Ben Keel, Jack Rummler, Kate Tanabe
MUSA 650: Geospatial Machine Learning in Remote Sensing
May 5, 2023

Damage Detection: Binarized classification models to detect hurricane flood damage

1. Introduction

With the increasing ramifications of climate change, natural disasters have become more frequent, causing intense damage to critical infrastructure. In the US, the most damaging hurricanes are now three times more frequent than they were 100 years ago.[1] Flooding is a pervasive effect of hurricanes, bringing intense rainfall, high winds, and storm surges. Storm surges, which are caused by the combination of high winds and low pressure, can lead to coastal flooding, adding to the damage.

After a hurricane event, flood damage assessment is critical to ensure that planners and emergency responders can assess the scale of damage, efficiently allocate recovery resources, and identify hot spot areas in need of assessment. Traditionally, damage assessment is done by ground surveying after a flood event, which can often be a laborious process, requiring intense use of capital and resources. However, a faster way to accomplish damage assessment is by using remote sensing classification and machine learning techniques to identify damage.

Leveraging the power of machine learning in remote sensing after a flood event, planners can identify infrastructure that has been damaged and see if there is a spatial pattern to infrastructure damage. This allows more effective deployment of recovery resources quickly after a disaster. To accomplish this, we are building models that can predict whether a building has been damaged or undamaged after a flooding event. These images have binarized, ground truth labels (no_damage and damage), and we are aiming to optimize the accuracy and loss of the models. This means that we want to correctly predict the damage status and minimize the difference between actual and predicted outcomes.

By successfully validating models on these metrics using this data, we can assume that if new data is introduced, it would accurately predict if a building had flood damage, allowing for more effective deployment of resources. This, in turn, saves local governments and emergency management organizations time and resources. Moreover, our analysis is quite granular, planners could use these models and look at the spatial patterns of predicted damage, and be able to derive policy conclusions about at-risk areas to invest in climate resilient mitigation for future disaster.

[1] "How Climate Change Makes Hurricanes More Destructive," Environmental Defense Fund, accessed May 5, 2023, https://www.edf.org/climate/how-climate-change-makes-hurricanes-more-destructive.

2. Data

We got our data from a Kaggle dataset[2] that has satellite images of hurricane flood damage, specifically after Texas' Hurricane Harvey in 2017. The images are various cropped overhead views of homes surrounded by forests, grass, roads, and other man-made infrastructure, and details from the training set may not be present in the validation and test sets. We are utilizing three of the data folders given: train_another, validation_another, and test, which have 5000, 1000, and 1000 of each label, respectively. We believe that with 10000 total training images, and 2000 for validation and testing, we have a robust data source to derive the accuracy of our methodology.

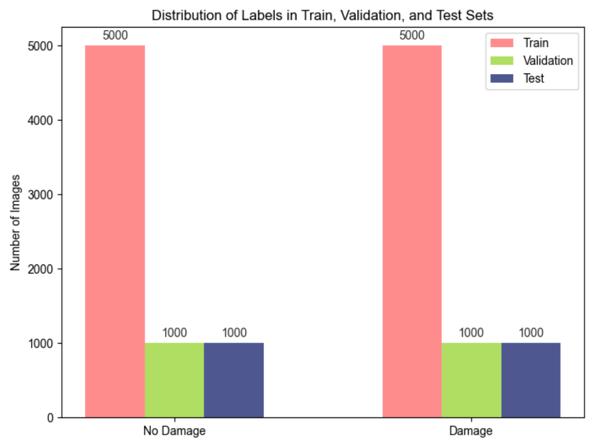


Figure 1. Distribution of data

The data was collected using satellites soon after Hurricane Harvey happened, and were geo-coordinates using aerial-view windows, which shows the data's ability to understand spatial distribution of flood damage.

