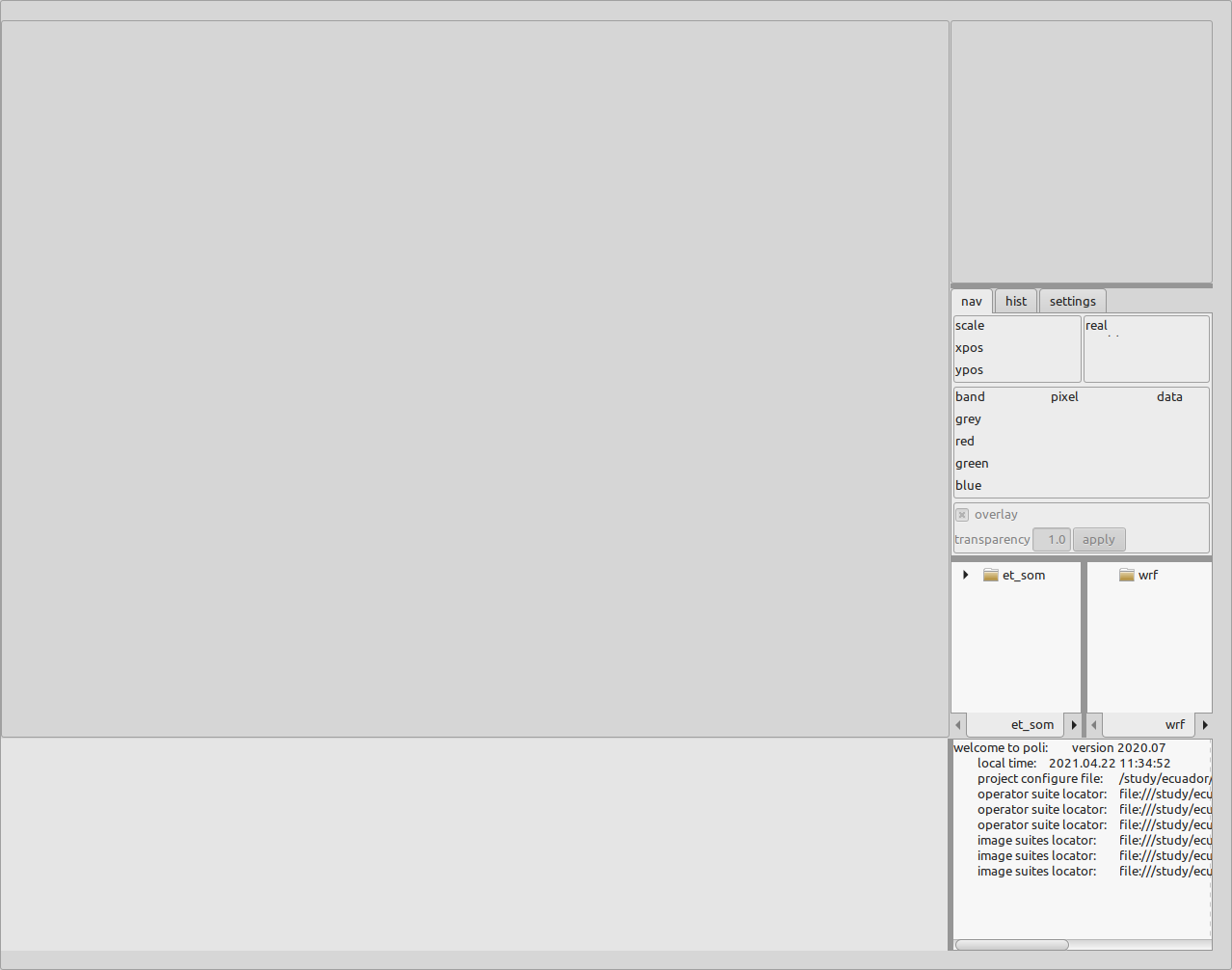
**Brief Tutorial on running SOM with POLI-GUI**

The Python On Line Imaging (POLI) application can be implemented in batch, streamed, or graphical modes. Each mode has its advantages. This tutorial shows how to use the graphical user interface (GUI) with the AI Self-Organizing Map (SOM) operators using output from the Weather, Research and Forecasting (WRF) model.

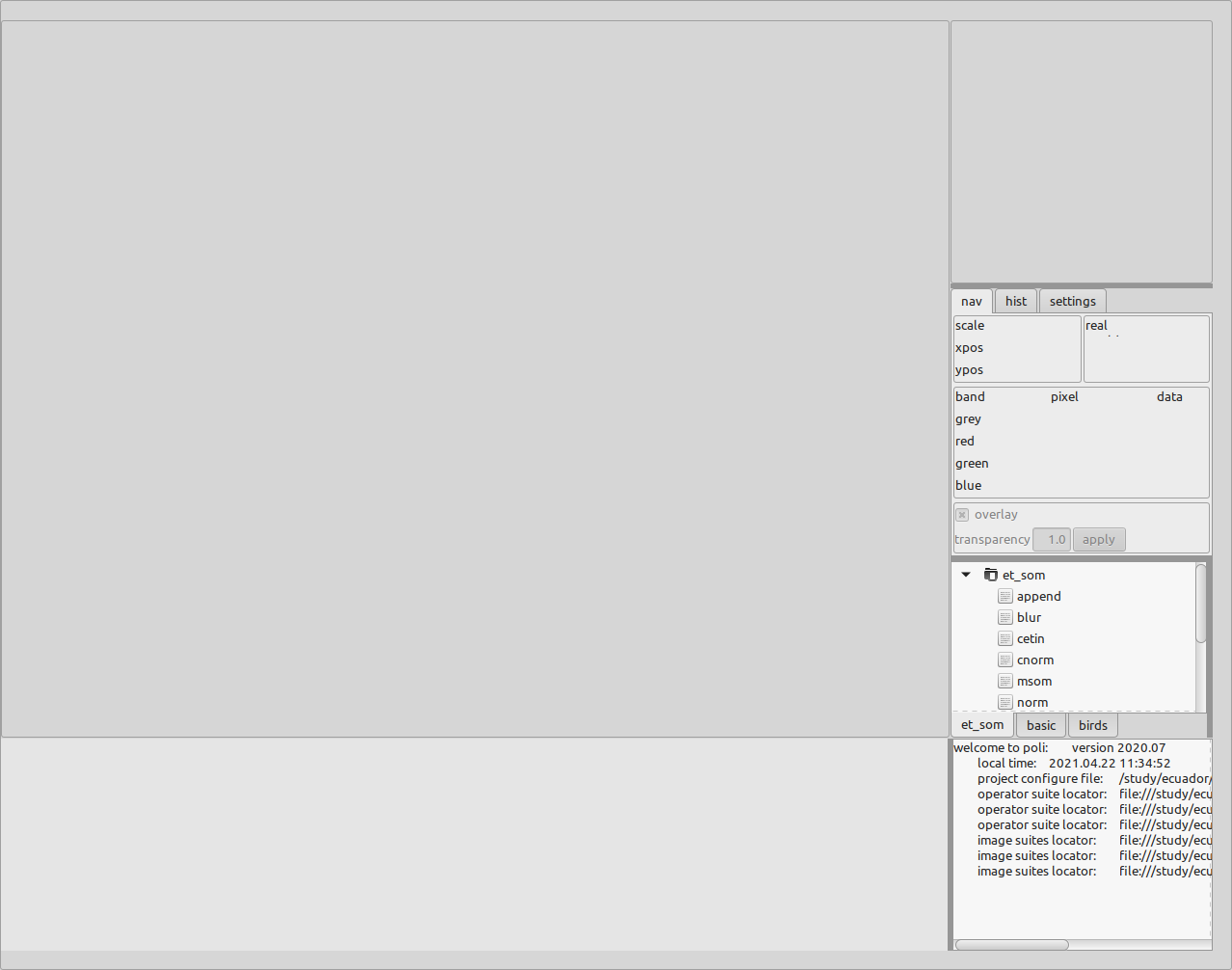
Once POLI has been installed, environment variables set, and project directories updated (see POLI install README.txt) invoke POLI from a command line:

> $POLI\_HOME/bin/work.py $POLI\_HOME/projects/default\_poli

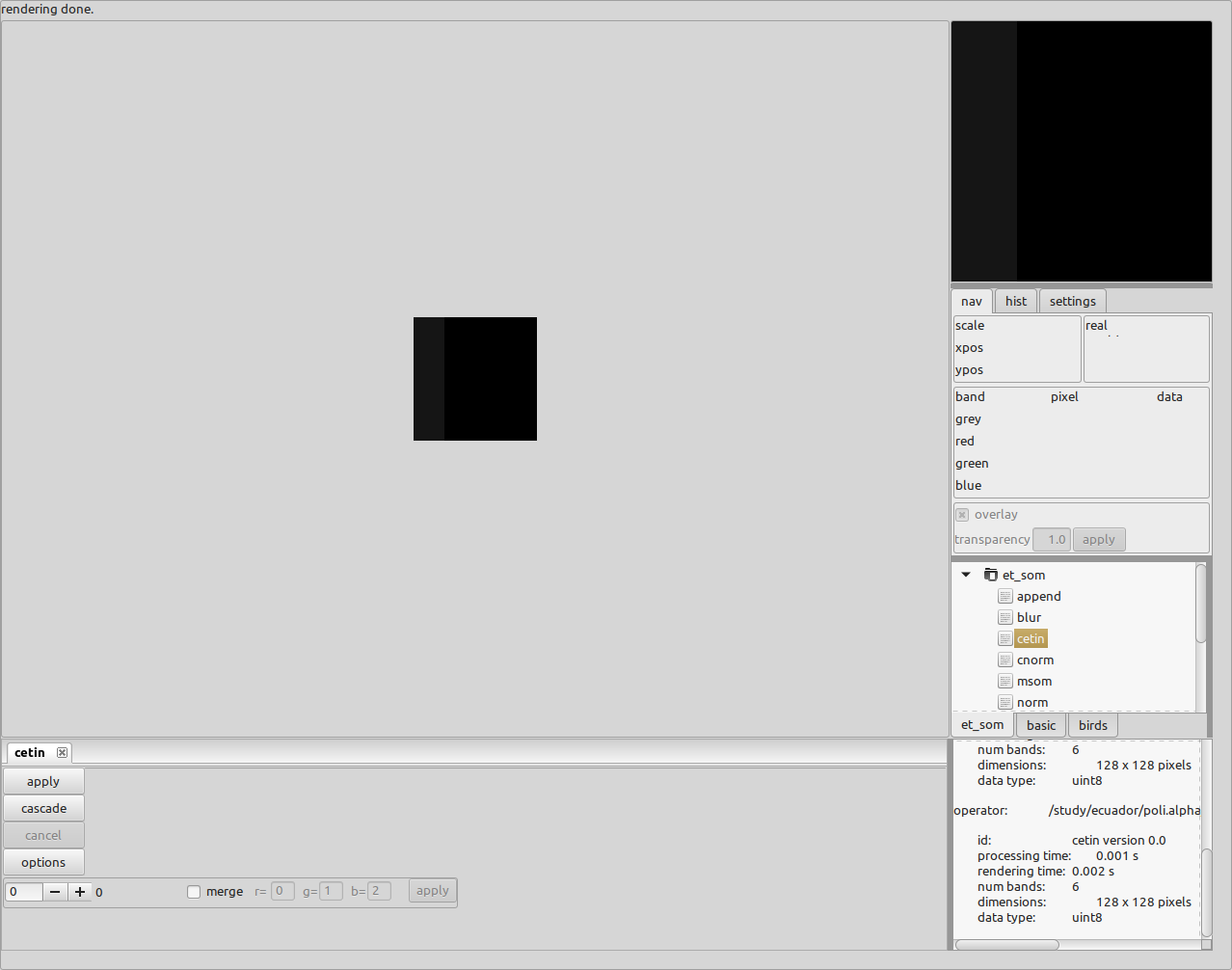
A blank POLI frame should appear:



