Individual Report

Flooding Damages Detection from Post-Hurricane Satellite Imagery Based on Convolutional Neural Networks

Bixuan Huang
(Group 4, cooperate with Weining Hu)

Abstract

In this project, we applied convolutional neural networks (CNN) to detect flooding damages from the 2017 Hurricane Harvey satellite images. To train the model, we used a dataset posted on Kaggle consisting of images labeled with two types of condition, which are "damage" or "no damage." We experimented with three different CNN architectures, which are VGG-16, Resnet50, and a custom convolutional architecture. Our best model, the custom model, was able to achieve an accuracy of over 90% on the test set. In this paper we describe the dataset we used, presented related work, explain how we iterated on our model to come up with the final version, and finally evaluate our results.

1 Introduction

When a hurricane makes landfall, emergency managers need to assess the situation and flooding damage so that they can make efficient response to the disaster and allocate resources for future rescue and recover work. Traditional practice to assess the disaster situation and damage largely relies on emergency response crews and volunteers to drive around the affected area and make assessments accordingly. However, this process is labor intensive and time-consuming. Thanks to the blooming development in the filed of computer vision studies, scientists find another way to improve the efficiency of building disaster assessment. More specifically, they applied image classification algorithms to distinguish damaged buildings and flooding area from the ones still intact. In this project, we explored varied state-of-the-art convolutional neural networks and use them to train and classify the 2017 post-Harvey Hurricane satellite images.

1.1 Dataset

We obtained the data[1] from Kaggle, which gives 128 X 128 X 3-pixel RGB satellite images of the Greater Houston area before and after Hurricane Harvey in 2017. The dataset also labels all images into two classes, either no damage or damage. The training set[2] consists of 10,000 examples, while the validation set[3] is composed of 2,000 examples. The test set[4] contains 9,000 images. The class distribution of the dataset is displayed in Table 1:

[1] https://www.kaggle.com/kmader/satellite-images-of-hurricane-damage
[2] The training set folder is named as "*train_another*."
[3] The validation set folder is named as "*validation_another*."
[4] The test set folder is named as "test_another."

Labels / Folders	Damage	No Damage
Training set	5,000	5,000
Validation set	1,000	1,000
Test set (Unbalanced)	8,000	1,000

Table 1: Distribution of the dataset

As it is shown in Figure 1, each of the training set and validation set has an even distribution across the two classes. Therefore, there is no need to apply image augmentation to increase the images from the underrepresented class. The dataset contains a balanced test set (titled as "test" in the zip file) and an unbalanced test set (titled as "*test_another*"). To preserve the original distribution of the two classes, we chose to test our models on the unbalanced dataset rather than the balanced test set. A few example images from the dataset are show in Figure 1.

Damage No Damage

Figure 1: Example images from the dataset

1.2 Problem Statement

The goal of this project is to predict whether a given image from the hold-out test set contain flooding damage or no damage. Additionally, we compare the performance differences between a custom architecture with fewer convolutional layers and the pre-trained models with more convolutional layers. Our evaluation matrix is the accuracy for each class, supplemented with a confusion matrix that highlights which class is better recognized than the other.

1.3 Related Work

The dataset was released by two students, Quoc Dung Cao and Youngjun Choe, from the University of Washington. They were the first ones who applied image classification algorithms to classify the 2017 Hurricane Harvey satellite images. In December 2018, the *Institute of Electrical Engineering Journal* accepted their research paper and published their work, which highlighted the findings that convolutional neural networks can automatically annotate flooded/damaged area on post-hurricane satellite images with approximately 97% accuracy.[5] Cao and Choe trained their networks using the *Keras* framework and on a shorter version of the VGG-16 network.

[5] Cao, Quoc Dung, and Youngjun Choe. "Building Damage Annotation on Post-Hurricane Satellite Imagery Based on Convolutional Neural Networks." *arXiv preprint arXiv:1807.01688* (2018). URL: https://arxiv.org/abs/1807.01688

2 Individual work - Custom Model

To make new contributions to the existing work that were completed by Cao and Choe, my teammate used *PyTorch* as our framework to train neural networks on. In addition to VGG-16, she also trained our model using another pre-trained model, Resent50. Lastly, I also built a custom model with less convolutional layers to compare with the pre-trained models.

