Automatic Disaster Detection from Aerial Imagery

Richard Chen
Department of Computer Science
Stanford University
chardch@stanford.edu

Angelica Pando
Department of Computer Science
Stanford University
apando@stanford.edu

Abstract

The process to coordinate rescue and relief efforts after a disaster is generally manual and time consuming. Manual inspection of aerial imagery is an improvement upon that, but is also slow and error-prone if the impacted area is expansive. We propose a two step Convolutional Neural Network (CNN) framework that identifies buildings in affected areas and determines whether or not they are likely to be damaged. This framework improves upon existing work by identifying damaged structures from raw aerial imagery without requiring heuristic based methods for detection and combining that with building damage evaluation. We apply this to Hurricane Maria, which damaged more than a third of the homes in Puerto Rico.

1 Introduction

After a crisis or natural disaster occurs, relief mobilization usually begins by deploying large rescue teams to perform windshield surveys. Quickly identifying impacted areas is a crucial problem when deciding where and when to send aid and focus rescue efforts, potentially saving lives or further infrastructural damage. Our project aims to address this with automatic damage assessment.

The input to our CNN-based framework is a raw aerial RGB image. The final output is a set of building bounding boxes and a binary label indicating whether or not the building in the bounding box is damaged. Our framework is split into two components: building detection and building damage classification. The aerial image is fed into a CNN object detection model, which outputs bounding boxes around buildings. The bounding boxes are used to crop out images of the individually detected buildings in the input and then fed to a CNN classification model to output labels of damaged/not damaged.

2 Related work

This problem has been approached from a few different perspectives using deep learning techniques, but they have focused on individual stages of the building damage analysis problem. In Doshi et. al. [9], larger "grid" areas are classified as damaged or not damaged based on the pixel difference between satellite images before and after the disaster. With each pixel classified as building vs non-building, they rely on significant pixel differences before and after the disaster, which may not occur when some structural damage occurs but buildings are still standing, or where their original segmentation is not accurate, as represented in their results. To compensate for this, the authors split images in grids, which they manually annotated and then evaluated damage within. Another approach developed by Cao et. al.[7] focused on individual building damage classification, which takes images of single buildings and then classifies them as damaged or not damaged. The dataset they used was from Hurricane Harvey. However, this relies on having pre-processed satellite images that contain only one building. They reported exceptional results, however upon inspection, their test images of damaged buildings had surrounding brown flood water, which is not the same as our case. The contribution of our paper is to combine both of these approaches and have a framework go from unprocessed imagery to building extraction and damage classification.

3 Dataset and Features

As we could not find a single dataset with building outlines, building images, and damage coordinates, we combined several data sources into a unified dataset to use throughout the framework: raw aerial RGB imagery taken soon after Hurricane Maria landed in Puerto Rico [16], Puerto Rico building footprints [4], and damage coordinates after Hurricane Maria, as assessed by FEMA [6]. Additionally, the building detection model included aerial building footprint imagery in Christchurch, New Zealand from Aerial Imagery for Roof Segmentation [8] We also processed a satellite imagery from Digital Globe [1], however it needed significant further manual processing as most tiles were either too blurry, covered in clouds, or required significant orthorectification. As such, the aerial imagery ultimately used in this paper is solely from NOAA and AIRS.

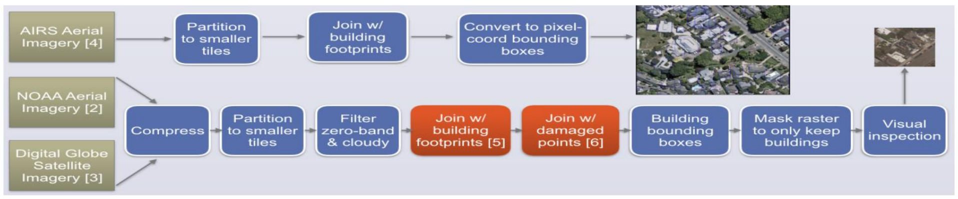
AIRS Aerial Imagery [4]
Partition to smaller tiles
Join w/ building footprints
Convert to pixel-coord bounding boxes
NOAA Aerial Imagery [2]
Digital Globe Satellite Imagery [3]
Compress
Partition to smaller tiles
Filter zero-band & cloudy
Join w/ building footprints [5]
Join w/ damaged points [6]
Building bounding boxes
Mask raster to only keep buildings
Visual inspection

Figure 1: Creating a dataset based on Hurricane Maria and Puerto Rico.