The SOM and pre-processing operators are in the panel indicated by the arrow below. Click on the tree “et\_som” to view the operators. Note that the panels are adjustable by dragging the dark gray lines.

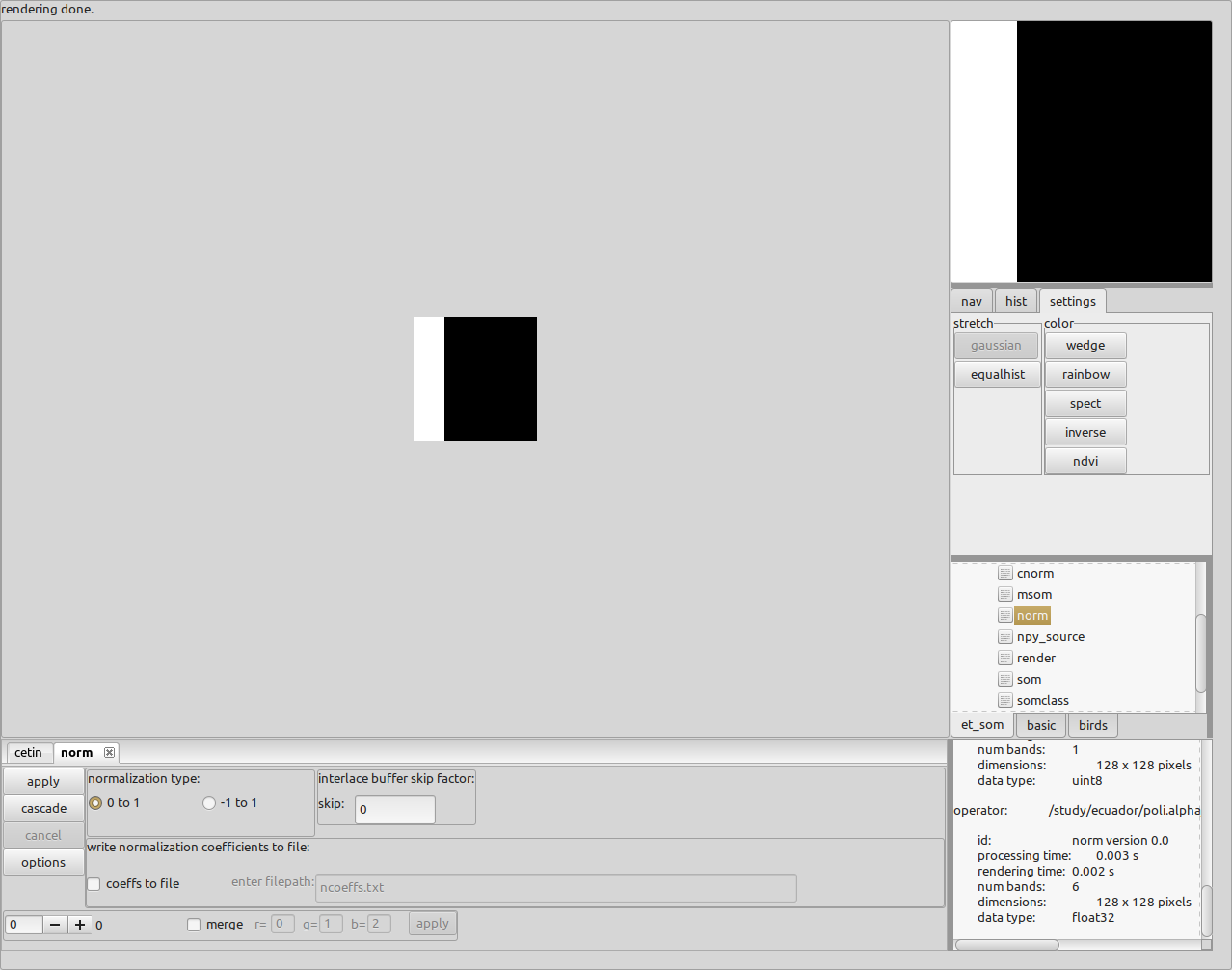


We start the processing with an example using a constructed image, “cetin” . The cetin operator generates a 128x128 data image with 6 buffers (variables) in a way which defines 16 distinct classes. Here we will use this image as a “source” for the processing. Double click on the “cetin” operator in the panel and press the “Apply” in the newly loaded panel. Use the “-” or “+” buttons (indicated by arrow) to scroll through the different buffers.

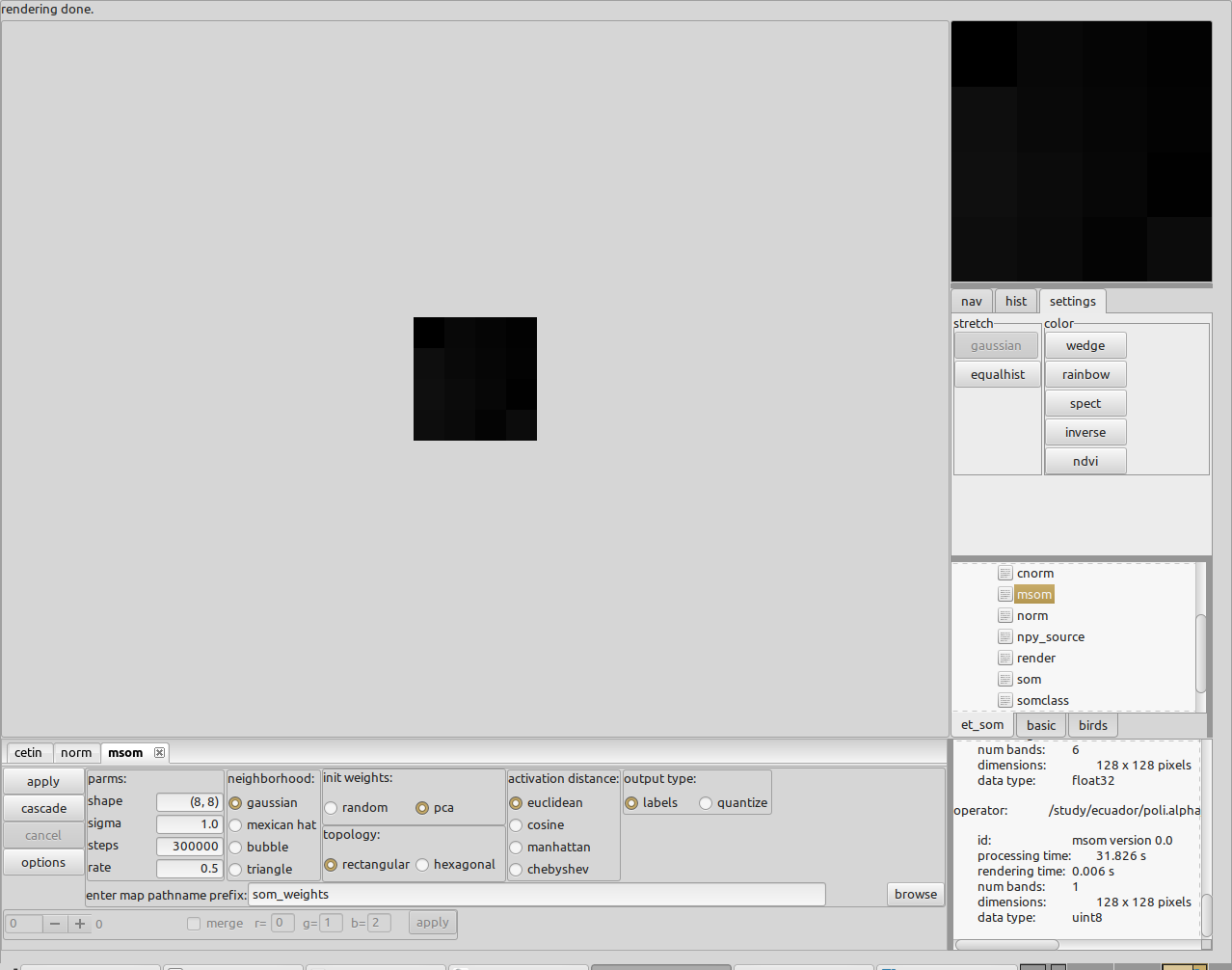


For this example it is not necessary to normalize, but we do so for demonstration sake. Go to operator tree and double click on the “norm” operator. The “norm” operator can normalize from -1 to 1 or from 0 to 1. We use the 0 to 1 option. It also has a parameter which skips buffers for normalization, and is needed when we later concatenate feature space data. The last parameter writes the normalization coefficients to file for later use when applying the classifications to target data. Since we are not using the classifications later, leave this unchecked.

