CLASSIFYING BUILDING DAMAGE FROM IMAGES

Based on Satellite Imagery after Hurricane Harvey 2017

Springboard Capstone Three

Fall 2020

CONTEXT

Hurricanes and extreme storms pose a risk to human health and causes significant damage to buildings and assets across the world every year.

After a hurricane event, relief efforts spend significant time and money evaluating damage to buildings manually.

Image: apnews.com

QUESTION

Can we classify buildings as damaged using satellite imagery?

DATA

Satellite images of buildings and structures in the Houston, TX area collected shortly after Hurricane Harvey in 2017.

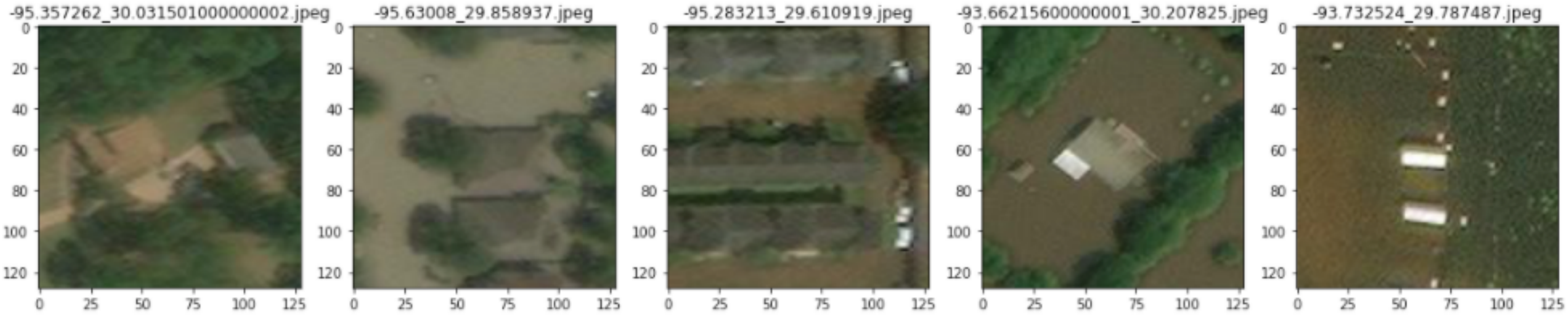
Training set: 5,000 damage / 5,000 no_damage
Test set: 8,000 damage / 1,000 no_damage
Validation set: 2,000 damage / 2,000 no_damage

Data citation: Quoc Dung Cao, Youngjun Choe, “Detecting Damaged Buildings on Post-Hurricane Satellite Imagery Based on Customized Convolutional Neural Networks”, IEEE Dataport, 2018. [Online]. Available: 10.21227/sdad-1e56 accessed on October 18, 2020

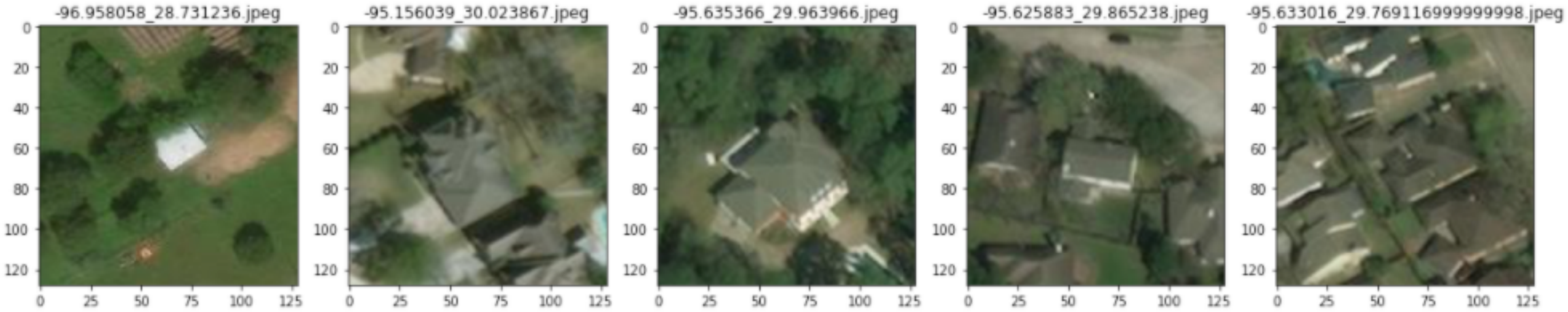
Image: Billings Gazette

DATA

Images labeled as damage:



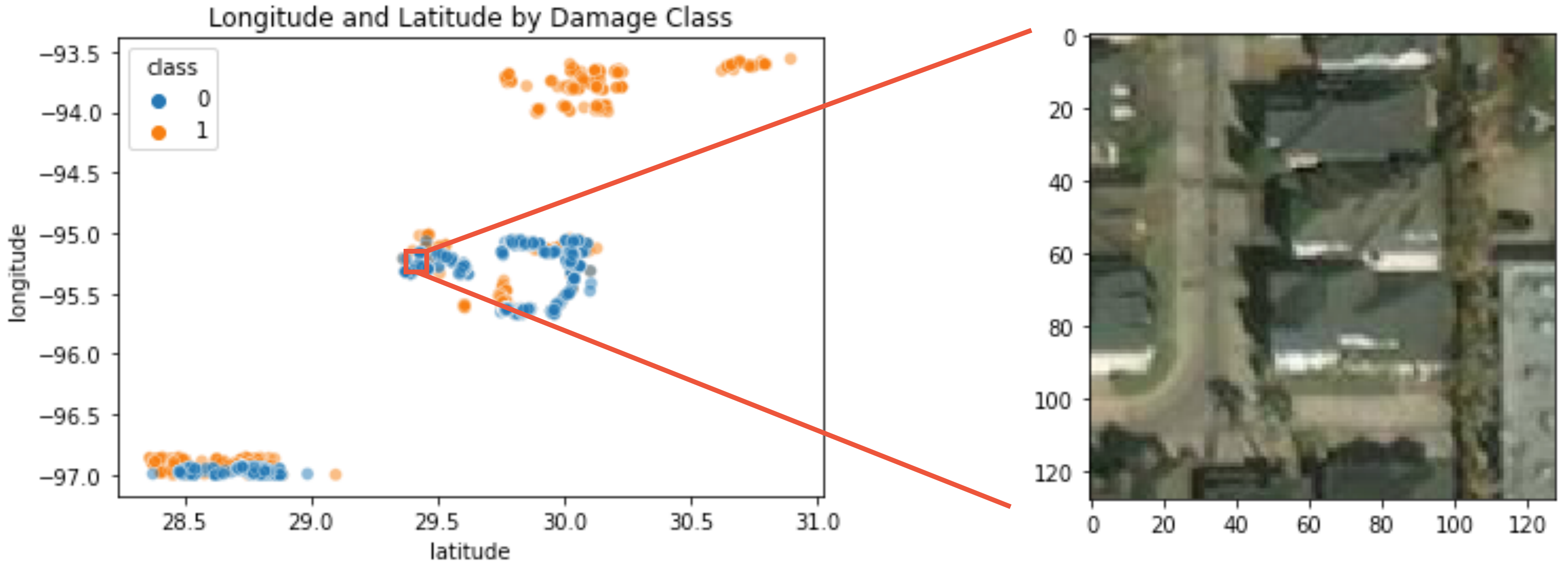
Images labeled as no_damage:



EXPLORATORY ANALYSIS

Latitude and Longitude

Latitude and longitude by damage class shows the spatial relationship of the images in the training set. It is possible to visually identify spatial areas where all buildings were damaged and areas where there is a mix of damaged and not damaged buildings.



EXPLORATORY ANALYSIS

Simple Feature Extraction

Simple logistic regression models are capable of describing this dataset relatively well using only basic features extracted from the image dataset including compressed file size and statistics describing raw pixel data.