3 Individual work – Detail

3.1 Model Architecture

As for the custom model, we built a network that contains 8 layers with weights. As illustrated in Figure 11, the model follows the block design pattern, where one convolutional layer (filter of 3×3 or 6×6)), one ReLU activation layer, one Batch Normalization layer, one pooling layer (factor 2, Max Pooling or Average Pooling) are stacked as a single learning block. The whole model is formed by three learning blocks, followed by two fully connected layers. We chose Stochastic gradient descent (SGD) as the model optimizer and CrossEntropyLoss as the function loss. The figure below displays only the major weighted layers of the network. Activation function layers and dropout layers are eliminated from the diagram for aesthetic purposes.

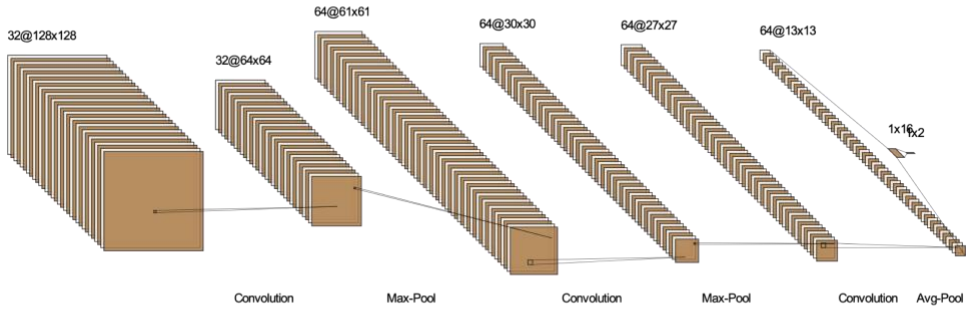

Figure 2. LeNet Style Diagram of the Custom Network[6]

3.2 Hyperparameters Tuning

Once we had established a successful base architecture, we sought to improve our performance through hyperparameter searches. The following present the main hyperparameters adjusted.

Batch Size: Based on the batch size above for the pre-train model, we started modeling by loading 50 images to the Dataloader. Less batch size limits our choice in Mini-Batch Size, so we increased the batch size. With the experience of increasing the batch size in the Dataloader to train pre-train model, to spend our time wisely, we decided to limit the batch size to 100.

[6] The diagram was created using this online tool: http://alexlenail.me/NN-SVG/LeNet.html

Learning Rate: Similarly, the performance of the Torch-lr-finder did not perform as good as we expected. We experimented to find the best learning rate by tracking and comparing the learning rate manually and finally, we determined that 0.01 was the most appropriate learning rate for the custom model.

Mini-Batch Size: We experimented with a batch size as large as 100. However, the performances are less desirable than using a smaller mini-batch size. Our results show that a smaller mini-batch size below 25 achieves better performances than using a larger batch size. In the final model, we used the size of 4 for mini-batch.

4 Results

To assess the performance of our models, we use a combination of accuracy and confusion matrix.

4.1 Accuracy

I am able to achieve good results with the accuracy rate of custom model as more than 90%. This result reveals that even a small convolutional neural network can perform well on this particular dataset.

4.2 Confusion Matrix

I examined the confusion matrix for the test set to understand better where misclassifications occur. The custom model achieved approximately 95% accuracy in identifying damage images.