[2] https://www.kaggle.com/datasets/kmader/satellite-images-of-hurricane-damage

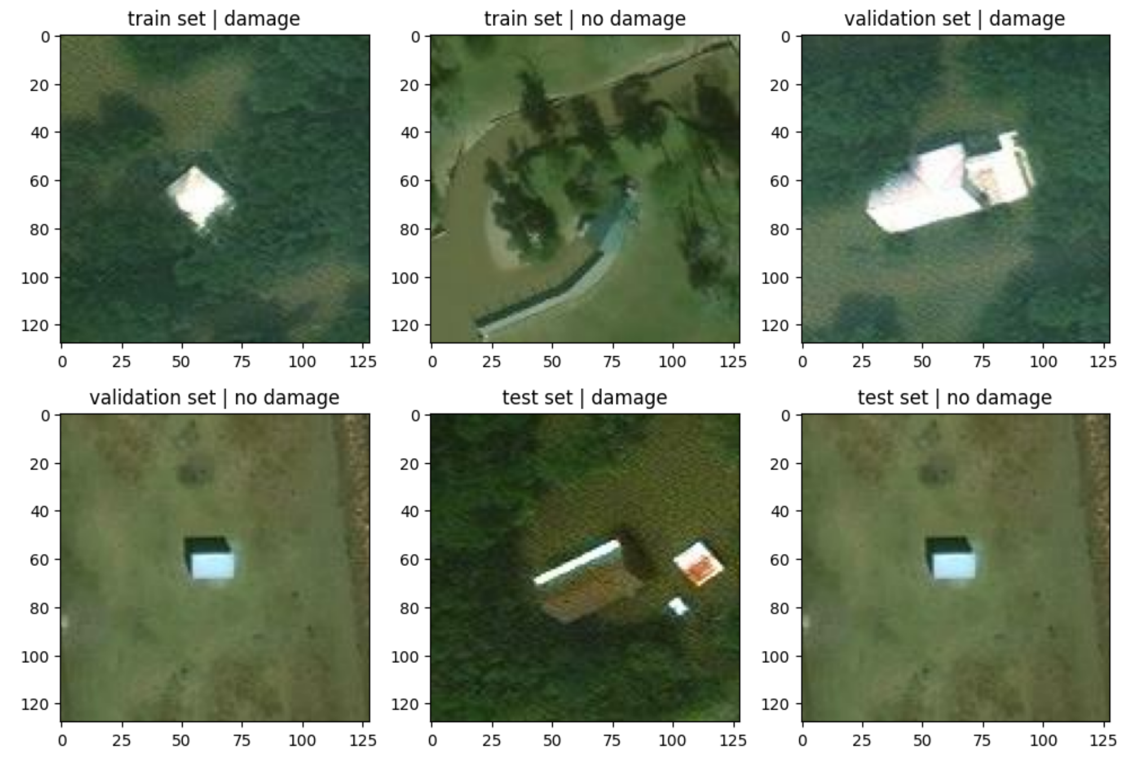


Figure 2: Plots of each label from each data set

3. Methods

Our methodology is to complete a binary classification using a variety of deep learning models. Specifically, we chose three different types of convolutional neural networks to experiment with: a traditional CNN optimized for our data, transfer learning using VGG16, and a residual neural network. We hypothesized that with their strength in image classification tasks, tested model architectures, and ability to address different problems within remote sensing problems, these three models would give us results that would optimize our accuracy and loss metrics.

CNN Model

Our first model is a simple CNN that uses convolutional layers and fully connected layers, while incorporating other layers such as batch normalization and dropout and others to help normalize the data, max pooling to consolidate the convolutions and filters into smaller, sharper signals, and then flattens the data set into multiple dense layers until the final binary output layer.

To make the training set models more generalizable and to give the CNN model a fighting chance with the other more advanced methods, and given the uncertainty of image variation among the training, validation, and test sets, we added a step of data augmentation that adjusted the inputs with zoom and rotation.

