Dataset Access Guide

This guide provides detailed instructions for accessing and using the recommended datasets for your satellite imaging disaster monitoring project.

Setting Up Required Accounts

NASA Earthdata Account

1. **Registration**: Visit https://urs.earthdata.nasa.gov/users/new
2. **Verification**: Confirm your email address
3. **API Access**: Generate API keys from your profile page
4. **Python Setup**: `python from earthdata import Auth auth = Auth(username='your_username', password='your_password') # Or using API key auth = Auth(api_key='your_api_key')`

Copernicus Open Access Hub

1. **Registration**: Visit https://scihub.copernicus.eu/dhus/#/self-registration
2. **Verification**: Confirm your email address
3. **Python Setup**: `python from sentinelsat import SentinelAPI api = SentinelAPI('your_username', 'your_password', 'https://scihub.copernicus.eu/dhus')`

Google Earth Engine

1. **Registration**: Visit https://signup.earthengine.google.com/
2. **Approval**: Wait for account approval (1-2 business days)
3. **Python Setup**:
   `python import ee ee.Authenticate() # First-time setup ee.Initialize()`

Accessing Forest Fire Datasets

NASA FIRMS Data

```
 # Using Earth Engine (recommended for your hardware constraints)
import ee
ee.Initialize()
```

```
# Get FIRMS active fire data
firms_collection = ee.ImageCollection('FIRMS')
recent_fires = firms_collection.filterDate('2024-01-01', '2024-05-01')

# Alternative direct download approach
import requests
from datetime import datetime, timedelta

yesterday = datetime.now() - timedelta(days=1)
date_str = yesterday.strftime('%Y-%m-%d')

# FIRMS URL (requires NASA Earthdata login)
url = f"https://firms.modaps.eosdis.nasa.gov/api/area/csv/[YOUR_API_KEY]/
VIIRS_SNPP_NRT/world/1/{date_str}"
response = requests.get(url)
with open('recent_fires.csv', 'w') as f:
    f.write(response.text)
```

Global Forest Burn Severity Dataset

```
 # Using Google Earth Engine
import ee
ee.Initialize()

# Access GFBS dataset
gfbs = ee.ImageCollection('projects/sat-io/open-datasets/GFBS')

# Filter by year and region
region_of_interest = ee.Geometry.Rectangle([longitude_min, latitude_min,
longitude_max, latitude_max])
filtered_gfbs = gfbs.filterBounds(region_of_interest).filterDate('2015-01-01',
'2016-01-01')

# Export a sample to Google Drive for analysis
task = ee.batch.Export.image.toDrive({
  'image': filtered_gfbs.first(),
  'description': 'gfbs_sample',
  'scale': 30,
  'region': region_of_interest
})
task.start()
```

Google FireSat

```
 # Using Google Earth Engine
import ee
ee.Initialize()
```

```
# Access FireSat data (check latest collection name in GEE catalog)
firesat = ee.ImageCollection('projects/google/firesat')

# Filter and visualize
recent_fires = firesat.filterDate('2024-01-01', '2024-05-01')
```

Accessing Flood Datasets

SEN12-FLOOD Dataset

```
 # Direct download approach
import requests
import os

# Create directory for dataset
os.makedirs('sen12_flood', exist_ok=True)

# Download dataset index (check website for latest URL)
index_url = "https://clmrmb.github.io/SEN12-FLOOD/dataset_index.json"
response = requests.get(index_url)
with open('sen12_flood/index.json', 'w') as f:
    f.write(response.text)

# Download sample images (modify based on index)
# Note: For your hardware constraints, download only what you need
```

NASA LANCE Global Flood Products

```
 # Using Earth Engine
import ee
ee.Initialize()

# Access NRT Global Flood product
flood_collection = ee.ImageCollection('MODIS/061/MCD12Q1')
recent_floods = flood_collection.filterDate('2024-01-01', '2024-05-01')

# Alternative: Direct download with NASA Earthdata credentials
import requests
from requests.auth import HTTPBasicAuth

url = "https://lance.modaps.eosdis.nasa.gov/imagery/subsets/?project=flood"
response = requests.get(url, auth=HTTPBasicAuth('your_username',
'your_password'))
```

ETCI-2021 Flood Detection Dataset

```
# Using Hugging Face Datasets
from datasets import load_dataset

# Load the dataset (will download to cache)
flood_dataset = load_dataset("blanchon/ETCI-2021-Flood-Detection")

# Access samples
sample = flood_dataset['train'][0]
```

Accessing Hurricane/Tornado Datasets

HURSAT Data

```
# Direct download approach
import requests
import os

# Create directory for dataset
os.makedirs('hursat', exist_ok=True)

# Example URL for a specific year (check NOAA website for latest URLs)
year = 2023
url = f"https://www.ncei.noaa.gov/data/hurricane-satellite-hursat/archive/v6/
{year}/"
# Note: You'll need to navigate the directory structure

# For NetCDF processing
import xarray as xr

# Open a sample file
ds = xr.open_dataset('hursat/sample.nc')
```

