Ayushi Kate
COE379L
13 November 2025

Project 02 Model Report

Hurricane Damage Binary Classification

1. Project Overview

The primary objective of this project was to analyze a dataset about buildings affected by Hurricane Harvey in Texas. The dataset had satellite images labeled damaged and non-damaged building images. The labels were used to develop and evaluate different neural network models based on different architectures to classify buildings as damaged or not damaged. The aim of this project is to develop the best performing binary classification model and evaluate their performances.

Data Preparation

The data preparation phase began with an examination of the initial dataset composition. First, I loaded in all the images so they could be readable through Python functions. I confirmed that there were 14170 damaged images, and 7152 non-damaged images.

I inspected the images initially by gathering a random sample of 10 damaged and 10 non-damaged images and displaying them with their labels to visualize them. I also took a random sample of 5 damaged and non-damaged images to inspect their size. After multiple runs, it looked like all of the images were of size 128 x 128.

Splitting the Data
The dataset was split into 80% training data and 20% testing data. I generated a random sample of damaged and non-damaged images and separated them into their own folders. I created a train folder and test folder in which there were separately labeled damaged and non-damaged images. This left me with 11336 Damaged Training images, 5721 Non-damaged Training images, 2834 Damaged Testing images, and 1431 Non-damaged Testing images.

2. Model Training and Evaluation

2.1. Training Procedure

1. **Classic Artificial Neural Network (ANN)**
   The model is a fully connected ANN that takes an input of 128x128x3 RGB images and flattens them into a vector of values. There are two hidden layers, the first with 128 neurons and the second with 64 neurons, both using ReLU nonlinear activation

functions. The output layer consists of 2 neurons with a softmax activation function which does the final binary classification. It is trained using the Adam optimizer and evaluates over sparse categorical cross-entropy loss.

2. **LeNet-5 Architecture**
   The LeNet model is a convolutional neural network which takes an input of 128x128x3 RGB images and first feeds them through a 5x5 convolution layer with 6 filters and ReLU activation. Next is a 2x2 average pooling layer that reduces the dimensions of the layer. The second stage is another 5x5 convolution layer with 16 filters and the same 2x2 pooling layer. After flatting the result, it is further passed through two fully connected layers, one with 120 neurons and the other with 84 neurons. The output layer consists of 2 neurons with the softmax activation function for final binary classification.
3. **Alternate LeNet-5 Architecture**
   The alternate architecture is a convolutional neural network that is better for handling images. It begins with a 3x3 convolutional layer with 32 filters and ReLU activation followed by a 2x2 max pooling layer to reduce dimensions of the layers. Afterwards, it continues with convolutional layers of 64, 128, and 128 filters followed by 2x2 max pooling layers. After feature extraction, the feature map is passed through a dropout layer, and it is finally fed through a fully connected layer with 512 neurons. The binary classification is done through a single sigmoid output neuron to generate a 0 and 1 for binary classification.

2.2. Model Evaluation

Model	Accuracy
ANN (Basic)	0.7889800667762756
LeNet-5	0.9317702054977417
Alternate LeNet-5	0.9807737469673157

2.3. Class Prediction

Overall, the models performed well in the binary classification task. The alternate LeNet-5 architecture had the best accuracy out of the three models in predicting the class. The classic LeNet-5 architecture was a close second, and performed fairly similar to the alternate architecture. I am pretty confident in these models, but especially in the alternate LeNet-5 architecture. It was able to achieve a really high accuracy on the validation set, so I am able to trust its ability to generalize to other data. However, more validation is necessary to guarantee the model’s performance in other contexts, perhaps with other datasets to confirm its results. Regardless, I am largely confident in the reliability of this model given its performance with this dataset.

3. Model Deployment & Inference

3.1. Inference Server Building

1) Navigate to the project directory inside the Project2 folder
2) Build Docker Image.
   Initializes the docker image on port 5000, as defined in the Dockerfile.
   ```
   docker compose build
   docker compose up
   ```
3) You can also pull the prebuilt docker image from Docker Hub using the below command. You will then have to connect the image to a running port.
   ```
   docker pull ayushik7/lenet-new:v1
   docker run -d --rm -p 5000:5000 ayushik7/lenet-new:v1
   ```

3.2. Endpoints Summary

There are two endpoints implemented on this model.

GET /summary	POST /inference
Returns a JSON summary of the metadata of the model.	Returns the binary classification in JSON of an image whether it is damaged or not.
Example Command: curl http://localhost:5000/summary	*Example Command:* curl -X POST -F "file=@/data/damage93.539521_30.982434.jpeg" http://localhost:5000/inference
Example Output: {"description":"Classify images predicting whether the building has been damaged by a hurricane","input_shape":[null,128,128,3],"model_name":"Alternate-LeNet5","number_of_parameters":2601666,"output_shape":[null,2]}	*Example Output:* {"prediction":"damage"}