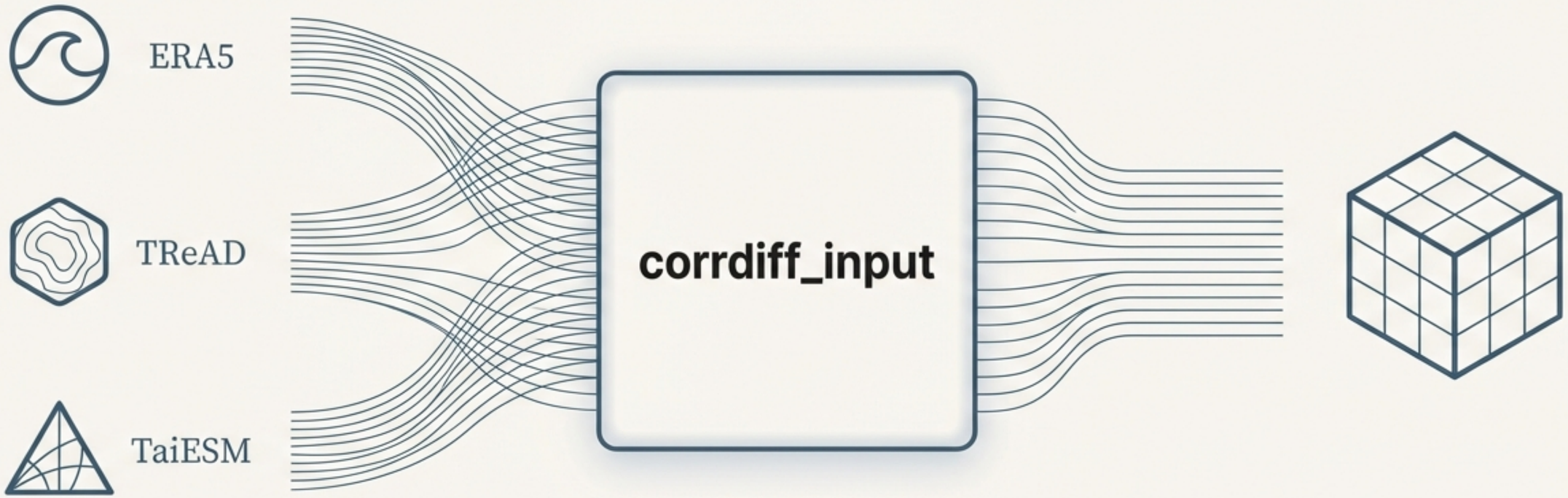
corrdiff_input: The Data Preparation Pipeline for CorrDiff

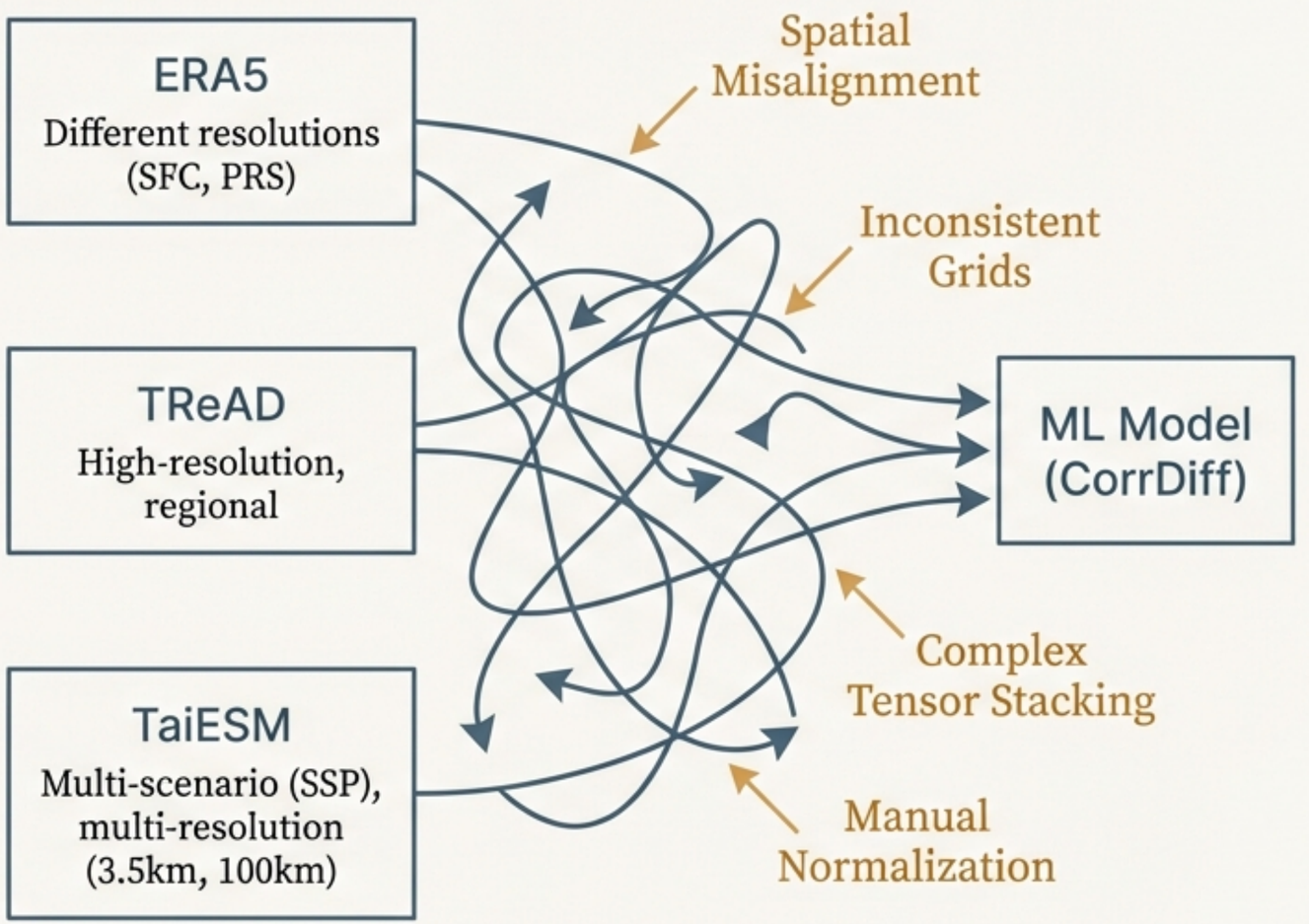
From Raw Climate Data to Model-Ready Tensors.



