Flood Debris Classification System — Week 2 Milestone Report

1. Project Problem Statement

Floods often cause massive disruption in urban areas by leaving behind debris, mud, and stagnant floodwater that block roads and endanger public safety. After such disasters, manual street inspections for damage and cleanup assessment are slow, labor-intensive, and hazardous for workers.

To overcome these challenges, our project aims to **develop an AI-powered image classification system** that can analyze post-flood street images and automatically categorize them into three actionable classes:

- **Safe:** Clean and drivable streets without visible debris or floodwater.
- **Debris Present:** Streets with moderate waste, trash, or obstacles requiring attention.
- **Immediate Cleaning Required:** Roads heavily impacted by floodwater, mud, or large debris that demand urgent cleanup.

This system helps **local authorities, municipal corporations, and disaster response teams** make data-driven decisions and prioritize cleanup operations more efficiently, improving **response time and safety** in flood-affected zones.

2. Dataset Description

Dataset Sources

1. **Roadway Flooding Dataset**
   - Contains images of roads under normal and flooded conditions.
   - Helpful for identifying floodwater and damage severity.
2. **Disaster Waste Dataset**
   - Contains images of disaster debris, garbage piles, and damaged streets.
   - Useful for recognizing waste and obstacles on roads.

About the Dataset

- The images from both datasets were **curated and organized into three main folders**:
  - Safe — normal road and clean street scenes.
  - Debris Present — streets containing visible trash, waste, or small debris.
  - Immediate Cleaning Required — flooded roads, mud-covered surfaces, and large debris.

Dataset Preparation

- All images were resized and normalized for consistent input to the AI model.
- **Data Augmentation** techniques were applied (rotation, flipping, zooming, brightness adjustment) to:
    - Prevent model overfitting.
    - Improve generalization for real-world conditions.
    - Increase dataset diversity.
- Each image was manually verified to ensure correct labeling under respective classes.

3. Model Training and Evaluation

Step 1: Model Setup

- The model was developed using **Google Teachable Machine** for quick AI prototyping and user-friendly training.
- Each class folder (Safe, Debris Present, Immediate Cleaning Required) was uploaded for supervised classification.

Step 2: Model Training

- Several training sessions were conducted, adjusting parameters to balance accuracy and efficiency.
- The model was trained for **multiple epochs** until the accuracy curve stabilized and overfitting was minimized.
- **Real-time preview** helped in understanding class separation quality.

Step 3: Data Augmentation

- Teachable Machine's internal augmentation features (rotation, zoom, brightness variation) were used.
- Additional manual augmentation was performed using tools like ImageDataGenerator for more variation.

Step 4: Evaluation and Results

- Model tested with unseen images to verify robustness.
- Achieved consistent **accuracy (above 90%)** on the validation dataset.
- Successfully predicted correct classes for test images under varied lighting and conditions.
- **Confusion Matrix** analysis showed minimal overlap between "Debris Present" and "Immediate Cleaning Required," indicating strong model learning capability.

Step 5: Model Export

- Trained model exported in **TensorFlow Lite (.tflite)** and **TensorFlow.js (.json)** formats for easy integration with web or mobile apps.
- Model and related assets stored in GitHub for version control.

4. Model Storage and Management

- **File Organization:**
  - /model/ → Trained model files (.tflite, .json, metadata).
  - /dataset/ → Sorted datasets for each class.
  - /test_images/ → New images used for testing.
  - /results/ → Output screenshots and metrics.
  - /docs/ → Reports, README, and progress documentation.
- **Version Control:**
  - Model checkpoints and datasets uploaded to **GitHub repository** for transparent tracking.
  - Future model versions will include retraining with additional real-time flood images for better accuracy.
- **Model Reusability:**
  - Stored models are compatible with TensorFlow.js (web) and TensorFlow Lite (mobile) for deployment flexibility.

5. Week 2 Summary and Achievements

Key Outcomes

- Successfully **collected, cleaned, and labeled** datasets for all three target classes.
- Applied **data augmentation** to enhance model generalization.
- Trained and validated a **robust AI model** capable of flood debris classification.
- Assigned floodwater images under "Immediate Cleaning Required" to ensure the model recommends actionable cleaning operations.
- Exported and stored trained model files with proper folder structuring and version tracking.

6. Next Steps (Week 3 Plan)

🔷 1. Interface Development

- Design a **user-friendly interface** (web or mobile) to upload street images.
- Integrate the model using **TensorFlow.js** (for browser) or **TensorFlow Lite** (for Android app).
- Display results dynamically with a confidence score and class label.

🔹 2. Advanced Testing & Optimization

- Conduct testing on mixed datasets captured under:
  - Different lighting conditions.
  - Various camera devices (phones, drones, CCTV).
- Improve inference speed and reduce model size for deployment efficiency.

🔹 3. Deployment

- Deploy the model on a **cloud-based platform** (Google Cloud / AWS) or host locally using Flask or Streamlit.
- Ensure scalability for real-time flood monitoring.

🔹 4. Documentation & Final Presentation

- Capture training screenshots, evaluation results, and interface demos.
- Prepare final **technical report** including methodology, results, and impact.
- Present the model demonstration to evaluators, highlighting its real-world utility.

7. Attachments

- **Trained Model File:** (TensorFlow Lite & JS formats)
- **Week 2 Report Document (This file)**
- **Dataset Sample Folders** (Safe / Debris Present / Immediate Cleaning Required)
- **README + GitHub Repository Link**