CS 838 Project Final Report
Post-Disaster Building Damage Assessment

Mehmet Furkan Demirel, Matthew Dutson, Shri Shruthi Shridhar
Department of Computer Sciences
University of Wisconsin–Madison
{demirel, dutson, shridhar2}@wisc.edu

Abstract

When a natural disaster strikes, it can drastically affect communication and transportation. Satellite imagery allows authorities to pinpoint and categorize damage to coordinate an effective response. However, manually sifting through images to localize damage is prohibitively labor-intensive. In this work, we use computer vision to automatically detect and annotate damaged buildings. We achieve competitive performance by using a modified U-Net architecture with a class-weighted loss. The idea for this project was motivated by the xView2 competition, in which we participated.

1. Introduction

Quick and accurate situational information is critical when coordinating an effective disaster response. In order for responders to effectively allocate resources, they need to know the location and severity of damage over the affected area. Disruption of local communication and transportation infrastructure makes on-the-ground damage assessment dangerous and slow. Satellite imagery offers an effective means to asses damage without physically visiting the affected area. However, because disasters often cover a large land area, analysts must manually sift through an enormous amount of data to acquire actionable information.

The goal of the xView2 competition is to address this analytic bottleneck, calling on machine learning practitioners to develop deep models which can automatically localize and quantify post-disaster building damage. The challenge uses the xBD dataset [6]. Released in June 2019 by the Defense Innovation Unit and other humanitarian organizations, this dataset contains high-resolution (1024 by 1024) pre and post-disaster satellite images annotated with building locations and damage scores. It contains 850,000 building polygons from six different types of natural disaster across 15 countries. The training set has 9168 pairs of images with labeled building polygons and the test set has 933 pairs of unlabeled images.

Figure 1: An example of pre and post-disaster satellite imagery in xBD

The competition uses the Joint Damage Scale (JDS) for scoring and evaluation [6]. Each pixel is annotated on a scale from 0 to 4, where 0=no building, 1=undamaged, 2=minor damage, 3=major damage, and 4=destroyed. Submissions are scored using a combined F1 score: 30% localization F1 plus 70% damage classification F1. Localization F1 is computed by considering any prediction 1-4 as "positive." Damage classification F1 is computed by taking the harmonic mean of the F1 scores for classes 1-4.

2. Related Work

Sublime et al. propose an almost fully unsupervised step-by-step method for processing very high resolution images [9]. They introduce a clustering algorithm to process image segments and to find multi-scale clusters on VHR images matching the different scales of interest. Their method focuses on object-based analysis rather than pixel-based analysis.

Vetrivel et al. demonstrate that integrating convolutional neural networks and 3D point clouds yields better performance compared to using a CNN independently [10]. They apply a "multiple-kernel-learning framework" to combine

image and scene characteristics for classification and illustrate the significance of 3D point cloud features.

A CNN change detection framework was developed by researchers from CrowdAI and Facebook [5]. Their deep learning model can automatically predict the magnitude of damage in an area after a disaster. Their framework relies only on readily-available datasets for common man-made features in satellite imagery (e.g. buildings and roads). They compute relative change between multiple snapshots captured before and after a disaster. This is in contrast to other research which applies convolutional neural networks on manually-labeled, disaster-specific datasets.

Bai et al. present a deep learning algorithm for semantic segmentation of high-resolution remote sensing images and pixel-based fine-scale damage mapping [3]. Their method uses a U-Net architecture to map earthquake and tsunami damage. Sublime and Kalinicheva [8] propose a damage assessment model which uses a single-shot multibox detector (SSD) with pre-training and data augmentation.

3. Methods

3.1. Data Preprocessing

The competition dataset defines buildings as lists of vertex coordinates. In the first phase of preprocessing, we converted these lists of vertices into categorical masks. Some buildings had no explicit damage label; for these we assumed the "undamaged" class. Following the mask generation we performed a validation split. Given the relatively small number of training images, we only set aside 10% for validation.

The training dataset represents only a handful of natural disasters. The competition organizers do not clearly state whether the test set is drawn from this same set of disasters. This presented us with a difficult choice. If the test set were *not* drawn from the same set of disasters as the training set, then randomly sampling training instances for validation could lead to validation scores which overestimate performance on the test set. This could be mitigated by placing certain disasters exclusively in the validation set. However, were the test set drawn from some unknowable distribution different than the training set, then the machine learning problem would not be well posed. For this reason, we randomly sampled validation instances from the training set without considering the disaster or disaster type (resulting in approximately stratified sampling over disasters).