A model trained using compressed file size as an independent variable along with latitude and longitude was able to perform with 0.82 test accuracy and an F1 score of 0.89

	model	test accuracy	train accuracy	precision	recall	f1 score
0	lr_latlong	0.706556	0.5079	0.892946	0.761125	0.821783
1	lr_fsize	0.821000	0.8156	0.972909	0.821500	0.890817
2	lr_avgpixel	0.818889	0.8151	0.973958	0.818125	0.889266
3	lr_medpixel	0.831222	0.8252	0.972721	0.833500	0.897745
4	lr_stdpixel	0.826333	0.8248	0.972961	0.827625	0.894428
5	lr_allpixel	0.804667	0.8210	0.978094	0.798125	0.878992
6	lr_hptune	0.827556	0.8258	0.973282	0.828750	0.895220
7	rf_stdpixel	0.918444	0.9957	0.991610	0.916000	0.952307
8	rf_stdpixel_GSCV	0.919556	0.9586	0.990429	0.918375	0.953042
9	lr_pca	0.795000	0.8354	0.972789	0.794444	0.874618

CNN MODEL SELECTION

Start with Simple Layers

An initial CNN model was trained with a very simple structure and then improved upon over several modeling iterations. Each model trained was compiled with the RMSprop optimizer and categorical crossentropy loss function.

By altering the number of convolutional layers, the number of filters, and the number of epochs trained, the best performing model was identified.

This model had a test accuracy of 0.94

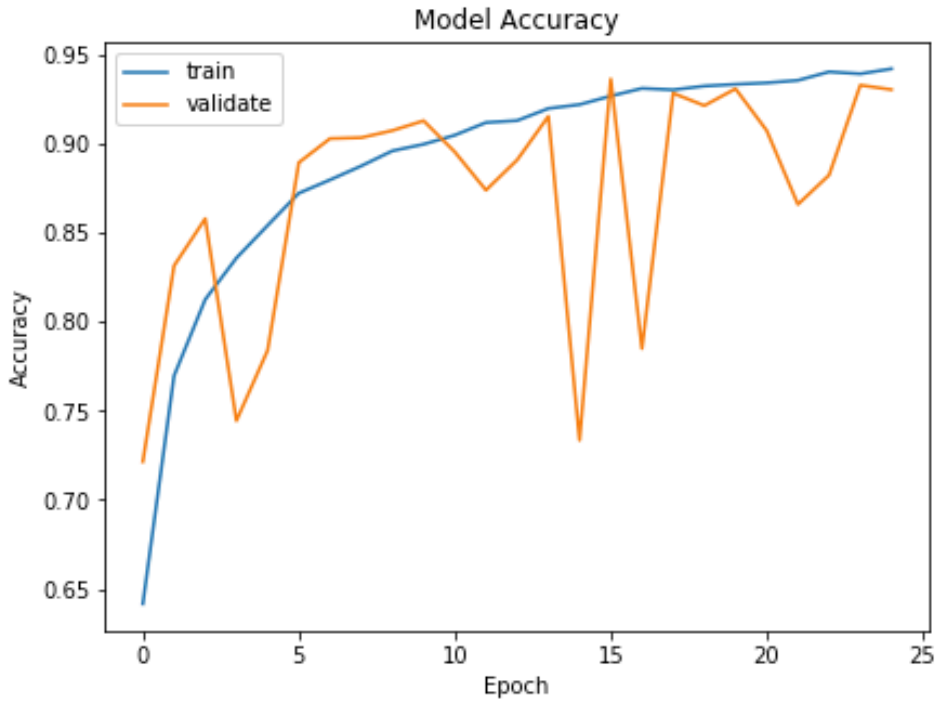
	model	test accuracy	test loss	train accuracy	train loss
0	model_1x4_e5	0.856333	0.436132	0.7502	0.537973
1	model_1x12_e5	0.888333	0.350369	0.8722	0.366229
2	model_2x12_e5	0.790667	0.550664	0.8479	0.396608
3	model_1x12_e25	0.943333	0.142658	0.9504	0.144678
4	model_1x12_e25_AUG	0.864111	0.340066	0.9041	0.240608
5	model_1x32_e25	0.890667	0.316850	0.9461	0.138461

CNN MODEL SELECTION

Best Model

The best performing model included 1 convolutional layer with 12 filters and trained over 25 epochs. While the validation accuracy fluctuating during training, the training accuracy continued to improve steadily across epochs.