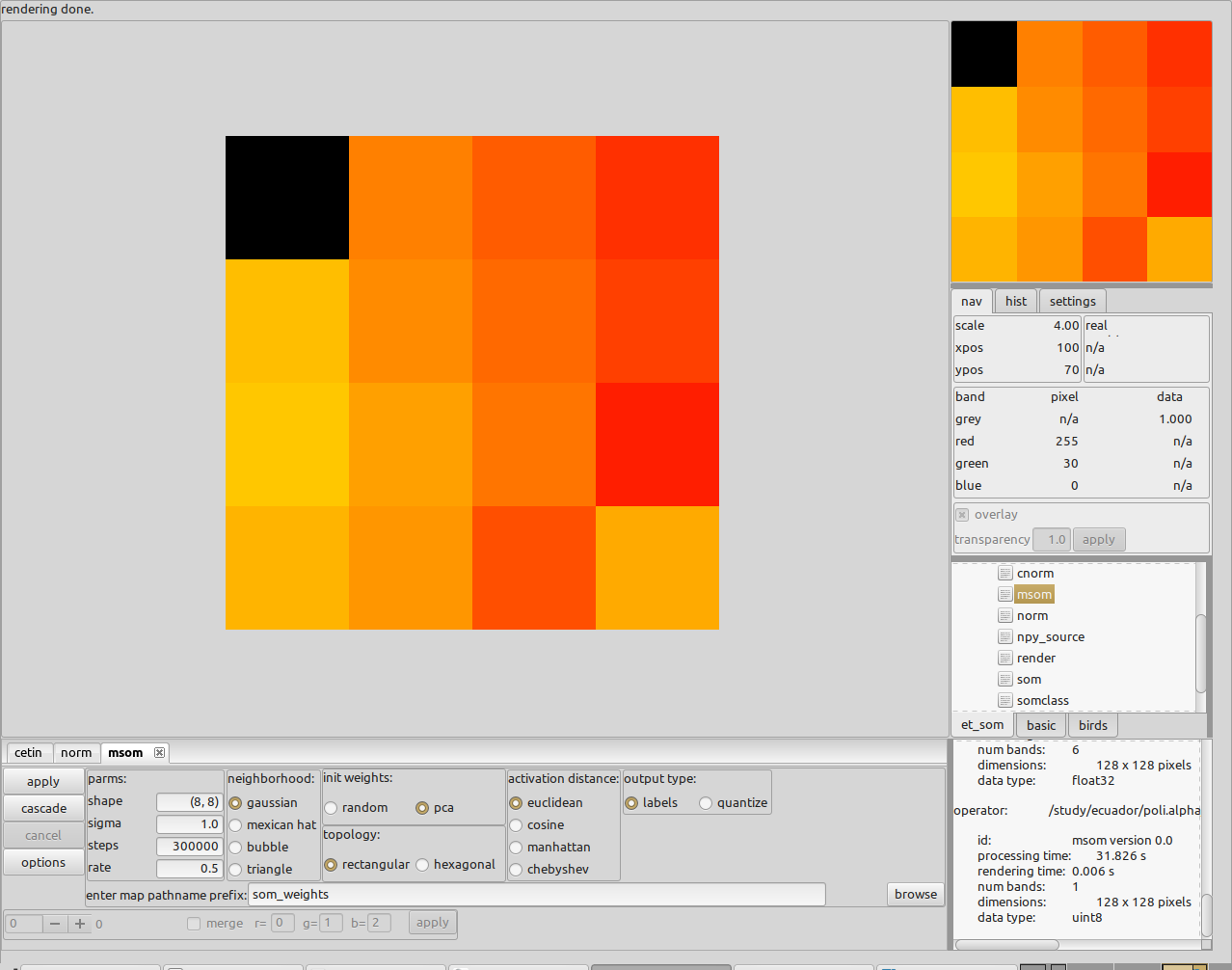
A key point here is that the operators use the previous operator’s output as input. Here we are using “cetin’s” output as the input for the “norm” operator”. Press the “apply” on the “norm” operator and to see results scroll through the buffers. You can also enlarge the image by using the mouse scroll wheel while hovering over the image.



The operator panel has two SOM implementations, we start with the “msom” operator. Double click on the “msom”. The SOM operators have parameters that indicate net geometry, steps, methods, initialization and neighbor influences. These values are outside the scope of this tutorial, however users should take the time later to get a good understanding of these parameters, as they become important in for specific implementations. For this example, press the “apply” button on the “msom” operator. The operator takes a few seconds to complete and a progress report is mode at the top of the POLI frame.

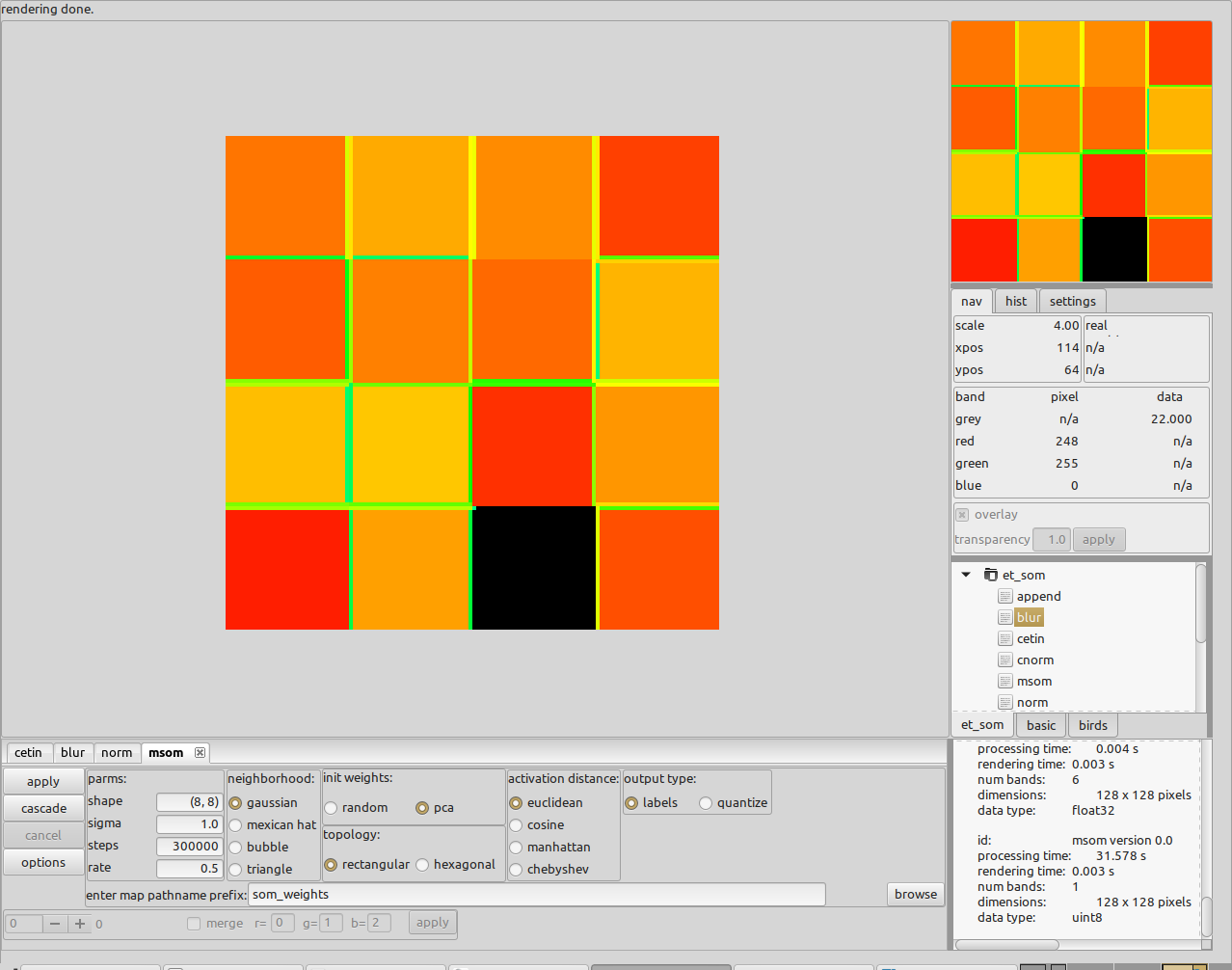


The rendered image looks empty, but since the display is showing classification labels as gray levels from 0 to 15 the image appears black. To colorize the gray levels press the “rainbow” button on the “settings” panel as indicated by the arrow below. The values of the class label can be interactively viewed by first pressing the “nav” tab and then hovering over the image. Class values given by the grey band data value in the “nav” panel. Enlarge the image by using the mouse scroll wheel. Classes and their weights are written to file as text, in this case som\_weights.labels.