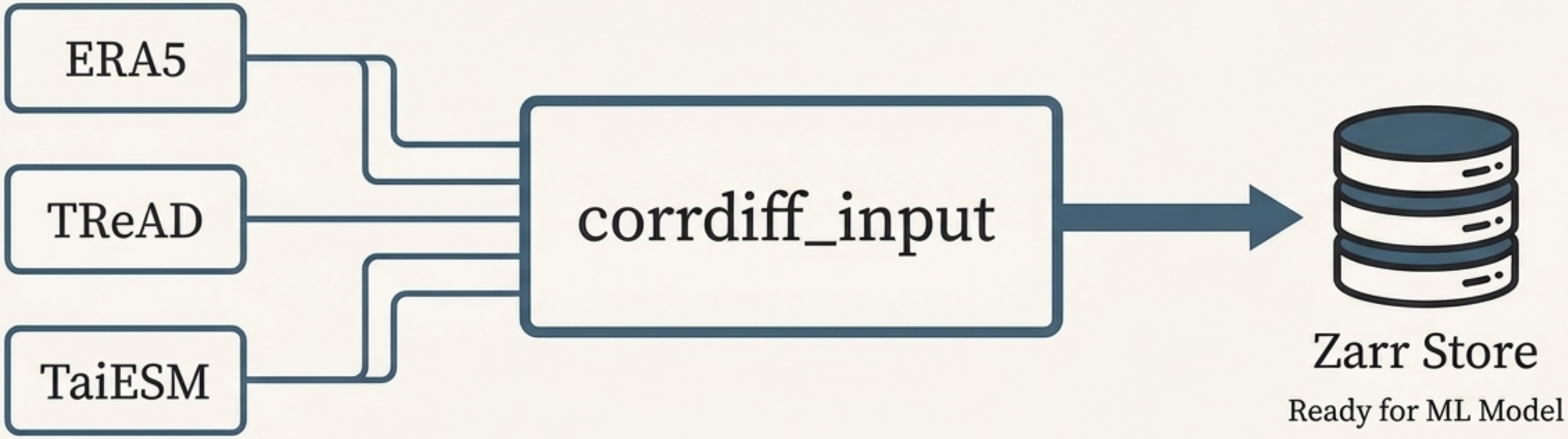
Generate robust, consistent datasets for ConDiff experiments from ERA5, TReAD, and TaiESM climate model outputs.

The Climate Data Labyrinth

- **Multiple Sources, Multiple Formats:** Ingesting raw NetCDF files from disparate sources like ERA5, TReAD, and TaiESM requires bespoke handling for each.
- **Spatial & Temporal Inconsistency:** Data arrives on different grids and with varying time coordinates, demanding complex and error-prone regridding and alignment.
- **Model-Specific Requirements:** CorrDiff expects precisely structured, normalized, and channel-stacked tensors, a final hurdle that can consume significant research time.



