Notes 2025-04-08

SMAP-HB / WRF-Hydro Project

Table of contents

```r
library(reticulate)
```

```
Warning: package 'reticulate' was built under R version 4.4.3
```

```r
py_require("pyyaml")
py_require("numpy")
py_require("pandas")
py_require("xarray")
py_require("netcdf4")
```

```python
import numpy as np
import pandas as pd
import xarray as xr
import os
```

This week

- ☐ Check fusion of dynamic data / look for other errors that could cause the data to lack the temporal component
- ☐ Update GitHub
- ☒ Clean up tile naming system: currently directories named tile_Y_X – switch to be tile_X_Y
- ☒ Update geospatial_env environment yaml to include new packages I installed: hvplot and pyyaml

Model architecture changes

1. Try increasing the size of the block where I add the dynamic data
2. Try adding the dynamic data at the beginning
3. Try adding the dynamic data in the middle
4. Try adding more epochs

Fix tile names

- Change from tile_Y_X to tile_X_Y
  - Note: data are saved with vars (time, lat, lon), which is a little confusing since lat is y... leave it for now

UNet edits - add more convolutions after dynamic data

- "unet1.py"
- Changes:

```
# Replace original layer
self.fuse_conv = nn.Sequential(
  nn.Conv2d(self.down_channels[-1] + in_channels_dynamic, self.down_channels[-1], kernel_size=1),
  nn.ReLU(inplace=True)
  )

# With 2 layers
self.fuse_conv = nn.Sequential(
  nn.Conv2d(self.down_channels[-1] + in_channels_dynamic,  # input channels
            self.down_channels[-1], kernel_size=3, padding=1),
  nn.ReLU(inplace=True),
  nn.Conv2d(self.down_channels[-1], self.down_channels[-1], kernel_size=3, padding=1),
  nn.ReLU(inplace=True)
  )
```

