

Getting Started: Setting Up the Extreme Weather Classification Pipeline

This guide is your hands-on resource for replicating the Extreme Weather Image Classification project. It covers the necessary environment setup, data acquisition, and a walkthrough of the core scripts in the repository.

Part I: Environment Setup

Before running the code, you need a stable environment. The original project was built using Python 3.9+ and PyTorch (for the modeling).

Clone the Repository: Use Git to pull down the entire project structure from the original source.

```
git clone  
[https://github.com/AryanThodupunuri/extreme-weather-classification.git](https://github.com/AryanThodupunuri/extreme-weather-classification.git)  
cd extreme-weather-classification
```

1.

Create Environment: It is strongly recommended to use a virtual environment (like `conda` or `venv`) to manage dependencies.

```
# Using conda  
conda create -n weather-cnn python=3.9  
conda activate weather-cnn
```

2.

Install Dependencies: All required libraries (PyTorch, torchvision, pandas, scikit-learn, etc.) are listed in the `requirements.txt` file found in the `Supplemental/` folder.

```
pip install -r Supplemental/requirements.txt
```

3.

Part II: Data Acquisition and Preprocessing

The project utilizes satellite imagery data from the Harvard Dataverse. Due to the size (9,081 images), the images themselves are **not stored** in the GitHub repository.

1. **Download the Data:** Locate the raw image file archive linked in the `Supplemental/data_link.txt` file (or directly via the Harvard Dataverse source). Download and extract the full dataset.

Structure the Data Folder: The main script (`scripts/classification_pipeline.py`) expects the extracted image files to reside in a specific location, typically a folder named `data/raw_images` within your project's root directory. Ensure your structure looks like this:

```
extreme-weather-classification/
```

```
|— scripts/
|— data/
|   |— raw_images/ (<- Place all 9,081 images here)
|— Supplemental/
```

- 2.
3. **Run Preprocessing Script:** The first section of the main script handles the crucial steps: class consolidation (5 categories to 2), image resizing (to 224x224), deduplication, and the stratified 80/10/10 train/validation/test split.
The script will output structured data folders (`data/train`, `data/val`, `data/test`) ready for model ingestion.

Part III: Model Training and Analysis

The core logic for training the Baseline CNN, MobileNetV2, and EfficientNetV2 is contained within the primary script.

Execute the Main Script: Run the `classification_pipeline.py` script from your project root. This script is designed to accept command-line arguments (e.g., `--model baseline`, `--model mobilenet`) to specify which architecture to train and evaluate.

Example: Train and evaluate the MobileNetV2 transfer learning model
`python scripts/classification_pipeline.py --model mobilenet`

- 1.
2. **Review Results:** After training is complete, the script automatically saves the performance metrics (confusion matrices, classification reports) to the `reports/` folder.
3. **Generate Interpretation (Grad-CAM):** The final section of the script includes functions to load the best-performing model and generate **Grad-CAM** visualizations. You can call these functions separately or ensure they are enabled during the final evaluation run.
The resulting heatmaps will be saved in the `visuals/` folder, which is necessary for the final documentation component of your assignment.

Your final submission will be judged on the clean, reproducible execution of these steps and the clear presentation of the resulting artifacts.