The original GeoTIF tiles available from NOAA are 20,000 x 20,000 pixels. In order to process them more efficiently, we compress and retile them into smaller 1,024 x 1,024 pixel images. We then filter the resulting tiles to remove blank images,. All tiles retain geocoding data, so we can map each tile to its corresponding lat/lon bounding box and to the crowdsourced OpenStreetMap building footprints from [4]. For damage ground truth, we downloaded GDB vectors with observations from a FEMA [6] assessment, including coordinates the location and level of damage. Joining all data sources together gives us tiles annotated with a variable number of buildings for building detection, and smaller images with one building per image labeled as either damaged or not damaged.

3.1 Building detection

The building detection training set also included the AIRS dataset, consisting of 951 tiff images containing 220,000 buildings with corresponding bitmasks to represent the building footprints. Each image is 10,240 x 10,240 pixels. Each image contains many buildings, so we tile the images into 16 equally sized compressed images, each 2560 x 2560 pixels, leading to 15,216 images.

We converted the bitmasks to bounding boxes using connected components with scikit-learn [17] and taking the minimum and maximum x,y coordinates with a margin of 10 pixels on each side, as the classification model can benefit from the surrounding appearance in detecting damage, as seen in [7].

We split the NOAA dataset into 18,767 images for the training, 2000 for the validation, and 347 for the test set. With the AIRS data, the training set has a total of 33,983 images. The test set contains only 347 images to maintain a balanced set of damaged and non-damaged buildings and after filtering for destroyed buildings, we were only able to obtain these test images. We decided to use 480 x 480 for the detection model due to the resolution that it preserves, while also fitting in memory and the smaller input size for the classification model, since it is a smaller portion of the overall image that contains one building.

3.2 Damage Classification

The damage classification dataset consisted of 128 x 128 pixel images containing a single building, derived from NOAA as above. In order to include some of the surroundings from each building, we crop a rectangle shape with a shapely [10] radius parameter of 0.00024 over the building footprint.

Each building image is labeled as either damaged or not damaged given the coordinates from [6]. Unfortunately, the labels from FEMA were not very accurate; there were clearly damaged buildings that went unreported by their visual assessment. Because of this, we visually assessed the dataset and removed images that were mislabeled or we could not visually identify as damaged or not damaged. The resulting dataset was of 2406 buildings for training, 428 for validation, and 384 for test. As there were more non-damaged buildings in the overall dataset, all train/validation/test sets consist of 60% damaged and 40% non-damaged buildings.

4 Methods

We run two CNNs in succession, one for building detection and another for building damage classification. The aim of the first model is to output a set of bounding boxes for individual building extraction. These can be used to mask the original aerial rasters and create smaller images containing only one building, to be classified as either damaged or not damaged by the second model.

4.1 Building Detection

The object detection detection model we use is a RetinaNet [14] with a ResNet101 [11] backbone. RetinaNet [14] consists of a backbone network and two convolutional subnetworks. The backbone network has a Feature Pyramid Network [13] built on top of it and extracts a convolutional feature map from the input image. The two subnetworks are both fully convolutional networks used for classification and bounding box regression. The loss function unique to this model is called the focal loss, which tries to address the issue of the heavy class inbalance between class objects and the background class in images and is used for the classification subnetwork. In practice it is weighted by a term α_t and defined as:

$$FL(p_t) = -\alpha_t(1-p_t)\gamma \log p_t.$$

where, p_t equals p, the predicted class probability, if the label equals the class of the example used for prediction and $1-p$ otherwise. The bounding box regression subnetwork uses a smooth $L1$ loss, defined as

$$L_\delta(a) = \begin{cases} \frac{1}{2}a^2 & \text{if} |a| \leq \delta \\ \delta(|a| - \frac{1}{2}\delta) & \text{otherwise} \end{cases} \tag{1}$$

We used the a default framework value of 3.0 for δ. The total loss is the sum of the focal loss and smooth $L1$ loss.

Our implementation uses a RetinaNet provided by the computer vision company Fizyr [3] and pretrained on the ImageNet dataset. It is implemented in Keras [2] with a Tensorflow [5] backend. We transfer learned on our combined training dataset of the NOAA Hurricane Maria and AIRS datasets and retrain the entire model. We attained improved mAP using both datasets, 0.49 vs 0.43 with only NOAA. We also visually inspected the results and found that including the AIRS images improved the true accuracy of the bounds since the NOAA dataset had many misaligned bounding boxes and those in the AIRS dataset's bounds were systematically more accurate. The images from the two datasets look visually similar.

