Classifying Building Damage from Aerial Imagery

Based on Post-Hurricane Harvey Satellite Images of Buildings
Springboard Capstone Three, Fall 2020

Context

Hurricanes and extreme storms can result in severe damage to buildings and infrastructure. Assessing damage is a critical step in post-hurricane relief efforts both for providing aid as well as for planning and policy development. As the effects of climate change become more apparent and extreme storms occur more frequently, assessing and understanding damage may be useful for predicting or preventing future damage.

Historically, building damage has been assessed manually on the ground, requiring significant time and resources. A model that can use easily-obtained aerial imagery to classify building damage could be an efficient and cost-saving asset.

Question

Can we classify buildings as damaged or not damaged using aerial images?

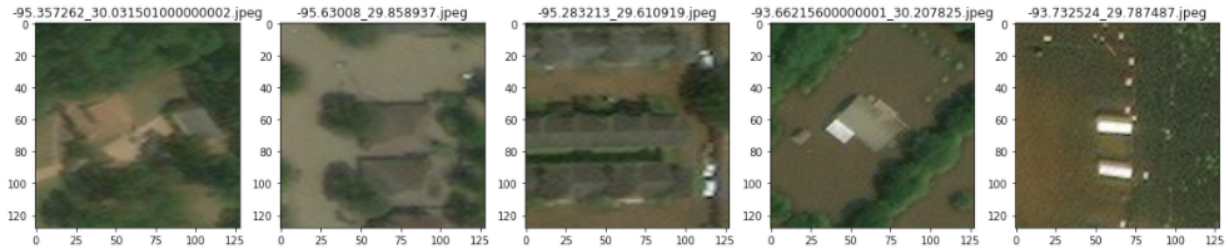
Data Source

Images for this project include satellite images of buildings and structures in the Houston, TX area collected after Hurricane Harvey in 2017. These images were sourced from the IEEE Dataport repository for open data. This dataset has been assembled and shared as part of a research initiative from the University of Washington. Data citation:

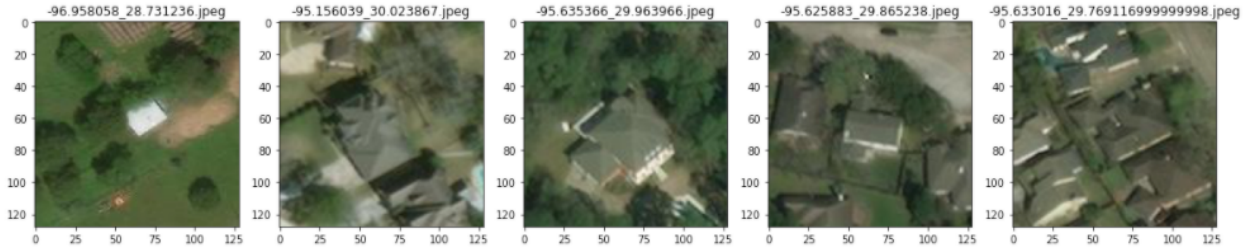
Quoc Dung Cao, Youngjun Choe, “Detecting Damaged Buildings on Post-Hurricane Satellite Imagery Based on Customized Convolutional Neural Networks”, IEEE Dataport, 2018. [Online]. Available: 10.21227/sdad-1e56 accessed on October 18, 2020

The dataset includes one training set, one validation set, and two test sets with images labeled as either “damage” or “no damage”. The damage class is indicated by 1 and the no damage class indicated as 0. Image file names include the latitude and longitude of the location the satellite image was taken.

Examples of “damage” images:



Examples of “no damage” images:



The following statistics describe each of the image sets that make up the dataset:

The training set includes 5,000 damage and 5,000 no damage images.

	class	latitude	longitude	file size	avg pixel	min pixel	max pixel
count	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000
mean	0.500000	29.711999	-95.526672	2762.533300	87.787732	10.183500	224.760600
std	0.500025	0.379762	0.657977	484.313286	17.587444	10.213856	32.752596
min	0.000000	28.363774	-97.001979	943.000000	19.122946	0.000000	85.000000
25%	0.000000	29.757811	-95.640076	2413.000000	77.439331	0.000000	207.000000
50%	0.500000	29.831086	-95.626598	2743.000000	85.260040	8.000000	234.000000
75%	1.000000	29.858590	-95.233276	3080.000000	94.686468	17.000000	255.000000
max	1.000000	30.895018	-93.559640	5647.000000	215.625610	162.000000	255.000000