A Unified Pipeline for Model-Ready Data



1 Load

Ingests raw NetCDF files.

2 Regrid

Aligns all data onto a unified WRF-style reference grid.

3 Construct

Normalizes, centers, and stacks channels into CorrDiff-compatible tensors.

4 Export

Outputs a consolidated, compressed Zarr store.

The Anatomy of the Pipeline

Input Data

NC

data/era5/

NC

data/tread/

NC

data/taiesm3p5/

Module 1: Data Loading

`src/data_loader/`

Specialized loaders for ERA5, TReAD, and TaiESM handle unique file structures and metadata.

Module 2: Data Building

`src/data_builder.py`

Orchestrates regridding against the reference grid (`ref_grid/wrf_208x208_grid_coords.nc`).

Module 3: Tensor Construction

`src/tensor_fields.py`

Handles normalization, channel stacking (e.g., `cwb_*`, `era5_*`), and metadata creation (center, scale, masks).

Module 4: Main Driver

`src/corrdiff_datagen.py`

The central script that executes the entire pipeline.

Output:

`corrdiff_dataset_...zarr`

Engineered for Consistency and Power

Unified Reference Grid

Creates or loads a consistent **208×208 WRF-style grid**, eliminating spatial misalignment from the start.

Multi-source Dataset Loading

Natively supports ERA5 (SFC & PRS), TReAD high-resolution fields, and TaiESM climate scenarios (3.5km & 100km).

Robust Regridding

Employs bilinear interpolation and nearest-cell extrapolation for accurate spatial alignment.

CorrDiff-Ready Tensors

Automatically constructs `cwb_*` and `era5_*` tensors with associated metadata (center, scale, variable names).

Efficient Zarr Export

Outputs compressed Zarr datasets, ideal for handling large, multi-year climate data with Dask.

Built-in Validation

Includes tools to verify data format and time coordinate consistency across files.

Generate Your First Dataset in One Command

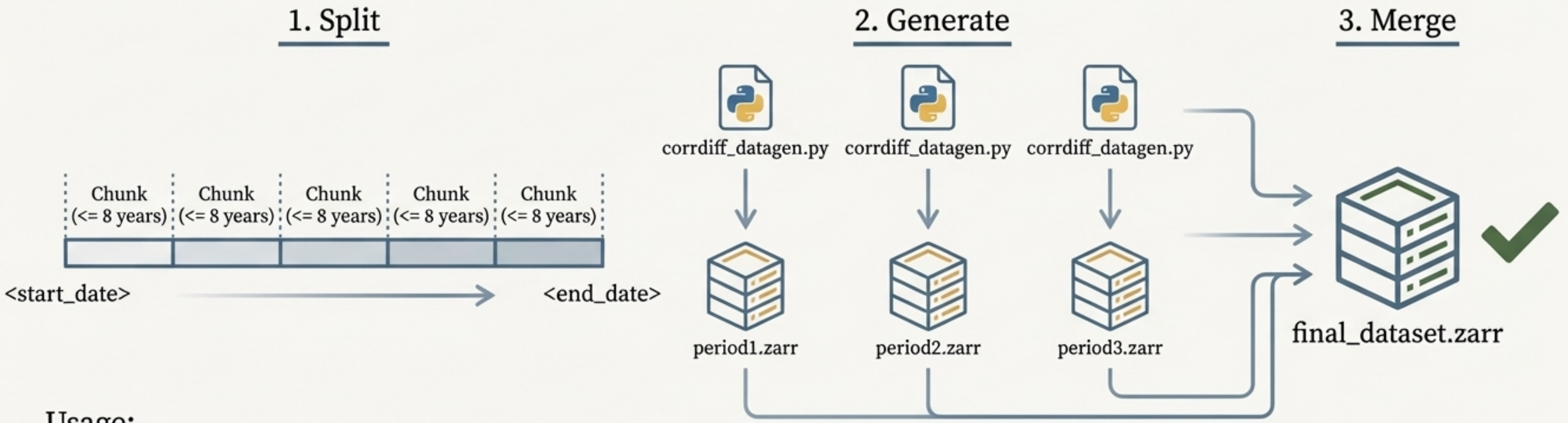
```
**Installation**
# Install all required dependencies from the environment file
conda env create -f env/corrdiff_input.yml

**Usage (CWA Mode: TReAD + ERA5)**
# Generate a dataset for a specific date range
python src/corrdiff_datagen.py 20180101 20180131

**Usage (SSP Mode: TaiESM 3.5km + 100km)**
# Generate a dataset for a specific SSP scenario
python src/corrdiff_datagen.py 20180101 20180131 ssp585
```

This single command loads, regrids, and constructs the CorrDiff-formatted tensors, saving the output to `corrdiff_dataset_<start_date>_<end_date>.zarr`.

