Automating Detection of Flood Damaged Buildings

Anne Evered
MUSA650 Final Project
Spring 2020

***Abstract*—Detecting flood damaged buildings has important applications in emergency response and assessment. However, done manually, the process of determining which buildings in a given area have sustained flood damage can be a labor-intensive and time-consuming one. This paper proposes a set of algorithms for automating this process. In particular, I investigate various machine learning techniques for determining whether a satellite image contains a flood damaged building. This paper uses satellite images from the Houston, TX area after Hurricane Harvey as a case study for model development, validation, and testing. Given the general success of Convolutional Neural Networks (CNNs) for image classification tasks, I primarily explore the application of these networks for this context. Based on this analysis, a CNN model with multiple drop out and regularization steps performs best, with 97% test accuracy, in determining which images showed flood damage. Building on these findings, similar models could be tested on other flood damage areas to test for the generalizability of this model to other storms and regions.**

I. Introduction

ASSESSMENT of damaged buildings is a critical step in emergency response after hurricanes and other severe weather events. Such assessments help guide where response and further evaluation is most needed, as well as provide an overall picture of infrastructure damage to a region. However, determining which buildings have sustained flood damage during hurricane events can be a difficult and time intensive process. When done via surveying of individual buildings, it may also require particular expertise in structural engineering or damage assessment.

Remote Sensing (RS) and Geographic Information Technologies (GIS) technologies provide opportunities for automating parts of this detection process. In particular, machine learning techniques have shown great success for satellite image classification tasks and could potentially be applied in this context.

This paper investigates the use of deep learning techniques for post-hurricane damage assessment, focusing on Houston, TX after Hurricane Harvey (2017) as a case study. Using pre-labeled satellite images from this region, I develop several models for evaluating whether a satellite image contains a building that has flood damage. In particular, I focus on the application of Convolutional Neural Networks (CNNs), which are typically very effective at image classification and segmentatioin tasks. These models proved to have high accuracy (97%) at this task, as well. In the following paper, I will discuss the dataset and models developed in more detail, as well as the relative performance of these models and the significance of these results. Overall, findings from this analysis suggest the potential for CNN and other deep learning models for supporting emergency management response.

This work was completed on May 4, 2020 as part of the completion of course curriculum for MUSA 650 'Machine Learning in Remote Sensing.' Anne Evered (e-mail: aeevered@seas.upenn.edu), is currently a masters student in Data Science at the University of Pennsylvania, PA 19104 USA. Project work was completed and submitted on May 8, 2020.

II. Related Works: Convolutional Neural Networks and Satellite Images

Convolutional Neural Networks have shown great effectiveness at tasks involving analysis of visual imagery, such object detection and image classification. These networks achieve high accuracy when compared with other methods, in part due to their capability for joint feature and classifier learning.[1]

One area CNNs have had high success in application is in the analysis of satellite images. These networks have been applied with high accuracy, for example, in the analysis of satellite images for post-disaster analysis. In recent work, neural networks have been used to automate coastal change detection because of sea level rise or severe storms.[2] Deep learning and remote sensing techniques have also been applied to assess damage and risk from earthquakes, landslides, tsunamis and wildfires.[3]

Of perhaps most relevance to this paper, the researchers who created the dataset used in this analysis performed their own analysis applying CNN techniques to differentiate flood damaged buildings from non-damaged buildings. Using these models, they achieve a testing accuracy of 97%.[3] This paper seeks to expand on these results by applying additional CNN networks to the dataset. In particular, I experiment with different regularization and drop-out steps, training parameters, and number of model layers.

III. Data

A. Dataset Overview

The primary dataset for this analysis consists of 23,000 RGB satellite images from Houston, TX after Hurricane Harvey hit the region in 2017. The original source and citation of the data can be found on the IEE data port [4]. For this project, though, I downloaded the cleaned and labeled version of the data from Kaggle: https://www.kaggle.com/kmader/satellite-images-of-hurricane-damage.

One of the remaining challenges with automated building detection is that even with advances in computer vision and remote sensing, substantial pre-processing work is often required to apply these techniques. For this reason, a pre-labeled, cleaned dataset is used rather than the raw satellite images.

Improved techniques for this pre-processing work constitute another area of important work.