The unbalanced test set includes 8,000 damage and 1,000 no damage images.

	class	latitude	longitude	file size	avg pixel	min pixel	max pixel
count	9000.000000	9000.000000	9000.000000	9000.000000	9000.000000	9000.00000	9000.000000
mean	0.888889	29.717114	-95.457869	2570.191667	85.440578	13.59100	216.152444
std	0.314287	0.404206	0.751128	425.628402	18.290089	10.66709	37.096237
min	0.000000	28.363781	-97.001948	1016.000000	13.880676	0.00000	70.000000
25%	1.000000	29.757398	-95.637082	2337.000000	74.184294	3.00000	190.000000
50%	1.000000	29.815798	-95.593778	2478.000000	82.332600	14.00000	225.000000
75%	1.000000	29.861814	-95.160405	2746.000000	93.497459	21.00000	253.000000
max	1.000000	30.994013	-93.528502	5484.000000	195.810074	120.00000	255.000000

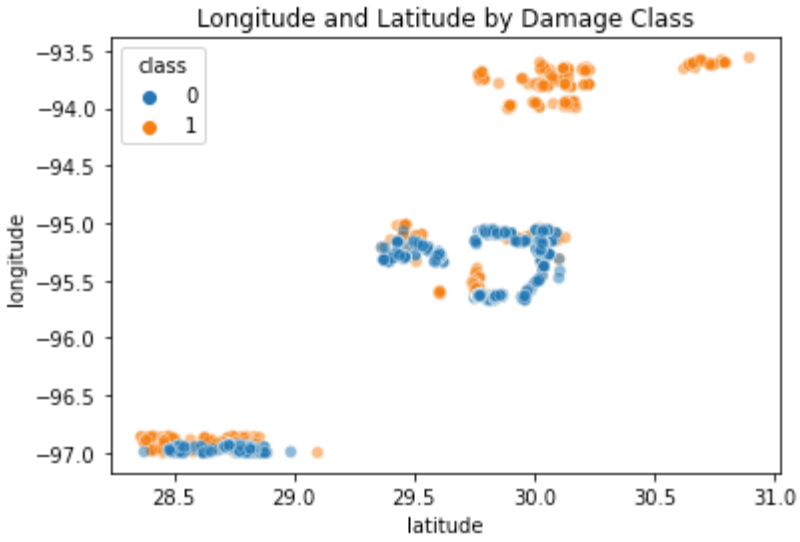
The validation set includes 1,000 damage and 1,000 no damage images.

	class	latitude	longitude	file size	avg pixel	min pixel	max pixel
count	2000.000000	2000.000000	2000.000000	2000.000000	2000.000000	2000.00000	2000.000000
mean	0.500000	29.715607	-95.528331	2765.711000	88.193913	10.43500	225.813500
std	0.500125	0.375348	0.662627	475.215775	17.757315	10.32226	33.049128
min	0.000000	28.371836	-97.001414	1070.000000	32.163715	0.00000	77.000000
25%	0.000000	29.758506	-95.639608	2422.000000	77.730309	0.00000	209.000000
50%	0.500000	29.829045	-95.626587	2744.000000	85.538002	9.00000	237.000000
75%	1.000000	29.858355	-95.243789	3081.000000	95.105476	17.00000	255.000000
max	1.000000	30.994087	-93.558326	5418.000000	198.853195	80.00000	255.000000

Exploratory Data Analysis

Exploratory data analysis helps to further understand the dataset and try and identify relationships and patterns before building a deep learning model. Using the training and test sets, features describing the latitude and longitude, as well as describing file size and various pixel statistics were extracted.

Plotting longitude and latitude 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.



Simple logistic regression models are capable of describing this dataset relatively well using only the basic features extracted during the exploratory data analysis phase. Using latitude and longitude alone, a logistic regression model was able to obtain 0.70 test accuracy score and 0.82 F1 score (model name lr_latlong).

Including the file size variable into the logistic regression increased test accuracy to 0.82 and F1 score to 0.89 (model name lr_fsize). This is a surprisingly well-performing model for an image dataset. The increase in performance with the inclusion of file size may be related to the process by which these images are compressed.

