Supplemental Material

Temporally Generalizable Land Cover Classification: A Re-current Convolutional Neural Network Unveils Major Coastal Change through Time

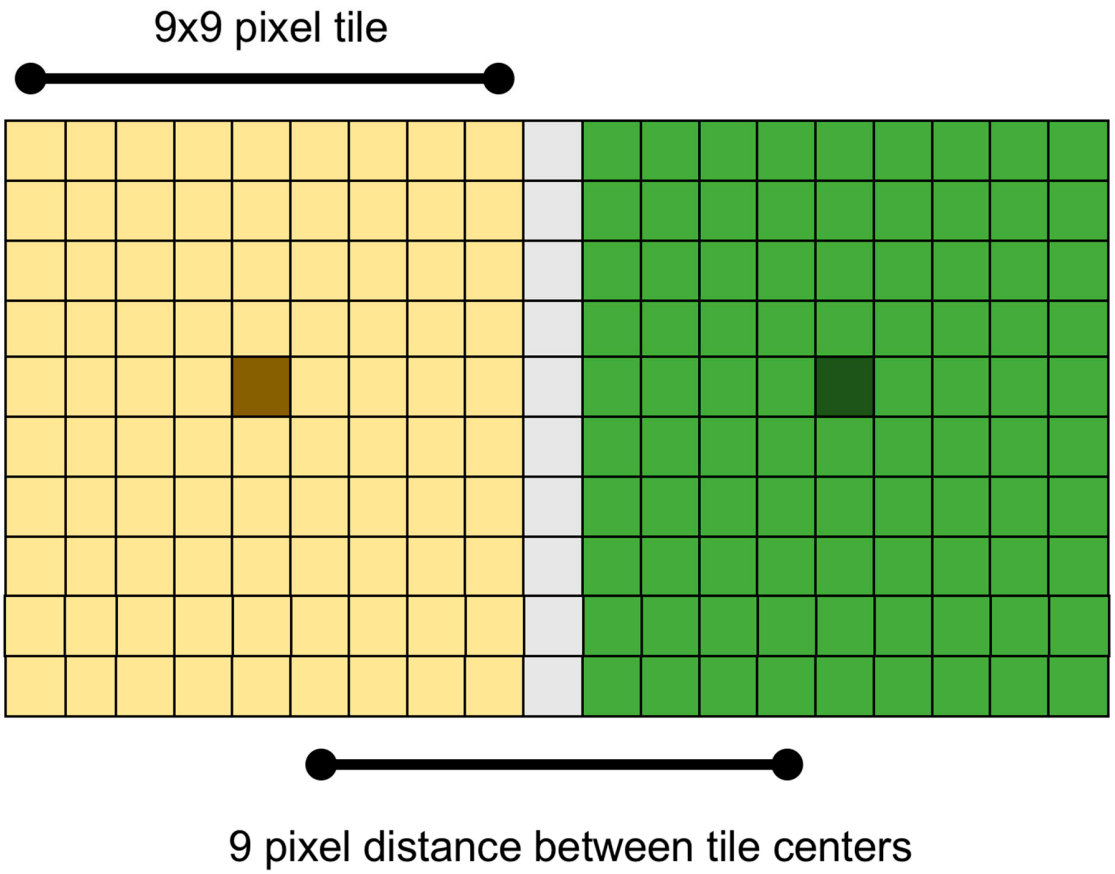

Figure S1. During the general of model testing, validation, and training data all testing, and validation patches was required to be 9 pixels away from other patch centers. Given that each patch was 9x9 this led to a margin of at least 1 pixel. This was enforced between testing, validation, and training patches, and within testing and validation patches. Training data patches were allowed to overlap with other training patches. In this figure the dark yellow (left) is the center pixel of a 9x9 patch and the dark green (right) is the center pixel of a different 9x9 patch. The grey in the middle is the minimum margin between patches.

