Ira Narang (in2933)
Nicole Olvera (no4342)
Rhea Samuel (rss3488)

Project 3 Report

Introduction:

In the event of natural disasters, it is crucial to respond efficiently and allocate resources accurately. This project focuses on developing an automated image classification system to assist in post-disaster analysis, specifically for Hurricane Harvey, by classifying satellite images of buildings as either damaged or not damaged. Given a labeled dataset of post-hurricane images from Texas, the objective is to explore and implement multiple neural network architectures for binary image classification and evaluate their performance. The project is divided into three key components: data preprocessing and visualization, model design, training and evaluation, and model deployment and inference. This work demonstrates the practical application of deep learning in impactful scenarios, with the potential to be refined and used in the future.

Data Preparation:

Data preparation began with code to load the data into Python data structures. We first organized the images into structured directories by our classes: **damage** and **no-damage**. Then, to effectively assess the model, we divided the data into 80% training and 20% testing for each class, with an additional 20% of the training set used for validation to monitor performance during training. Images were resized to a uniform size of 150×150 pixels to ensure our input dimensions remained fixed, and we also established a batch size of 32. In order to standardize the pixel intensity values, we scaled the pixel values from the [0, 255] range to [0, 1] using TensorFlow's **Rescaling** function. We finally examined the shape of the train and testing batch as well as their labels, and printed a portion of the batch to ensure all preprocessing changes were implemented correctly. Overall, these changes were implemented to help improve model performance.

Model Design:

Multiple neural network architectures, along with different variants, were explored and evaluated to achieve optimal classification performance. The sections below explain our implementation of each.

Artificial Neural Network (ANN):
We first explored a fully connected ANN as a baseline. The input images were flattened from 150×150×3 dimensions to a 1D array and passed through one or more **Dense** layers. We tested three configurations to observe the effect of increasing model depth and the number of perceptrons on classification accuracy:

- **Case 1**: A simple architecture with one hidden layer of 128 perceptrons using ReLU activation.
- **Case 2**: A deeper model with two hidden layers, 512 and 256 perceptrons.
- **Case 3**: A three-layer architecture, 512, 256, 128 perceptrons.

Lenet-5 CNN:
The Lenet-5 CNN model is an early convolutional neural network. Its simple and computationally efficient architecture is ideal for small datasets and resource-constrained devices. To improve performance on our dataset, we explored changes to both feature extraction and network layers across three cases:

- **Case 1:** Lenet-5 CNN with 6 and 16 filters, 120 and 84 perceptrons.
- **Case 2:** Lenet-5 CNN with 12 and 32 filters for more feature extraction capacity.
- **Case 3:** Lenet-5 CNN with 6 and 16 filters, and dense layers of 256 and 128.

Alternate-Lenet-5 CNN:
To further improve our image classification accuracy, an alternate variant of the classic Lenet-5 convolutional neural network (CNN) [1] was implemented. This modified architecture incorporated adjustments such as increasing the filter sizes to 32 for the first layer, 64 for the second layer, 128 for the third layer, and 128 for the last layer. There were 4 layers in total, each with a Max Pooling layer, and increased the dense layer size to 512 perceptrons. Two training configurations were assessed:

- **Case 1**: Alternate Lenet-5 CNN with four layers as described above with 10 epochs
- **Case 2**: Alternate Lenet-5 CNN with four layers as described above with 20 epochs

Model Evaluation:

We evaluated the models described above and their results are summarized below:

Artificial Neural Network:

	Case 1	**Case 2**	**Case 3**
Total Parameters	8,640,257	34,692,097	34,724,865
Accuracy	0.7237	0.7590	0.6668

Increasing the depth and number of perceptrons in our ANN models initially improved performance, with Case 1 achieving 72.37% accuracy and Case 2 reaching the highest accuracy of 75.90%. However, adding a third hidden layer in Case 3 caused the accuracy to drop to 66.68%, likely due to overfitting and the inability of ANNs to capture spatial patterns in image data. These results highlighted the limitations of fully connected ANNs for image classification and motivated us to explore convolutional neural networks (CNNs) for better performance.

Lenet-5 CNN:

	Case 1	**Case 2**	**Case 3**
Total Parameters	2,232,846	4,460,038	47,771,258
Accuracy	0.9791	0.9764	0.9659

As observed, accuracy was highest at 97.91% in Case 1, our baseline, but then proceeded to decrease slightly for each following case (2 & 3). Rather than improving performance as intended, the increased

[1] Cao QD, Choe Y. Building damage annotation on post-hurricane satellite imagery based on convolutional neural networks. arXivOrg 2020. https://arxiv.org/abs/1807.01688

Ira Narang (in2933)
Nicole Olvera (no4342)
Rhea Samuel (rss3488)

complexity led to signs of overfitting. Nonetheless, all CNN variations significantly outperformed the ANN models, affirming the strength of convolutional architectures for this task. Overall, our baseline Case 1 performed the best.

Alternate- Lenet-5 CNN:

	Case 1	**Case 2**
Total Parameters	3,453,634	3,453,634
Accuracy	0.9425	0.9877

As expected, the accuracy when testing with 20 epochs was higher than when testing with 10 epochs, with a 4.52% increase in accuracy. These results suggest that the alternate Lenet-5 architecture effectively improved performance in our image classification task, achieving the best accuracy overall.

The Alternate Lenet-5 CNN model emerged as the optimal choice due to its balanced complexity and high accuracy. The architectural modifications enabled our model to reach an accuracy of 98.77%, surpassing the classic Lenet-5 CNN model with an accuracy of 97.91% and significantly outperforming the ANN baseline with an accuracy of 75.90%. Its robust performance indicated enhanced feature extraction and an effective decrease of overfitting.

Model Deployment:

The selected Alternate Lenet-5 CNN model was deployed as a web-based inference server using Docker. Docker facilitated the creation of a lightweight, reproducible environment, making model deployment straightforward and portable. To deploy our model, we created a Docker image using a Dockerfile that installed dependencies and included our Flask-based API. Docker Compose simplifies container management by defining services within a docker-compose.yml file.
In order to deploy our server,

```
docker-compose up -d --build
```

must be typed in the terminal. Once the server is running, the curl commands can be tested.
The Flask application provided two key endpoints:

- **Summary Endpoint:** Accessed using `curl -X GET http://localhost:5000/summary`, providing metadata about the model including its name, version, a mini description, and achieved accuracy.
- **Inference Endpoint:** Accessed using `curl -X POST -F "file=@image.jpeg" http://localhost:5000/inference`, enabling users to submit images for classification. This endpoint returns clear predictions indicating whether buildings depicted have “damage” or “no damage”. A proper image path must be included for this endpoint to work.

Overall, this deployment strategy demonstrated an efficient & reliable method for applying deep learning models in real-world scenarios, facilitating automated and accurate post-disaster damage assessments.