Hurricane Damage Classification Project Report

Krish Chaudhary

1. Data Preparation

The dataset used for this project contained satellite imagery of buildings affected by Hurricane Harvey, classified into two categories: damage and no_damage. The original images varied in size and quality, so we applied consistent preprocessing to standardize inputs for the neural networks. Specifically, each image was resized to **128x128 pixels** and converted to **RGB format**, resulting in input tensors with shape (128, 128, 3). These were normalized by dividing pixel values by 255.0. The training, validation, and test sets were generated using ImageDataGenerator with proper shuffling and batching, ensuring the model did not learn from imbalanced or ordered data.

Additionally, we manually inspected several images to identify and account for anomalies such as grayscale entries, corrupt files, or unreadable formats, which we excluded. Data generators were configured to handle class imbalance and data augmentation was not included in this implementation to maintain the integrity of the test set.

2. Model Design

We explored three primary neural network architectures in our effort to accurately classify hurricane damage from satellite imagery:

- **Fully Connected Dense Neural Network (ANN)**: This served as our baseline model. The input image was flattened into a 1D vector and passed through several dense layers with ReLU activations, ending in a sigmoid output layer. Despite achieving high training accuracy, this model overfit to the majority class (no_damage). It lacked the ability to capture spatial features essential for visual tasks.
- **LeNet-5 Convolutional Neural Network**: Adapted from the classical LeNet-5 architecture, this CNN included two convolutional layers with ReLU activations, followed by max-pooling, a flattening layer, and two dense layers. It dramatically improved performance on the minority class. The convolutional layers allowed the model to learn spatial hierarchies of features, which are critical for recognizing visual patterns of structural damage.
- **Alternate LeNet-5 (Modified CNN)**: This variant introduced additional regularization and increased kernel complexity. However, it did not consistently outperform the original LeNet-5 and occasionally exhibited instability across epochs. After experimentation, we

determined that the original LeNet-5 offered the best trade-off between training time and performance.

For all models, we used binary_crossentropy loss, the Adam optimizer, and monitored val_accuracy and val_loss. Training incorporated **EarlyStopping** and **ModelCheckpoint** callbacks to prevent overfitting and retain the best-performing weights.

3. Model Evaluation

The best-performing model was the **LeNet-5 CNN**. While it had a slightly lower overall accuracy (~0.52) compared to the dense ANN (~0.66), it significantly improved performance on the minority class (damage). Specifically, the LeNet-5 achieved:

- **F1-score (damage): 0.39**
- **Recall (damage): 0.45**
- **Precision (damage): 0.33**

In contrast, the ANN predicted only the majority class, leading to an F1-score and recall of 0.00 for damage. Thus, LeNet-5 provided more meaningful results by learning actual features of hurricane damage.

A confusion matrix confirmed this improved balance, and weighted averaging across metrics showed a macro-F1 of ~0.49 and weighted F1 of ~0.53. While not perfect, the model captured essential spatial patterns of damage, and we are reasonably confident in its ability to generalize to similar post-disaster imagery.

4. Model Deployment and Inference

To deploy the best model (LeNet-5), we implemented a **Flask-based inference server** with two RESTful endpoints:

- **GET /summary**: Returns a JSON payload describing the model architecture, including input shape, output shape, model type, and total number of parameters.
- **POST /inference**: Accepts a raw image file in binary format and returns a classification prediction: { "prediction": "damage" } or { "prediction": "no_damage" }.

The server loads the trained model (lenet_best.keras) and processes incoming images with the same preprocessing pipeline used during training. Inference is performed with a binary threshold of 0.5.

This application was packaged using **Docker** to ensure cross-platform compatibility and ease of deployment. A Dockerfile defines the image environment, and a docker-compose.yml file simplifies startup and exposes the app on port 5000. The model image was pushed to Docker Hub (x86 architecture) to meet deployment specifications.

To run locally:

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

To make requests:

```
curl http://127.0.0.1:5000/summary
curl -X POST -H "Content-Type: application/octet-stream" --data-binary @sample.jpg
http://127.0.0.1:5000/inference
```

These endpoints enable scalable and reliable access to the model's classification abilities, fulfilling the requirements for automated, server-based disaster inference workflows.

Conclusion

This project demonstrates the full pipeline of machine learning for social good—from data preprocessing and neural network modeling to real-world deployment in a containerized environment. Our LeNet-5 model, despite moderate accuracy, successfully identifies hurricane-induced damage with reasonable class balance. The Docker-based deployment ensures it can be integrated into larger emergency response systems or future AI research workflows with minimal overhead.