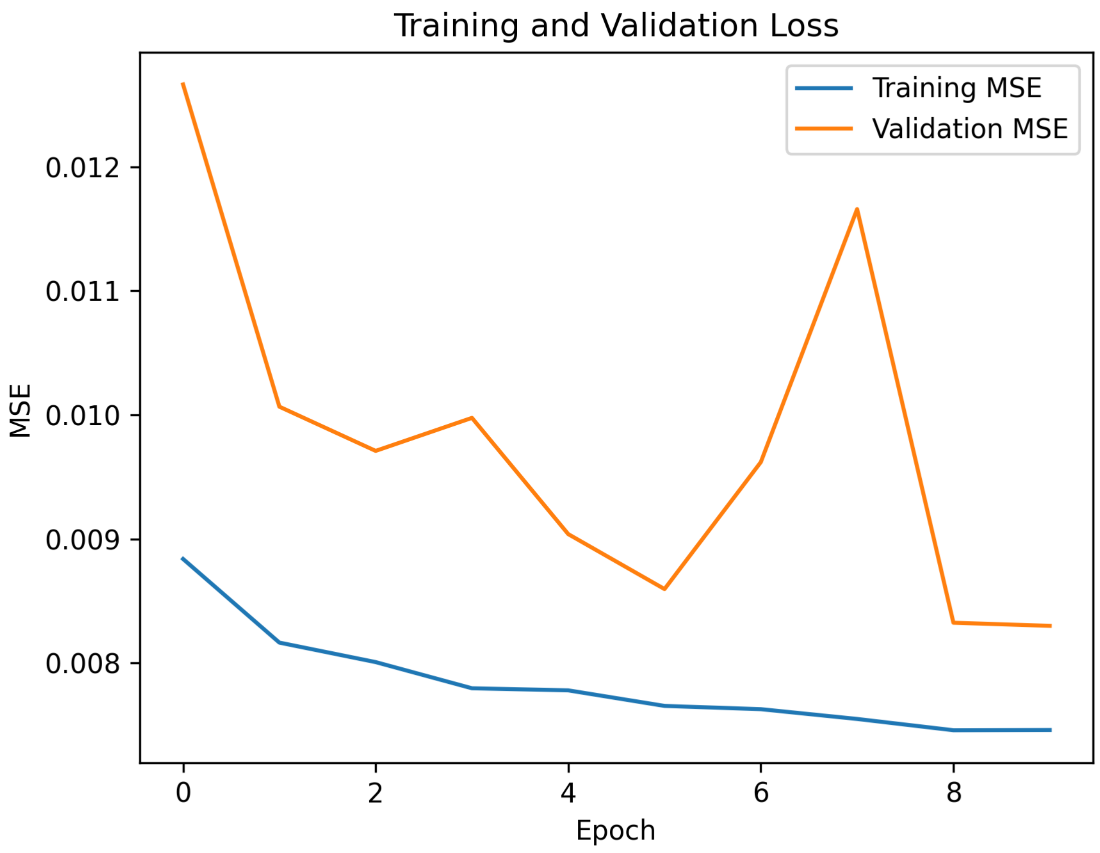

Figure 1: Unet1 training and validation curves

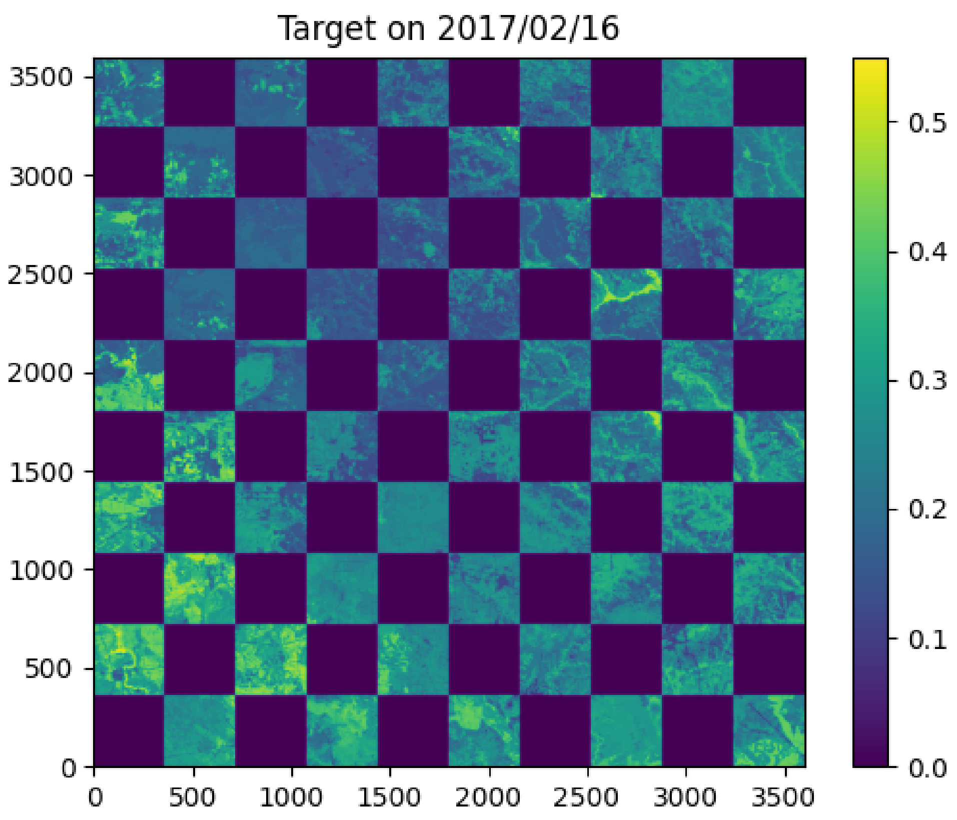

Figure 2: Target

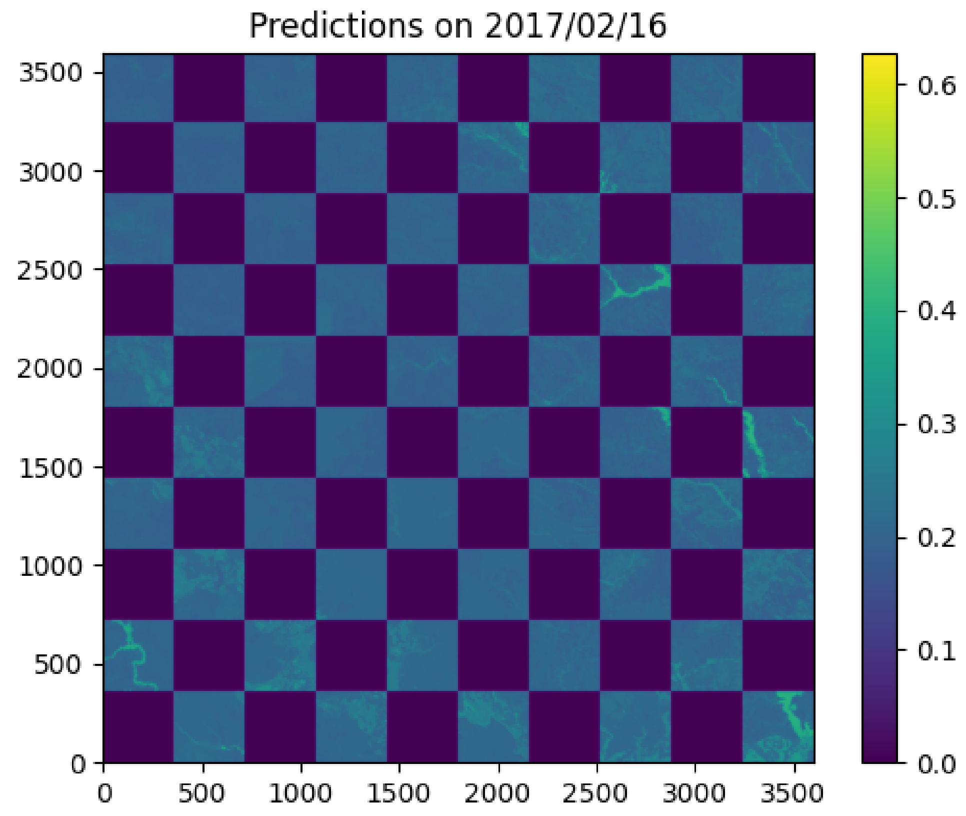

Figure 3: Prediction

First UNet edit

- Doesn't seem very effective

Info on time breakdown:

- Loading: 0.37s
- Training epoch takes ~1000s
- Validation takes ~900s

Should I omit validation to save time since it seems unlikely we're overfitting?

Why is my MSE so small compared to the DeepSoil paper?

- They weight the points around points where they have in-situ data more highly when calculating MSE

$$L_{\text{MSE}} = \frac{1}{B}\sum_{k=1}^{B}\frac{1}{N}\sum_{i=1}^{N}\left(w_i \cdot \left|x_{k,i} \cdot mask_{k,i} - y_{k,i} \cdot mask_{k,i}\right|^2\right)$$

Figure 4: DeepSoil MSE equation

- Maybe it's because they use a physics-based loss function that reduces the tightness of fit in order to ensure the results conform to physical expectation?

Aside: They use bulk density, residual soil water content, organic matter, and log saturated hydraulic conductivity – consider reducing what I include to match?

Sanity check: compare my MSEs to the mean model

- The MSE is only 0.00510 – maybe our study area has less variation than the DeepSoil one
  - But our MSE gets down to ~0.007 – so we aren't even doing as well as the mean D:

```
# Calculate mean and MSE for mean model
mean_over_time = np.mean(all_data, axis=0)
plt.imshow(mean_over_time, vmin=0, vmax=mean_over_time.max(), origin='lower')
global_mean_mse = np.mean((all_data - mean_over_time)**2)
global_mean_mse
```

- Try running with 20 epochs instead

unet2

- Since the first attempt hasn't shown any improvement in terms of reflecting temporal dynamics, try another approach for increasing the influence of the dynamic data
- Downsample the dynamic data and insert them at the beginning
- Didn't go down to 1x1 bottleneck, just to 11x11
- Layers: 180x180, 90x90, 45x45, 22x22, 11x11
- Channels: 64, 128, 256, 512
- 20 epochs

Something wrong with the SMAP 10 km inputs

- When I used hru2grid with 10,000 m resolution, axes were out of order in the output, so I just sorted them again
- Decided it was ok at the time, but maybe it was a sign of a larger issue

