Samuel Amoabeng (EID: sba 798) Jacob B Wendling (EID: jbw2976)

PROJECT 3 REPORT

PART 1

DATA PREPARATION AND VISUALIZATION

The dataset we are looking at is images of buildings and land after Hurricane Harvey. We are trying to use a Convolutional Neural Network to predict whether the image contains damage or no damage. The images are 128x128 pixels with RGB values.

Data preparation is made using different steps in data processing, first, all the libraries needed to perform this process. To prepare the data for training and evaluation of the neural networks, I loaded the dataset into Python data structures. I then investigated the datasets to determine the basic attributes of the images, such as dimensions, color channels, and number of samples. After this, I split the data into training, validation, and testing sets using appropriate proportions. Additionally, I performed preprocessing steps such as rescaling and normalization to ensure the data was suitable for training and evaluation of the neural networks.

```
train damage image count:  11336
test damage image count:  2834
len of overlap:  0
train no damage image count:  5721
test no damage image count:  1431
len of overlap:  0
```

The outcome shows 11336 training images of damage and 2834 test images of damage, on the other hand, there are 5721 training no-damage images and 1431 test no-damage images.

```
Files in train/damage:  14170
Files in train/no_damage:  7145
Files in test/damage:  14170
Files in test/no_damage:  7145
```

There were 14170 files containing train/damage and test/damage and 7145 files with train/no damage and test/no damage.

```
Found 21315 files belonging to 2 classes.
Using 17052 files for training.
Using 4263 files for validation.
```

The number of files found was 21315 which belongs to 2 classes. 17052 files were used for training our model and 4263 were used for validation.

PART 2

MODEL DESIGN, TRAINING AND EVALUATION

We explored three different model architectures in order to get a model that has high accuracy. We tested a standard convolutional neural network (CNN), the Lenet-5 Convolutional Neural Network (LCNN) architecture, and an Alternate-Lenet-5 CNN (ACNN) architecture described in the research paper given, which can be found in the references. The CNN model had 10 layers with 3 convolution layers and 3 dense layers (857373 total parameters). The LCNN model had 8 layers with 2 convolution layers and 3 dense layers (1739417 total parameters). The ACNN model had 11 layers with 4 convolution layers and only 2 dense layers, and the convolution layers had many more filters than the other models (2601153 total parameters). The final layer of all three models was the same, a single node with a sigmoid activation function for binary classification. The loss function was therefore a binary crossentropy function for this purpose. The specific construction and summary of the models can be found in the jupyter notebook in our github repository.

Following model design, the models were evaluated based on their performance in classifying the images in the dataset. Among the three architectures, the Alternate-Lenet-5 CNN architecture emerged as the top performer, achieving an accuracy of 98.0% on the test set. The CNN model also performed very well at 97.8% on the test set, but it had a lower validation accuracy, 96%, when training.

In conclusion, the Alternate-Lenet-5 CNN architecture proved to be the most effective model for the task, delivering outstanding accuracy metrics. We are very confident in this model to predict the damage state of these images with 98.0% accuracy.

To use this model for yourself, you can find the code on our github repository found in the references. The specific instructions and exact commands can be found in the README.md files.

1. Clone the repository
2. Navigate to the project 3 folder
3. Run *run_flask.sh*
4. In a new terminal, run *submit_image.py* with an image to get a prediction. The program will print the prediction of whether the image contained damage.

References

Research paper: https://arxiv.org/pdf/1807.01688.pdf

Github repository: https://github.com/Kwabena86/SW-DSIGN-FOR-RESP-COE379L