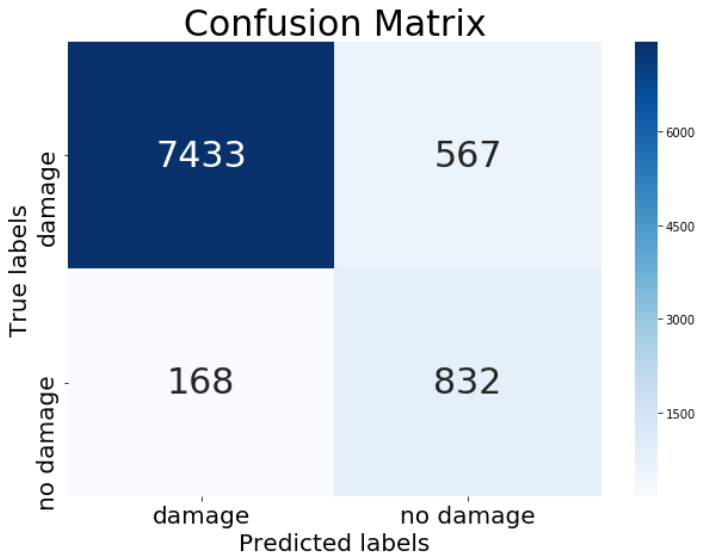

Figure 3. Confusion Matrix (Custom Model)

Models	Damage Accuracy	No Damage Accuracy	Damage Error	No Damage Error
Custom Model	92.91%	83.2%	7.08%	16.8%

Table 2. Classification accuracy and error across all classes

4.3 Loss and Accuracy Over Time

The custom model yielded a slightly higher validation loss than the training loss. It could be a sign that the custom model is overfitting. For future work, we can try to improve the custom model by adding more dropout/batch normalization layers, applying more data augmentation, or decreasing the number of parameters of the model.[7]

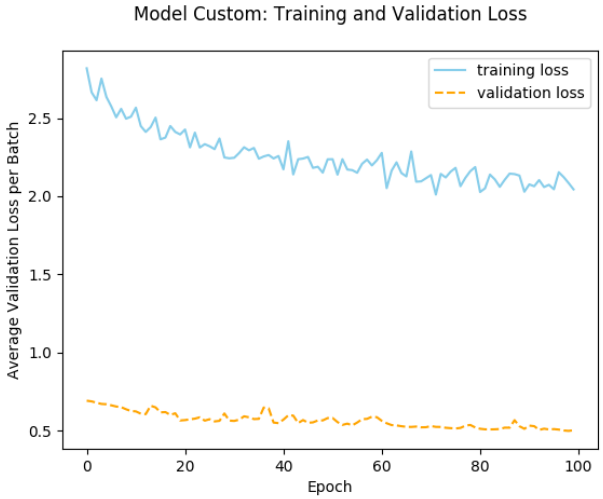

Figure 3. Train loss and validation loss comparison

5 Summary and conclusions

In this project, I designed a custom convolutional neural network from scratch to recognize flooding damage from images that were taken before and after Hurricane Harvey in 2017. The results demonstrated that my model was able to achieve acceptable results in comparison to the original work completed by Cao and Choe (their best model achieved a 97% accuracy). Among our three models, the custom model with less convolutional layers performed best in detecting flooding damage, while the VGG-16 model performed the best in classifying no flooding damage images.

To further improve model performance, I would like to implement more epochs to train our pre-trained models, offsetting the loss issues mentioned in section 4.3. Additionally, I wish to experiment with more measures to solve the overfitting issues occurred with the existing custom model.

6 Calculate contribution

Lines of code copied from the internet: 200

[7] https://groups.google.com/forum/#!topic/caffe-users/J5205HG7sog

Lines of code modify modified: 120
Lines of code added: 50
The percentage = $\frac{200-120}{200+50} \times 100$

7 References

Deep Learning Based Damage Detection on Post-Hurricane Satellite Imagery:
https://github.com/qcao10/DamageDetection

Binary Face Classifier using PyTorch:
https://hackernoon.com/binary-face-classifier-using-pytorch-2d835ccb7816

TORCHVISION.MODELS:
https://pytorch.org/docs/stable/torchvision/models.html

How can we release GPU memory cache?:
https://discuss.pytorch.org/t/how-can-we-release-gpu-memory-cache/14530