In this case, the original dataset was labeled by volunteers in a crowd-sourcing project. Each square-sized image contains a building that could either be labeled as Flooded/Damaged or Undamaged. Based on the labeling of these buildings, the images are divided into two categories (damage and no_damage). Examples from both categories are shown in Figures 1 and 2. Additional examples are included in Appendix A.

Fig. 1. Sample image from label "no damage."

Fig. 2. Sample image from label "damage."

These sample images (and the additional images in the Appendix) illustrate how subtle the differences are between the images showing flood damaged buildings and those showing buildings that do not have flood damage. Given how similar the *damage* and *no damage* images appear, I expected fairly low performance accuracy when applying the CNN models and was surprised at the high performance of these models on differentiating between the two types of images.

The particular locations of each of the images are based on known building coordinates from publicly available data. These building coordinates are provided in the name of each image. Figure 3 shows the latitude and longitude of each image, color-coded by the damage/no_damage label.

Lastly, the dataset is pre-divided into testing, training, and validation segments. In particular, the full dataset is divided into four parts: training data (balanced), validation data (balanced), an unbalanced test dataset and a balanced test dataset. The breakdown of the number of images in each label and testing/training/validation category can be seen in the following tables.

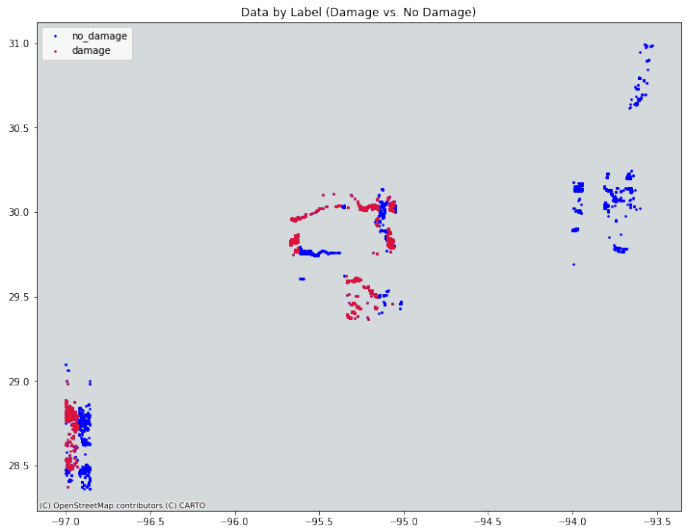


Fig. 3. Graph showing the latitude and longitude of each of the images in the dataset.

Label	Number of Images
damage	15,000
no damage	8,000

Test/Train/Validation Set	Number of Images
test (balanced testing)	2,000
test_another (unbalanced testing)	9,000
train_another (training data)	10,000
validation_another (validation data)	2,000

Test/Train/Validation Set	Label	Number of Images
test	damage	1,000
	no_damage	1,000
test_another	damage	8,000
	no$_d amage$	1,000
train_another	damage	5,000
	no_damage	5,000
validation_another	damage	1,000
	no_damage	1,000

B. Data Preparation

The prepared dataset was already relatively clean with the images already labeled and divided into training, testing and validation sets. Given this, minimal additional data preparation was done. Standardization such that each pixel value is between 0 and 1 was completed by dividing each value by 255. The predefined data partitions for training, validation, and testing data were used for all model development and testing.

IV. Methods

In approaching the task of creating a model to differentiate between flood-damaged and non-flood-damaged buildings, I focused primarily on Convolutional Neural Networks (CNN) and related models. As discussed above, the main reason for focusing on CNNs is that these models are typically very effective at image classification and detection tasks. Another advantage of CNN models over other machine learning techniques is that they are robust to shifts in the location of features in the training images. This is a desired model features here given that the buildings are not always in the center of the input images.

There are many parameters that can be adjusted when developing CNN models both for the structure of the model (number of layers, filter size) and for the training process. In building models to test, I started with a first convolutional layer with a relatively small filter size (3x3) based on the hypothesis that what differentiates the images are small and local features. To avoid overfitting, I also tried several models that had one or more dropout layers within the network.