3.2. Architecture

Due to its past success in segmenting satellite images, we chose a U-Net architecture [7, 1]. We implemented and trained the network from scratch in TensorFlow and Keras. We made several adjustments to the original implementation described in [7]. The first was replacing ReLU activations with leaky ReLU activations (leak ratio $\alpha = 0.3$). We found that this improved both accuracy and the speed of training convergence.

The original U-Net performs a single up-front padding of the input and then does not pad any convolutions. The up-front padding is defined such that the network output has the same size as the unpadded input. We found it was much less cumbersome to simply mirror-pad feature maps before each convolution. Although these approaches are not technically equivalent, we believe it is unlikely that this would cause a significant change in accuracy.

The xView2 competition is somewhat unique in that the model is provided with not one, but two images (before and after the disaster). The competition baseline model performs localization and damage detection in sequence, passing only the pre-disaster image to the localization model and only the post-disaster image to the damage model [2]. This is clearly non-optimal, especially for damage detection. Whether a building is damaged is a function not only of the building's appearance after the disaster, but also of the *change* in its appearance during the disaster. To allow our model to compare the pre and post-disaster images, we passed both to a single network as a six-channel input.

To motivate this choice, imagine that we want the network to learn to subtract the pre-disaster image from the post disaster-image (it is reasonable to think that damage has something to do with visual change). This operation can clearly be represented by a convolution (by strategically choosing the kernels' central weights from $\{-1, 0, +1\}$). Another advantage to the six-channel approach is that it makes the network robust to clouds or other obfuscation in one of the two images (see the last row of Figure 6 as an example).

3.3. Loss Function and Metrics

The pixels in the training data are highly skewed toward the "no building" and "undamaged" classes (see Figure 2). For this reason, using a standard cross entropy loss leads to a model which places a disproportionate emphasis on correctly recovering background pixels. In fact, we found that training with standard cross entropy led to a model which predicted background *everywhere*. We experimented with several methods for class rebalancing, namely:

- Weighting by inverse class frequency
- Manually adjusting class weights as hyperparameters
- The class balanced loss approach from [4]

The heavy skew in class frequencies also motivated the use of metrics other than accuracy (e.g. a model which predicts background everywhere would achieve 97.1% accuracy). We chose to track per-class precision and recall, damage and localization F1 scores, and the overall competition metric (see Section 1).

Class	Description	Frequency
0	No building	97.1%
1	Undamaged	2.20%
2	Minor damage	0.242%
3	Major damage	0.254%
4	Destroyed	0.168%

Figure 2: Pixel class frequencies

3.4. Training and Experiments

We trained our model over the course of several weeks on the Euler supercomputer (GeForce GTX 1080 GPU). With this hardware, a single training epoch takes about 45 minutes. We trained using two different approaches: (1) simultaneous and (2) separate training of localization and damage detection. Experiments with (2) are ongoing; therefore, we will focus here on the results obtained with approach (1).

Due to computational and time constraints, we were unable to perform a systematic grid search over the space of hyperparameters. Instead we performed what amounts to a coordinate descent search, optimizing each hyperparameter individually while holding the others constant. The following hyperparameters were varied:

- Optimizer (SGD or RMSProp)
- Learning rate and decay schedule
- Data augmentation (with or without)
- Class rebalancing strategy
- Manual class weight values
- Number of convolutional layers at each U-Net level
- Depth of the U-Net

After altering a specific hyperparameter, we observed and recorded its influence on training. Based on the results, the change was either discarded or preserved in future experiments. A complete history of these experiments can be seen at `https://bit.ly/2YCckdR`. Figure 3 shows the training configuration of the best-performing model.

Among these hyperparameters, the class rebalancing strategy seemed to have the greatest effect on performance. After trying all three approaches outlined in 3.3, we found that manually tuning the class weights produced the best results. These manual class weights consequently represented another set of hyperparameters requiring tuning.

