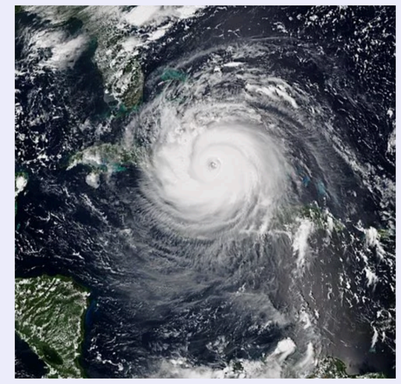
Deep Learning Based Damage Detection on Post-Hurricane Satellite Imagery

Quoc Dung Cao and Youngjun Choe

Department of Industrial and Systems Engineering

University of Washington, Seattle



INTRODUCTION

After a hurricane, damage assessment is critical to emergency managers and first responders. To improve the efficiency and accuracy of damage assessment, instead of using windshield survey, we propose to automatically detect damaged buildings using image classification algorithms. This could give the stakeholders useful information about the severity of the damage to plan for and organize necessary resources.

In this project, we propose to use a convolutional neural network (CNN) to automatically detect 'Damaged' vs 'Undamaged' buildings in the Greater Houston area during 2017 Hurricane Harvey (Figure 1) using satellite imagery (Figure 2). With our custom network, we are able to achieve 97.08% accuracy on the test set.

Through this project, we hope that other researchers can use the dataset and methodology to study and experiment with different uses of satellite imagery in disaster response. We also contribute a pre-trained architecture that achieves satisfactory result. It can facilitate transfer learning either in feature extraction, fine-tuning, or as a baseline to speed up the learning process in future development/events with similar properties.

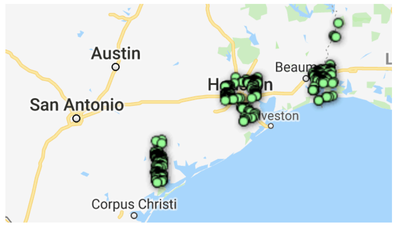


Figure 1: Affected areas during Hurricane Harvey

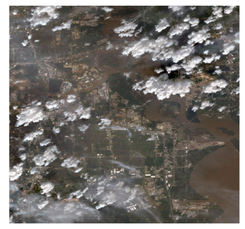


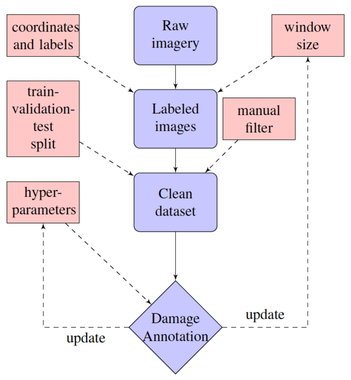
Figure 2: Satellite imagery

CHALLENGES

There are multiple challenges in damage detection. First, the satellite imagery resolution is not as high as the various state-of-the-art datasets commonly used to train neural networks. Second, the volunteers' annotation could be erroneous. Third, there are some inconsistencies in image quality. Since the same region can be captured multiple times in different days, the same coordinate may have different quality and orthorectification.

METHODOLOGY

We present here a framework of damage annotation.



1. Cropping window approach to generate data:
The building coordinates, which are either easily obtained publicly or available from crowd-sourcing projects, are used as the center of a window. The window is then cropped from the raw satellite imagery (Figure 2) to create a data sample (Figure 3).

2. Data processing:
We manually filter the bad-quality images (Figure 3) and remove duplicates to make sure the dataset contains only one good-quality image at each unique coordinate (Figure 4(a)).

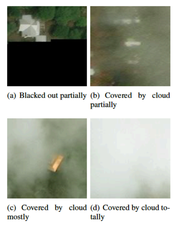
3. Image featurization and network architecture:
To featurize the images, we project them into the same feature dimension. The images are then fed through a CNN (both pre-built and customized) to further extract the right set of features. Our basis for determining the size and depth of a customized network is to monitor the information flow through the network and stop appropriately when there are too many "dead" filters (Figure 4 & 5).

4. Image classification:
We keep the training and validation sets balanced and leave the remaining data to construct 2 test sets, balanced and unbalanced (with a ratio of 1:8).

IMPLEMENTATION

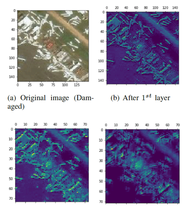
We train the neural network using the *Keras* library with TensorFlow backend with a single NVIDIA K80 Tesla GPU with 64GB memory on a quad-core CPU machine. The network weights are initialized through Xavier initializer. The mini batch size for stochastic gradient descent optimizer is either 20 or 32.

Since it is computationally costly to train the CNN repeatedly, we do not tune the hyper-parameters through a full grid search or full cross-validation. Only some reasonable combinations of the hyper-parameters are considered. Among the parameters, window size is truly a challenge. We do not try all the sizes with all hyper-parameters. We implement a simple model with all the window sizes and find that 128x128 window yields an ideal result.