In the following section, I will discuss three of the models I tested. The specific features of each of these models are based on the above reasoning, while also representing a diversity of approaches to this problem. For building each of the models below, I used the popular neural network framework Keras and ran the training, testing and validation processes in Google colab. The balanced training and validation segmentation provided in the dataset are used for each training/testing process. I use the unbalanced testing dataset for most of the model comparison given that it is larger, but also do a final test with the smaller balanced testing dataset in the last section. Accuracy score (i.e. what percentage of images in the test set are correctly identified) is used as the primary evaluation metric of the models. Based on a initial evaluation of the three models' relative performance, I then did further parameter tuning and performance evaluation with the best of these models.

A. Training and Evaluation

Each of the following models are initially trained using 20 epochs and batch size of 32 (prior to parameter sweep). I also use categorical crossentropy as the loss function and softmax as the activation for each of the models.

1) Model 1: The first model tested is a fairly simple CNN model with one pooling layer and two drop-out layers (see Fig. 4). This model gives a test accuracy of 95.089%. I have also provided additional metrics and the confusion matrix for this model below. Graphs of the validation accuracy and loss per epoch can be seen in Appendix C.

Metric	Value
Accuracy Score	95.089%
Precision Score	96.368%
Recall Score	98.175 %
F1 Score	97.263 %

	(predict) positive	(predict) negative
(actual) positive	704	29
(actual) negative	146	7854

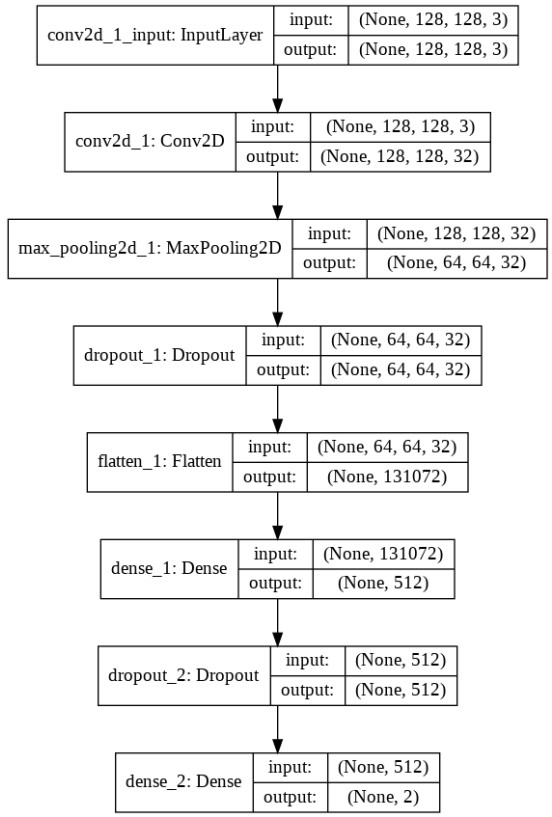


Fig. 4. Summary diagram of fewer layer CNN model, first model tested (Model 1).

2) Model 2: I also tested a model with converted grayscale images (with a dense layer first). A summary of this model can be seen in Figure 5.

This model did not appear to do as well at differentiatinng the damage. In comparison to Model 1, the test accuracy for this model was under 90% (83.756% on the unbalanced testing dataset). This suggests that some information is lost when removing the RGB channel information from the dataset. The validation accuracy and loss per epoch when trained with batch size of 32 over 20 epochs can be seen in the Appendix.

Again, additional metrics for this model are shown below and graphs of the validation accuracy and loss per epoch are included in the Appendix.

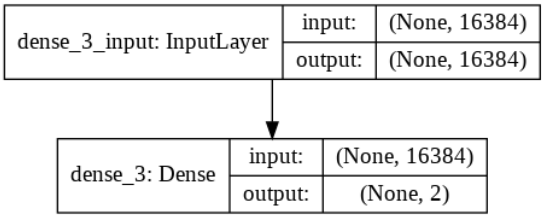


Fig. 5. Summary diagram of CNN model trained tested with grayscale images (Model 2).

Metric	Value
Accuracy Score	83.756%
Precision Score	91.867%
Recall Score	89.663 %
F1 Score	90.752 %

	(predict) positive	(predict) negative
(actual) positive	365	635
(actual) negative	827	7173

3) Model 3: After these initial two models, I added additional layers to the first model to see if this might improve the accuracy further. With additional layers added to the CNN (model shown in Figure 6), a test accuracy of 96.578% was achieved. Again, graphs of the validation accuracy and loss per epoch are included in the Appendix.