4. Results

Table 4 shows the per-class precision and recall for the best-performing model. Our model is very successful in predicting pixels that belong to class 0, 1, and 4. It is less successful when recovering instances of class 2 and 3. This is not surprising given the subtle visual cues associated with these classes (minor and major damage). Sample validation set predictions from this model can be seen in Figure 6.

Hyperparameter	Value
Optimizer	RMSProp
Learning rate and schedule	10^{-5} with decay to 10^{-6}
Data augmentation	None
Class weighting strategy	Manual
Manual class weights	[0.05, 1.0, 3.0, 3.0, 1.0]
Convolutions per level	2
U-Net depth	5

Figure 3: Best-performing training configuration

Class	Description	Precision	Recall
0	No building	0.9987	0.9542
1	Undamaged	0.4353	0.9109
2	Minor damage	0.3123	0.1159
3	Major damage	0.5660	0.1516
4	Destroyed	0.3639	0.6183

Figure 4: Validation results for the best-performing model

We also submitted our model's test predictions to the official xView2 competition website, and ranked in the top 50 for a few weeks until we were eventually outscored by other competitors. Table 5 shows our test scores compared to the competition baseline model. Our model outperforms the baseline model in damage detection and overall F1, and shows competitive performance in localization. It is worth mentioning that even though these scores were from our submission that produced the highest combined F1 score, other models achieved better individual performance on localization and damage.

Model	Combined	Localization	Damage
Ours	**0.491**	0.679	**0.410**
Baseline	0.284	**0.805**	0.061

Figure 5: xView2 competition scores

5. Conclusion

In this work, we have built and trained a modified U-Net architecture in order to automate the process of post-disaster building damage assessment. Our experiments demonstrate that our model shows competitive performance in building localization and damage classification.

Nevertheless, we believe that we can achieve even better performance with more experiments–including a more systematic hyperparameter search and different network architectures. To this end, we are currently performing exper-