hru2grid outputs vs numpy average

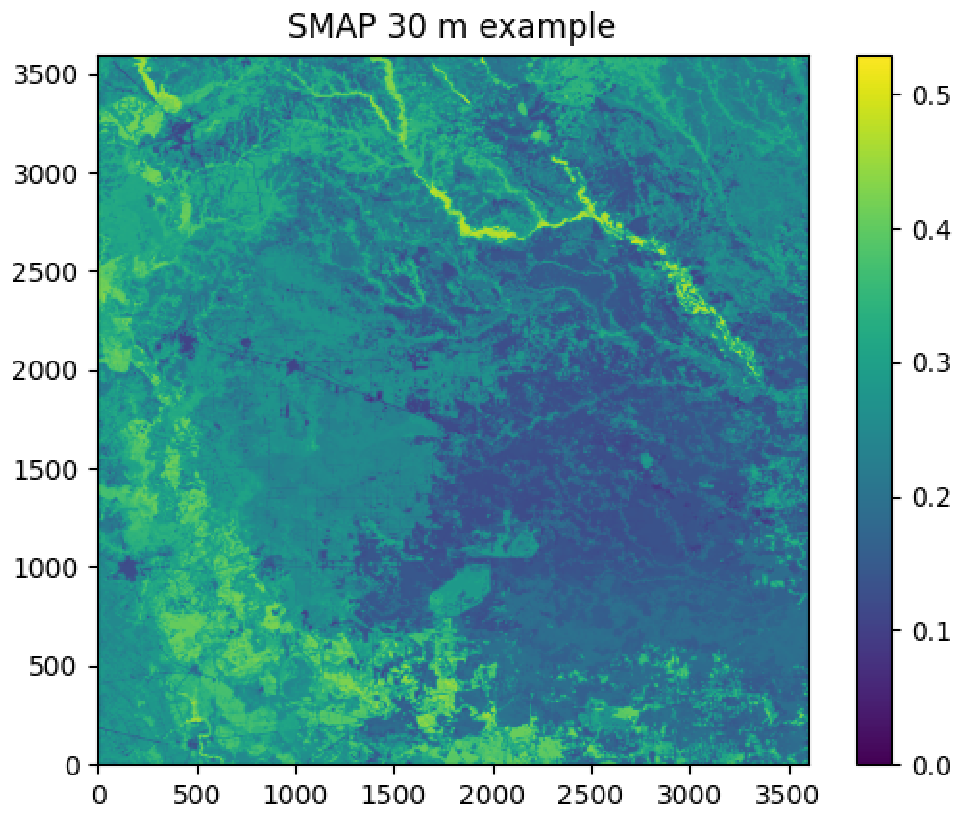

Figure 5: SMAP 30 m original example

hru2grid outputs vs numpy average

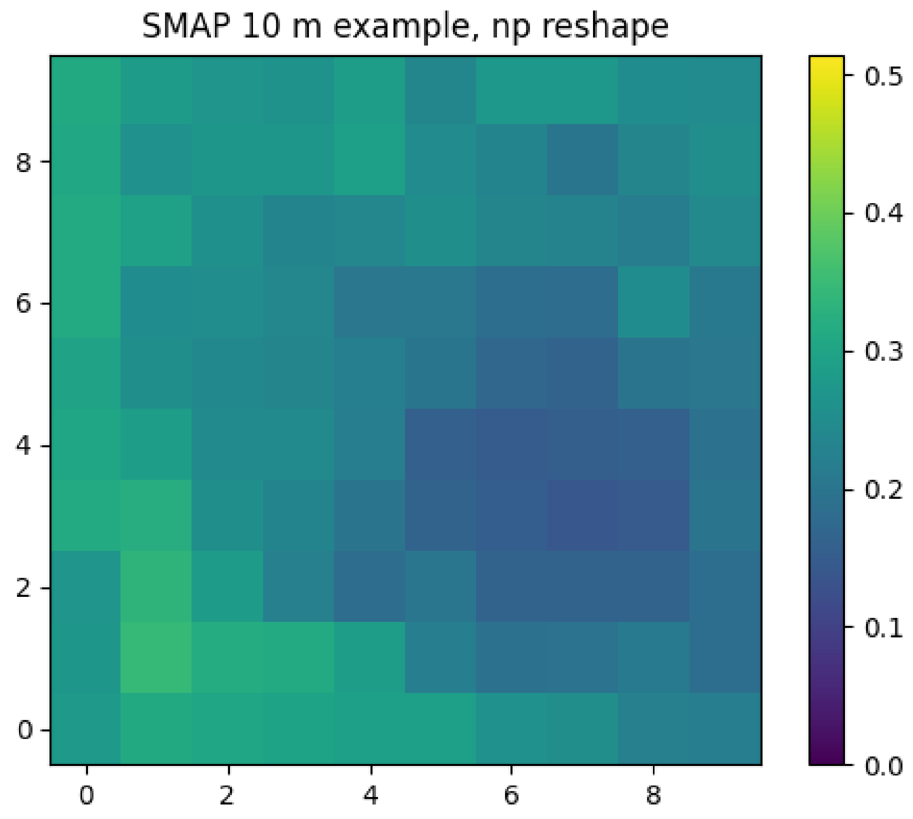

Figure 6: SMAP 10 km example, np reshape approach

hru2grid outputs vs numpy average

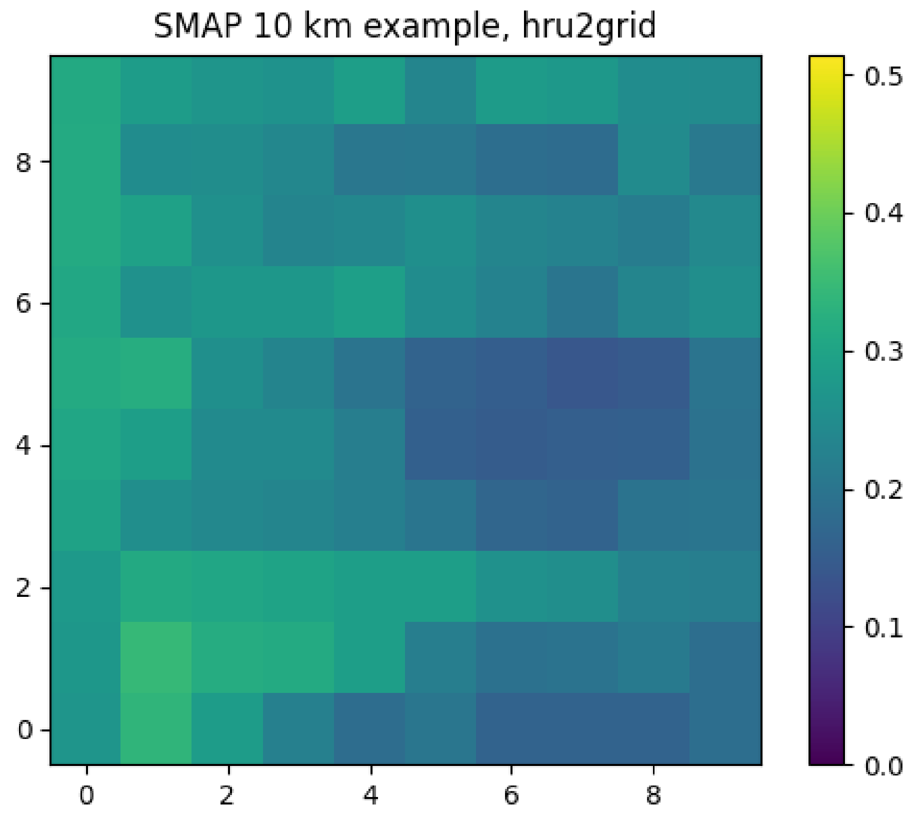