This model, similar to VGG-16, has multiple drop out and maxpooling layers. One reason for the higher accuracy than Model 1 may be that the additional drop out and pooling layers help to further prevent overfitting.

Parameter Tuning for Model 3

Based on initial evaluation of these models, I performed further parameter testing on the third model. The results of this additional evaluation are provided in the below table and in Figure 7.

Batch Size	Number of Epochs	Accuracy
32	10	96.211%
32	20	94.967%
32	50	86.667%
64	10	95.456%
64	20	91.422%
64	50	96.611%
128	10	91.377%
128	20	96.267%
128	50	97.667%

Of the parameters tested, batch size of 128 and 50 epochs gave the highest testing accuracy in the parameter sweep. Therefore, I use these parameters when training the model for the following final model evaluation metrics.

Based on this preliminary analysis, I trained and evaluated a final model (using Model 3 as the CNN network architecture and batch size: 128, epochs: 50 for the training parameters). The results of this final evaluation are shown in the following section, along with some of the images that the model misclassifies and why this might be the case.

B. Results

1) Performance on Unbalanced Testing Data: When tested on the unbalanced training dataset and with the tuned hyperparameters (number of epochs: 50, batch size: 128), Model 3 gives a testing accuracy of 96.877%. Figures 8 and 9 show the loss and accuracy on the test data (here the validation dataset) per epoch. The accuracy improves over time and the loss decreases over time as expected, particularly in the first 10-20 epochs.

Given that this testing dataset is unbalanced, checking other metrics such as Precision and Recall, as well as examining the ROC curve, in addition to the overall accuracy, is particularly important. Specifically, these metrics provide a check to make sure the high testing accuracy is not because the model is favoring the over-represented label. These metrics

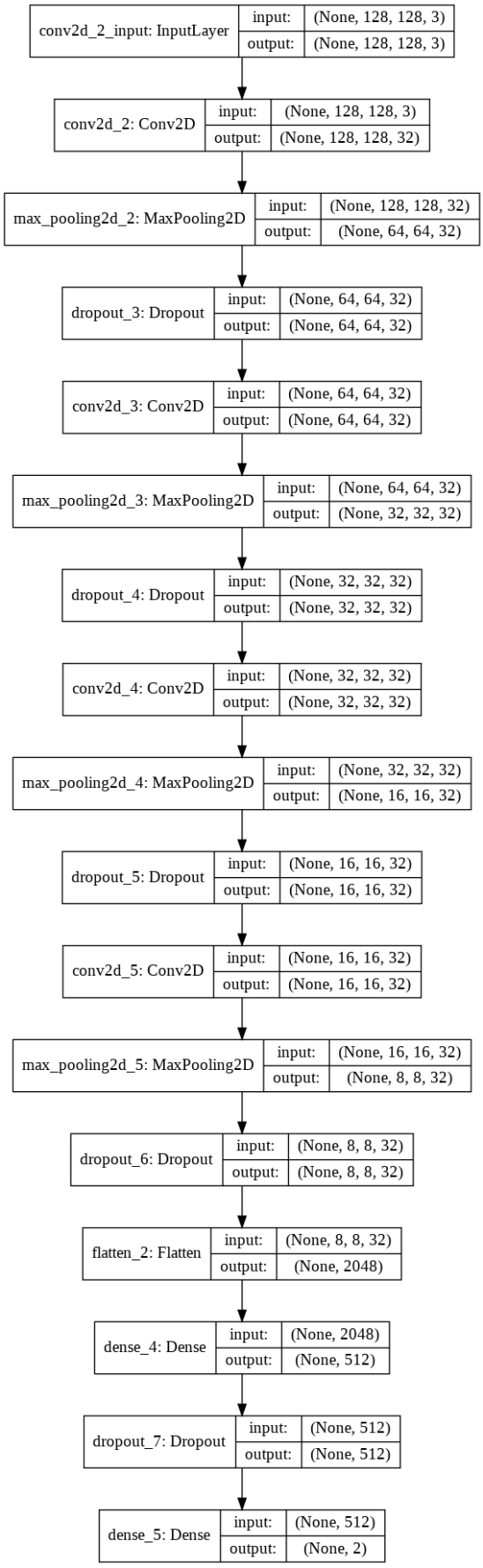


Fig. 6. Summary diagram of CNN model that gave the highest testing accuracy of models tested (Model 3).