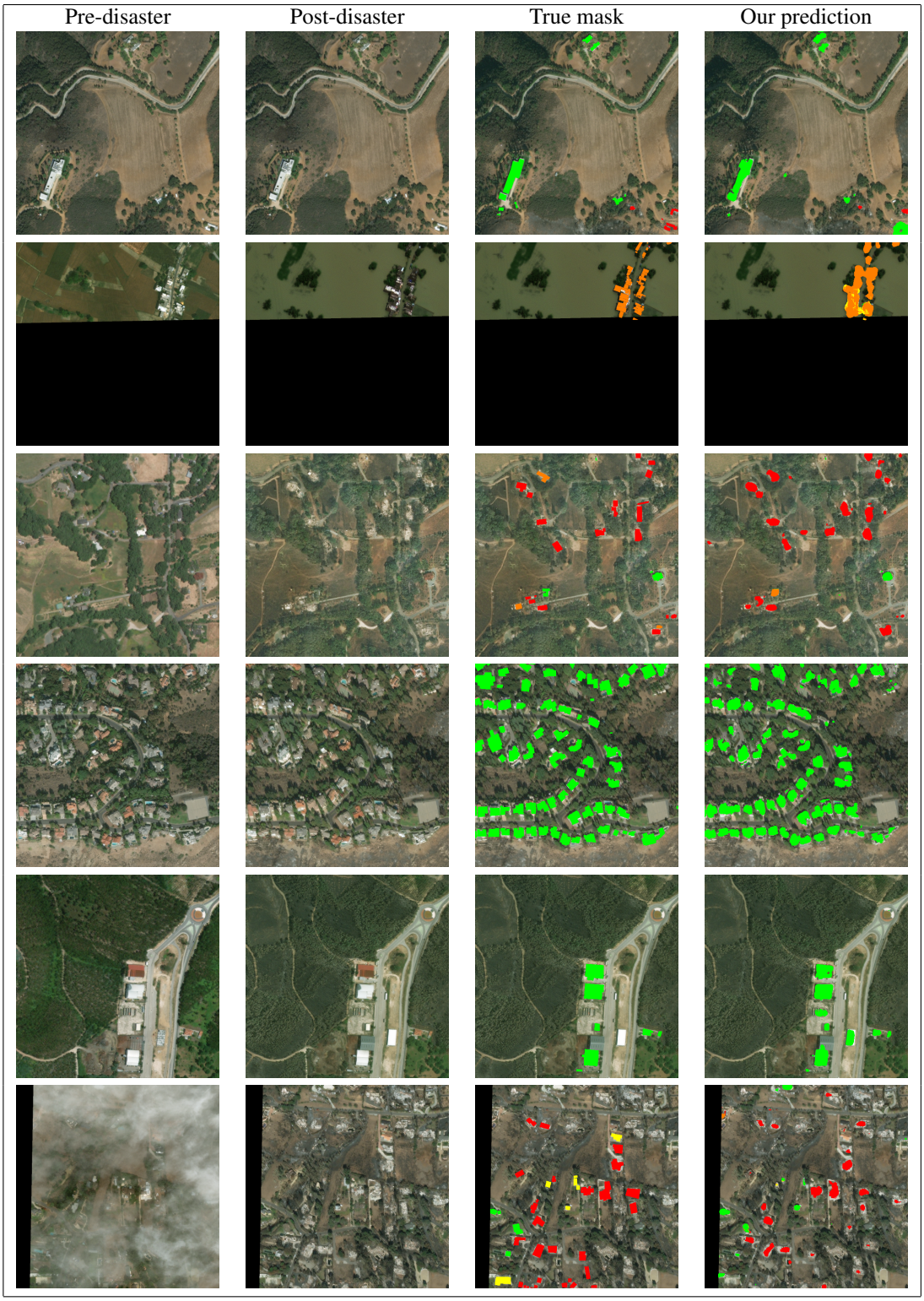

Figure 6: Validation predictions of the best model
■ no damage, ■ minor damage, ■ major damage, ■ destroyed

iments to train separate models for localization and damage detection, and have found promising preliminary results.

We are also experimenting with more drastic modifications to the U-Net architecture. For example, while the original U-Net does allow for some context around each pixel to be accounted for, it fails to capture true global context, especially for high-resolution images. To address this limitation, we are currently training a model which adds global pooling/unpooling layers at the "bottom" of the U-Net.

6. Source Code Link

`https://github.com/mattdutson/xview2`

7. Group Member Contributions

Mehmet was primarily responsible for running the training experiments and writing testing code, Matthew for implementing the model and training pipeline, and Shri for the data loading and augmentation pipeline. All group members assisted with background research, this report, and the project presentation.

References

[1] Spacenet on amazon web services (aws). *The SpaceNet Catalog*, Jan 2018.

[2] DIUx-xView/xview2-baseline, Nov. 2019. original-date: 2019-09-25T18:58:26Z.

[3] Yanbing Bai, Erick Mas, and Shunichi Koshimura. Towards operational satellite-based damage-mapping using u-net convolutional network: A case study of 2011 tohoku earthquake-tsunami. *Remote Sensing*, 10(10), 2018.

[4] Yin Cui, Menglin Jia, Tsung-Yi Lin, Yang Song, and Serge Belongie. Class-balanced loss based on effective number of samples. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9268–9277, 2019.

[5] Jigar Doshi, Saikat Basu, and Guan Pang. From satellite imagery to disaster insights. *arXiv preprint arXiv:1812.07033*, 2018.

[6] Ritwik Gupta, Bryce Goodman, Nirav Patel, Ricky Hosfelt, Sandra Sajeev, Eric Heim, Jigar Doshi, Keane Lucas, Howie Choset, and Matthew Gaston. Creating xbd: A dataset for assessing building damage from satellite imagery. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 10–17, 2019.

[7] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

[8] Jérémie Sublime and Ekaterina Kalinicheva. Automatic post-disaster damage mapping using deep-learning techniques for change detection: Case study of the tohoku tsunami. *Remote Sensing*, 11(9):1123, 2019.

[9] Jérémie Sublime, Andrés Troya-Galvis, and Anne Puissant. Multi-scale analysis of very high resolution satellite images using unsupervised techniques. *Remote Sensing*, 9(5), 2017.

[10] Anand Vetrivel, Markus Gerke, Norman Kerle, Francesco Nex, and George Vosselman. Disaster damage detection through synergistic use of deep learning and 3d point cloud features derived from very high resolution oblique aerial images, and multiple-kernel-learning. *ISPRS journal of photogrammetry and remote sensing*, 140:45–59, 2018.