DEEP LEARNING TO THE RESCUE!

Classifying Hurricane Havoc One Pixel at a Time

CECI NGUYEN & KLOE WANG
DCN558, WZ4994
COE379L
APRIL 10, 2025

Contents

Introduction

This project addresses the challenge of automatically classifying post-hurricane building damage using satellite imagery. With the increasing frequency of natural disasters and the vast areas they impact, manual damage assessment becomes inefficient and time-consuming. By leveraging Convolutional Neural Networks (CNNs), we aim to develop a system that can accurately distinguish between damaged and undamaged buildings. The final model is deployed as a web service using Flask and Docker to support real-time inference capabilities. This project aims to automate the classification process to aid emergency responders in assessing damage more efficiently and accurately.

Data Preparation

The dataset contains high-resolution satellite images labeled with various levels of building damage after natural disasters. For this project, we focused on a binary classification problem: damage vs. no_damage. In preparation for model training, all images were first converted to grayscale to reduce dimensionality and minimize computational load. This step was particularly important for enhancing the model's efficiency, as it allowed the model to focus on the essential features without being overwhelmed by the complexity of color information. The images were then resized to a uniform resolution of 128x128 pixels to standardize the input size across the model pipeline, ensuring that all images were consistent for training. Pixel values were normalized to the range [0, 1] to accelerate convergence during training and improve model performance. Finally, the dataset was split into training and testing sets using an 80/20 ratio, ensuring that the model had sufficient data for both learning and evaluation. The preprocessing pipeline ensured consistency in model inputs, reduced noise and computational overhead, and improved overall training efficiency, contributing to the success of the models tested.

Model Design

In this project, we explored three convolutional neural network models: a Baseline CNN, an Alternate LeNet-5, and the original LeNet-5. The Baseline CNN was a custom-built architecture that consisted of three convolutional layers, each followed by ReLU activation and max pooling, and two fully connected dense layers. The rationale behind this model design was to create a simple yet effective architecture that could strike a balance between computational efficiency and model performance. Despite its simplicity, the Baseline CNN performed surprisingly well, achieving around 96% accuracy on the test set. The Alternate LeNet-5 was inspired by the classic LeNet-5

architecture but modified to handle 128x128 grayscale input images instead of the original 32x32 size. Significant modifications included adding padding to the convolutional layers to preserve spatial dimensions, incorporating dropout layers for regularization, and using a sigmoid activation function for binary classification. This architecture was specifically designed to capture hierarchical spatial features while remaining computationally efficient, and it achieved the highest test accuracy of approximately 97%. The original LeNet-5 architecture was modified slightly to accommodate the 128x128 input images and binary output. While the LeNet-5 architecture had historically performed well on simpler datasets like MNIST, it struggled to adapt to the larger, more complex satellite images used in this project. Consequently, it achieved a lower test accuracy of around 91%. This underperformance was likely due to LeNet-5's limited ability to capture the complex spatial features in the larger images without additional tuning. Overall, the Alternate LeNet-5 model proved to be the most effective, offering the best combination of performance, accuracy, and deployability, and was chosen as the final model for deployment.

Model Evaluation

The evaluation of the models involved comparing the test accuracy of each architecture. The Alternate LeNet-5 model achieved the highest accuracy at approximately 97%, followed closely by the Baseline CNN at around 96%, while the original LeNet-5 reached 91%. The Alternate LeNet-5 model demonstrated consistent performance, showing minimal overfitting due to the inclusion of dropout and careful selection of convolutional parameters. This model also exhibited smooth convergence during training, which indicated that it had learned effectively from the training data. Furthermore, the model performed well on the validation set, suggesting strong generalization capabilities. Based on these results, confidence in the Alternate LeNet-5 model is high, especially considering that it performed well across multiple test batches and showed robustness in various training conditions. While the model's performance is promising, further evaluation using additional datasets from other hurricane events or geographic locations would be beneficial to confirm its robustness and generalizability in real-world scenarios. Nonetheless, given its high accuracy and consistent results, the Alternate LeNet-5 model is deemed reliable for post-hurricane building damage classification.

Model Deployment and Inference

To enable real-time predictions, the best-performing model, Alternate LeNet-5, was deployed using a Flask-based HTTP server inside a Docker container. This approach

ensures that the model can be easily replicated and run in any environment with minimal setup.

Deployment Steps:

Clone the Repository:

1. git clone https://github.com/CeciNguyen/Classifying-Hurricane-Havoc-One-Pixel-at-a-Time.git
2. cd Classifying-Hurricane-Havoc-One-Pixel-at-a-Time

Build the Docker Image:

3. docker-compose build

Launch the Docker Container:

4. docker-compose up -d
5. The Flask server will be accessible at http://localhost:5000.

Pull Prebuilt Docker Image (Optional):

(Alternatively, users can pull the prebuilt image from Docker Hub)

6. docker pull ssspro/hurricane-damage-model:latest
7. docker run -d -p 5000:5000 ssspro/hurricane-damage-model:latest

Available Endpoints:

- **GET /summary**: Returns metadata such as the model architecture, input/output shape, framework version, and test accuracy.
- **POST /inference**: Accepts a binary image file (PNG or JPEG) and returns a JSON response with the prediction (damage or no_damage).

Example Inference Request:

curl -X POST -F "image=@path_to_image" http://localhost:5000/inference

The model will return a JSON response indicating whether the building is damaged or not:

```
{
  "prediction": "damaged"
}
```

Conclusion

This project demonstrates how Convolutional Neural Networks can effectively classify post-hurricane building damage using satellite imagery. Among the models tested, the Alternate LeNet-5 architecture performed the best, offering the ideal balance of accuracy and computational efficiency. The model was successfully deployed as a containerized web service using Flask and Docker, enabling real-time classification of satellite images. With further training on diverse datasets, this system could become a valuable tool for emergency response teams during natural disasters like hurricanes.