

Project 03

Analyzing Hurricane Harvey's Impact with Machine Learning and Satellite Images

Oswaldo Salinas

Serena Shah

10th April, 2024

Data Preparation

For the project classifying satellite images post-Hurricane Harvey as damaged or undamaged, the dataset was prepped for training. Using `os` and `pathlib.Path`, directories for training and testing were set up. Image filenames were collected with `os.listdir` and sorted into damaged or undamaged categories. With `random.sample`, 80% of images were chosen for training and the rest for testing, avoiding overlap. The `shutil.copyfile` function then organized these images into their respective training or testing sets and categories in the `'data'` directory, streamlining dataset management for effective model training and evaluation. Following the organization and splitting of the data, the next phase involved preprocessing the images to make them suitable for training. The process begins with importing `TensorFlow` and the Rescaling layer from `TensorFlow Keras` to handle and preprocess the image data. Directory paths for both training (`train_data_dir`) and testing (`test_data_dir`) datasets are defined, in preparation for data manipulation. For training purposes, images are processed in batches of 32, with each image size being 128x128 pixels; a similar image size is maintained for testing, albeit with a smaller batch size of 2 to accommodate resource constraints or testing requirements.

Next is to load and categorize training data using `tf.keras.utils.image_dataset_from_directory`, scanning the specified directory for images and inferring class labels from the subdirectory structure. 20% of this data is allocated for validation with the `validation_split` parameter, ensuring a clear separation between training and validation datasets. A random seed ensures reproducibility across runs. After segmentation, a Rescaling layer normalizes pixel values to [0, 1], optimizing model training efficiency. This normalization applies to both training and validation datasets, resulting in `train_rescale_ds` and `val_rescale_ds`. The test dataset, loaded from `test_data_dir`, undergoes similar rescaling for consistency, producing `test_rescale_ds`. This comprehensive data preparation, from loading to normalization, establishes a strong foundation for neural network training and evaluation, ensuring optimal processing of images for all use cases.

Model Design

Artificial Neural Network (ANN)

The first model, an Artificial Neural Network (ANN) designed with Keras, mimics the human brain using layers of interconnected neurons to process data for tasks like classification and pattern recognition. By adjusting the connections or weights during training, it learns to model complex data relationships. The ANN for this project employs a sequential layout starting with a Flatten layer that converts input images into a 1-dimensional array. Following this, it has two Dense layers with 120 and 128 neurons respectively, both using the `relu` activation function for feature extraction and nonlinear transformations. The final output layer is a Dense layer with a single neuron and `sigmoid` function, outputting class likelihood probabilities. It's compiled using the `adam` optimizer and `categorical_crossentropy` for loss. Training occurs over 20 epochs with a batch size of 32 using the `.fit`

method on ``train_rescale_ds``, with ``val_rescale_ds`` as validation data to monitor overfitting and evaluate model generalization.

Lenet-5 Convolutional Neural Network (CNN)


LeNet-5, a convolutional neural network (CNN) model for image classification into categories implemented via Keras. It features a sequential architecture that processes images through a series of convolutional and pooling layers before making classifications through fully connected layers. It starts with a 3x3 convolutional layer with 6 filters and `'relu'` activation to extract basic features from 128x128x3 images, followed by a 2x2 average pooling layer to reduce feature map sizes. A second convolutional layer with 16 filters of size 3x3, `'relu'` activation, and another pooling layer aims at extracting more complex features. This layer aims to extract more complex features from the output of the previous pooling layer. The feature maps are then flattened into a 1-dimensional vector for fully connected layers: the first with 120 neurons, and the second with 84 neurons, both using `'relu'` activation. Flattening is necessary to transition from the 2 dimensional feature maps to a format suitable for dense layers. The final layer, with 3 neurons and `'softmax'` activation, outputs class probabilities. Training the LeNet-5 model involves using ``model_lenet5.fit`` on ``train_rescale_ds`` over 20 epochs with mini-batches of 32 images, optimizing memory and speeding up learning. Validation on ``val_rescale_ds`` assesses generalization to new data, helping prevent overfitting. The ``history`` object records training and validation metrics, indicating learning progress.

