Input: Raster bands are in different dimensions , need to do consistent because we are doing u-net modeling how to approach

Output: Because each satellite band comes in **different spatial resolutions** (for example, MODIS 250m, 500m, 1km bands or Sentinel bands at 10m/20m/60m), the raster layers don't line up.
For **U-Net**, all input channels must be **the exact same height × width**, so we need to make every band consistent before feeding them in.

The typical steps:

1. Resample / normalize resolutions

Make all bands match one target size (e.g., 256×256).
You can use:

- `rasterio.warp.reproject`
- or simple `cv2.resize` after reading as arrays
  Common choice: Upscale/downscale everything to the *highest* resolution band.

2. Chip the image

Large rasters → split into small patches (tiles), e.g.:

```
def chip_image(img, chip_size=256):
    chips = []
    for i in range(0, img.shape[1], chip_size):
        for j in range(0, img.shape[2], chip_size):
            chip = img[:, i:i+chip_size, j:j+chip_size]
            if chip.shape[1] == chip_size and chip.shape[2] == chip_size:
                chips.append(chip)
    return chips
```

3. Pad edge regions

If the last chip isn't exactly 256×256:

```
def pad_to_256(arr):

    h, w = arr.shape[1], arr.shape[2]

    padded = np.zeros((arr.shape[0], 256, 256))

    padded[:, :h, :w] = arr

    return padded
```

Input: help me inspect the tiff files, rasterio geotiffs for wildfires 10 of them in folder The last 7 are enivromental bands are the environmental data and everything else is days of the fire. Some fires are weeks long. Others are like two days. eacxh file has different number of bands, i have them saved in folder called Raster_Burn_data

Output: Got it! You have 10 GeoTIFFs, each with multiple bands: the first bands correspond to days of the fire (varying per file), and the **last 7 bands are environmental data**. You want to inspect them—probably to understand shape, size, number of bands, etc.

Here's a Python script using **Rasterio** and **NumPy** to inspect your GeoTIFFs and summarize them:

```
import rasterio

import os

import numpy as np


# Path to your folder

DATA_DIR = "/Users/advaitdarbare/test_475/Raster_Burn_Data"


# List all tiff files

tiff_files = [f for f in os.listdir(DATA_DIR) if f.endswith('.tif')]
```

```
# Inspect each file
for tiff in tiff_files:
    file_path = os.path.join(DATA_DIR, tiff)
    with rasterio.open(file_path) as src:
        num_bands = src.count
        width = src.width
        height = src.height
        crs = src.crs
        transform = src.transform

        print(f"\nFile: {tiff}")
        print(f"Shape: {height} x {width}")
        print(f"Number of bands: {num_bands}")
        print(f"CRS: {crs}")
        print(f"Transform: {transform}")

        # Show info about first and last 7 bands
        fire_bands = num_bands - 7
        print(f"Fire duration bands: 1 to {fire_bands}")
        print(f"Environmental bands: {fire_bands + 1} to {num_bands}")

        # Example: print min/max of first fire band and first env band
        first_fire_band = src.read(1)
```

```
        first_env_band = src.read(fire_bands + 1)

        print(f"First fire band min/max:
{first_fire_band.min()}/{first_fire_band.max()}")

        print(f"First env band min/max:
{first_env_band.min()}/{first_env_band.max()}")
```

✅ What this does:

- Lists all TIFFs in your folder.
- Prints **shape**, **number of bands**, **CRS**, **transform** for each file.
- Separates the fire bands from environmental bands.
- Gives a quick check of min/max values for the first fire band and first environmental band.

If you want, I can also make a **summary table of all 10 files** showing: filename, number of fire days, environmental bands, width, height—so you can quickly compare them.

Do you want me to do that?

Input: need to normalize the enviromental variabls what kinds of factors to use

Output: Based on the analysis of your files (where we saw the "Max" values for each band), here are the best normalization factors to use.

These numbers scale your data so that everything falls roughly between **0.0 and 1.0**, which prevents the model from getting confused by large numbers like Elevation (2000+) vs small numbers like Wind (0.5).