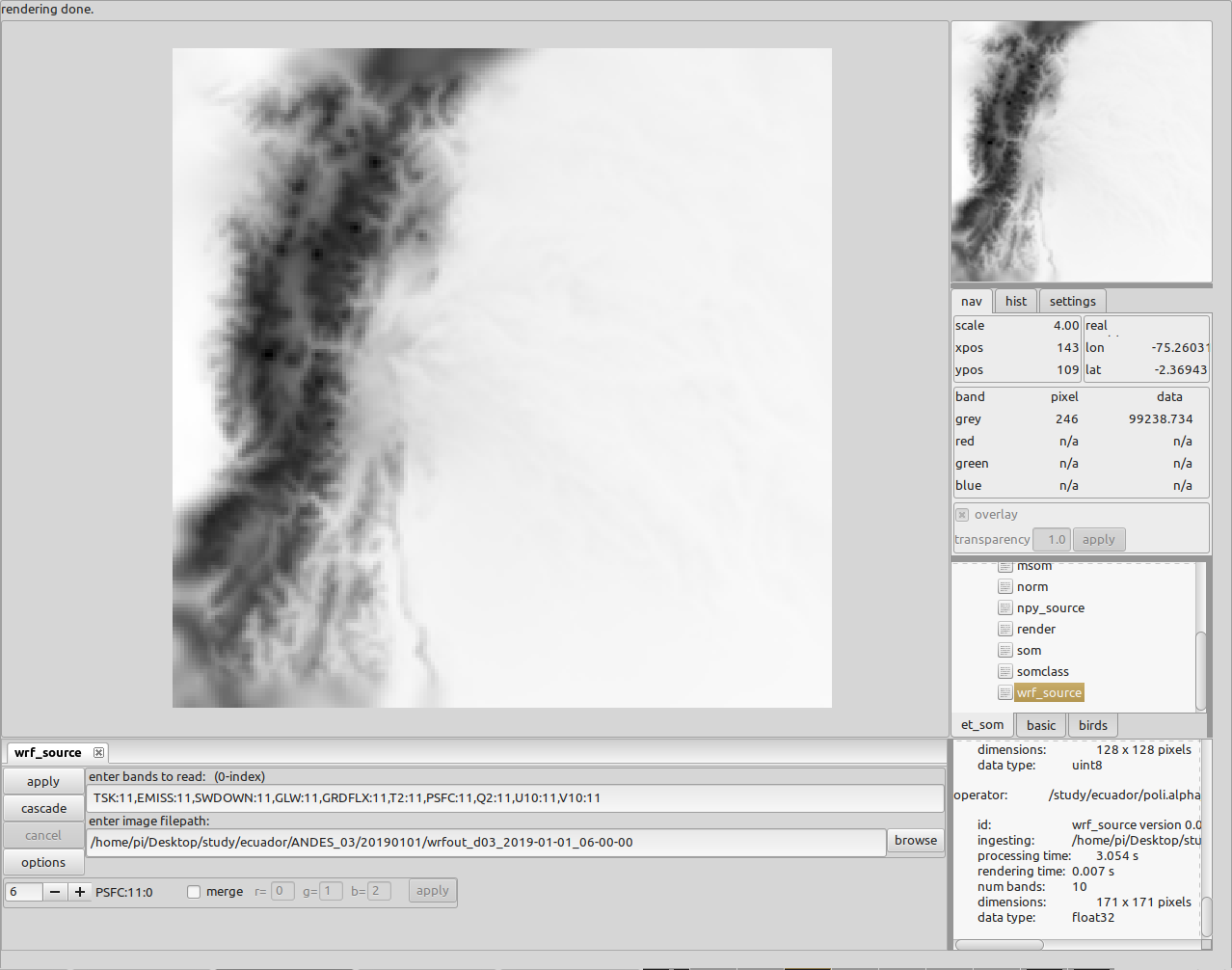
Pressing the “apply” button again will process the normalized data as before, but the outcome will be different as it will still give 16 distinct classes, but the labeling is different.

An interesting exercise is to blur the source “cetin” image in order to induce more classes. This is done by first clicking on the “cetin” tab and then selecting (double clicking) the “blur” operator from the operator tree. Press the “apply” button on the “blur” operator. Press the “norm” tab and then press “apply”, and finally press the “msom” tab and press “apply”. More classes have be identified due to the blurring as seen below.



The second SOM operator “som” can be used in place of “msom” and is left to the reader as an exercise. The reader can start from the beginning or can remove the “msom” operator by clicking on the “x” on the tab and then double clicking on “som” in the operator tree. Both “msom” and “som” operators output the class weights in the same format so that the same program can be used to apply these weights to target data.

We now move on to using WRF data as our source. Start with a blank POLI frame or remove existing operators by clicking on the “x” on all the operator panel tabs. From the operator tree double click on the “wrf\_source” operator. POLI is designed to accommodate different data sources, so for example to use a “JPEG” image the “source” operator would be used. Here we are using an operator specifically for the netCDF output from WRF. Since WRF outputs hundreds of variables we wish to selectively read the variables of interest and so a parameter must be set to specify which variables to use and the time slice.

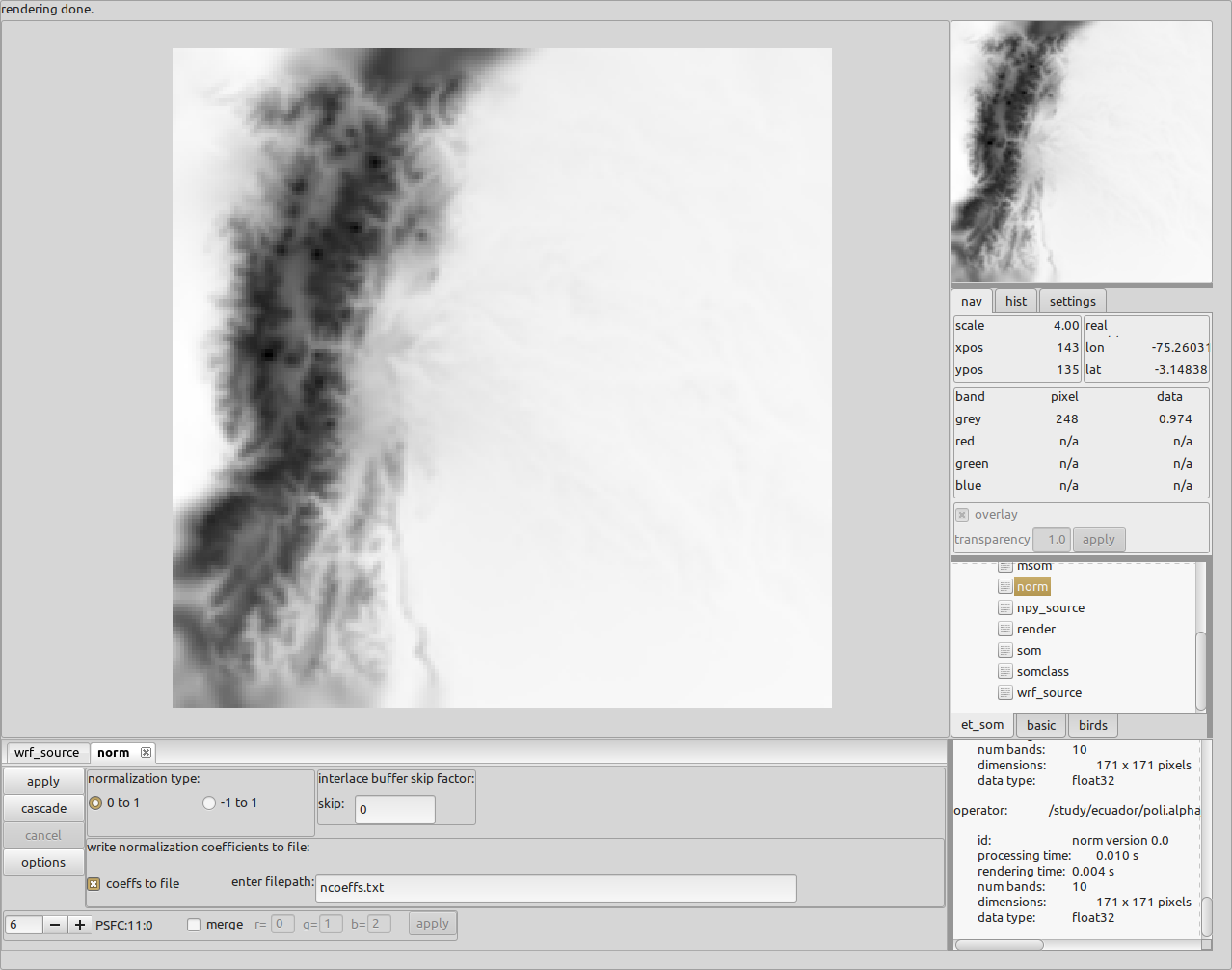


So, to specify the raw variables to calculate ETo at time slice 11 we use the string:

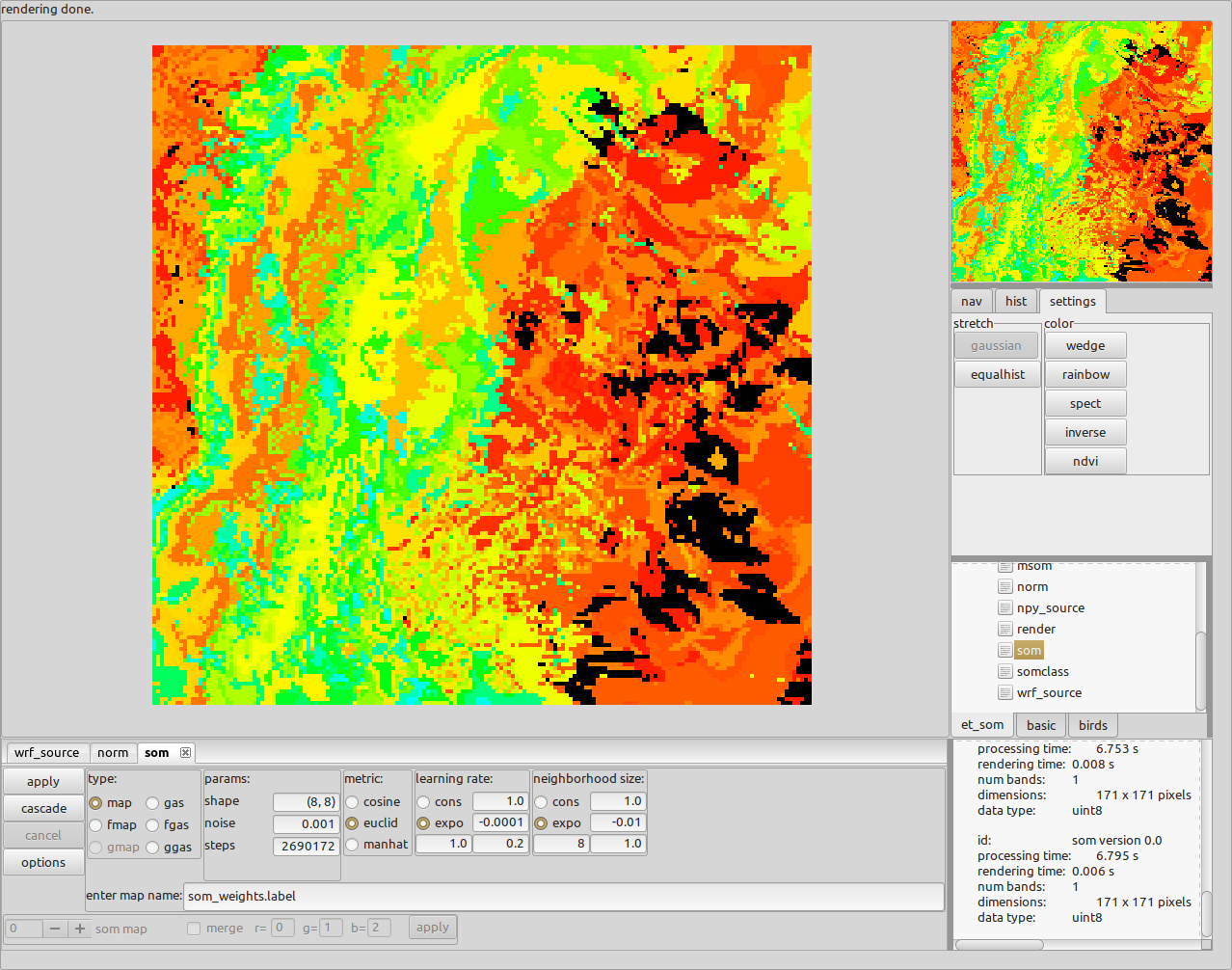
TSK:11,EMISS:11,SWDOWN:11,GLW:11,GRDFLX:11,T2:11,PSFC:11,Q2:11,U10:11,V10:11

