Project 3 Report: Building Damage Classification Using Neural Networks

Data Preparation

Our dataset consisted of satellite images of Texas buildings taken after Hurricane Harvey. Each image was labeled as either damage or no_damage. The primary goal of this project was to classify these images as accurately as possible using deep learning.

Data Preprocessing and Augmentation

To ensure the quality and performance of our model training, we constructed a robust and clear data preparation pipeline:

- **Data Organization**: We separated the dataset into two general classes based on the labels. A custom script was written to organize the dataset into subfolders for each class and divide it into an 80-20 train-test split.

- **Resizing and Normalization**: All images were resized to 128x128 pixels to maintain a consistent input size across the dataset. Pixel values were normalized between 0 and 1 to improve numerical stability, accelerate model convergence and improve accuracy.

- **Data Augmentation**: Real-time image augmentation was utilized to artificially increase the variability of the training set and enhance the model's generalization ability. The augmentation techniques included:

    - Random rotations up to 20 degrees

    - Width and height shifts up to 20%

    - Random zooming and shearing

    - Horizontal flips

    - Standardization through centering (mean subtraction) and normalization (division by standard deviation)

- **Validation Split**: During training, 20% of the training data was held out as a validation set to monitor the model's performance on unseen data and prevent overfitting. This would also show the training model's efficiency and figure out any improvements to be made

Dataset Overview:

Category	Count
Total Images	21322
Training Set	17057
Testing Set	4265

Image preprocessing and augmentation were handled using the ImageDataGenerator class from Keras, which provided an efficient pipeline for spontaneous augmentation and data feeding during both training and evaluation.

Model Design

For model design, three different deep learning architectures were explored for this binary classification task. It includes the following:

1. Fully Connected Artificial Neural Network (ANN)

This architecture was built using dense layers and is relatively simpler compared to CNNs.

Architecture:

- A Flatten layer to convert the 2D image into a 1D vector
- Three dense layers with 512, 256, and 128 neurons, respectively
- Dropout layers (rate: 0.5) for regularization
- A final sigmoid activation function for binary classification

2. LeNet-5 Convolutional Neural Network (CNN)

This model was inspired by the classic LeNet-5 architecture, traditionally used for digit recognition, but here adapted for satellite image classification.

Architecture:

- Three convolutional layers with increasing filters (6, 16, 120), each followed by max pooling
- A Flatten layer and a single dense layer with 84 neurons
- A dropout layer for regularization
- A final sigmoid output for binary classification

3. Alternate-LeNet-5 CNN

With the recent advancements in CNN architectures, we developed a deeper version of LeNet-5 with enhanced capacity for feature learning.

Architecture:

- Four convolutional layers with increasing filters (32, 64, 128, 128)
- Max pooling layers after each convolutional block
- Dense layers with dropout for regularization
- A sigmoid output layer for classification

This model added more depth and capacity, allowing it to learn more complex and abstract patterns from the input image data.

Model Evaluation

All models were trained for 20 epochs with early stopping and model checkpointing to retain the best-performing model based on validation accuracy.

Performance Comparison:

Model	Training Accuracy	Test Accuracy
Fully Connected ANN	66.5%	66.2%

LeNet-5 CNN	92.9%	92.4%
Alternate-LeNet-5 CNN	92.5%	92.5%

The **Alternate-LeNet-5 CNN** outperformed the other models with the highest test accuracy of **90.3%**. This indicates strong generalization to unseen data, particularly considering the diversity in post-hurricane imagery. It also ensures the model training is efficient and has minimal overfitting compared to other models.

Model Confidence

We are confident in the performance of the Alternate-LeNet-5 model. It remained stable across both training and test datasets, showing minimal signs of overfitting. The narrow gap between training and test accuracy suggests a well-balanced model with reliable performance.

Model Deployment and Inference

Deployment Overview

To serve the trained model in real-world scenarios, we wrapped the inference application in a **Docker** container and created a simple web server using **Flask**.

- **Docker**: A Dockerfile was created to build an image with Python, TensorFlow, Flask, and all necessary dependencies. This setup guarantees reproducibility and platform independence.
- **Flask Server**: The server exposes two REST API endpoints:
  - GET /summary: Returns model metadata such as structure and training accuracy
  - POST /inference: Accepts an uploaded image, preprocesses it, and returns the predicted class (damage or no_damage)

This deployment model enables easy scalability and portability, making it highly suitable for integration into emergency response systems and other decision-support tools.

Ending Remarks

Through the exploration of various data augmentation techniques, training different types of models and deployment engineering, we developed an effective, high-performance model for classifying hurricane-induced building damage from satellite imagery. The **Alternate-LeNet-5 CNN** demonstrated competitive classification performance. With more fine-tuning, more intense training and hypertuning, the model could potentially be deployed into real life scenarios including cases such as disaster assessment platforms.

Overall, this project demonstrates the real-world potential of deep learning for humanitarian and environmental applications, particularly in the context of disaster response and recovery.