Figure 7: SMAP 10 km example, hru2grid

- Every group of 3 rows vertically is reversed

Raw hru2grid output

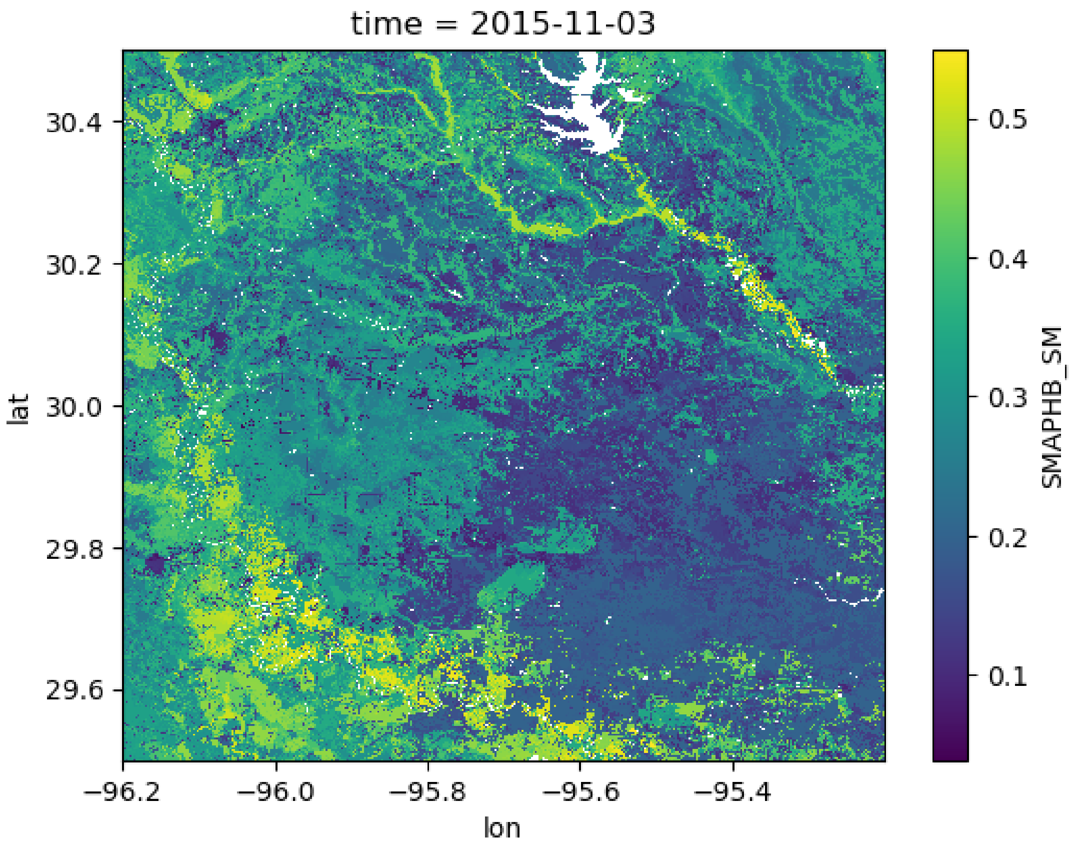

Figure 8: SMAP 30 m example

```
example_10km_smap = example_10km_smap.sortby("lat")
example_10km_smap = example_10km_smap.sortby("lon")
example_10km_smap.isel(time=2).plot()
```