To specify the data file, drag and drop from a file browser into the filepath text panel or use the “browse” button to find the file. The above image shows the surface pressure for 2019/01/01 over Ecuador for time slice 11. As before we wish to normalize the data so we double click on the “norm” operator but this time we want to save the normalization coefficients for later use, so we click on the “coeffs to file” check box. Press the “apply” button to normalize.

It is important to note that for data values bigger than eight bits, the rendered pixels are stretched to 0-255 range making the data look as if it were normalized when it is not. This can be seen by comparing the above image with the normalized version below.

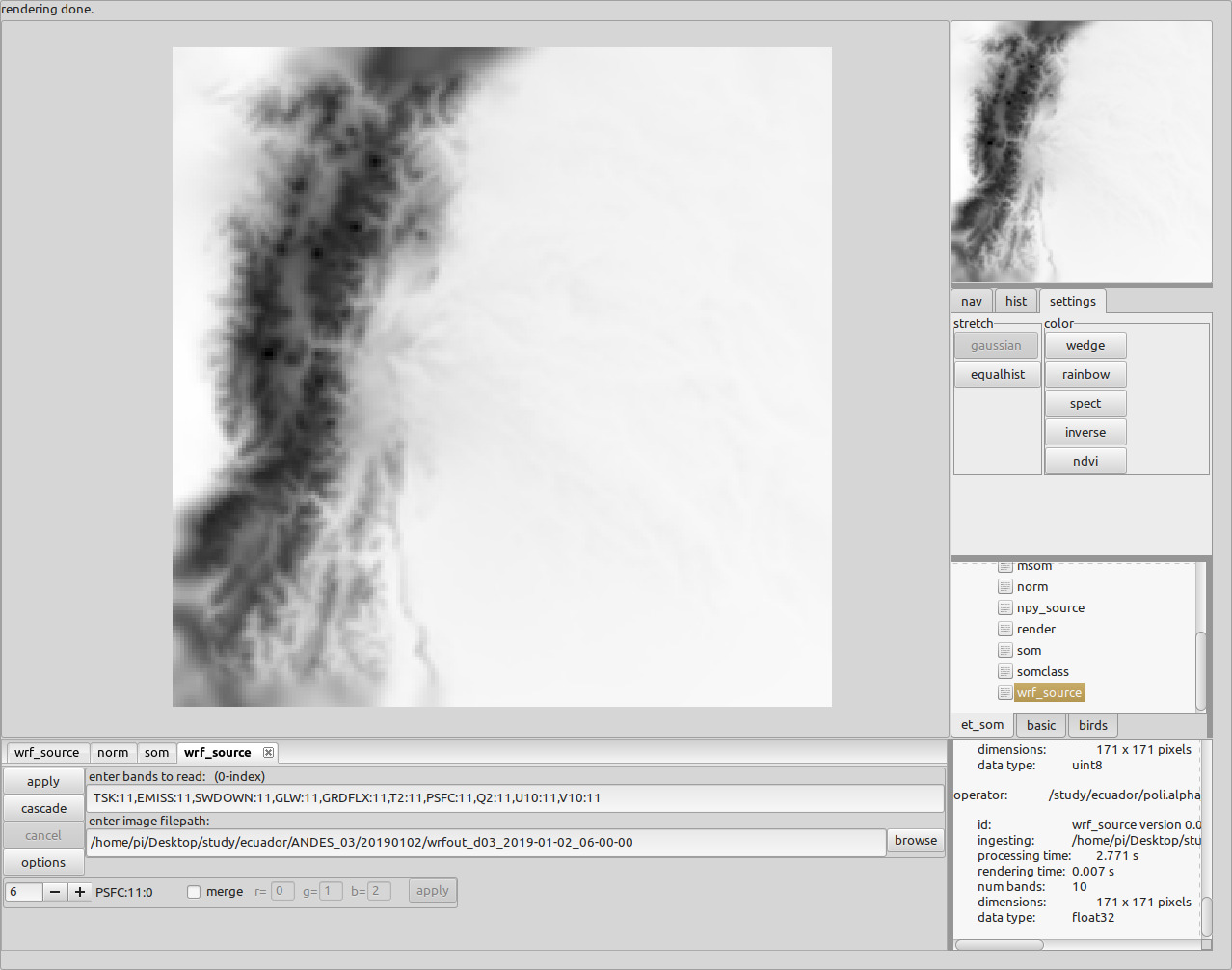


We now train on this very limited data set using the SOM operator “som”. Go to the operator tree and double click on the “som” operator. The “som” parameters are different than the “msom” parameters and again due diligence is required as to how to use these. The most notable difference is that “som” operator can use a three-dimensional net grid as opposed to “msom” which can only use two-dimensional grid. For example, you can specify a 4x4x4 grid as (4,4,4) in the “shape” text panel.



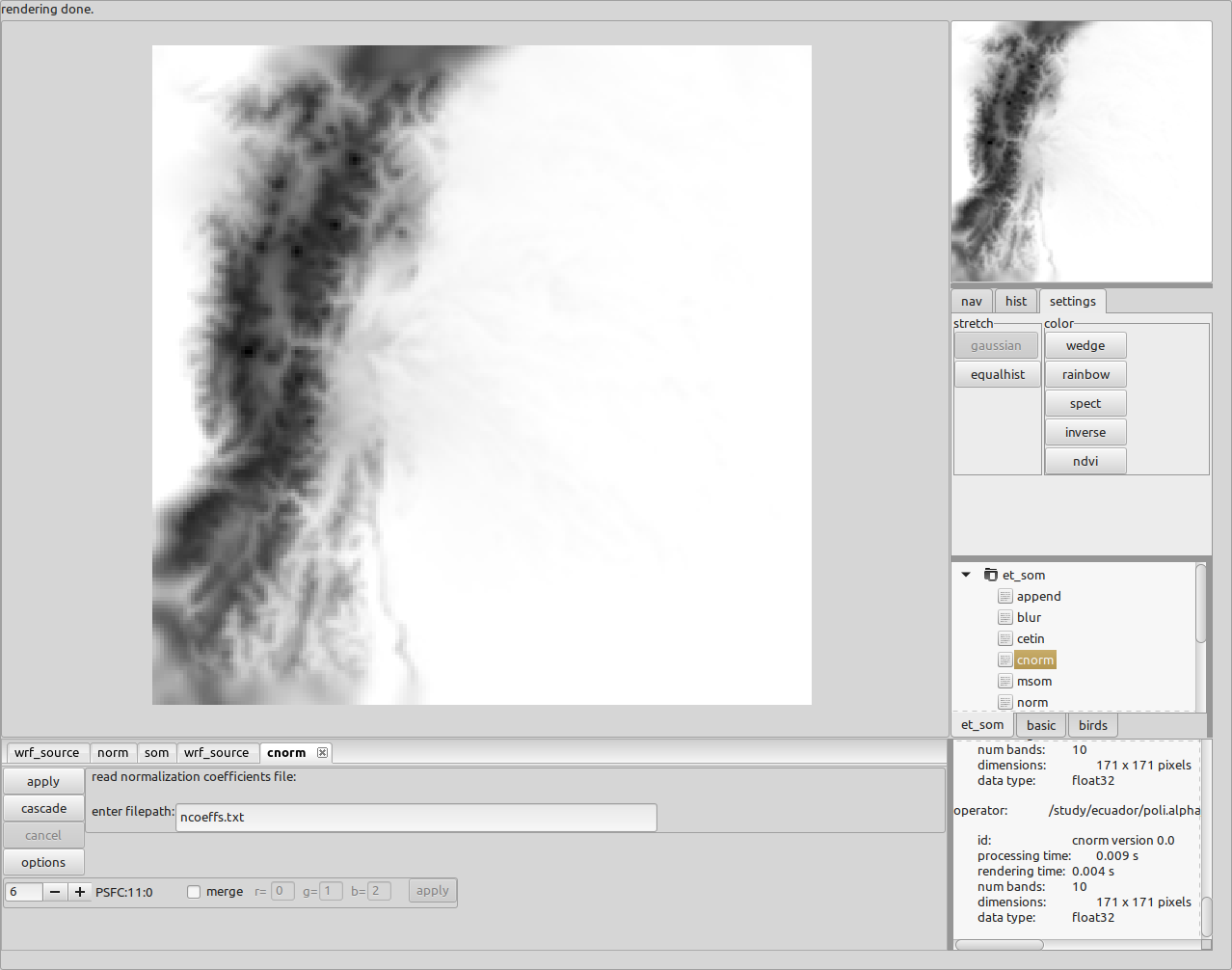
The image above shows a clustering of 64 classes and has written out the weights in the file “som\_weights.labels”. As a very shallow example, we now apply these weights to another data set, say the next day. Instead of starting fresh, we can use “wrf\_source” again and start a new stream. In this way, crude image comparison can be made by selecting the operator tabs back and forth.

With the “som” operator displayed, double click on the “wrf\_source” operator. Copy and paste the string that specifies the variables/time into the “bands to read” text panel. Drag and drop the next day’s data set as shown below.

