Setting Up Earth2MIP: A Step-By-Step Guide

Adam Lizerbram

June 2025

1. Set up your virtual environment. You may use venv, conda, whatever you prefer. On the TIDE cluster I had issues with setting these up so I downloaded the libraries to my .local folder which worked as well. To ensure your virtual environment is set up correctly, run the commands

```
which python3
which pip
```

   which should each return a path to your virtual environemnt.

2. Make sure pip, setuptools, and wheel is up to date.

```
pip install --upgrade pip setuptools wheel
```

3. Clone the earth2mip repository and traverse to it.

```
git clone https://github.com/NVIDIA/earth2mip
cd earth2mip
```

4. Install all requirements. This may take a while to run.

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

5. Install additional packages.

```
pip install ruamel.yaml
```

6. Download the files necessary to run the model. Replace `[link]` with this link.

```
wget '[link]'
unzip fcnv2_sm.zip
```

   This folder includes files `global_means.npy`, `global_stds.npy`, `weights.tar`, and `metadata.json`. All of these files are required to load the model later. These means and stds are used if the model uses the ERA5 dataset, which was used to train the model. If you use your own data, make sure to replace these npy files with means and stds that match your custom data. This is automatically done in the provided scripts. Store the original npy files in a separate folder, because you will need them if you want to rescale the output to match the ERA5 dataset distributions.

7. Now try running `test_earth2mip.py`.

```
python3 /your/path/to/test_earth2mip.py
```

Make sure to replace the file path with your own. You may encounter the following errors. They can be fixed by editing the earth2mip package.

   (a) `No module named 'apex'`
   In earth2mip/networks/fcnv2/sfnonet, replace the line `from apex.normalization import FusedLayerNorm` with `FusedLayerNorm = nn.LayerNorm`. This completely avoids needing the apex package at all. This error should not occur if you have an NVIDIA GPU available to use.

   (b) `WeightsUnpickler Error`
   In earth2mip/networks/fcnv2_sm.py, on line 155, add `weights_only=False` to the arguments in the `torch.load(...)` function.

8. If `test_earth2mip.py` ran successfully, now take a look at `run_all_fcnv2_pipeline.py`. This script executes the other scripts `randinitconditions_fcnv2.py`, `visualize_fcnv2.py`, and `animate_sequence.py`. It can perform the following steps: `predict` (generate a new forecast), `visualize` (create images of each timestep for each variable in the forecast), `animate` (compile images into an animated gif), `clear forecasts` (delete .nc files from the ./forecasts/ directory).

   The script `randinitconditions_fcnv2.py` will use FourCastNetv2 to generate an inference starting from randomly generated initial conditions from the given distribution for the provided number of timesteps. The forecast will be saved as a NetCDF file in the ./forecasts/ directory.

   The script `visualize_fcnv2.py` will create plots for each timestep for all specified variables. There are numerous variables that FourCastNetv2 predicts, and a list of all variables is printed out. This script will also create a folder hierarchy to neatly organize all of the images in the form of distribution/variable/frames.

   The script `animate_sequence.py` will use the images generated from the previous script and compile them into an animated gif for each variable and save it in the corresponding folder distribution/variable.

   Make sure to specify the following variables inside `run_all_fcnv2_pipeline.py`:

   (a) `predict`: boolean to indicate whether to generate a prediction.
   (b) `visualize`: boolean to indicate whether to visualize the results.
   (c) `animate`: boolean to indicate whether to animate the visualized results.
   (d) `clear_forecasts`: boolean to indicate whether to delete all .nc files in the ./forecasts/ directory for each iteration. Recommended to avoid accidentally using too much memory.
   (e) `distributions`: list of distributions to run the full pipeline on.
   (f) `distr_names`: corresponding list of names of the distributions that you want to be displayed in the images. Can use LaTeX code if the names are surrounded by $ as you would in LaTeX. Make sure both `distributions` and `distr_names` are the same length.
   (g) `mean, std, a, b, df`: parameters for each distribution (mean, std used for normal and lognormal; a, b used for uniform; df used for chi-sq).
   (h) `visual_dir`: the directory to save the generated images and animations to.

(i) `model_dir`: the directory which contains the necessary files for the model (`global_means.npy, global_stds.npy, metadata.json, weights.tar`).

(j) `era5_stats_dir`: the directory which contains the original `global_means.npy` and `global_stds.npy` files. This is used to rescale the output data from FCNv2 to match the ERA5 dataset distributions.

(k) `timesteps`: the number of predictions to make including the initial step (if you want 5 predictions after the initial step, set `timesteps = 6`).

(l) `variables`: list of variables to visualize.

(m) `forecast_filepath`: directory and filename to store forecasts in. This is located inside the loop, which gives it a new filename each iteration.

Once all of the variables have been assigned and all paths changed to match your own, you should be able to use this pipeline to easily run any number of predictions using FourCastNetv2 using any custom initial data. You can run the pipeline with the following command:

```
python3 /your/path/to/run_all_fcnv2_pipeline.py
```

Enjoy forecasting with Earth2MIP!