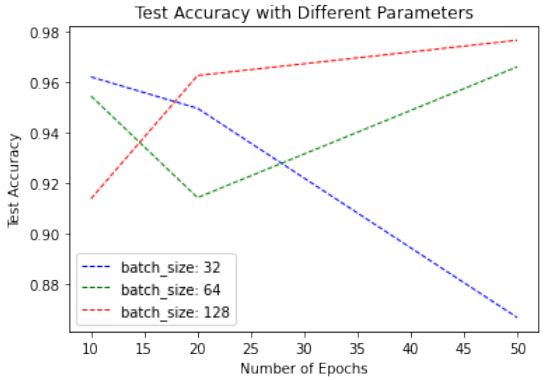


Fig. 7. Plot showing testing accuracy using different epoch numbers and batch sizes for Model 3.

can be seen in the table below and the ROC curve in Figure 10.

Metric	Value
Accuracy Score	96.877%
Precision Score	99.272%
Recall Score	97.200 %
F1 Score	98.225 %

v

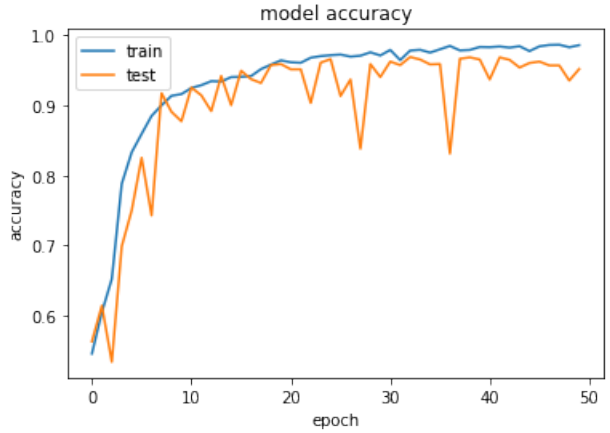


Fig. 8. Training and testing (validation data) accuracy by epoch for Model 3 with tuned hyperparameters.

Accuracy by Label

The following table provides the number of correct predictions in each of the two label categories. The accuracy is slightly higher (97.200%) for the damage label category than for the no damage label category (94.300%).

True Label	Correct Predictions	Total	Accuracy
damage	7776	8000	97.200%
no damage	943	1000	94.300%

Presented as confusion matrix:

	(predict) positive	(predict) negative
(actual) positive	943	57
(actual) negative	224	7776

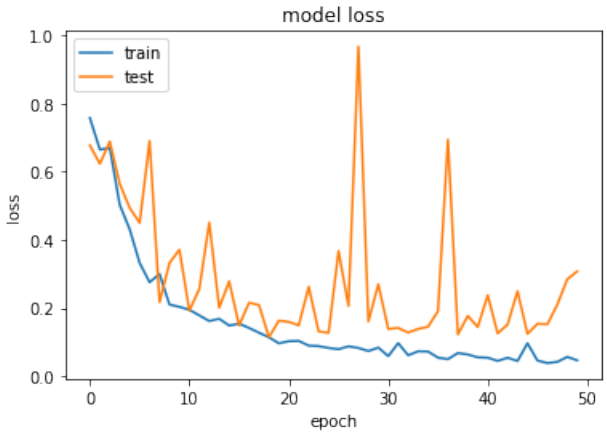


Fig. 9. Training and testing (validation data) loss by epoch for Model 3 with tuned hyperparameters.

