Using Machine Learning to Model Global Terrestrial Carbon Flux

Donovan Murphy[1,2], Brianna Bonner[1], Taimoor Arshad[1], Grey Nearing[1,2], Craig Pelissier[2]

[1] The University of Alabama, [2] NASA GSFC

A. Groundwork

This project evaluates the use of machine learning to meet the challenges of **scalability and continuity** in carbon flux monitoring and prediction. Today, the measurement of terrestrial gas fluxes is limited in spatial representation, evidenced by the over eight hundred sites capturing carbon information in a globally distributed yet patchy network, FLUXNET. Using the **eddy covariance** method, these sites compute **net ecosystem exchange** (NEE), a measure of the vectoring of carbon dioxide through their ecosystems. Though these sites deliver high quality, *in situ* data, they are inherently restricted to hyper-local observation, limiting accurate inferences toward a finer spatial resolution. Additionally, many ecoregions lack FLUXNET site representation, including the Pacific coast of South America, Central Asia and the Middle East, and the vegetation-rich archipelagos of the Sunda Shelf (see Fig. 1). These absences are based on sites' geographic distribution and independent of ecosystem type.

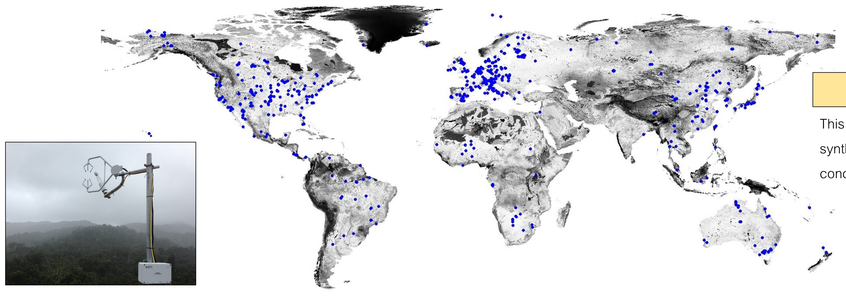


Fig. 1: FLUXNET sites. Darker areas indicate lower representation in the network.
Inset: Anemometer on site in Denmark.

Eddy Covariance

This method estimates gas exchange rates by synthesizing windspeed and trace gas concentrations.

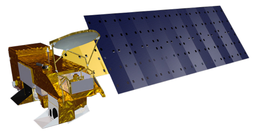
B. Method

Iterating from the sparse data at FLUXNET sites, our team has taken a nonparametric approach to infer carbon flux at the global scale. This method allows for data gathering at a granularity unavailable to FLUXNET analysis alone.

The following summarizes this project's workflow and objectives:

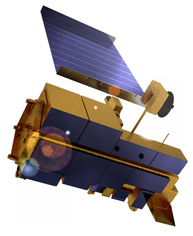
1. **Assemble** a training/testing data product of *in situ* FLUXNET measurements between 2007 and 2014;
2. **Assemble** a training/testing data product integrating net ecosystem exchange with remotely sensed data of related variables covering 2007 through 2014;
3. **Train and evaluate** four machine learning algorithms and protocols using both data products:
   a. Find the best performer(s) at predicting NEE;
   b. Assess model sensitivities to specific input variables;
4. **Determine the significance of differences in predictive power** by passing the data products from (1) and (2) through machine learning algorithms to understand if these techniques are **scalable-to-globe***;
5. **Generate global maps** of terrestrial carbon flux over the tested time period.*

**In progress*



C. Data Products

1. *In Situ Training Matrix* ("*In Situ*")
   - 8 years of daily observations
   - 130 FLUXNET sites contributing all variables
   - 10 state variables and net ecosystem exchange (NEE)



2. *Remote Sensing Training Matrix* ("RS")
   - 8 years of observations at 8-day temporal resolution
   - 159 FLUXNET sites contributing NEE
   - 7 remotely-sensed variables at 0.5-degree spatial resolution
     - Reconstructed solar-induced fluorescence (RSIF);
     - Normalized Difference Vegetation Index (NDVI);
     - Day and Night Land Surface Temperature (LST-Day, LST-Night);
     - Air Temperature (AT);
     - Surface Air Pressure (SurfPres);
     - Cloud Fraction (CloudFrc);

NDVI, LST, AT, SurfPres, and CloudFrc courtesy MODIS and AIRS on EOS Terra (above right) and Aqua (left) provided by NASA GES DISC and NASA/USGS LPDAAC. RSIF courtesy Pierre Gentine (Columbia University). This project uses MATLAB as the primary tool for data preparation, analysis, and visualization.