We also decided to retrain the entire network after experimenting with retraining only the subnetworks, as seen in Table 1 with the frozen backbone.

4.2 Damage Classification

As a baseline, we use the CNN architecture and pre-trained model developed by Cao and Choe [7]. The model was trained on images containing a single building from satellite images taken after Hurricane Harvey in Houston, which makes this a very similar task to ours. However, the damage extent between Hurricanes Harvey and Maria and the infrastructure in Houston and Puerto Rico differs enough that this baseline model has an accuracy of 61.4% on our test set.

The baseline model has four convolutional layers with a ReLU activation function, followed by two fully-connected layers and an output layer with a sigmoid activation function. It is trained on 10,000 images using Adam optimization with mini-batches of size 20 and learn rate of 0.001, 50%

dropout, and L2 regularization in the fully-connected layer with $\gamma = 10^{-6}$. It uses accuracy as the performance metric and a binary cross-entropy function to compute loss, defined as

$$L(y, y') = -\sum_i (y'_i \log(y_i) + (1 - y'_i) \log(1 - y_i)).$$

After running several transfer learning experiments on this model, we arrived at the architecture below for the second stage in our framework. Given our small dataset and the relatively poor baseline performance on our test set, making the network deeper with an additional fully-connected layer resulted in a significant performance boost. The convolutional layers have the following dimensions: 148x148x32, 72x72x64, 34x34x128, and 15x15x128, with 3x3 convolutions and 2x2 max-pooling. The output is then flattened onto a layer of 6272 nodes, followed by a 1024 dense layer, another 512 layer, and finally the output layer with a 1D label vector. To address overfitting, we use a 0.5 dropout at every CONV/MAXPOOL layer and following the first two fully connected layers.

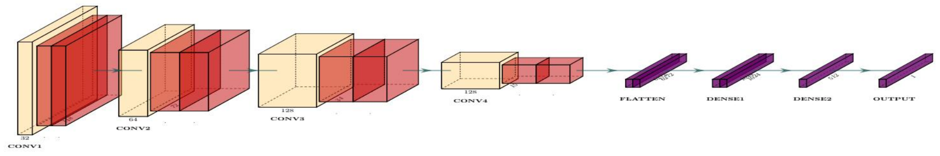

Figure 2: Damage classification model architecture.

5 Experiments & Results

5.1 Building Detection

We experimented with different RetinaNet backbones, which portions of the model to update in training, batch sizes, learning rates, anchor boxes and training datasets to use when training the RetinaNet model. We chose to focus on the RetinaNet [14] model due to its demonstrated advantages in accuracy over the current state of the art two-stage detection models, such as variants of Faster R-CNN [19] and Feature Pyramid Networks [13], while maintaining the speed of previous one-stage models, such as YOLO [18] and SSD [15]. The best performing model from our experimentation had a ResNet101 backbone with batch size of 2, learning rate of 0.0001 with decay when the loss plateaued, and training data consisting of the NOAA and AIRS datasets.The metric we used to evaluate this model was mean average precision (mAP), which is the mean across all classes of the average precision of the predicted bounding boxes ranked by predicted class confidence with a correct bounding box if the intersection of union, IoU, threshold is met. [12]. We used a threshold of 0.25 for the IoU. The rationale behind this and the bounding box margin mentioned previously was that we do not need very precise bounding boxes. We care more about obtaining the detection, since we add a margin when cropping out the individual buildings. For backbone network selection, we focused on ResNet [11] due to the easier training of a large network provided by the residual connections. We experimented with ResNet50 and ResNet101 and whether or not to freeze the backbone of the

	ResNet101	ResNet50	Frozen ResNet101
Total loss	1.5277	1.6626	1.8822
Regression loss	1.2484	1.3483	1.5300
Classification loss	0.2793	0.3143	0.3522

Table 1: RetinaNet Backbone: ResNet101 vs ResNet50 vs frozen ResNet101, 20 epochs, lr=1e-4

network during training. As seen in Table 1, the non-frozen ResNet101 attains lower loss, which is expected since it is a larger model and we have sufficient training data. We did not see issues of overfitting when using the full training dataset, with both training and validation mAP converging to about 0.743, using the 0.25 IoU threshold, and to 0.493 when using the 0.5 IoU threshold.