Our architecture used two convolution layers among 11 total layers. The convolutions each used a 3x3 kernel with a step of 1, using 4 filters, then 8, and were separated by a batch normalization layer for generalization, a max pooling layer (2x2) to focus any signal the convolutions picked up, and a dropout layer again for generalization. The architecture of our CNN model is presented below:

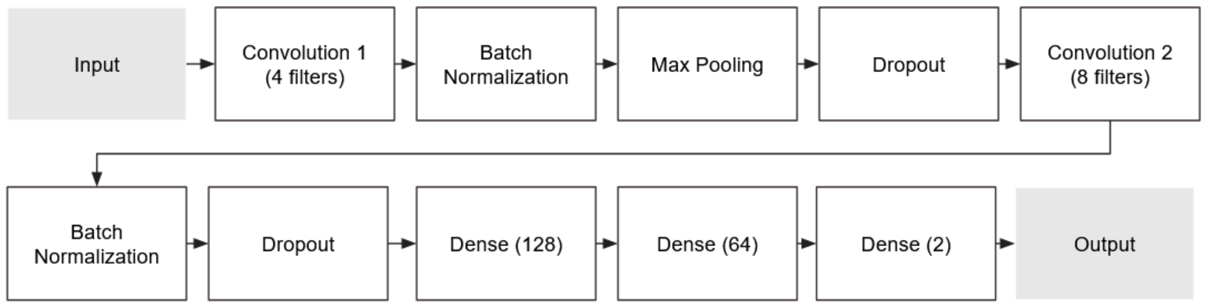


Figure 3. CNN Architecture

<u>VGG16 Model</u>

Our second model uses transfer learning and builds off a convolutional neural network architecture. Transfer learning allows us to leverage a pre-trained model and apply it to another dataset, with the opportunity to add or adjust layers to fit the new dataset. Transfer learning can save time and resources, while often resulting in better performance outcomes.

We used a pre-trained VGG16 model to classify the satellite images of hurricane damage. VGG16 is a type of convolutional neural network (CNN) that uses small convolutional filters and 16 weighted layers. However, there are 24 total layers: 13 convolutional layers, 5 max pooling layers, 1 flatten layer, and 3 dense layers. The convolutional layers are consistently followed by a max pooling layer, which down-samples the images and reduces the risk of overfitting. VGG16 is unique in that it uses smaller filters and strides with padding and max pooling, rather than having a large amount of hyperparameters.[3] We used the Adam optimizer and categorical cross entropy loss. The architecture of our VGG16 model is presented below.

[3] Great Learning, "Everything You Need to Know about VGG16," *Medium* (blog), September 23, 2021, https://medium.com/@mygreatlearning/everything-you-need-to-know-about-vgg16-7315defb5918.

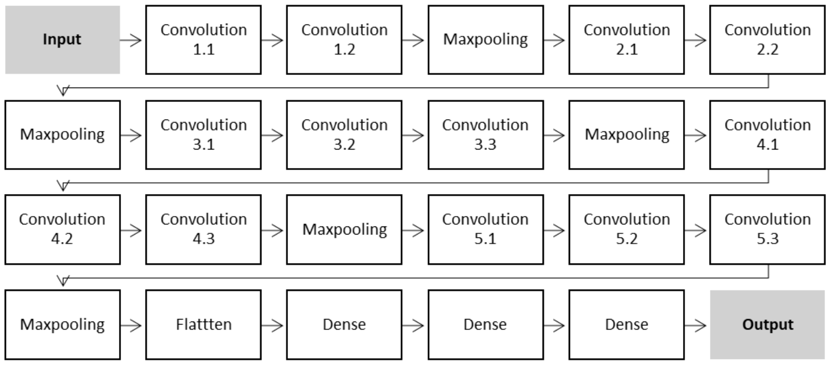


Figure 4. VGG16 architecture

ResNet Model

Our final model is a residual neural network, which is an iteration of a convolutional neural network. ResNet architectures are known for their ability to handle deep networks for feature extraction and classification. ResNet models are good for handling the vanishing gradient problem, which is the issue of when the model during training optimizes to a point where the gradient becomes close to zero, essentially leading to poor performance.[4] The ResNet architecture adds "residual connections" between layers, directly adding one output of a layer to another layer further down the network, allowing for gradients to flow more easily through the network, permitting deeper learning of the images.