Scale Confidently: Handling Multi-Year Datasets Without OOM Errors

🚨 **Problem:** Generating datasets for long time ranges (> 8 years) can cause out-of-memory (OOM) errors, even on large servers.

✅ **Solution:** The `datagen_n_merge.sh` script automates a robust chunking and merging strategy.



Usage:

```
# For TReAD + ERA5
./datagen_n_merge.sh <start_date> <end_date>

# For all future TaiESM SSP scenarios
./datagen_n_merge.sh <start_date> <end_date> all
```

A Complete Toolkit for Data Management and Validation

Inspect & Preview

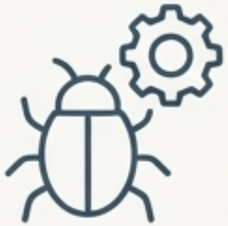
Quickly view the structure and preview data slices of any Zarr file.

```
python src/helpers/dump_zarr.py <input_zarr_file>
```

Filter & Slice

Filter datasets by specific date ranges for analysis or subsetting.

```
python src/helpers/filter_zarr.py
```



Debug Regridding

Enable NetCDF dumps to visually inspect pre- and post-regridding artifacts.

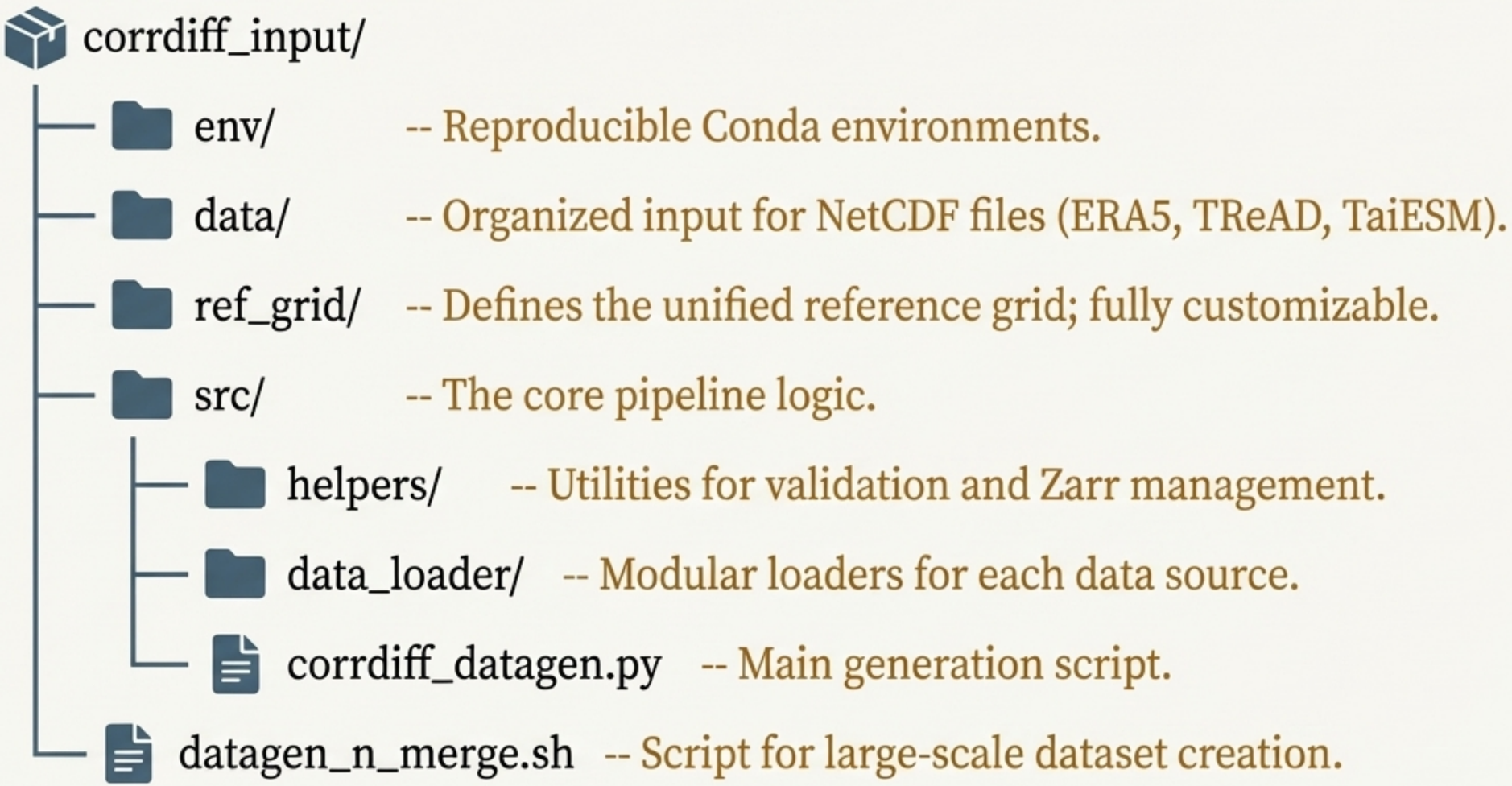
```
DEBUG = True in src/corrdiff_datagen.py
```

