Hurricane Damage Prediction with Neural Networks

Jackson Thetford, Mihiro Suzuki

Introduction:

This project focuses on using deep learning techniques to classify satellite images into "damaged" or "non-damaged" buildings after Hurricane Harvey. The dataset used is a collection of satellite images from Texas. This project included data preprocessing for image analysis, exploration of various neural network architectures such as ANN, LeNet-5, an alternative LeNet-5 architecture, VGG-16, ResNet, and Xception and also included evaluation of model performance. The models are deployed with Flask , providing an HTTP interface for image classification.

Data preparation:

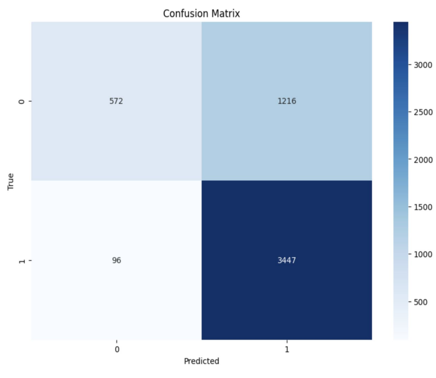
After the dataset was loaded, it was visualized to understand its characteristics, such as image size and format. The data that was used for this project can be found in the /data directory of this repository. It contains 21322 (128px, 128px) RGB images. Each image was classified as "damage" or "no_damage". The data was then converted into Python data types, specifically into NumPy arrays, and stored into two variables: data_all for the images and labels_all for their corresponding labels. The RGB values were normalized to the range [0, 1] for model training. Finally, the dataset was divided into two parts: one for training and the other for testing. This division was used to evaluate the performance of models on new, unseen data.

Models Used:

- **ANN (Artificial Neural Network)**: A dense, fully-connected network with varying layers and perceptrons.
- **LeNet-5 CNN:** A convolutional neural network based on the classical Lenet-5 architecture, adjusted for our image dimensions.
- **Alternate-LeNet-5 CNN:** Inspired by a variant discussed in this research paper, though customized for our specific dataset.
- **VGG:** A custom model with the VGG-16 architecture as described by this paper.
- **ResNet:** A custom model with the ResNet architecture as described by this paper.
- **Xception:** A custom model with the Xception architecture from Google as described by this paper.

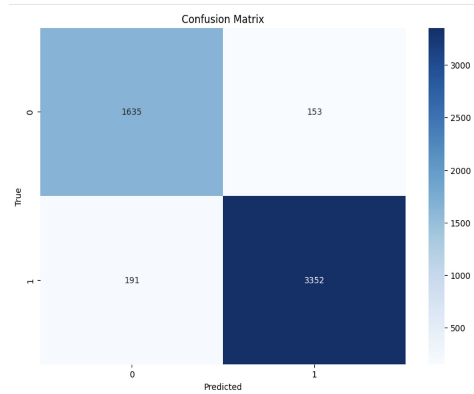
Model Design

ANN



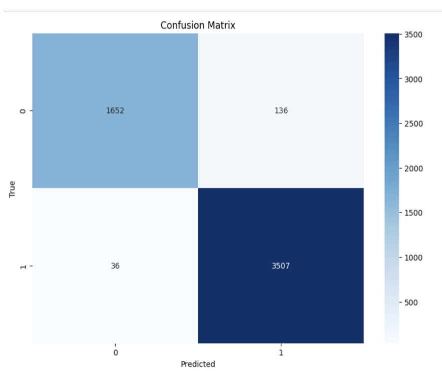
The test accuracy was around 75%. It incorrectly predicts damage (false positive) more often than it incorrectly predicts no damage.

Lenet-5 CNN



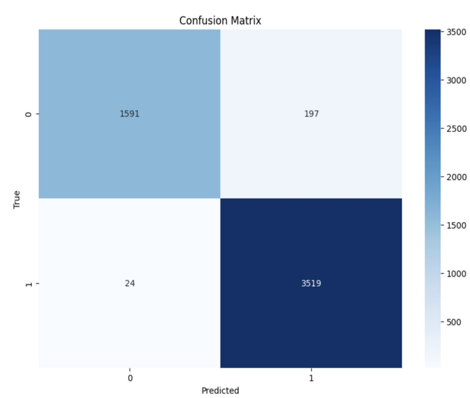
The test accuracy was around 93%. It also has a slight bias towards false negatives but is significantly more accurate than ANN

Alt-Lenet-5 CNN



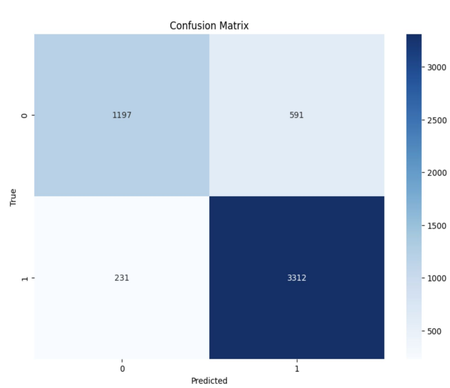
The test accuracy was slightly higher than the original CNN at around 97% accuracy. It also has a slight tendency to false positives

VGG 16



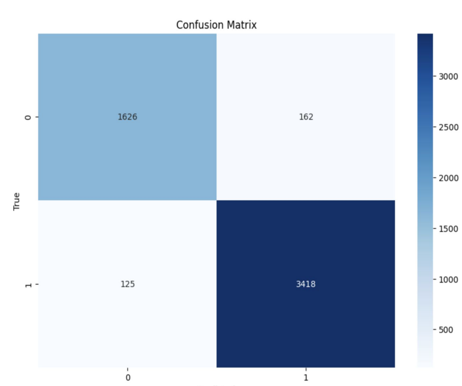
Test accuracy was around 96% accuracy with a slight tendency to false positives

Res Net



The test accuracy was 85% and has a tendency for false positives.

Xception



Test accuracy was around 95% accuracy with a slight tendency to false positives

Model evaluation:

Models were evaluated based on their accuracy score on the validation set. The alt_lenet model was the best model. While the Lenet model actually out performed the alt_lenet model by 0.5% on the original testing set, through further testing that both models were subjected to, the alt_lenet model was found to be slightly better, averaging an accuracy of 97.3%. We are very confident in our model as during training the accuracy and validation accuracy stayed within 2% of each other. This gives our confidence that our model is not overfit to the training data and that it has accurately captured the nuances of the “damage” and “no-damage” features in the images.

Model deployment and inference:

This model is deployed via an inference server as flask app. The API endpoints are as listed below. This API serves as an interface for classifying images using pre-trained TensorFlow models. It allows users to classify images as "damaged" or "not damaged", change the underlying model, and retrieve information about the current model.

API Endpoints

Endpoint	Method	Description
`/model/info`	GET	Provides basic information about the currently loaded TensorFlow model, including its parameters.
`/model/models`	GET	Lists the available TensorFlow models that users can switch to, indicating the default model.
`/model/predict`	POST	Classifies an image uploaded by the user as either "damaged" or "not damaged".
`/model/change`	POST	Changes the TensorFlow model used by the server to a specified model.
`/model/summary`	GET	Provides a textual summary of the currently loaded TensorFlow model's architecture.
`/help`	GET	Provides an overview and usage examples for the available API endpoints.

This allows the user to experiment with different models and feed in images to test with. See the file inference_example.ipynb for examples of use for each API endpoint in depth.The default do for the inference server is the “alt_lenet” model as it was found to be the best performing, however all models shown in page 2 are available through the “/model/change” endpoint.