Raw hru2grid output

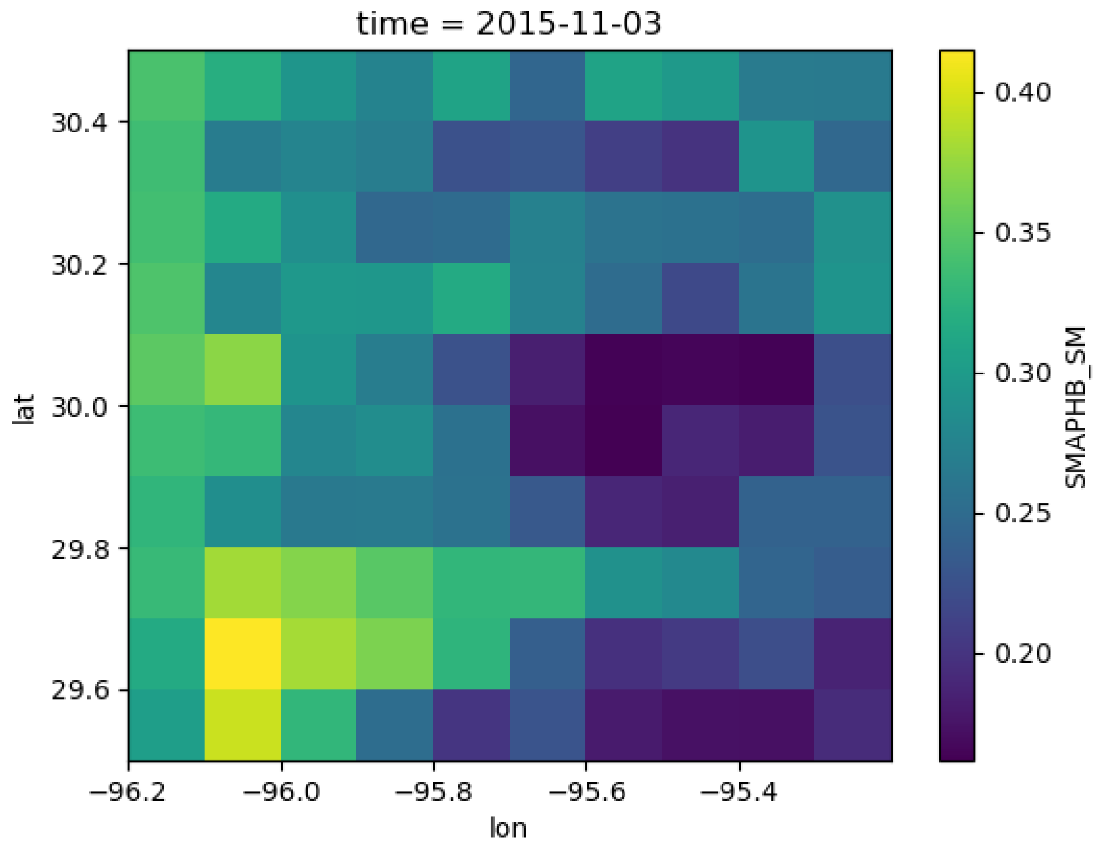

Figure 9: SMAP 10 km example, sorted

```
example_10km_smap.lat.values.sort()
example_10km_smap.lon.values.sort()
example_10km_smap.isel(time=2).plot()
```