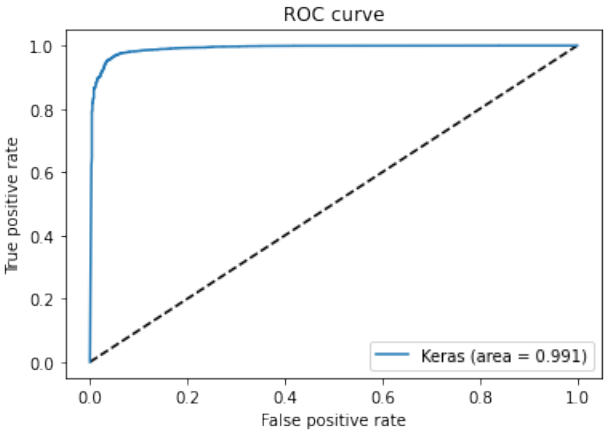


Fig. 10. ROC curve for Model 3 on unbalanced label data.

Example Misclassifications

While CNN models can be difficult to interpret in terms of which features of the image are leading the model to classify an image one way or another, looking at which images the model *did not* correctly classify can provide some insights. Figures 11 and 12 provide examples of images in the unbalanced testing data that were not correctly classified by Model 3. Figure 11 shows an image that was labeled in the dataset as having no damage, but was misclassified as having damage, whereas Figure 12 shows an image that was labeled as damage but was misclassified as having no damage.

Appendix B provides additional examples of misclassified images. Based on these examples, it would seem that, as expected, the model may be more likely to misclassify images where it is ambiguous even to a human whether there is flood damage to the shown building. For example, Figure 11 may suggest a possible data quality issue as it is unclear on whether this image should have originally been labeled as damage or no_damage.

2) Performance on Balanced Testing Data: Lastly, as a final test, I also ran the tuned model (Model 3 architecture, tuned

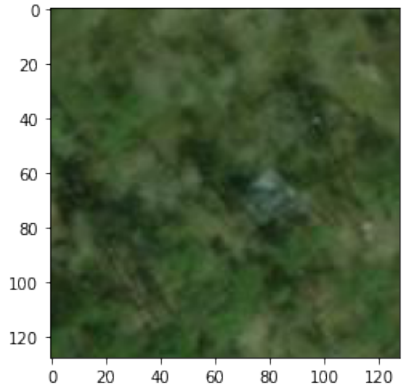


Fig. 11. This is an image labeled as having no damage. It was mislabeled as having damage.

Fig. 12. This is an image labeled as having damage. It was mislabeled as having no damage.

hyperparameters) on the balanced test data. Again, this test dataset has 1,000 images labeled as showing flood damage and 1,000 images labeled as showing no flood damage. While this balanced data set is not as large in terms of number of images as the unbalanced test dataset, I did this final check to confirm that the balanced testing dataset was not skewing the results heavily, giving a misleadingly high accuracy. This final test gives a testing accuracy of 97.600%. Precision, Recall and F1 Scores, along with the confusion matrix, are provided below. These outputs suggest that the model also performs well on the balanced data.

Metric	Value
Accuracy Score	97.600%
Precision Score	97.600%
Recall Score	97.600 %
F1 Score	97.600 %

	(predict) positive	(predict) negative
(actual) positive	976	24
(actual) negative	24	976

V. Discussion

A. Discussion of Results

In summary, like previous work, this project shows how convolutional neural networks show great promise at being able to accurately differentiate between images that show post-hurricane flood damage and those that do not. Again, based on an initial look at the images, I was surprised at how well the CNNs were able to perform on this task given how subtle the differences between the labeled images seem.

However, a fairly complex CNN is required, with the fewer layer model not performing as well as the more complex one. In addition, the RGB channels of the images appear important in achieving a high accuracy value. This may be because the color channels help the model initially identify what is a building and what is landscaping.

One interesting result is that for all of the models that performed relatively well, the validation accuracy is very high even on the first epoch. This may suggest a high redundancy in the data, with the model not learning much additional information in subsequent epochs.

B. Areas for Further Analysis

This project focused on only one region and time frame as a case study: Houston, TX after Hurricane Harvey in 2017. Both the testing and training process were restricted to using images from this area and time. Therefore, a fitting next step for further analysis would be to test the models discussed on satellite images from other regions and contexts. For example, it would be interesting to see how the classification model performs on building images from Houston, TX after a different storm or severe weather event or, for instance, from New York City after Hurricane Sandy. This may also help to identify which aspects of the images the model is most relying on for classification. The models could also be trained and tested on larger datasets.