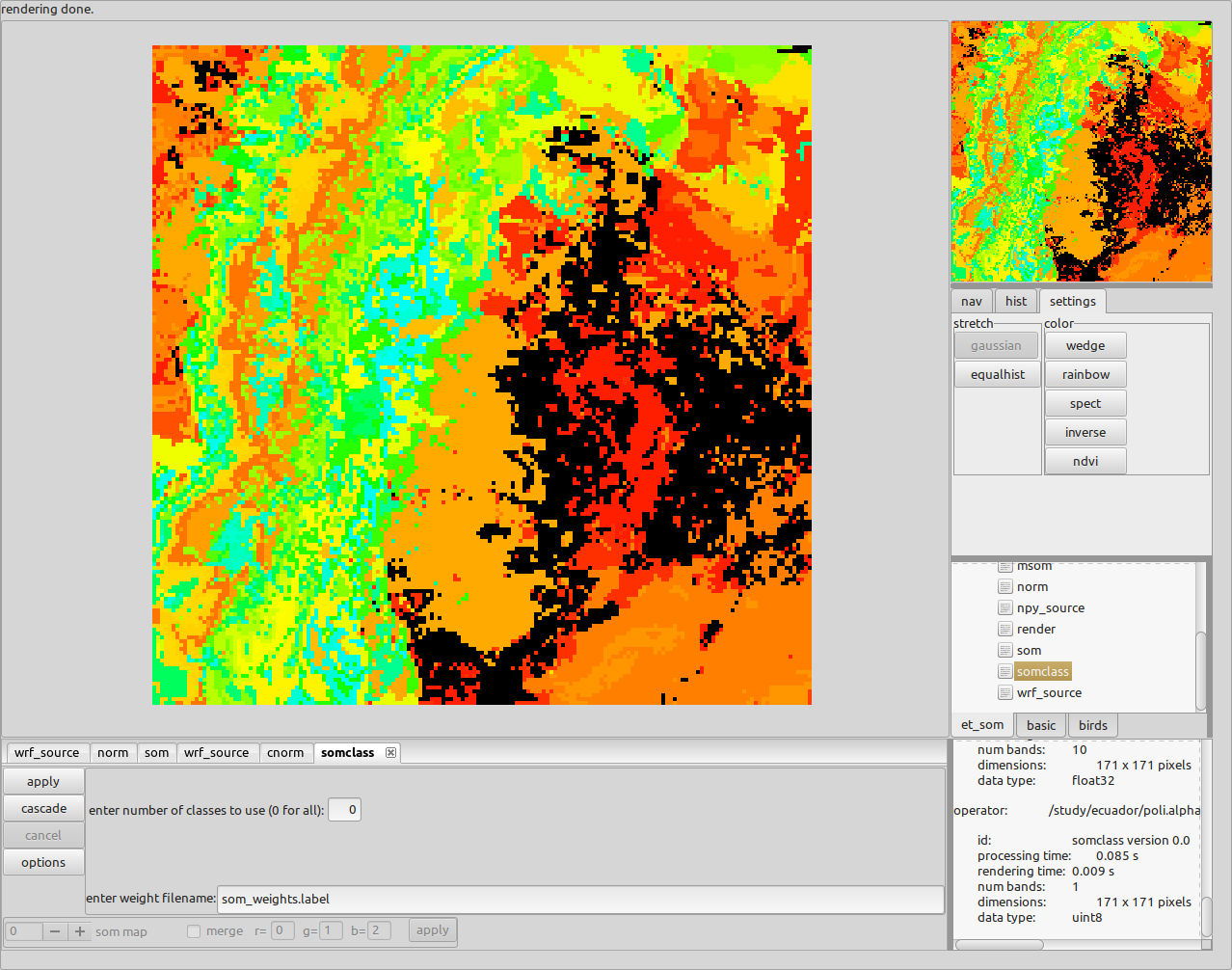


Be aware that the 0th buffer (variable) will be displayed, while the above image shows the 6th buffer.

Now we normalize the data as before, but this time we need to use the same normalization coefficients that were used in the training, which were saved to the file “ncoeffs.txt” by the operator “norm”. Instead of the operator “norm”, double click on the operator “cnorm” and specify the “ncoeffs.txt” in the filepath text panel. Press the “apply” button.

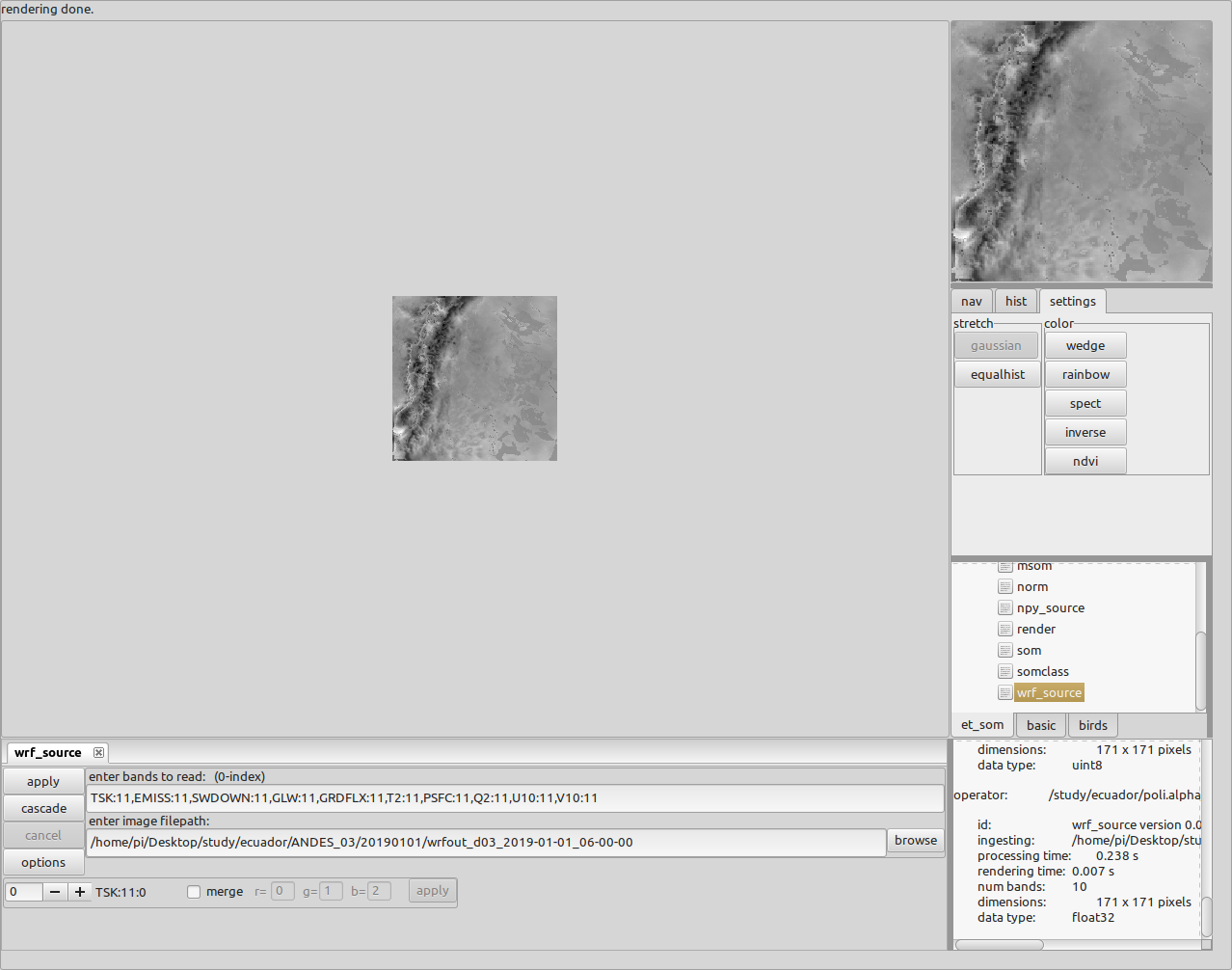


The operator “som\_class” accepts weights from either “msom” or “som” operators and applies those weights to the target data set, in our case data from 2019/01/02. Double click on “som\_class” in the operator tree to apply the weights. Both SOM operators rank the classes on the number of pixels, so it is possible to specify some number of top classes to use. Set the number of classes to use to 0 to use all classes and specify the weight file to use (“som\_weights.label” in our case). Press “apply” button.