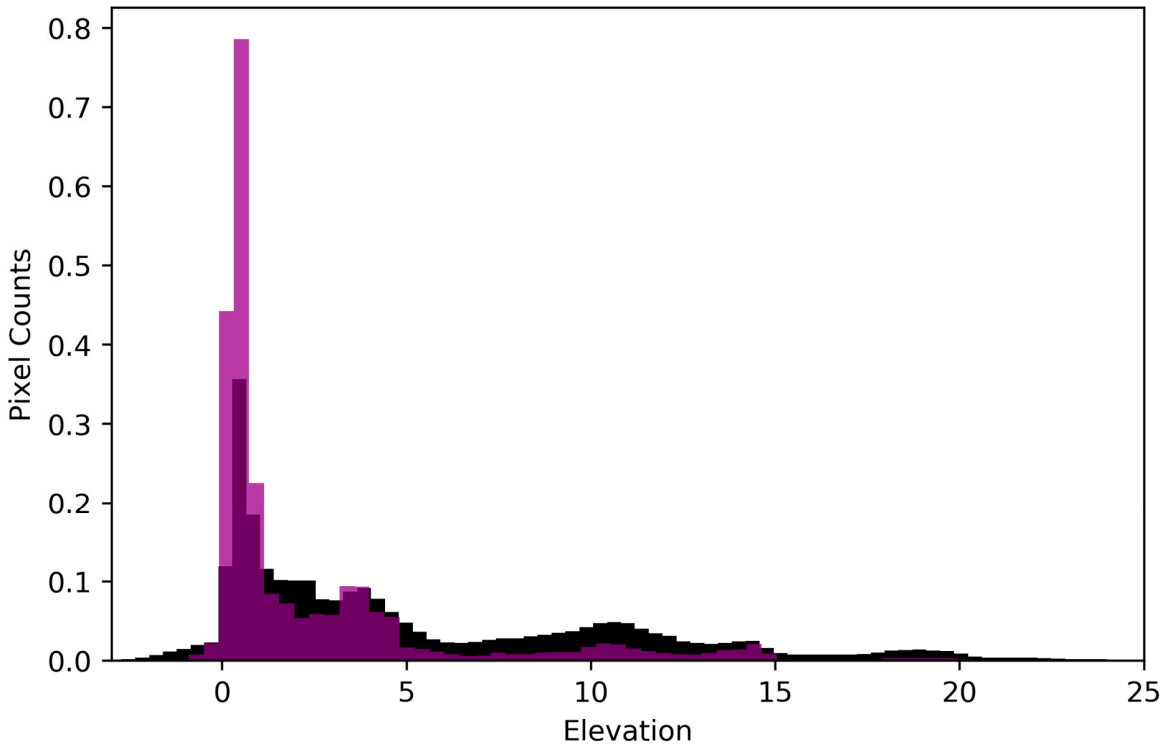

Figure S2. Distribution of the elevations of all pixels (black) and pixels that were forest or agriculture in 1989 and were classified as wetland in 2011 (purple). Pixels that shifted to wetland had a much higher likelihood to be at lower elevation, particularly under 1 meter.

General Notes on Model Development

We find that complex software engineering projects like this can be extremely challenging for non-engineers and even though our team was largely composed of engineers it was still a long onboarding process. Given the size of the code base needed for data processing, visualization, and model development we strongly recommend all of the best practices provided here: https://journals.plos.org/plosbiology/article?id=10.1371/journal.pbio.1001745. And particularly in remote sensing, we found that constant visualization and unit testing is critical given the dimensionality of the data and the complex processing it must undergo in order to successfully be ingested into a model.

We found many times throughout this project that minor changes to the hyperparameters were less impactful that increasing the quality of data going into training the model. While not a novel finding we want to emphasize this for practitioners who need operational models that investment in quality data is likely to pay more dividends that time fine-tuning and already good model.

Table S1. Architecture for recurrent neural network as output as printed from the

`model.summary()` method in keras showing the layer name, the shape of that tensor, and the

number of trainable parameters in each layer. The tile_input layer shows the input of 5 time steps by 7 image bands. The output from the final recurrent layers is run through a dense layer, and output into the six final classes.

```
Layer (type)                 Output Shape              Param #   Connected to
==================================================================================
rnn_input (InputLayer)       (None, 5, 7)              0
_____________________________________________________________________________
lstm_1 (LSTM)                (None, 5, 12)             960       rnn_input[0][0]
_____________________________________________________________________________
concatenate_1 (Concatenate)  (None, 5, 19)             0         rnn_input[0][0]
                                                                 lstm_1[0][0]
_____________________________________________________________________________
lstm_2 (LSTM)                (None, 5, 12)             1536      concatenate_1[0][0]
_____________________________________________________________________________
concatenate_2 (Concatenate)  (None, 5, 31)             0         rnn_input[0][0]
                                                                 lstm_1[0][0]
                                                                 lstm_2[0][0]
_____________________________________________________________________________
lstm_3 (LSTM)                (None, 5, 12)             2112      concatenate_2[0][0]
_____________________________________________________________________________
concatenate_3 (Concatenate)  (None, 5, 43)             0         rnn_input[0][0]
                                                                 lstm_1[0][0]
                                                                 lstm_2[0][0]
                                                                 lstm_3[0][0]
_____________________________________________________________________________
lstm_4 (LSTM)                (None, 5, 12)             2688      concatenate_3[0][0]
_____________________________________________________________________________
concatenate_4 (Concatenate)  (None, 5, 55)             0         rnn_input[0][0]
                                                                 lstm_1[0][0]
                                                                 lstm_2[0][0]
```

			lstm_3[0][0]
			lstm_4[0][0]
lstm_5 (LSTM)	(None, 5, 12)	3264	concatenate_4[0][0]
concatenate_5 (Concatenate)	(None, 5, 67)	0	rnn_input[0][0]
			lstm_1[0][0]
			lstm_2[0][0]
			lstm_3[0][0]
			lstm_4[0][0]
			lstm_5[0][0]
lstm_6 (LSTM)	(None, 5, 20)	7040	concatenate_5[0][0]
concatenate_6 (Concatenate)	(None, 5, 87)	0	concatenate_5[0][0]
			lstm_6[0][0]
lstm_7 (LSTM)	(None, 5, 20)	8640	concatenate_6[0][0]
concatenate_7 (Concatenate)	(None, 5, 107)	0	concatenate_5[0][0]
			lstm_6[0][0]
			lstm_7[0][0]
lstm_8 (LSTM)	(None, 5, 20)	10240	concatenate_7[0][0]
concatenate_8 (Concatenate)	(None, 5, 127)	0	concatenate_5[0][0]
			lstm_6[0][0]
			lstm_7[0][0]
			lstm_8[0][0]
lstm_9 (LSTM)	(None, 64)	49152	concatenate_8[0][0]

```
_________________________________________________________________________________________________
dense_1 (Dense)                 (None, 64)           4160        lstm_9[0][0]
_________________________________________________________________________________________________
landcover (Dense)               (None, 6)            390         dense_1[0][0]
=================================================================================================
Total params: 90,182
Trainable params: 90,182
Non-trainable params: 0
```