	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

Several other logistic regression models were trained, as well as random forest classifers. The logistic regression models continued to show F1 scores staying just under 0.90. The random forest classifiers had very high F1 scores, but also showed high training accuracy suggesting overfitting.

Finally, using PCA for dimension reduction identified 763 components which allowed a reduction from 49,000. The final logistic regression model using the PCA components resulted in an F1 score of 0.87. While these metrics are impressive, the use of deep learning will likely be able to improve on performance even further.

These are the metrics we will try to beat with a convolutional neural network:

	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

Convolutional Neural Network

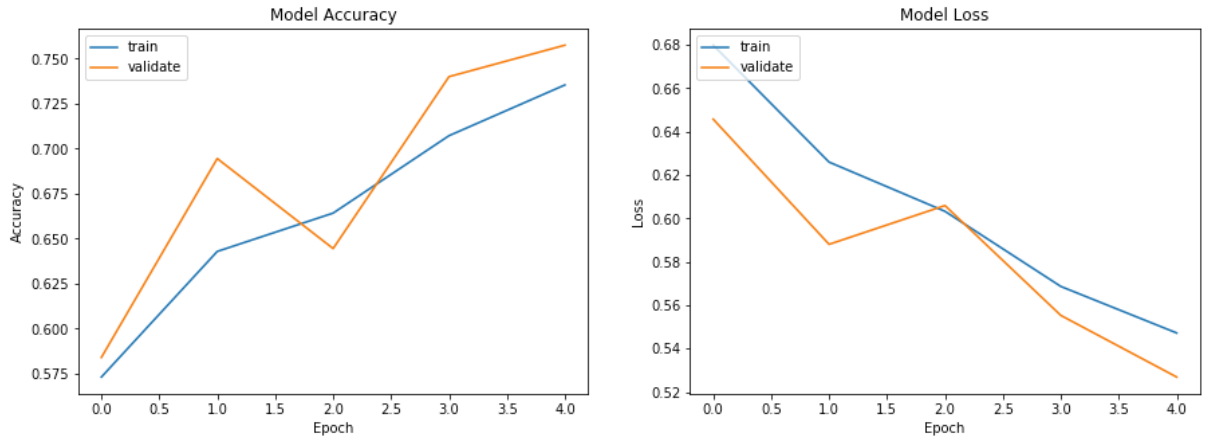
An initial CNN model was trained with a very simple architecture and then improved upon over several modeling iterations. Each model trained was compiled with the RMSprop optimizer and categorical crossentropy loss function.

The initial CNN model trained on this dataset was a very simple model with only 1 convolutional layer and 4 filters. This model also included a drop out layer prior to the dense layers and was trained across 5 epochs.

model_1x4_e5

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 128, 128, 4)       112
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 64, 64, 4)         0
_________________________________________________________________
dropout (Dropout)            (None, 64, 64, 4)         0
_________________________________________________________________
flatten (Flatten)            (None, 16384)             0
_________________________________________________________________
dense (Dense)                (None, 64)                1048640
_________________________________________________________________
dense_1 (Dense)              (None, 2)                 130
_________________________________________________________________
activation (Activation)      (None, 2)                 0
=================================================================
Total params: 1,048,882
Trainable params: 1,048,882
Non-trainable params: 0
_________________________________________________________________
```

While very simple and fast to train, this initial model showed a test accuracy of 0.85.

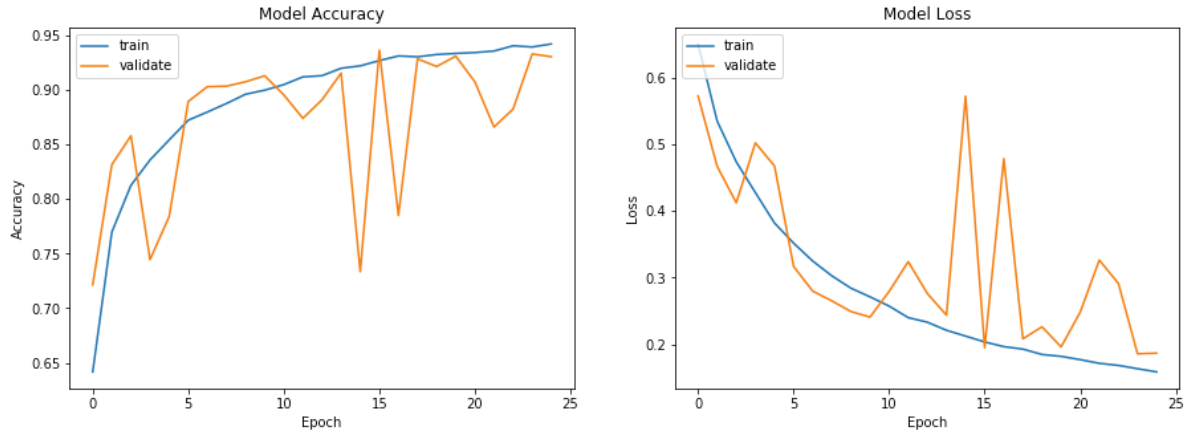


Successive model architectures were trained using different combinations of convolutional layers and filters. For example, increasing the number of filters from 4 to 12 resulted in an increase in test accuracy to 0.89. See the BuildingDamage_Modeling notebook for all model specifications and results.

Ultimately, the best performing model was not the most complicated. The best model included 1 convolutional layer with 12 filters and trained over 25 epochs. This model had a test accuracy of 0.94 which exceeds the accuracy obtained by the basic logistic regression classifiers.

model_1x12_e25