Raw hru2grid output

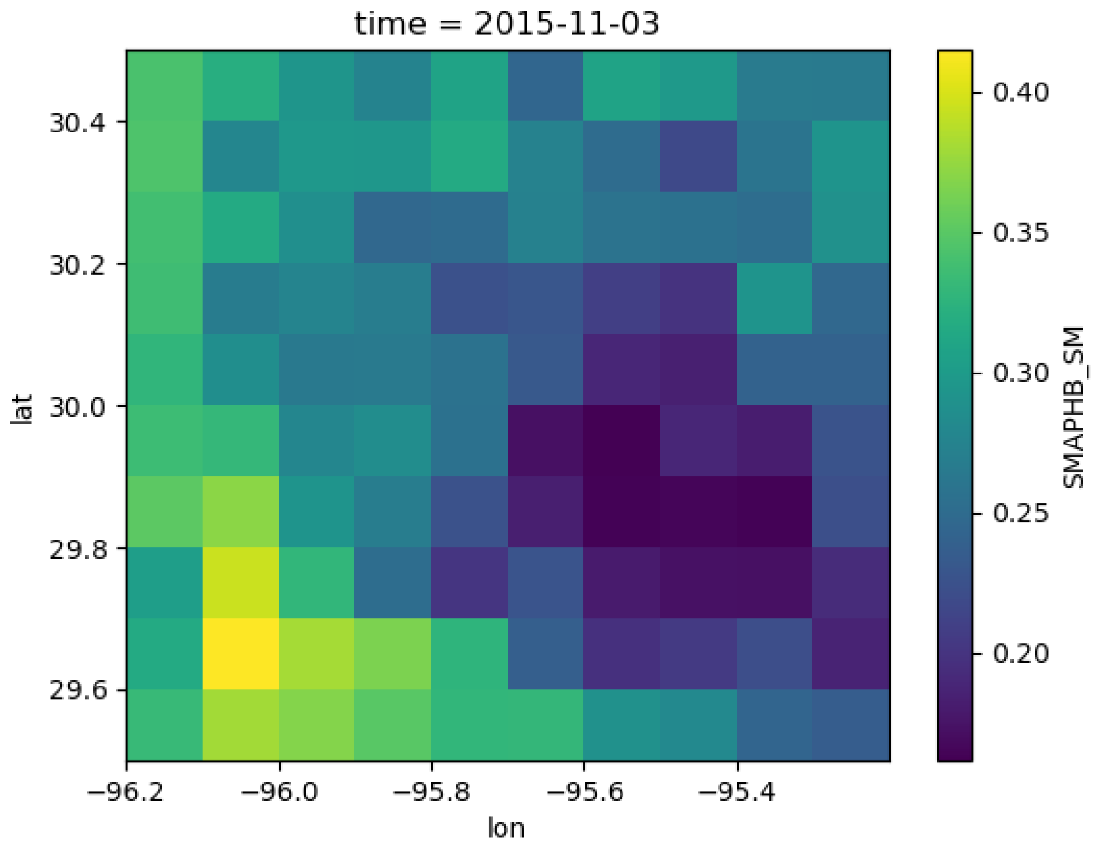

Figure 10: SMAP 10 km example, sort lat/lon but not values

Fix the modified SMAPHB functions

- Seems like the data are correct, but the lat/lon labels are out of order
- Look into this more

Temporary fix: just resample the 30 m data with numpy

- Instead of using hru2grid, use another approach applied to the 30 m data
- Main issue with np method is that it doesn't use geographical info - might have issues with study area where shape of earth comes into play
  - Fast, though – do this for now and later go back to edit the modified SMAP script so that it doesn't scramble the lat and lon
- When more time, use xarray approach to update all dynamic.nc files in tiles folder instead

Having fixed SMAPHB 10km data, run unet3

- Try running unet2 (the one with dynamic data added at the beginning) after having modified the data loader to take the numpy file of fixed SMAPHB 10 km data instead of the original dynamic tiles
- Added one more layer for unet3
- Layers: 180x180, 90x90, 45x45, 22x22, 11x11, (5x5)
- Channels: 64, 128, 256, 512, (1024)
- 20 epochs