Additional deep learning techniques could also be applied to this problem in future analysis. Data augmentation could be used, for example, in the data preparation stage to reduce possible overfitting and expand how generalizable the models are. For instance in their paper, Cao, Quoc Dung et al., discuss applying random rotation, horizontal flip, vertical and horizontal shift, shear, and zoom to the training images.[3] Additional techniques such as UNet or transfer learning could also be applied.

Further improvement in the speed and accuracy of pre-processing work needed to run these types of CNN networks (i.e. creating and labeling testing and training datasets), will increase the impact of automated detection of post-storm building damage. Particularly given that damage assessment and response after severe weather events is often time sensitive, in order for the application of machine learning to have a positive impact in this realm, an efficient pre-processing data pipeline is critical. This step will help expand the reach and benefit of these new tools for aiding post-hurricane and severe storm emergency response.

ACKNOWLEDGMENTS

I would like to thank Dr. Guray Erus, teaching assistant Mikhail Hayhoe, and fellow students from MUSA650 Spring 2020 for providing the education, guidance, and feedback necessary for this project.

APPENDIX A
ADDITIONAL EXAMPLE IMAGES

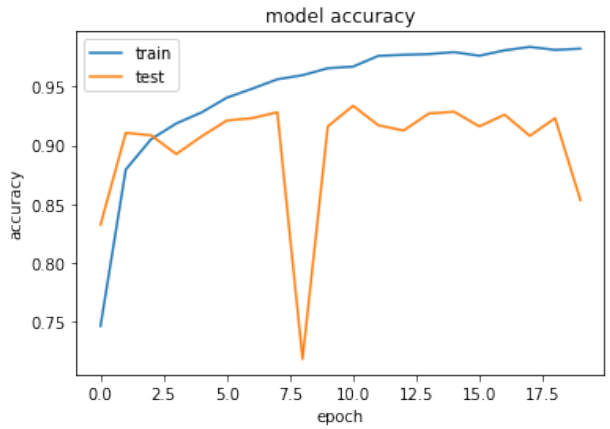
Additional sample images from both label categories.

APPENDIX B
ADDITIONAL MISSCLASSIFICATION IMAGES (MODEL 3)

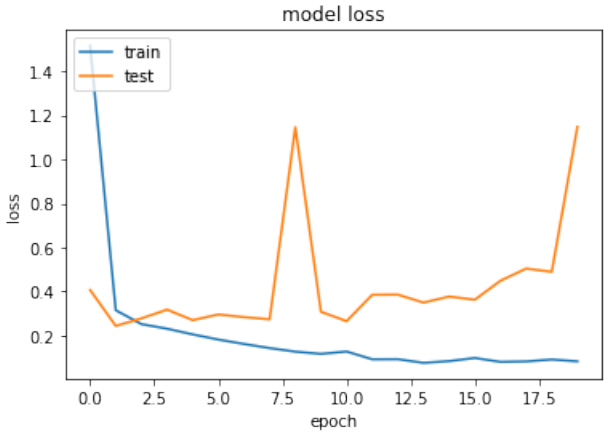
Additional examples of images misclassified by Model 3.

APPENDIX C
ADDITIONAL PERFORMANCE PLOTS

A. Additional Model Performance Graphs - Model 1

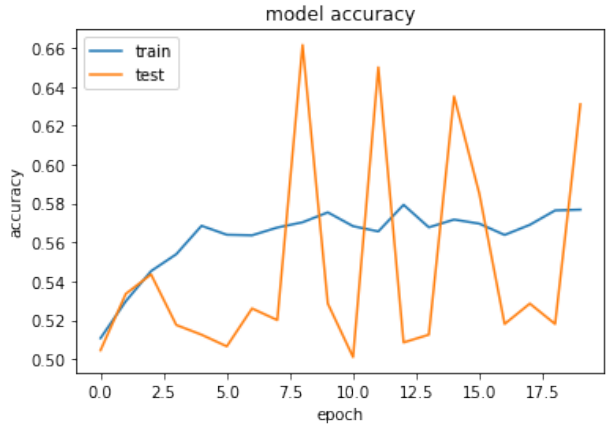


Training and testing (validation data) accuracy by epoch for Model 1.