TorNet Dataset

```
# Check MIT website for latest download instructions
# This is a new dataset, so access methods may evolve

# Example download approach
import requests
import os

# Create directory for dataset
os.makedirs('tornet', exist_ok=True)
```

```
# Download dataset (placeholder URL - check official source)
url = "https://tornet.mit.edu/download"
# Follow specific instructions from the TorNet website
```

NOAA Historical Hurricane Tracks

```
 # Direct download of shapefile
import requests
import os
import zipfile

# Create directory for dataset
os.makedirs('hurricane_tracks', exist_ok=True)

# Download historical tracks
url = "https://www.ncei.noaa.gov/data/international-best-track-archive-for-climate-stewardship-ibtracs/v04r00/access/shapefile/IBTrACS.NA.list.v04r00.points.zip"
response = requests.get(url)
with open('hurricane_tracks/tracks.zip', 'wb') as f:
    f.write(response.content)

# Extract the zipfile
with zipfile.ZipFile('hurricane_tracks/tracks.zip', 'r') as zip_ref:
    zip_ref.extractall('hurricane_tracks')

# Read with GeoPandas
import geopandas as gpd
tracks = gpd.read_file('hurricane_tracks/IBTrACS.NA.list.v04r00.points/IBTrACS.NA.list.v04r00.points.shp')
```

Cloud-Based Processing Strategies

Google Earth Engine for Large Datasets

```
 # Example: Processing Sentinel-1 SAR data for flood detection
import ee
ee.Initialize()

# Define region of interest
roi = ee.Geometry.Rectangle([longitude_min, latitude_min, longitude_max, latitude_max])

# Get Sentinel-1 collection
collection = ee.ImageCollection('COPERNICUS/S1_GRD') \
    .filterBounds(roi) \
```

```
    .filterDate('2024-01-01', '2024-05-01') \
    .filter(ee.Filter.listContains('transmitterReceiverPolarisation', 'VV'))

# Function to detect water
def detect_water(image):
    vv = image.select('VV')
    water = vv.lt(-14)  # Threshold for water detection
    return water.rename('water').copyProperties(image, ['system:time_start'])

# Map function over collection
water_collection = collection.map(detect_water)

# Visualize
water_vis = {'min': 0, 'max': 1, 'palette': ['white', 'blue']}
```

Google Colab for Analysis

```
 # In Google Colab notebook
# Install required packages
!pip install earthengine-api sentinelsat rasterio geopandas

# Authenticate Earth Engine
import ee
ee.Authenticate()
ee.Initialize()

# Download small samples for detailed analysis
# Example: Export a small region from Earth Engine to Colab
```

Data Management Strategies for Limited Hardware

Selective Data Loading

```
 # Example: Load only specific bands or regions
import rasterio

# Open dataset with windowed reading
with rasterio.open('large_satellite_image.tif') as src:
    # Read only a small window
    window = rasterio.windows.Window(0, 0, 500, 500)
    data = src.read(1, window=window)
```

Cloud Storage Integration

```
# Example: Save results to Google Drive from Colab
from google.colab import drive
drive.mount('/content/drive')

# Save results
import matplotlib.pyplot as plt
plt.imsave('/content/drive/MyDrive/satellite_project/results/fire_detection.png',
fire_mask)
```

Chunked Processing

```
# Example: Process large raster in chunks
import rasterio
import numpy as np
from rasterio.windows import Window

with rasterio.open('large_image.tif') as src:
    # Get metadata
    profile = src.profile

    # Create output file
    with rasterio.open('output.tif', 'w', **profile) as dst:
        # Process in 1024x1024 chunks
        for col in range(0, src.width, 1024):
            for row in range(0, src.height, 1024):
                width = min(1024, src.width - col)
                height = min(1024, src.height - row)

                window = Window(col, row, width, height)
                data = src.read(window=window)

                # Process chunk
                processed_data = your_processing_function(data)

                # Write to output
                dst.write(processed_data, window=window)
```

Troubleshooting Common Access Issues

NASA Earthdata Login Issues

- Ensure your account is verified
- Check if you've accepted all required data use agreements

- Try regenerating your API key

Copernicus Open Access Hub Timeouts

- Implement exponential backoff in your requests
- Use sentinelsat's built-in retry functionality
- Consider using mirror sites like Alaska Satellite Facility

Google Earth Engine Memory Errors

- Reduce the spatial or temporal extent of your analysis
- Use .clip() to limit analysis to your region of interest
- Export intermediate results instead of keeping them in memory

Large File Downloads

- Use aria2c or wget with resume capability
- Split downloads into smaller chunks
- Use cloud storage as intermediate storage