In the ResNet architecture, we use convolutional layers and residual blocks with batch normalization and ReLU activation functions after each convolutional layer. The skip functions are the ones that help gradients move throughout the network more fluidly, allowing for deeper learning and preventing the vanishing gradient problem. There are four stages of the architecture: the first stage has one convolutional layer with three residual blocks and three stages after have two residual blocks with a stride of 2 in the first block, each stage with increasing input. The model is compiled using categorical cross entropy and the Adam optimizer.

Given the size of the ResNet model architecture, it is best to check out using this link.

[4] Rachel Zhiqing Zheng, "Beginners' Guide to Image Classification: VGG-19, Resnet 50 and InceptionResnetV2 with TensorFlow," Medium, April 29, 2020, https://towardsdatascience.com/beginners-guide-on-image-classification-vgg-19-resnet-50-and-inceptionresnetv2-with-tensorflow-4909c6478941.

4. Results

CNN Model

The CNN model was somewhat successful in classifying the hurricane damage images. When used to predict the test set images, it has a loss of 0.267 and an overall accuracy of 0.884. Increasing the convolutions further was not effective in increasing the overall accuracy. Interestingly, the validation set accuracy was sometimes higher than the training set during the different epochs. We hypothesize this is due to the data augmentation providing enough image variability to adapt better to unknown data. However, some validation set results were much worse. So overall, the data augmentation helped with the CNN's generalizability to new data, but if it ended on the wrong epoch, then the model may turn out much worse.

Because our following model was VGG16, we refrained from refining the CNN model past this stage of accuracy, else it may resemble our other method too closely and not provide us with a distinct example.