D. Algorithm Development

Artificial neural networks (ANN)

At each epoch in the time series, training data are presented to a hidden layer of **weighted "neurons"**, an internal collection of parameterized processing nodes that adjust in selective power throughout training. As the data are processed, relationships develop between local or remotely-sensed state variables and net ecosystem exchange. As training time progresses, **neurons become better or worse predictors of NEE and are selected accordingly**, having "learned" the relative associations among the variables. The tuning of these neurons results in a computational network able to consistently and accurately predict NEE given any modulation of the input variables used at training.

Gaussian process regressions (GPR)

Given an input of training data, Gaussian (normal) distributions are fitted between every pair of states (i.e., all variable data for one observation and the next). The means of these distributions become the interpolants between the states. At each epoch of training, new Gaussian means are found between each subsequent pair of states, resulting in a **smooth predictive model encompassing all possible states** that, when provided with data lacking NEE, will fit a NEE value based on its associated variables.

Random forests and tree-bagging (TBG)

This class of algorithm extends the concept of decision trees, where training data are evaluated against parameterized rules that govern its descent through a "tree" of possible associations, resulting in NEE output that matches to the specific distribution of input variables. In a random forest, **subsets of training data are selected at random** and "bagged" or **aggregated**. Decision trees are then fitted to these subsets. At each classification node (i.e., branch) throughout this training, features are selected to continue at random, ending with an ensemble of decision-makers **more capable at accounting for input variance** and therefore of more accurate NEE prediction.

Recurrent neural networks and LSTM (RNN)

A big-data extension of ANN, an RNN uses neural network architecture as outlined above with two significant differences. First, once input data pass through the hidden layer(s) of neurons, a portion of this processed data joins the next epoch's input and **re-enters the network (recurrence)**, super-tuning the neurons toward faster, more accurate prediction. Second, RNNs are capable of routing input through gated state storage of accessible memory. **Long short-term memory** (LSTM) is one instance of this capability. With an LSTM implementation, states are preserved throughout training, allowing associations made at previous epochs to be available or "**remembered**" at the present one.

*(NB: Before running each algorithm, data structures are divided into **training** and **test sets** randomly or by selection, dependent on the algorithm; pseudocode is not provided to benefit readership and inclusivity.)*

E. Validation and Testing

We applied two validation protocols to evaluate each algorithm's performance as NEE predictors. 11 statistics are calculated as models are tested at each site. Full-record statistics are generated at completion as holistic criteria for the learning method itself.

K-fold (local, site-specific)

In this approach, the algorithm progresses through the data structure on a site-by-site basis and data partitioning is governed by the time dimension (i.e., observations). Prior to training, the number of validation partitions is parameterized ("5" was chosen as this k-value). For each site and at each epoch, five observations are selected; the algorithm trains on four and is evaluated on the fifth.

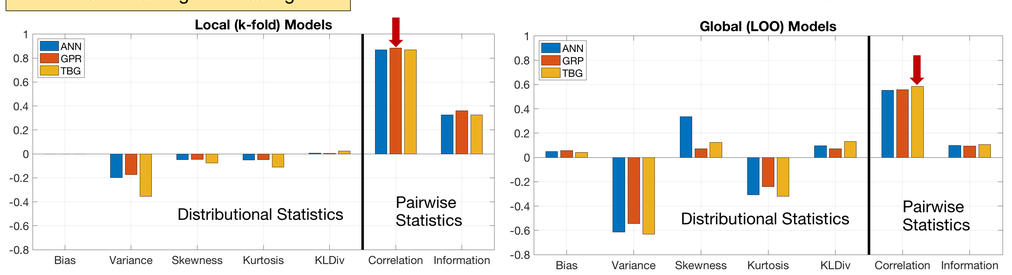
"Leave-one-out" or LOO (global, cross-site)

As the algorithm increments through the sites, training data are gathered at all sites save the one at the current increment. For example, if the neural network is working at the 13th site, input data for sites 1-12 and 14-159 will be used to train, with the data at site 13 reserved for testing.