The batch size of 2 was constrained by being the largest batch size we could use before running out of memory on a Quadro P1000 GPU. Training with a larger batch size of 8 and 16 on a CPU took much longer to train. We tested a few learning rates, and decided upon $1e-4$ since that gave a final loss of 1.569, while 1e-5 gave 1.932, 1e-3 did not not converge, and 1e-6 was too slow to converge and stopped.

After training our final model, with the hyperparameters mentioned above, for 20 epochs, the total loss loss, classification loss, regression loss oscillated around 1.523, 0.2793, 1.2484 respectively, while the validation mAP converged to 0.743 with the 0.25 IoU threshold. The results can be seen in Figure 3. Figure 4 shows an example of the inference output on an image from the NOAA validation set (red are predicted, green are labels). Visually it appears to capture most of the buildings. In images where the performance is not as good, we find blurry images, tall sky scrapers or very large buildings and generally buildings with significantly different appearances. This may be due to the their scarcity in the training set.

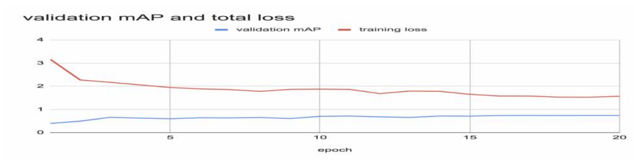

Figure 3: Training loss and validation mAP

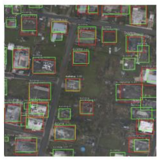

Figure 4: Building detection inference on NOAA imagery. Red are predictions; green are labels

5.2 Damage Classification

We applied transfer learning on the baseline model using the Keras library [2] with TensorFlow backend on an NVIDIA K80 Tesla GPU with 26GB memory on a quad-core CPU machine. We started by freezing the convolutional base in the baseline model and re-initializing the top fully connected layers, then training with Adam (learn rate 0.001) with mini batches of size 32. As in the original model [7], the original 128 x 128 pixel images are projected onto 150 x 150 pixel images as input to the model to account for geo-feature to pixel inconsistencies. We also experimented further training the last convolutional layers, using the original baseline weights as initialization values.

Our original dataset used a "zoom" shapely [10] envelope radius parameter of 0.00016 around the building footprint ground truth, which proved to be too close to the building as tuning schemes resulted in little improvement. A radius parameter of 0.00024 yielded building images of similar proportions as those used to train the baseline models, and significantly improved performance. As our training and validation sets are very small, the fully connected layers were prone to overfitting. To address this, we added on-the-fly data augmentation using Keras' ImageDataGenerator and an additional dropout stage (with dropout of 0.5) in the fully-connected 1024 layer. As this is a relatively small model on a small dataset, we were able to iterate quickly and perform numerous experiments, a subset of which is summarized below. We chose the model with 3 fully-connected layers and 3 frozen

Data Augmentation	Zoom	FC Layers	Frozen CONV	Epochs	Accuracy	F1
No	0.00016	2	4	100	53.9%	61%
Yes	0.00016	2	4	100	52.9%	58%
Yes	0.00024	3 (addtl Dropout)	3	100	80.5%	84.1%
Yes	0.00024	3	3	100	83.3%	86.3%
Yes	0.00024	3	3	200	83.6%	86.7%
Yes	0.00024	3	2	100	83.9%	87.6%
Yes	0.00024	3	3	400	84.1%	86.8%

Table 2: Damage classification model experiments sample results. Columns: whether data augmentation was used or not, shapely zoom radius parameter when extracting building images from original rasters, number of fully connected layers at top, number of frozen convolutional layers, number of train epochs, and resulting accuracy and F1 scores.

convolutional layers as our final model, given it has the highest accuracy at 84.1% and lowest loss of 0.4. It has an F1 score of 86.8%, recall of 89.3%, and a 84.5% precision. Interestingly, the model in which only two convolutional layers were frozen (meaning the last two conv layers were further trained) has a higher F1 score due to its high recall of 97%. We decided to favour the higher accuracy model given this model's much lower precision of 79.6%. We also experimented using RMSProp

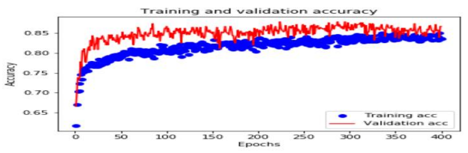

Figure 5: Training and Validation Accuracy

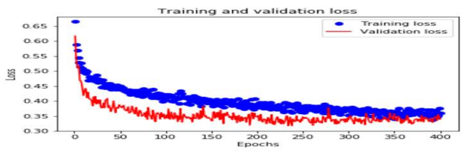

Figure 6: Training and Validation Loss

instead of Adam as an optimizer, however it resulted in noisy learning and a lower validation accuracy plateauing around 68%.

