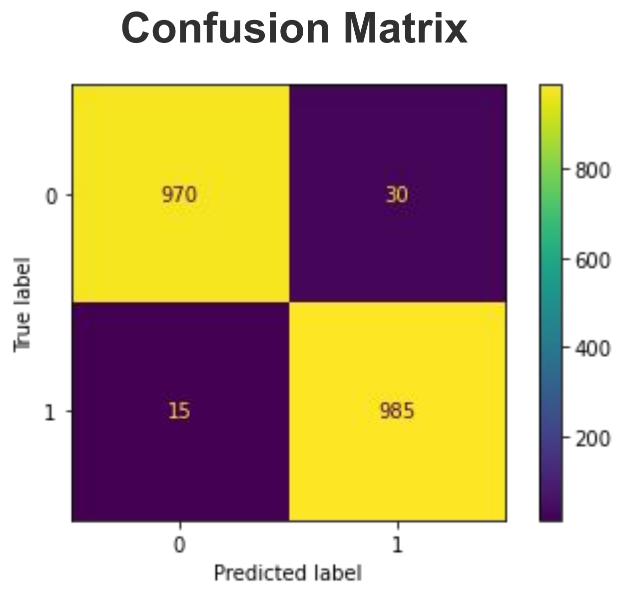
Sensitivity/TPR: 0.97

Specificity/TNR: 0.985

Precision/PPV: 0.985

Fallout/FPR: 0.015

FNR: 0.03

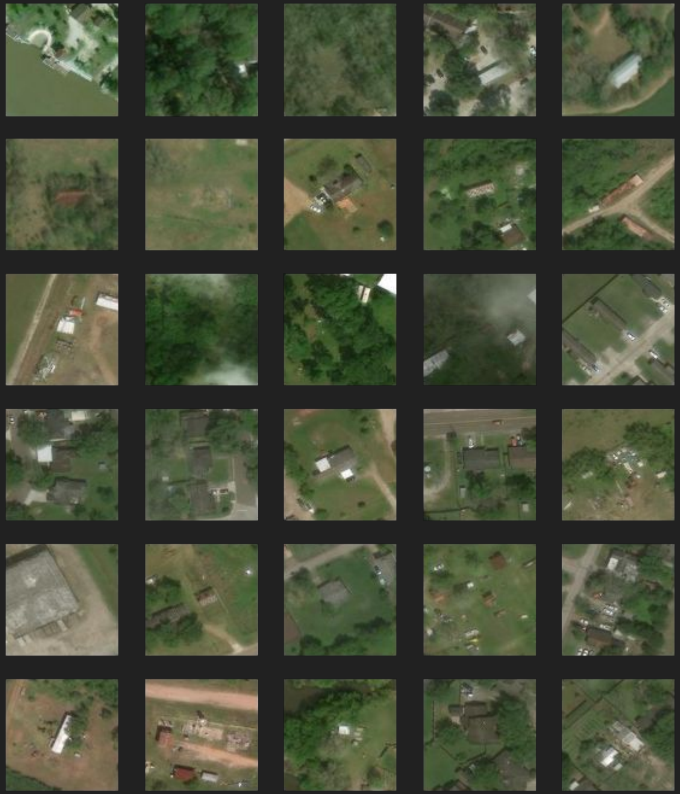


0 = no damage, 1= damage

Selected Model: Incorrectly Classified Images

False Positives

(30 total)



False Negatives

(15 total)

False Positives tend to have surfaces that are mistaken for flood waters, or have junk around them that is mistaken for damage. False positives also tend to be rural structures.

False negatives appear to be mostly large or non-residential structures, have a lot of variation in the ground surface, and/or no obvious flood water

Selected Model: Correctly Classified Images

True Positive Examples

Tend to have obvious flood water, or scattered materials.

Some of these aren't obvious to the human eye.

True Negative Examples

Tend to be obviously not flooded (visible ground)

Also examples that do have water or surfaces that look like flood water to the human eye.

Conclusion: Opportunities

Improve training data:

- More labelled data
- Imagery from after floods have subsided
- Imagery from neighborhoods with mixed impacts
- Imagery from different cities

Improve modeling:

- Cross validation
- Hyperparameter optimization

Conclusion: Using the Model

Damage classification as step in automated pipeline:

- Ingest/clean aerial imagery
- Crop images of structures based on MS Building Footprints data
- Classify images
- Plot locations/damage assessment on interactive map

Much faster than crowdsourcing/manual review: classified 2,0000 structures in seconds on desktop computer

Hurricane Ida - Crowdsource Damage Assessment App

327 GIS staff required to ID 197,712 damaged structures