Running Kratzert's Model on CAMELS

My modified version of the CAMELS dataset is uploaded as the `CAMELS` folder at https://github.com/Okstate-WaDE/streamflow-with-NH-model. This dataset is a barebones version of CAMELS that includes the data necessary to train/evaluate models, a list of the 531 basins that Kratzert et al. trained their models on (`basins`), and a pretrained NeuralHydrology model (`nh_model`). It also fixes four misformatted files scattered throughout the dataset. The process I followed for setting it up is detailed below.

I started off by downloading the CAMELS dataset from the NSF website at https://gdex.ucar.edu/dataset/camels/file.html. The most important files for this project are

- `basin_timeseries_v1p2_metForcing_obsFlow.zip`
- `camels_clim.txt`
- `camels_geol.txt`
- `camels_hydro.txt`
- `camels_name.txt`
- `camels_soil.txt`
- `camels_topo.txt`
- `camels_vege.txt`

and other files like `camels_attributes_v2.0.pdf` and `readme.txt` are helpful for understanding attributes. I created a folder called `CAMELS` to store the dataset in. After extracting `basin_timeseries_v1p2_metForcing_obsFlow.zip`, I moved the `basin_mean_forcing` and `usgs_streamflow` folders (from `basin_dataset_public_v1p2`) into the `CAMELS` folder. I created a folder within `CAMELS` called `basins` (for later), then another folder called `camels_attributes_v2.0` and moved all seven `camels_****.txt` (static attributes) files into `camels_attributes_v2.0`.

Next, I downloaded a NeuralHydrology model that worked for the CAMELS dataset. I decided to use the regional models mentioned in "HESS Opinions: Never train a Long Short-Term Memory (LSTM) network on a single basin" (Kratzert, Gauch, Klotz, and Nearing). For this, I needed `basin_list.txt`, a list of the 531 basins their model is trained on, which I have uploaded on GitHub. I stored it in `CAMELS/basins`. After downloading `never-paper.tar.gz` from https://doi.org/10.5281/zenodo.10139248, I extracted it and went to the subdirectory `single-basin-vs-regional-model/run_dirs/regional_model`, where there were several folders. Each folder contains the data for a pretrained model. I chose the first option, `531_basins_multi_forcings_temporal_split_ensemble_member_7_0510_195908`, copied it to a more accessible place (for example, within the `CAMELS` folder), and renamed the folder to `nh_model`. To ensure that the model could run on my system, I had to modify the `config.yml` file within `nh_model`:

- `data_dir` was changed from `camels-data` to the path to my `CAMELS` folder (so in this case, `/Users/*****/Documents/CAMELS`)
- `device` was changed from `cuda:0` to `cpu` since my laptop didn't have a CUDA GPU
- `run_dir` (much further down in the file) was changed from the original (very long) path to the new run directory, i.e. `/Users/*****/Documents/CAMELS/nh-model`
- `test_basin_file`, `train_basin_file`, and `validation_basin_file` went from `/home/gsnearing/hypertuning_experiments/basin_lists/531_basin_list.txt` to the location of `basin_list.txt` on my machine, which was `/Users/*****/Documents/CAMELS/basins/basin_list.txt`
- `train_dir` was changed from its original long path to the `train_data` folder within the new run directory, `/Users/*****/Documents/CAMELS/nh-model/train_data`

To evaluate the model on CAMELS, I installed the NeuralHydrology library for Python. On a computer with Python installed, I ran `pip install neuralhydrology` (sometimes, `python -m pip install neuralhydrology` or `py -m pip install neuralhydrology` works instead). The full documentation is available at https://neuralhydrology.readthedocs.io, but since I only needed to evaluate a pretrained model I ran the command (NOTE: won't work yet) `nh-run evaluate --run-dir /Users/*****/Documents/CAMELS/nh-model` (the last argument would be wherever the pretrained model from `never-paper.tar.gz` was stored). The evaluation ran smoothly at first but crashed roughly a quarter of the way through.