6 Discussion/Conclusion/Future Work

We found that combining two CNN model performing different tasks to ultimately achieve one objective can yield positive results. However, performance would likely have been better with more training data. Raw satellite imagery requires a non-trivial amount of pre-processing and visual inspection for which we did not have enough resources nor time. We chose Hurricane Maria to test current literature performance on a different environment (climate and infrastructure) than those previously evaluated (mainland US). Damage classification base model performance was poor in spite of similarity of the task; however, transfer learning on images from new environment proved effective. A greater diversity of disaster types and environments would make the model more robust. This deep learning technique can have a big impact in helping first responders identify first-pass, worst-hit areas immediately following a disaster.

We found that the best performing algorithm for the building detection was a RetinaNet with ResNet101 backbone, learning rate of 0.0001, achieving a mAP@0.25 of 0.743 and mAP@0.5 of 0.493. As for damage classification, our best performing experiment yielded an accuracy at 84.1% and loss of 0.4, with an F1 score of 86.8%.

Regarding future work, gathering more data and cleaning the dataset further is the next step that we would take. Misaligned bounding boxes, blurry resolution, and blank portions that still had ground truth building bounding boxes likely affect the model performance. We noticed this when replacing DigitalGlobe dataset with the cleaner NOAA dataset. Additionally, the damage labels from FEMA were unreliable, as many buildings that were clearly destroyed were not included in their damage assessment. Further improvements can also be made on both the building detection model as well as the classification model, such as training more complex networks for the two subnetworks of RetinaNet and testing other architectures for the classification model, both using more computing resources.

7 Contributions

Richard experimented with and trained the building detection model. Angelica tested different architectures and trained the damage classification model and designed the poter. Both authors contributed to the image data preprocessing for the aspects related to our respective components, such as bounding box conversion, individual building crop extraction, and tiling and filtering the NOAA and AIRS datasets. Source code can be found at https://github.com/chardch/DisasterDetection/.

References

[1] Digital globe. hurricane maria satellite images.

[2] Keras: Deep learning for humans. https://github.com/keras-team/keras. Software available from https://github.com/keras-team/keras.

[3] Keras implementation of RetinaNet object detection. https://github.com/fizyr/keras-retinanet.

[4] Openstreetmap bundled by geofabrik. puerto rico building footprints. http://download.geofabrik.de/.

[5] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[6] Federal Emergency Management Agency. Hurricane maria damage assessments, 2017.

[7] Quoc Dung Cao and Youngjun Choe. Deep learning based damage detection on post-hurricane satellite imagery. *CoRR*, abs/1807.01688, 2018.

[8] Qi Chen, Lei Wang, Yifan Wu, Guangming Wu, Zhiling Guo, and Steven L. Waslander. Aerial imagery for roof segmentation: A large-scale dataset towards automatic mapping of buildings. *CoRR*, abs/1807.09532, 2018.

[9] Jigar Doshi, Saikat Basu, and Guan Pang. From satellite imagery to disaster insights. *CoRR*, abs/1812.07033, 2018.

[10] Sean Gillies. Manipulation and analysis of geometric objects in the cartesian plane.

[11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.

[12] Paul Henderson and Vittorio Ferrari. End-to-end training of object class detectors for mean average precision. *CoRR*, abs/1607.03476, 2016.

[13] Tsung-Yi Lin, Piotr Dollár, Ross B. Girshick, Kaiming He, Bharath Hariharan, and Serge J. Belongie. Feature pyramid networks for object detection. *CoRR*, abs/1612.03144, 2016.

[14] Tsung-Yi Lin, Priyal Goyal, Ross Girshick, Kaiming He, and Piotr Dollar. Focal loss for dense object detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PP:1–1, 07 2018.

[15] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott E. Reed, Cheng-Yang Fu, and Alexander C. Berg. SSD: single shot multibox detector. *CoRR*, abs/1512.02325, 2015.

[16] National Oceanic and Atmospheric Administration. Hurricane maria aerial imagery, 2017.

[17] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine Learning in Python . *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[18] Joseph Redmon, Santosh Kumar Divvala, Ross B. Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. *CoRR*, abs/1506.02640, 2015.

[19] Shaoqing Ren, Kaiming He, Ross B. Girshick, and Jian Sun. Faster R-CNN: towards real-time object detection with region proposal networks. *CoRR*, abs/1506.01497, 2015.