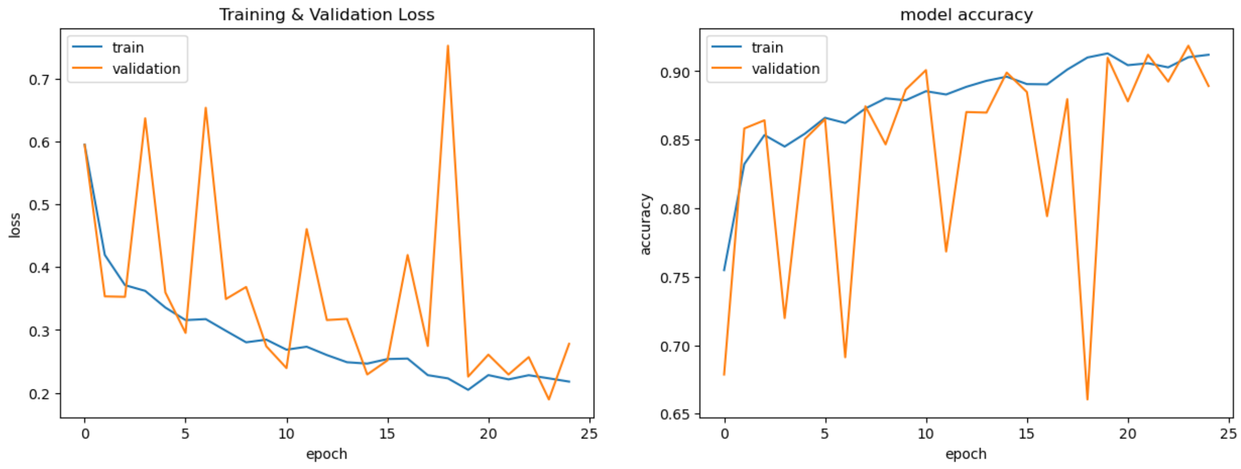


Figure 5. CNN Results

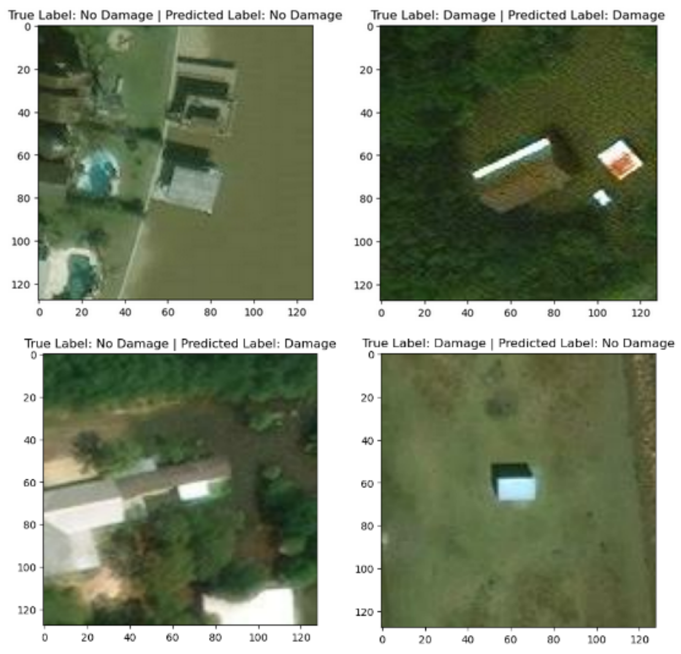

True Label: No Damage | Predicted Label: No Damage
True Label: Damage | Predicted Label: Damage
True Label: No Damage | Predicted Label: Damage
True Label: Damage | Predicted Label: No Damage

Figure 6. CNN Classification Results

VGG16 Model

The VGG16 model was very successful in classifying the hurricane damage images. The overall loss was 0.3577, meaning that there were very few errors in the predictions. The overall accuracy was very high at 94%, with some epochs reaching 100% accuracy. The epoch with the lowest accuracy still came in relatively high at 86% with a loss of 0.7603. With such high accuracies, we do run the risk of overfitting which could result in a loss of generalizability or ability to predict on new data. However, the validation datasets still had low rates of loss and high accuracy levels. Validation loss ranged from 0.22 to 0.39 and validation accuracy ranged from 90% to 94%. We present plots of the losses and accuracies below.