This error was due to a few misformatted files within the CAMELS database. I debugged by modifying the `neuralhydrology/datasetzoo/camelsus.py` file within my locally installed NeuralHydrology library to print out which file it was reading as it ran. The culprit files were all located within the `CAMELS/basin_mean_forcing/maurer` folder, specifically at

- `03/02108000_lump_maurer_forcing_leap.txt`
- `09/05120500_lump_maurer_forcing_leap.txt`
- `11/07067000_lump_maurer_forcing_leap.txt`
- `15/09492400_lump_maurer_forcing_leap.txt`

Each file contains an incorrect header (in the fourth row) that leaves out the labels for the `Year`, `Mnth`, `Day`, and `Hr` columns, as well as missing some capitalization in the other columns. I replaced the faulty headers with the correct header that I found from other files: `Year Mnth Day Hr Dayl(s) PRCP(mm/day) SRAD(W/m2) SWE(mm) Tmax(C) Tmin(C) Vp(Pa)`. I couldn't fit it on one line in this report, but make sure it is only one line in the `.txt` file. After fixing this, I was able to use `nh-run evaluate --run-dir /Users/*****/Documents/CAMELS/nh-model` with no issues.

Converting CAMELS to a GenericDataset

The CAMELS dataset is in a very specific format, and while it would be possible to add new data to it, it's much easier to format new data in a NeuralHydrology's built-in `GenericDataset` format (especially because it uses `.csv` and `.nc4` files instead of `.txt`). To ensure compatibility and get familiar with the format, I started by converting the necessary CAMELS data from the `CamelsUS` format into a `GenericDataset`. As opposed to the base CAMELS dataset that has data split between the `camels_attributes_v2.0`, `basin_mean_forcing`, and `usgs_streamflow folders`, the `GenericDataset` wants static attribute data in `attributes` and both forcing and streamflow data in `time_series`. I won't go too deep into the code that I wrote for this conversion, but I do have a copy of `CAMGEN` (CAMELS but generic) on GitHub.

First, the static attributes. This was the easiest part because a `GenericDataset` wants its attributes in `.csv` files, and CAMELS has attributes in `.txt` files that are formatted like a CSV (comma-separated values) but with semicolons separating entries instead of commas. I wrote a simple Python script to read a file from `camels_attributes_v2.0`, strip any pre-existing commas (this was only an issue in `camels_name.txt`, which contained entries like `Fish River near Fort Kent, Maine`; removing commas did not harm clarity or readability), replace all semicolons with commas, and write the result to a `.csv` file in the `attributes` folder of my generic CAMELS dataset, `GENCAM`.

Next, the meteorological forcing data. I took heavy inspiration from `neuralhydrology/datasetzoo/camelsus.py` in the NeuralHydrology API on GitHub, since they already had code for processing the CAMELS `.txt` files into a Pandas `DataFrame`. After indexing the data by `DateTime`, replacing all "/"s with "per"s in the column names (this was

because the `.nc4` format didn't like forwards slashes, so `prcp(mm/day)_daymet` became `prcp(mmperday)_daymet`), I concatenated the Daymet, Maurer, and NLDAS forcings together and moved onto the streamflow data.

Once again, I used `camelsus.py` as my guideline. I opened the `.txt` files within `usgs_streamflow` using Pandas's `read_csv` function, indexed by `DateTime`, and extracted the `QObs` (observed streamflow) column as a Pandas `Series`. I added this column to the `DataFrame` with all the forcings (and it fit in nicely since they all used the same index), then converted the `DataFrame` to an Xarray `DataArray` that I saved in `.nc4` format. I repeated this process, both forcing and streamflow data, for every basin in CAMELS using a simple `for` loop.

After converting the raw data from CAMELS into a more generic format, I also had to create a new run directory and reconfigure it. I copied `nh_model` from `CAMELS` into `GENCAM` and modified `config.yml` so that `dataset` was changed from `camelsus` to `generic` and any relevant path variables (such as `data_dir`, `test_basin_file`, etc) were updated. I also had to replace every instance of "/" in the `dynamic_inputs` and `target_variables` lists with "per" so that it would match the changes I made to the forcing data. Using a basic replace all function, I made a similar modification to `train_data_scaler.yml` in the `train_data` folder. After these changes, I ran `nh-run evaluate --run-dir /Users/*****/Documents/CAMGEN/nh-model` and the model evaluated without issues. As previously stated, `CAMGEN` is available on GitHub but the `config.yml` file's relevant path variables will need to be filled in.

Combining Open-Source Data with CAMELS

The end goal of this project was to evaluate the pretrained NeuralHydrology model on basins outside of the CAMELS dataset. After getting familiar with the GenericDataset format, it was time to get data for these basins. For the static attributes, I chose to use NLDI data from PyNHD. For the meteorological forcings, I looked at PyDaymet, but due to an internal API, that no longer works. Currently working on using PyGridMET instead.