```
_________________________________________________________________
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
_________________________________________________________________
```



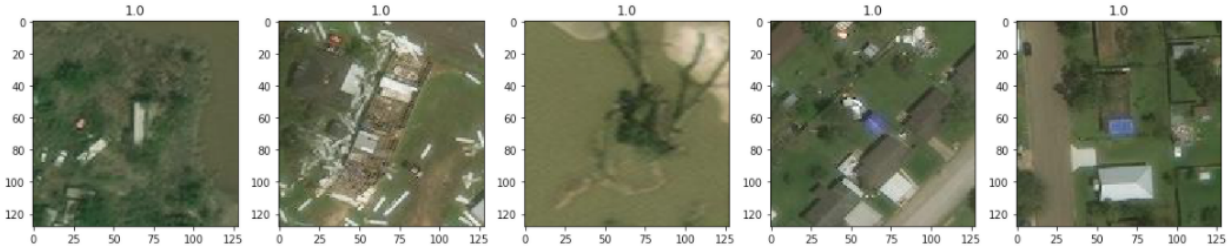
Metrics of all models trained and evaluated on the test set:

	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

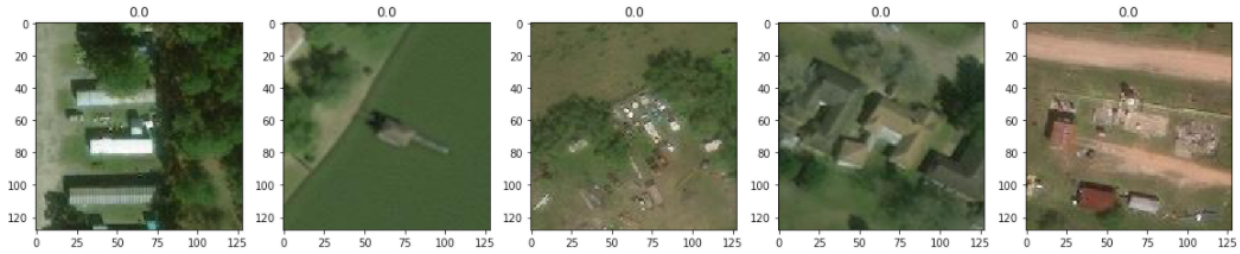
Model Results

The best performing CNN model misidentified 510 out of 9,000 images including 406 damaged images and 104 not damaged images.

A few of the incorrect classifications, images labeled as damaged but identified as not damaged:



A few of the incorrect classifications, images labeled as not damaged but identified as damaged:



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.

With a test accuracy of 0.94 when evaluated on the test set, 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 traveling around the area.

Future Work

There are many opportunities to further improve on this model. With more time available, the model can be trained across more epochs and likely improve on performance as it is built now. Additionally, more tuning of model parameters such as the optimizer and learning rate, as well as further assessment of additional convolutional layers, will also likely result in better performance.

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.