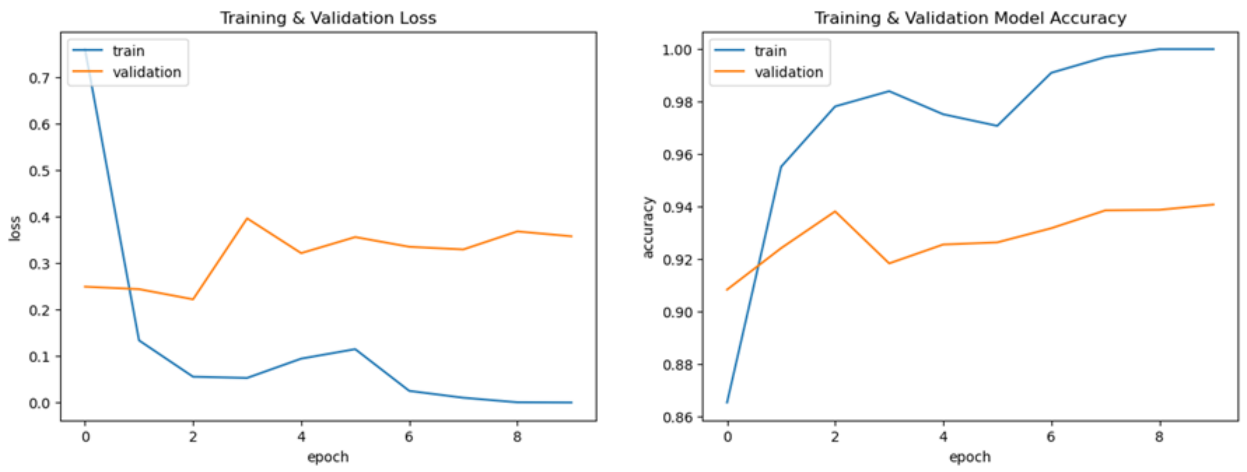
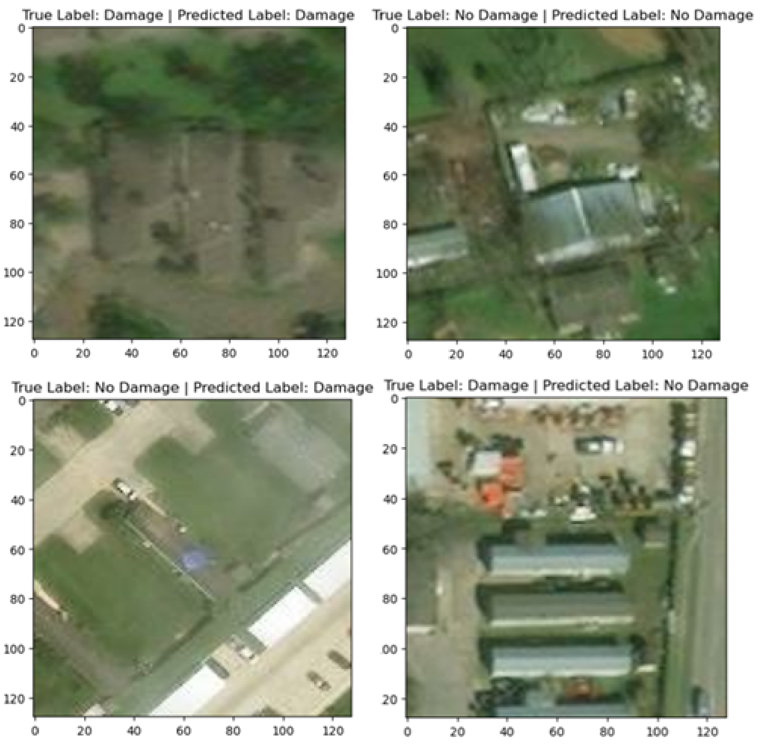


Figure 7. VGG16 results

Even though our model had high accuracy overall, it did make some incorrect predictions. We present examples of correct predictions for damage and no damage. We also show examples of incorrect predictions: one where the model predicted damage, but the outcome was no damage and one where the model predicted no damage, but the outcome was damage.


True Label: Damage | Predicted Label: Damage
True Label: No Damage | Predicted Label: No Damage
True Label: No Damage | Predicted Label: Damage
True Label: Damage | Predicted Label: No Damage
Figure 8. VGG16 predictions

<u>ResNet</u>

Of the three models, ResNet was the weakest in both loss and accuracy metrics. The test loss was 0.673, which is a pretty decent loss score, but worse than both the CNN and VGG-16 methods. Moreover, the test accuracy was 63%, which was the lowest accuracy score by far.

For the model training and validation process, a total of 10 epochs and a batch size of 32 was used. The plots of training and validation data are presented below.