Verify Source Data

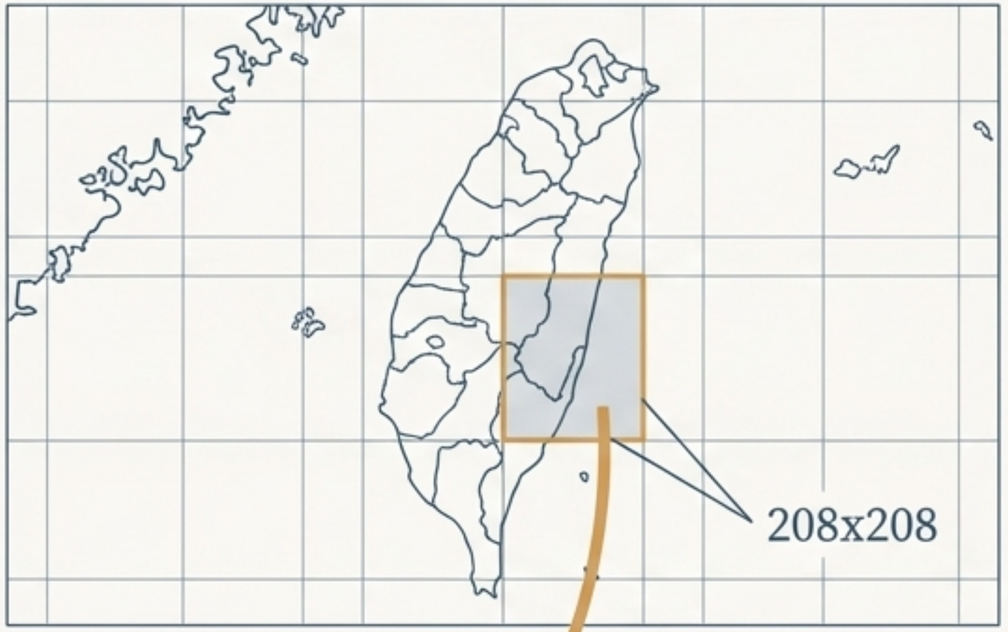
Check low-resolution file formats and time coordinate consistency before processing.

```
python src/helpers/verify_lowres_fmt.py ...
python src/helpers/verify_time_coord.py ...
```

An Organized and Extensible Project Structure



Customize the Reference Grid for Your Research



Step 1: Generate a New Grid

Modify grid dimensions `ny` and `nx` in the grid generation script.

```
# In ref_grid/generate_wrf_coord.py
ny, nx = 208, 208 # Desired grid dimensions
```

Then, run the script to create your custom coordinate file.

```
cd ref_grid
python generate_wrf_coord.py
```

Step 2: Update the Pipeline Configuration

Point the data builder to your new reference grid file.

```
# In src/data_builder.py
CWA_REF_GRID = "../ref_grid/wrf_NEW_GRID.nc"
SSP_REF_GRID = "../ref_grid/ssp_NEW_GRID.nc"
```

Accelerate Research and Ensure Reproducibility

- **Automates** multi-source climate dataset preparation.
- **Handles** large datasets efficiently via Dask & Zarr.
- **Ensures** spatial & temporal consistency across all inputs.
- **Provides** a full suite of debugging and inspection tools.
- **Outputs** standardized, CorrDiff-ready tensors.
- **Extensible** to new climate models and custom grids.

Get Started and Contribute

View on GitHub

github.com/bentian/corrdiff_input

One-Line Installation

```
conda env create -f
env/corrdiff_input.yml
```

Contributions Welcome

Pull requests and issues are welcome. If integrating new datasets or grids, please document them clearly.