Alternative Lenet-5 Convolutional Neural Network (CNN)

The Alternate-Lenet-5 model, a variant of the Lenet-5 architecture, introduces modifications for enhanced complexity and performance. It features an increased number of filters in its convolutional layers, ranging from 32 to 128, compared to Lenet-5's 6 to 16 filters, enabling the extraction of a wider and more complex feature set. This is particularly advantageous for distinguishing between damaged and undamaged buildings. Alternate-Lenet-5 employs max pooling to preserve dominant features, whereas Lenet-5 uses average pooling, which might blend away critical details. Additionally, Alternate-Lenet-5 integrates a dropout layer to prevent overfitting by intermittently omitting neurons during training, a strategy not utilized in Lenet-5. This adjustment potentially enhances Alternate-Lenet-5's robustness and ability to generalize on new data. While Alternate-Lenet-5's advanced design may lead to superior damage assessment accuracy in satellite imagery, it demands greater computational resources. In contrast, Lenet-5's simpler architecture is less resource-intensive but might fall short in capturing the detailed features needed for high precision, making the choice dependent on the project's computational constraints and accuracy requirements.

Model Evaluation

Evaluating three models for classifying satellite images of buildings as damaged or undamaged provides clear insights into the efficacy of different machine learning architectures. The Artificial Neural Network (ANN) model, with its simple dense layer design, has the highest trainable parameter count at approximately 5.9 million.



Although this suggests a high capacity for learning, it also increases the risk of overfitting and requires significant computational power. The ANN model struggles with accuracy, showing early plateauing in validation accuracy and achieving a final test accuracy of just 66.39%, indicating difficulties in handling the complexities of satellite imagery. On the other hand, the LeNet-5 model incorporates convolutional layers and uses average pooling, with a more modest parameter count of around 1.74 million. This design balances complexity with computational demand. During training, LeNet-5 consistently improves, evidenced by a steady rise in validation accuracy and achieving a notable final test accuracy of 95.17%. This superior performance demonstrates the convolutional neural network's (CNN) strengths in image processing.

LeNet-5 and traditional ANNs differ primarily in architecture and application scope. LeNet-5, as a CNN, is tailored for image recognition tasks, featuring layers specifically designed for automatic feature extraction and spatial data processing, such as convolutional and pooling layers. In contrast, traditional ANN architecture consists of fully connected layers and are more versatile, handling a wide range of data types but lacking the inherent spatial processing advantage of CNNs. Consequently, LeNet-5 excels in image-related tasks by leveraging spatial hierarchies, while ANNs potentially offer broader applicability.

The Alternate-LeNet-5 model, utilizing a deeper convolutional structure and max pooling, offers an advanced approach with around 804k parameters, showcasing high learning efficiency and the highest final test accuracy of 98.44%. Its architecture enables nuanced data interpretation, leading to precise classifications. This comparison underscores the limitations of dense-only architectures like the ANN model in processing spatial data and highlights the advantages of convolutional models, particularly the Alternate-LeNet-5's architectural depth and feature extraction capabilities, in achieving superior accuracy in image classification.

Model Deployment

The deployment process involved containerizing the pre-trained model within a Docker image to ensure consistency and ease of deployment across different environments. The command `docker pull serenashah/ml-proj03-api` is used to download the Docker image named `serenashah/ml-proj03-api` from Docker Hub. This image features the high-performing Alternate LeNet-5 model and an inference server for processing and predicting incoming image data. The command `docker run -it --rm -p 5000:5000 serenashah/ml-proj03-api` starts a container from the pulled image. This container hosts an inference server on port 5000 for image classification, accepting HTTP requests. Using `curl localhost:5000/models`, an HTTP GET request targets the `/models` endpoint to fetch details on the deployed model. To classify an image, it must be serialized into a transmittable HTTP format, often by converting it into a list or array of pixel values, stored in the `input_image` variable. The Python code snippet provided demonstrates how to send this serialized image to the inference server using a POST request: `rsp = requests.post("http://172.17.0.1:5000/models", json={"image": input_image})`. The server receives the image data, processes it through the Alternate LeNet model, and returns a prediction indicating whether the image shows a damaged or undamaged building.