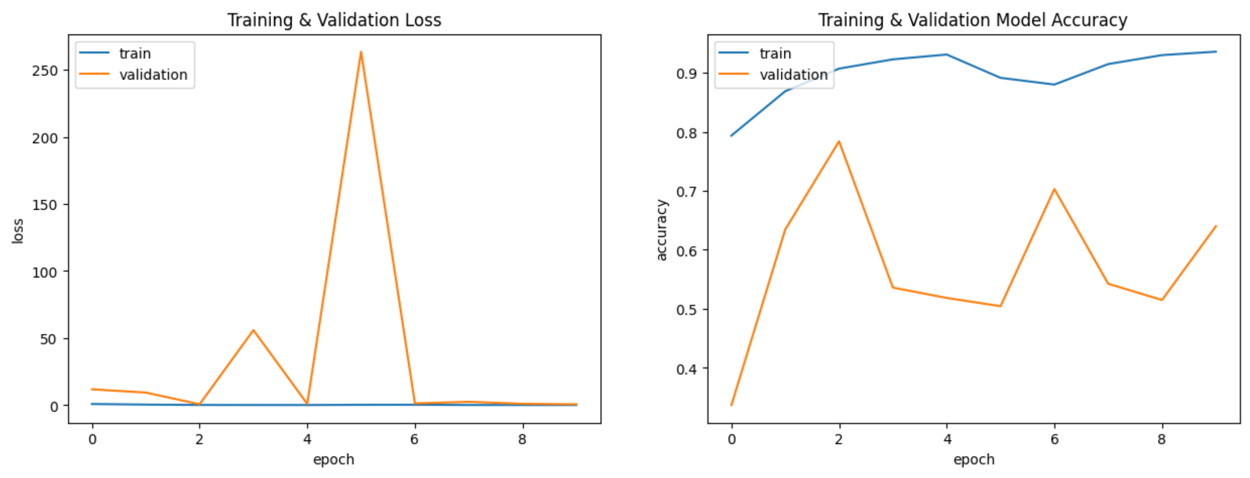


Figure 9. ResNet results

We see that the training loss stayed constantly low and the training accuracy had a relatively upward trajectory. However, the validation loss and accuracy experienced pretty disjointed outcomes. Most shockingly, the third and fifth epoch had loss scores of over 55 and 260, respectively; however, the loss function eventually evened out toward the final few epochs. These random spikes indicate that the model is being overfit to the training data throughout the training and validating process, and it may also indicate insufficient regularization techniques. The high fluctuation in the validation accuracy may indicate the model's inability to generalize to data outside of the training set.

5. Discussion

We derived several interesting insights from our modeling process. Interestingly, the loss is less on our CNN model than on the one that uses transfer learning. This could be due to the fact that our model is only trained on hurricane damage images, whereas the VGG16 transfer learning model will incorporate them into its established weights, and thus be more inflexible to new data but ultimately more accurate. The high accuracy of the VGG16 model likely comes from its generalizability to several contexts, as the model has been trained on a variety of data. However, model options like ResNet, which also have provenly high success for deep learning, saw really low accuracy. While we expected to see a higher accuracy even after multiple modeling iterations, there are several reasons why it may have not performed as well. For one, we could have utilized different transfer learning models for ResNet as well (e.g., ResNet50) which with predetermined architectures could have greatly benefited the model's generalizability and accuracy. Additionally, the model may have benefited from more parameters (e.g., data augmentation), though given the model's already intense computational demands (3 hours to run 10 epochs), our parameterization was limited.

Our main takeaways from this process:

- CNN is a strong choice to reduce loss, likely due to the fact that we had a large dataset with a binarized outcome, so it may benefit from the simplistic architecture.
- VGG-16 was our overall most accurate, given the pre-trained nature of transfer learning and its proveness to be accurate on unseen data. With customized parameters, we could really increase the accuracy.
- ResNet was our weakest model, but also the most complex, which indicates that our model may have been overfitting to the training data and could have benefitted from a different architecture.

We also would like to note that we experimented with using image segmentation for classification, specifically using a U-NET approach, but we eventually deemed it unnecessary for this particular project, as we wanted to focus only on image classification.

6. Conclusion

Overall, we built three deep learning models that all had wavering degrees of accuracy, ranging from 63 to 94 percent. While some models performed better in accuracy and loss than others, we believe that they all demonstrate the insightfulness of identifying and detecting damage to critical infrastructure after a natural disaster. As floods become more dramatic and intense with the ramifications of climate change, the need for high-technical and quick responsive technologies and emergency management methods is critical. We would definitely recommend at least our first two deep learning models, and potentially our third with more parameter optimization, as ones that could be highly useful in a planning and emergency response application to assess flood damage data using the power of remote sensing.

If you would like to see our code, check our Jupyter Notebook.