Unet with dynamic added at top - old SMAPHB 10 km

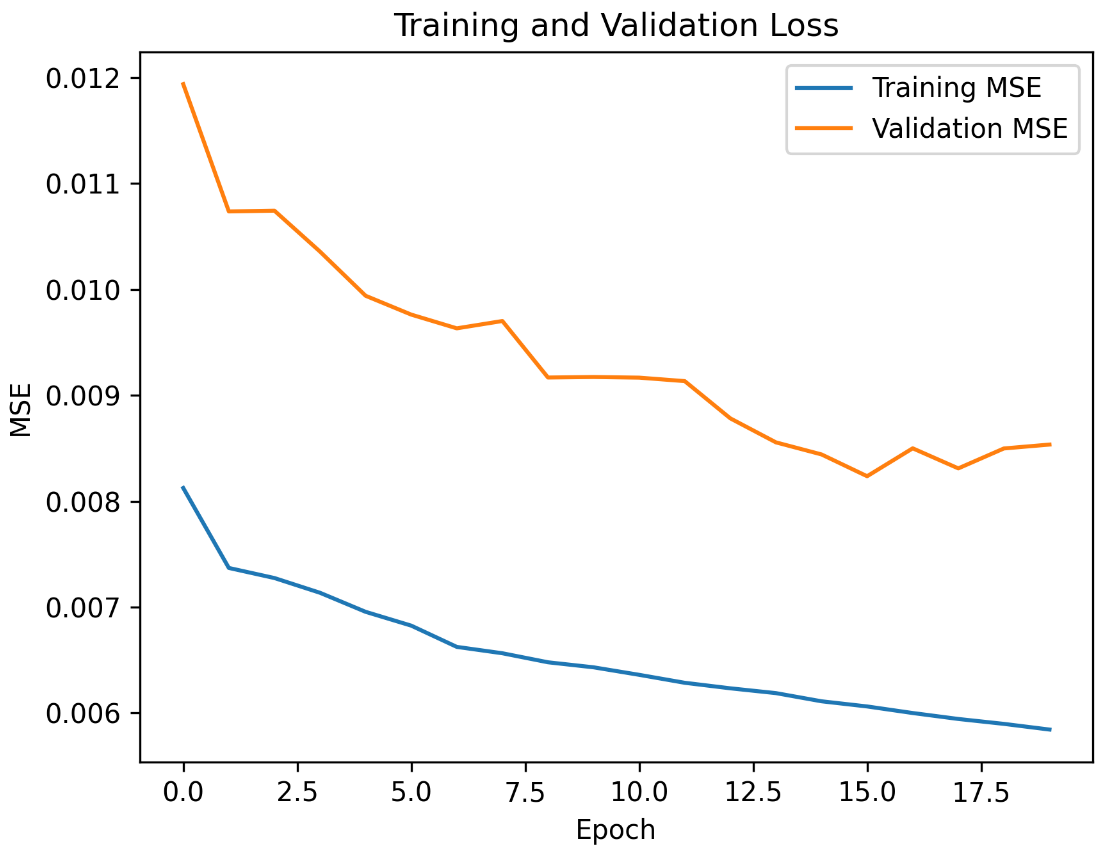

Figure 11: Training curve for model with messed up SMAPHB 10km

Unet with dynamic added at top - fixed SMAPHB 10 km

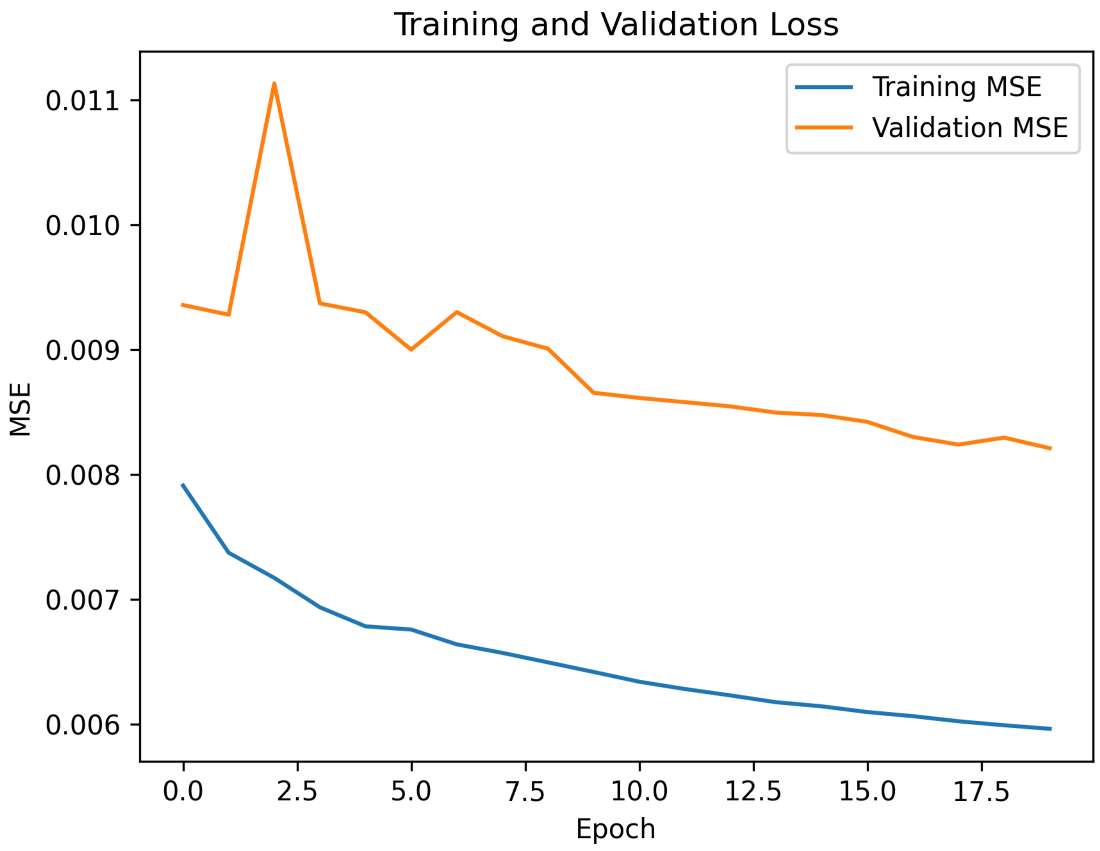

Figure 12: Training curve for model with fixed data