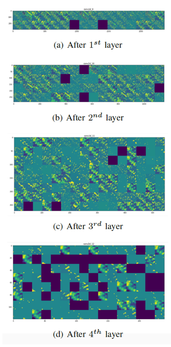
(a) Blacked out partially (b) Covered by cloud partially (c) Covered by cloud mostly (d) Covered by cloud totally

Figure 3: Examples of bad-quality cropped images



(a) Original image (Damaged) (b) After 1^{st} layer (c) After 2^{nd} layer (d) After 3^{rd} layer

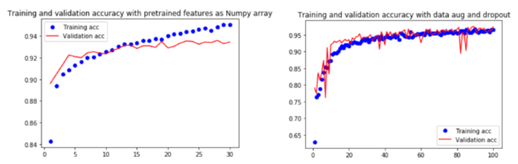
Figure 4: Information flow within one CNN filter



(a) After 1^{st} layer (b) After 2^{nd} layer (c) After 3^{rd} layer (d) After 4^{th} layer

Figure 5: Information flow in all CNN filters

To counter over-fitting, which is a recurrent problem of deep learning, we also adopt data augmentation in the training set through random rotation, horizontal flip, vertical and horizontal shift, shear, and zoom. This can effectively increase the number of training samples to ensure more generalization to achieve better validation and test accuracy. Furthermore, we also employ dropout and L2 regularization in the densely connected layer (Figure 6).



(a) Over-fitting happens after a few epochs (b) Little sign of over-fitting

Figure 6: Fighting overfitting with image augmentation and dropout

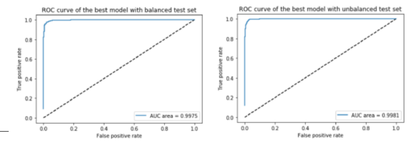
RESULTS

The table below demonstrates the performance of various models. The best performing model is our fresh model with data augmentation and dropout using Adam optimizer, which can achieve 97.08% accuracy on the unbalanced test set. The AUC metric is also computed and shows a satisfying result of 99.8% on the unbalanced test set.

Model	Val. Acc.	Test Acc. (Balanced)	Test Acc. (Unbal.)
CNN	95.8%	94.69%	95.47%
Leaky CNN	96.1%	94.79%	95.27%
CNN + DA + DO	97.44%	96.44%	96.56%
CNN + DA + DO (Adam)	**98.06%**	**97.29%**	**97.08%**
Transfer + DO	93.45%	92.8%	92.8%
Transfer + DA + DO	91.1%	88.49%	85.99%
LR + L2 = 1	93.55%	92.2%	91.45%
Transfer + DA + FDO	96.5%	95.34%	95.73%
Leaky+Transfer + DA + FDO +L2	96.13%	95.59%	95.68%
Leaky+ Transfer + DA + FDO +L2 (Adam)	97.5%	96.19%	96.21%

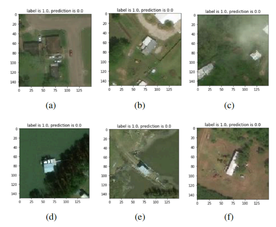
Legend: CNN: Convolutional Neural Network; Leaky: Leaky ReLU, else default is ReLU; DA: Data Augmentation; LR: Logistic Regression; L2: L2 regularization; (Adam): Adam optimizer, else default is RMSprop optimizer; DO: 50% drop out in the densely connected layer; FDO: Full drop out, i.e 25% drop out after every max pooling layer and 50% in the densely connected layer; Transfer: Transfer learning using VGG-16 architecture

TABLE I
MODEL PERFORMANCE



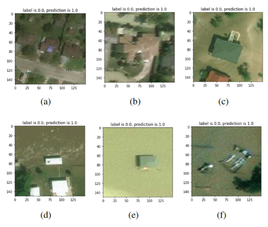
(a) AUC of balanced test set (b) AUC of unbalanced test set

Figure 7 shows some of the false positive cases. We hypothesize that the algorithm could predict the damage through flood and debris detection. Under such hypothesis, the cars in the center of Figure 7(a), the lake water in Figure 7(b), the cloud covering the house in Figure7(c), and the trees covering the roof in Figure 7(f) can potentially mislead the model. For the false negative cases in Figure 8, it is harder to make sense out of the false prediction. Even through visual inspection, we cannot see Figures 8(a)(b) as being damaged. Figures 8 (e)(f) are clearly damaged but the algorithm misses them.



(a) (b) (c) (d) (e) (f)

Figure 7: False positive cases



(a) (b) (c) (d) (e) (f)

Figure 8: False negative cases

CONCLUSION

We demonstrated that through deep learning, automatic detection of damaged buildings can be done satisfactorily. Although our data can be specific to the geographical condition and building properties in the Greater Houston area during Hurricane Harvey, the model can be further improved and generalized to other future disaster events in other regions if we can collect more positives samples from other past events and negative samples from other areas.

We also wish to expand the research to road damage annotation which could help plan effective transportation routes of food, medical equipment, or fuels to the victims.