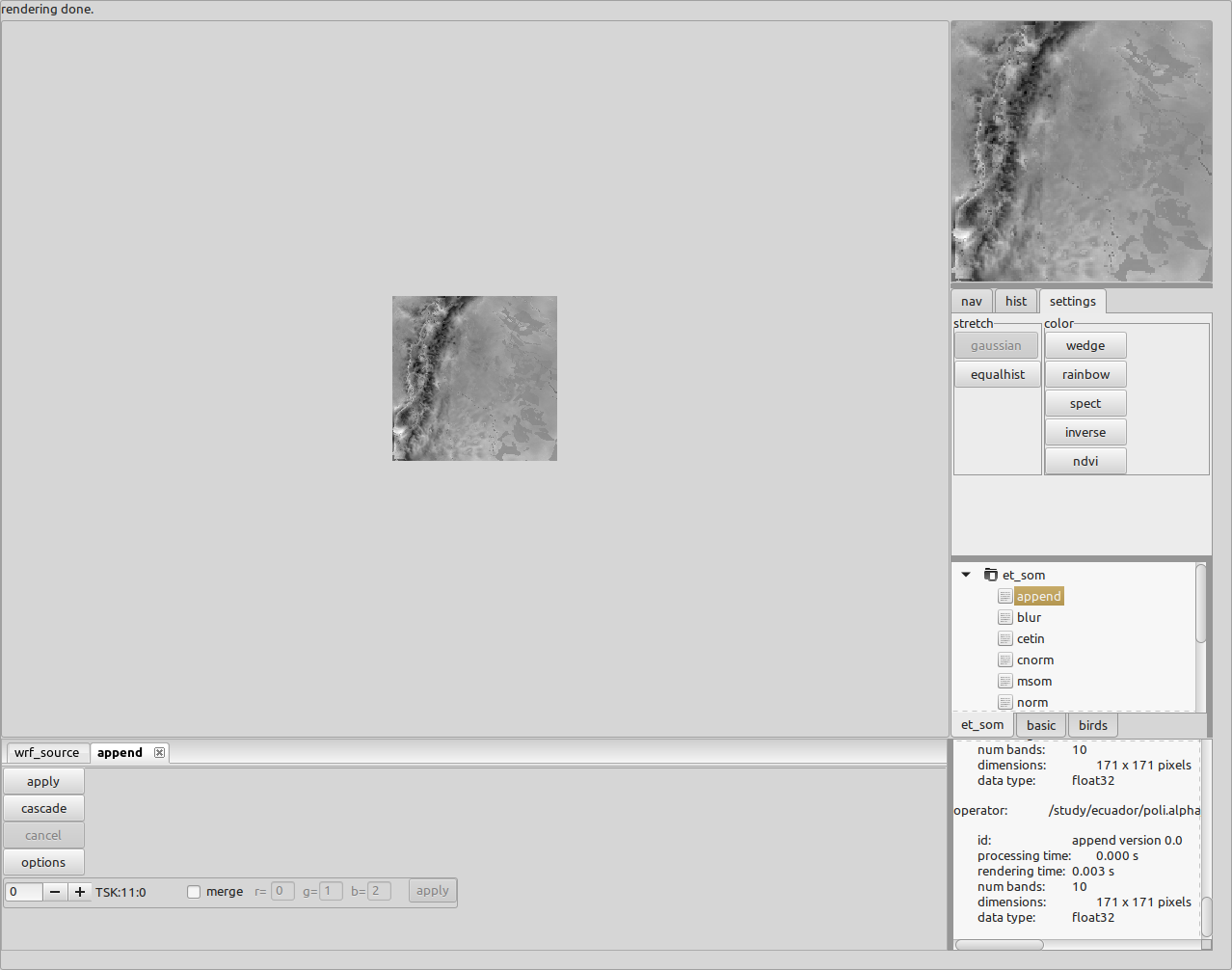


So far, this exercise has only used one data set to train on. In order to train on more daily data sets we append the data together and then implement a SOM operator. We start as before with the “wrf\_source” operator but now implement the “append” operator and iterate up to some number. The following images will demonstrate the iteration.

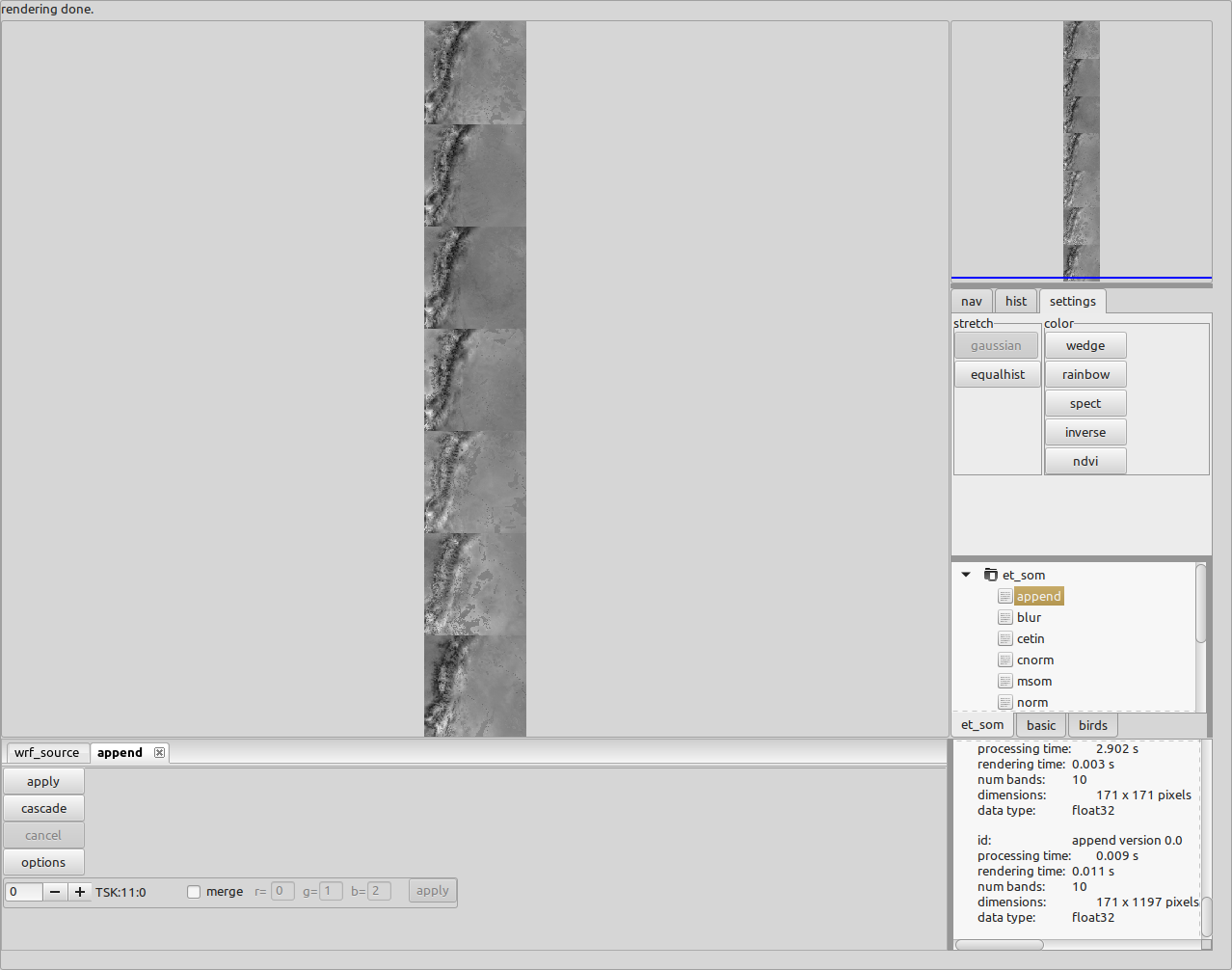
Start with first data set:



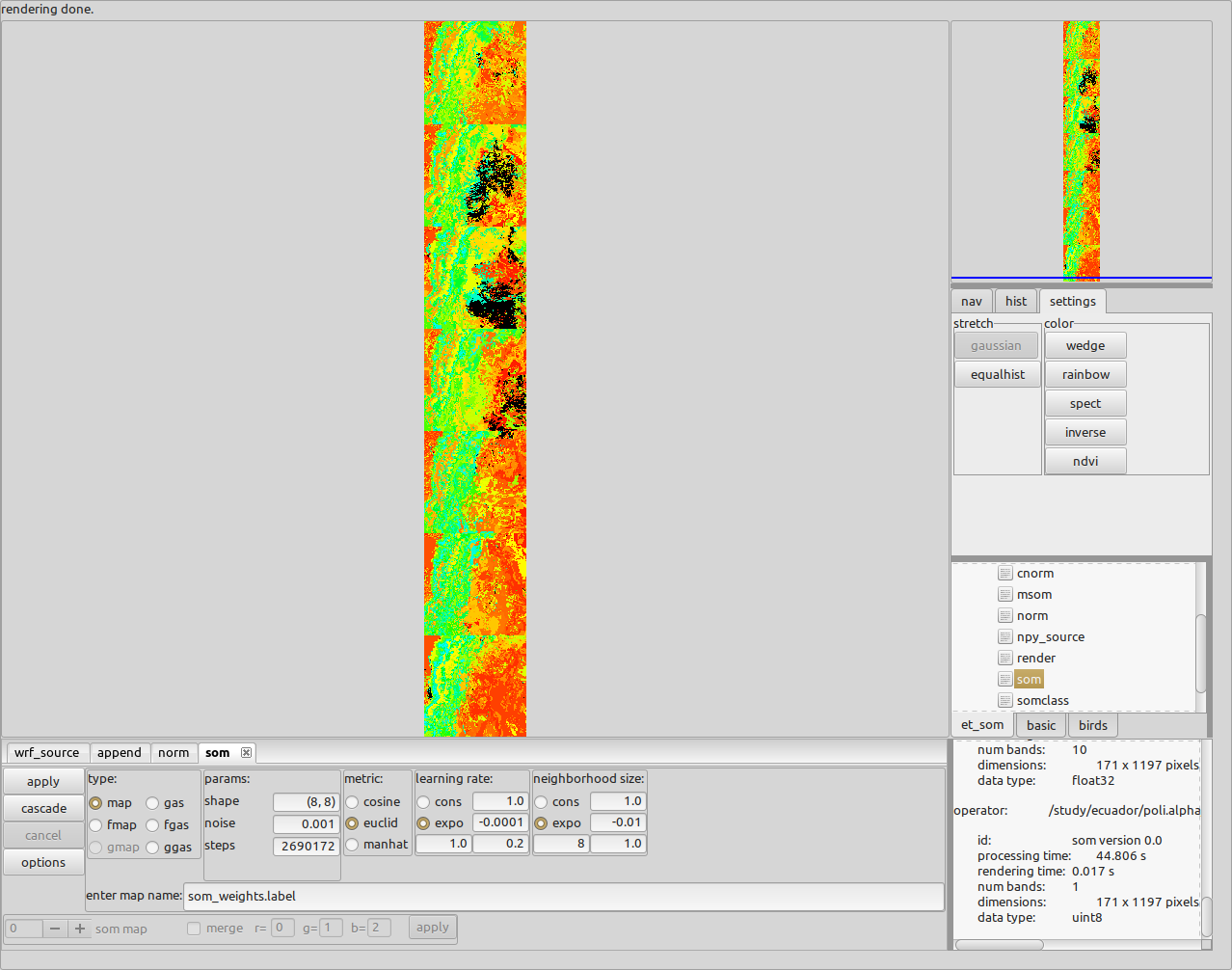
Next double click on “append” from the operator tree and press “apply”



Iterate now by going back to “wrf\_source” and drag/drop the next days data set. Go to the “append” operator and press “apply”. Repeat up to some number, say 7 times.



We proceed as before with normalization and training.



Application of these class weights to other data sets is done as before.