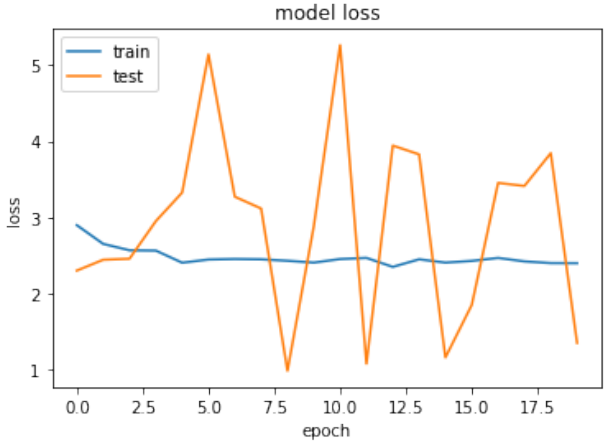


Training and testing (validation data) loss by epoch for Model 1.

B. Additional Model Performance Graphs - Model 2

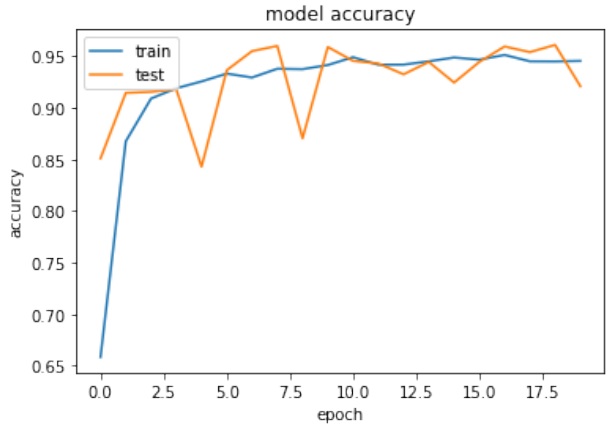


Training and testing (validation data) accuracy by epoch for Model 2.

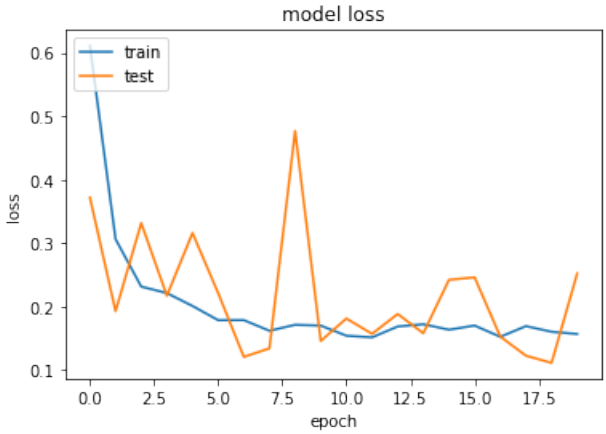


Training and testing (validation data) loss by epoch for Model 2.

C. Additional Model Performance Graphs - Model 3



Training and testing (validation data) accuracy by epoch for Model 3.



Training and testing (validation data) loss by epoch for Model 3.

REFERENCES

[1] Gua, Jiuxiang, et al., "Recent Advances in Convolutional Neural Networks." In: *CoRR* https://arxiv.org/pdf/1512.07108.pdf (2017).

[2] Park, Sung Jae et al., "Machine Learning Application for Coastal Area Change Detection in Gangwon Province, South Korea Using High-Resolution Satellite Imagery." In: *J. of Coastal Research* pp. 228-235. https://bioone.org/journals/Journal-of-Coastal-Research/volume-90/issue-sp1/SI90-028.1/Machine-Learning-Application-for-Coastal-Area-Change-Detection-in-Gangwon/10.2112/SI90-028.1.short (2019).

[3] Cao, Quoc Dung and Youngjun Choe., "Building Damage Annotation on Post-Hurricane Satellite Imagery Based on Convolutional Neural Networks." In: *CoRR* http://arxiv.org/abs/1807.01688 (2018).

[4] Quoc Dung Cao, Youngjun Choe, "Detecting Damaged Buildings on Post-Hurricane Satellite Imagery Based on Customized Convolutional Neural Networks", IEEE Dataport, 2018. [Online]. Available: http://dx.doi.org/10.21227/sdad-1e56. Accessed: May. 07, 2020.