Sensitivity Analyses

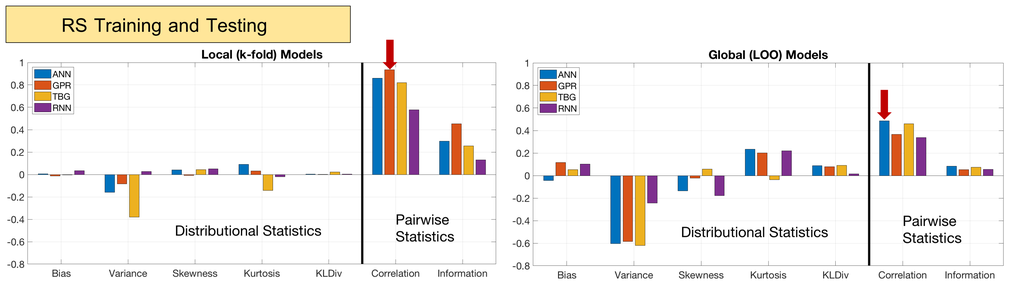
An auxiliary objective in this project was to determine which input variables contribute most to model outcome to optimize data selection. The algorithms were trained as above and tested with k-fold validation, but with output statistics accounting for the relative weighting of the input classes (i.e., variable columns in the data structure) throughout the process.

In Situ Training and Testing



Figs. 2-3: Full-record, out-of-sample test statistics for *In Situ* Training Matrix. In this context, "Correlation" provides a best estimate of performance.

RS Training and Testing

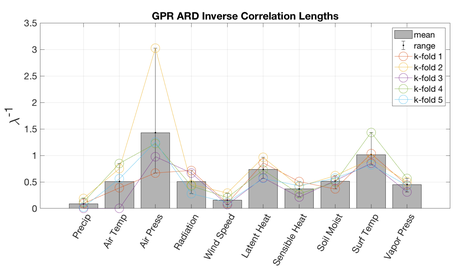


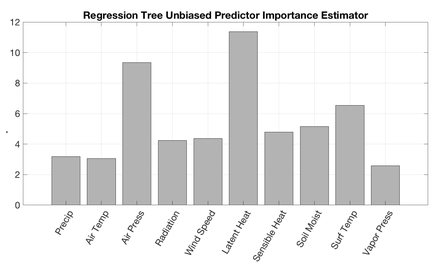
Figs. 4-5: Full-record, out-of-sample test statistics for RS Training Matrix. In this context, "Correlation" provides a best estimate of performance.

Performance Results

After analysis of first-attempt test statistics, a specific algorithm class was not found to be superior across all validation and testing methods, though **ANN was found to be most consistent**. Notable findings include the following:

1. In both k-fold validations, **all algorithms performed well** as indicated by **high correlation values and low variance.**
2. The result most applicable to the scalability objective of this project occurs in Fig. 5. **ANN performed best with RS input with a correlation coefficient of 0.50**, an acceptable preliminary value.
3. LOO models underestimate variance. This is due to the fact that **global models fail to capture complete variability across sites**, a fault that can be mitigated by more rigorous site classification schemes (e.g., IGBP land cover grouping), additional remote sensing input layers, and including a higher number of monitoring sites contributing NEE during training.





Figs. 6-7: Example sensitivity analyses for *In Situ* state variables in standalone GPR and TBG models.

Input Sensitivity Results

1. For *In Situ* variable analyses, **air pressure, latent heat, and surface temperature** contributed most to prediction and were weighted most heavily.
2. ANN displays the highest uniformity of response, with **RSIF, LST-Day, and air temperature the highest contributors.**
3. Sensitivity analyses between *In Situ* and RS agree, as many high responders match (RSIF is a proxy measure for latent heat).

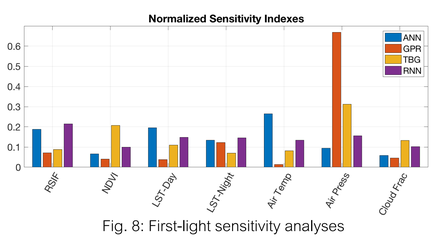


Fig. 8: First-light sensitivity analyses for RS variables in all models.

Contact

● Donovan Murphy | dlmurphy5@crimson.ua.edu

Acknowledgements

Our team thanks the following for their support: Dr. Daniel Duffy and the NASA Center for Climate Simulation at GSFC, Dr. Milton Halem (University of Maryland, Baltimore County), Pierre Gentine (Columbia University), Dr. Stefan Kern and Remon Sadikni (Integrated Climate Data Center, University of Hamburg), Fluxdata, The MathWorks, Inc., Department of Geological Sciences at The University of Alabama.

Contact

● Donovan Murphy | dlmurphy5@crimson.ua.edu