NOTES:

As can be imagined from the above append iteration, large scale implementation is cumbersome for the POLI/GUI application. In order to train over months or years requires POLI/BATCH implementation. You could then use the weights in POLI/GUI to look at target data sets.

The above example took just one time slice out of the daily data set. This shallow implementation of considering only one hour can be expanded to 24 hours by extending the feature space on an hourly basis. That is, the feature space now is a concatenation of hourly variables, reflecting a daily state instead of one hour. This is also best done in a POLI/BATCH implementation.

The variables used above are the raw variables used to calculate the actual variables used in the ETo equation. Our POLI/BATCH scripts do the conversion and then train and apply on actual variables.

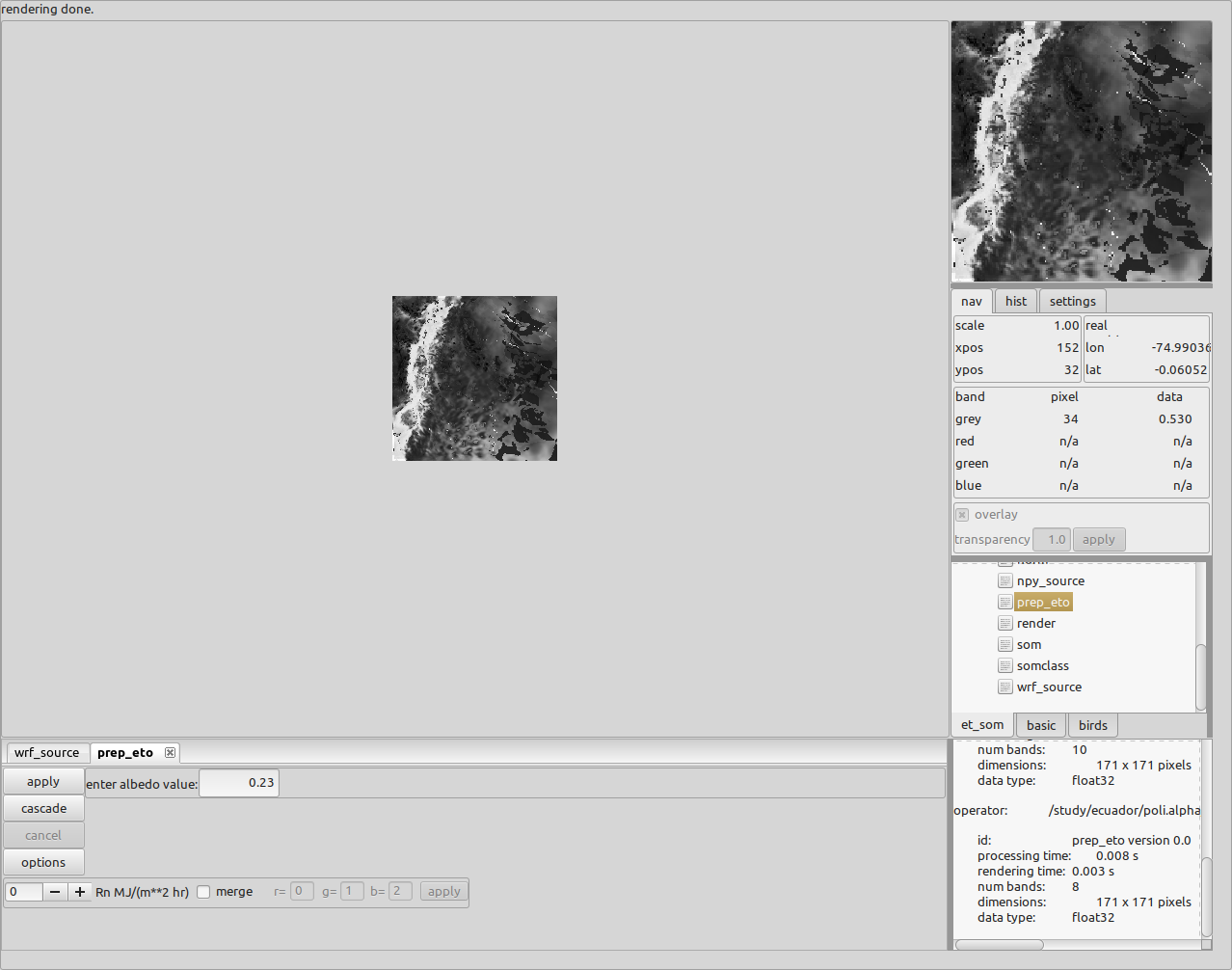
Actual variable are presented in our paper.

Use the operator “prep\_eto” to convert the raw variables to actual variables in the GUI as an example. As before use, “wrf\_source” with the same band string:

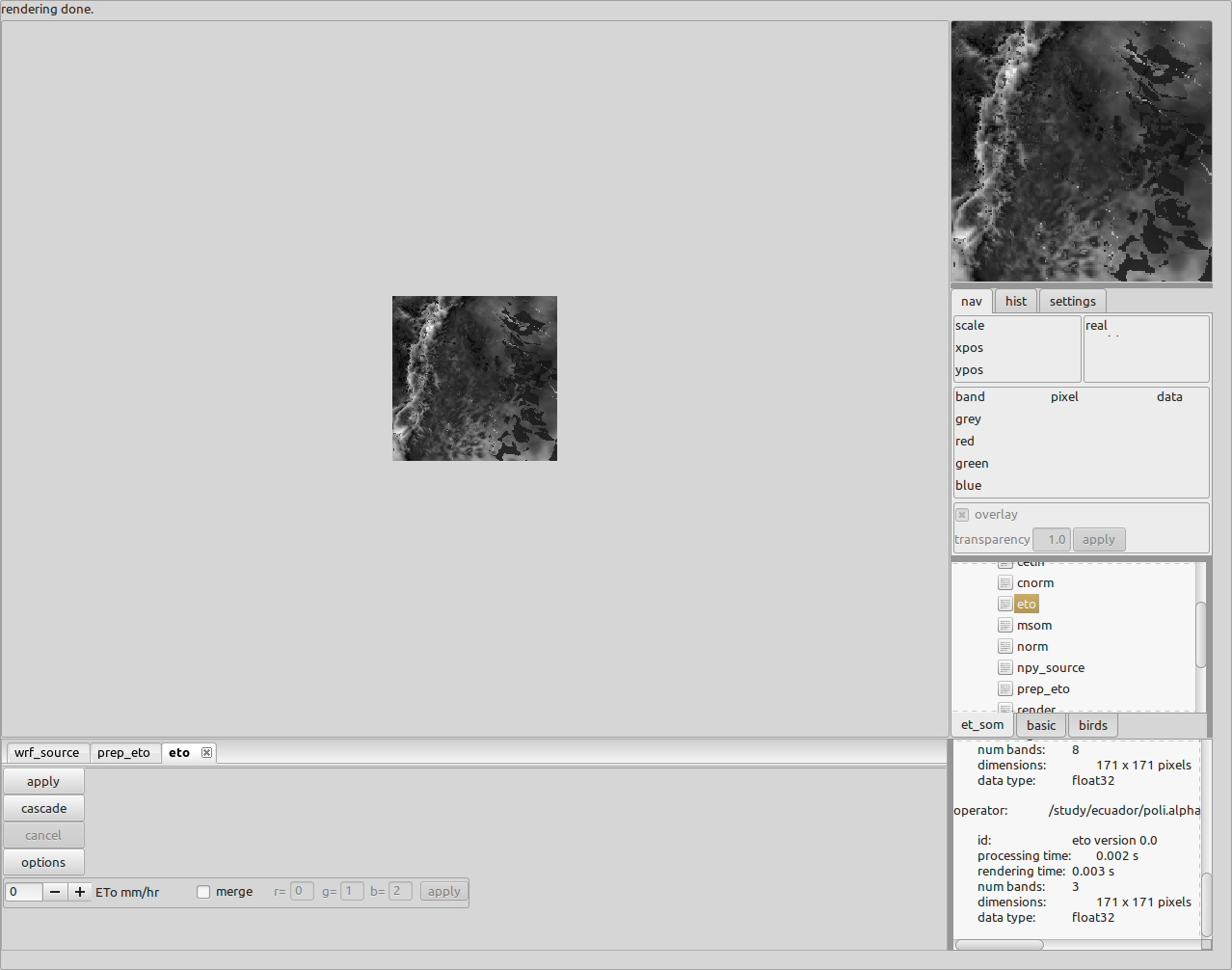
TSK:11,EMISS:11,SWDOWN:11,GLW:11,GRDFLX:11,T2:11,PSFC:11,Q2:11,U10:11,V10:11

Drag and drop a WRF output file into the filepath text panel. Press apply. Go to operator tree and double click on the “prep\_eto” operator. Press apply. The units of the buffers are indicated by the arrow.

Scroll through buffers with the “-” and “+” buttons.



To calculate the ETo use the “eto” operator. This will render ETo values for the instantaneous values given in the WRF output file.



However, the ETo equation in the FAO paper uses an hourly average. In order to obtain an hourly average, specify two hourly band slices in “wrf\_source” operator by using a band string like :

TSK:10,EMISS:10,SWDOWN:10,GLW:10,GRDFLX:10,T2:10,PSFC:10,Q2:10,U10:10,V10:10,

TSK:11,EMISS:11,SWDOWN:11,GLW:11,GRDFLX:11,T2:11,PSFC:11,Q2:11,U10:11,V10:11

This string is asking for the raw variables at two time slices., 10 and 11. The operator “prep\_eto” sees the extra variables and will prepare both time slices and then takes the average of the actual variables.

The script “collect\_data.py” shows how, for each daily WRF datafile, to get the two hourly slices throughout the day, convert from raw to actual (prep\_eto) and concatenate the hourly variables to construct a feature space that encapsulates the entire day, appends each day and then normalizes across same variables in different time slices, for presentation to the som as training data.

The scripts “train\_minisom.py” or “train\_som.py” then cluster the feature space and provides the labels (classes) with neural weights to be used by “apply\_class.py” on target data sets (other WRF output files). The “apply\_class.py” scripts provides a visualization for humans, and a numpy file, of the classes for each day of the target data sets. Finally, the produced numpy file is used by the script “cluster\_labels\_msom.py” or “cluster\_labels\_som.py” to cluster pixels with similar class migrations, rendering the final product of pixels with similar ETo behavior.