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_4 (Conv2D)            (None, 128, 128, 12)      336
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 64, 64, 12)        0
_________________________________________________________________
dropout_3 (Dropout)          (None, 64, 64, 12)        0
_________________________________________________________________
flatten_3 (Flatten)          (None, 49152)             0
_________________________________________________________________
dense_6 (Dense)              (None, 64)                3145792
_________________________________________________________________
dense_7 (Dense)              (None, 2)                 130
_________________________________________________________________
activation_3 (Activation)    (None, 2)                 0
=================================================================
Total params: 3,146,258
Trainable params: 3,146,258
Non-trainable params: 0
_________________________________________________________________
```

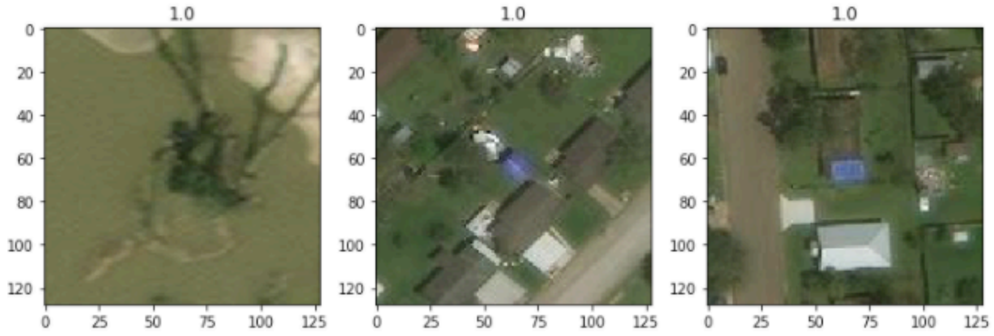


KEY TAKEAWAYS

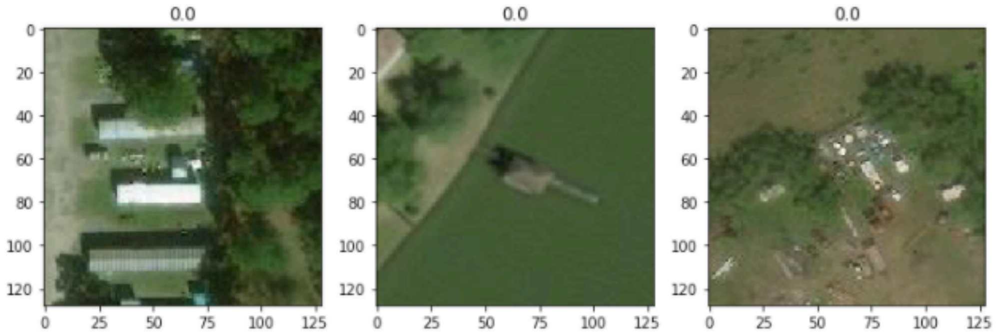
Misidentified Images

The best performing CNN model misidentified 510 out of 9,000 test images including 406 damaged images and 104 not damaged images. While some of these are obvious mistakes, others are difficult to determine damage with the human eye indicating the model's mistakes are reasonable ones.

Images labeled as damaged that were incorrectly predicted as not damaged.



Images labeled as not damaged that were incorrectly predicted as damaged.



KEY TAKEAWAYS

Model Uses

With a test accuracy of 0.94, this model could prove useful for identifying buildings and structures that have been damaged or flooded after a severe hurricane event.

Evaluating damage from aerial images will **allow relief efforts to assess damage and identify areas that may have been impacted harder than others.**

This process would be a significant **improvement in time and cost** compared to typical solutions which involve manually identifying damage from vehicles.

FUTURE WORK

Model Improvement

Further tuning of model hyperparameters, as well as additional adjustments to possible CNN layers may increase performance. Training the model for more time with more epochs will also likely result in higher accuracy.

Overfitting

The current CNN model may be on the verge of overfitting. Continued evaluation of additional dropout layers or other techniques to address overfitting may be necessary.

Data Augmentation

Increasing the size of the dataset by transforming images, such as changing color or rotation, may help improve performance and reduce overfitting.

Expand Dataset

This model has been trained using images of one particular area after one particular storm. Collection of more data from other events both spatially and temporally will allow this model to generalize more effectively on new data and ultimate be more useful.