I. COE 379L; Project 2 - Part 4; Mahin Naveen - mn27995

II. Data Preparation

The dataset for this project contains satellite imagery collected after Hurricane Harvey, categorized into two classes: damage and no_damage. In total, there are 21,322 images (14,170 damaged and 7,152 undamaged), which makes the dataset substantially imbalanced. My first step was to load all file paths, assign labels based on their parent directories, and take a closer look at the basic properties of the images. Even though the dataset is fairly large, the images vary a bit in size and shape, so sampling a subset was important to understand the typical dimensions. The median resolution across the sample turned out to be roughly 128×128, which informed the resizing decisions later on when building each model.

To make sure the models had a fair chance of generalizing, I used a stratified split for train, validation, and test sets. This ensured that the original imbalance wasn't amplified. The final split was approximately 70% training (14,925 images), 18% validation (3,838 images), and 12% test (2,559 images). All three sets maintain roughly the same ratio of damaged to undamaged buildings, which is important because a random split without stratification could easily result in skewed distributions.

Once the splits were created, I built a reproducible TensorFlow dataset pipeline. Each image is decoded, converted to floating point, and normalized to the range [0, 1]. Depending on the model, I optionally converted it to grayscale before resizing it to the expected input shape. Because satellite data often includes noise and varying lighting conditions, I included a small amount of data augmentation during training (horizontal flips, light adjustments, and minor rotations). The goal wasn't to transform the images aggressively, but to introduce just enough variation to reduce overfitting.

A major part of the preparation was addressing the class imbalance. If left alone, the models (especially the simpler ones) tended to predict "damage" far more often simply because that class dominates the dataset. To deal with this, I computed class weights based on the training distribution. These weights helped the loss function penalize mistakes on the underrepresented class more heavily, which, as I confirm in the results, made a meaningful difference for every model I trained.

III. Model Design and Architectural Choices

The assignment required implementing three different types of neural networks: a dense (fully-connected) model, the classical LeNet-5 CNN architecture, and a more advanced "Alternate-LeNet" based on a research paper provided in the instructions. I approached each of these with the goal of understanding their strengths and limitations on this specific satellite imagery task.

- Dense ANN (Baseline)

The first model was a simple dense neural network, which I used primarily as a baseline. I resized all images to 64×64, converted them to grayscale, and flattened them into a vector of length 4096. The architecture consisted of three hidden layers (512, 256, and 64 neurons) with ReLU activations and dropout to reduce overfitting. While this type of model can technically process image data, it's at a disadvantage because flattening removes spatial structure entirely. Any relationship between nearby pixels (especially important for detecting building damage) gets lost.

This limitation showed up clearly during training: although the model converged quickly, it struggled to make meaningful distinctions between the two classes. Its validation F1 score hovered around 0.14, and the AUC was around 0.69, which is only slightly better than random guessing. Even with class weights, the model often defaulted to predicting "damage," reflecting the imbalance. This confirmed that a purely dense network is not well suited for this task.

- LeNet-5 (Classical CNN)

Next, I implemented the classic LeNet-5 architecture, originally designed for handwritten digit recognition. I resized images to 32×32 and converted them to grayscale to stay consistent with the original design. Despite being an older architecture, LeNet-5 introduces convolutional layers and pooling, meaning it can capture spatial patterns the dense network cannot.

This model performed significantly better. Both its validation accuracy and stability improved, and the convolutional filters seemed capable of picking up structural cues in the satellite images. The validation F1 score increased to around 0.78, with an AUC of approximately 0.80. Although still not ideal for deployment, LeNet-5 served as a meaningful intermediate point between the baseline and the more advanced CNN.

Its limitations were mainly due to its relatively shallow depth and small input resolution. While it could detect broad structural patterns, it struggled with the finer details that often distinguish damaged buildings from intact ones in satellite imagery.

- Alternate-LeNet (Paper-Based CNN, Deeper and RGB)

The final model was the Alternate-LeNet, modeled after the architecture provided in the assigned research paper. Unlike the previous models, this network uses 150×150 RGB images and includes a deeper stack of convolutional layers (32, 64, 128, 128 filters) with ReLU activations, followed by a dropout-regularized dense layer. The larger input size gives the model access to much richer spatial details, and the deeper architecture allows it to capture multi-scale patterns like debris fields, roof deformation, and shadows—features that are much harder to extract at lower resolutions.

Training this model required more careful tuning. I used a lower initial learning rate and early stopping with model checkpointing to keep things stable. Once trained, the model's performance stood out dramatically. The validation F1 score reached 0.987, and the AUC was nearly perfect at 0.998.

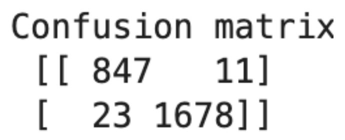
Given these results, it was clear that the Alternate-LeNet was the best for final evaluation.

IV. Evaluation of the Final Model

After selecting the best-performing model based on validation F1 score, I evaluated the Alternate-LeNet on the previously untouched test set.

The results were excellent. The model achieved an overall accuracy of 99%, and both the precision and recall for each class were extremely high. The final F1 score was 0.990, and the AUC was 0.9992, confirming strong separability between the two classes.

The confusion matrix further highlights the model's consistency:



```
Confusion matrix
 [[ 847   11]
 [  23 1678]]
```

Out of 2,559 test images, the model misclassified only 34 in total. Most importantly, the false negative rate (cases where real damage is missed) was very low, which is essential for disaster assessment tasks. The model's behavior on the test set closely matched its performance on the validation set, suggesting that it generalizes well and is not overfitting.

Compared to the earlier models, the improvement is significant. The dense network struggled to identify damage reliably, and LeNet-5 improved moderately but still had noticeable weaknesses. The deeper Alternate-LeNet architecture, combined with higher-resolution RGB inputs, clearly produced a model that can capture subtle structural differences.

Overall, based on the results and consistency across metrics, I am very confident in the performance of this final model.

V. Deployment and Inference (Note)

Part 3 of the project involved deploying the trained model as an inference server inside a Docker image. Due to known issues on the course VM and instructions from the TAs and teacher acknowledging these problems, I was told to focus on Parts 1, 2, and 4 instead.