

# A Framework for Applying Neural Networks to Eddy Covariance Data

Dr. June Skeeter

june.skeeter@ubc.ca

University British Columbia

# Eddy Covariance

Semi-continuous, ecosystem-scale measurements of energy, water, and trace gas fluxes.

- Noisy, voluminous data sets
  - Frequent gaps
  - Observational bias
- Well suited for machine learning!



**Burns Bog EC Station  
Delta, BC**

# Neural Networks

**Universal approximators:** can map any continuous function to an arbitrary degree accuracy.

- With enough hidden nodes, will fit **any** pattern in a dataset
  - Care must be taken to ensure the patterns are real
  - Early stopping allows
- Well suited for non-linear, multi-variate response functions
  - Capable **interpolation** and **extrapolation**

# Commonly Cited Limitations

Issue	Solutions
Over-fitting	<ul style="list-style-type: none"><li>- Model ensembles</li><li>- 3-way cross validation</li><li>- Pruning inputs</li></ul>
Black boxes models	<ul style="list-style-type: none"><li>- Plot partial derivatives</li><li>- Feature importance</li></ul>
Computationally expensive	<ul style="list-style-type: none"><li>- Tensorflow</li><li>- GPU processing</li></ul>

# Objective

Provide a framework for applying NN models to EC data for descriptive analysis and inferential modelling.

- The [github repository](#) linked to this presentation has functional examples that can be used to apply NN models.
  - Runs in Python and Tensorflow
    - *GPU support not required*

# Example Data

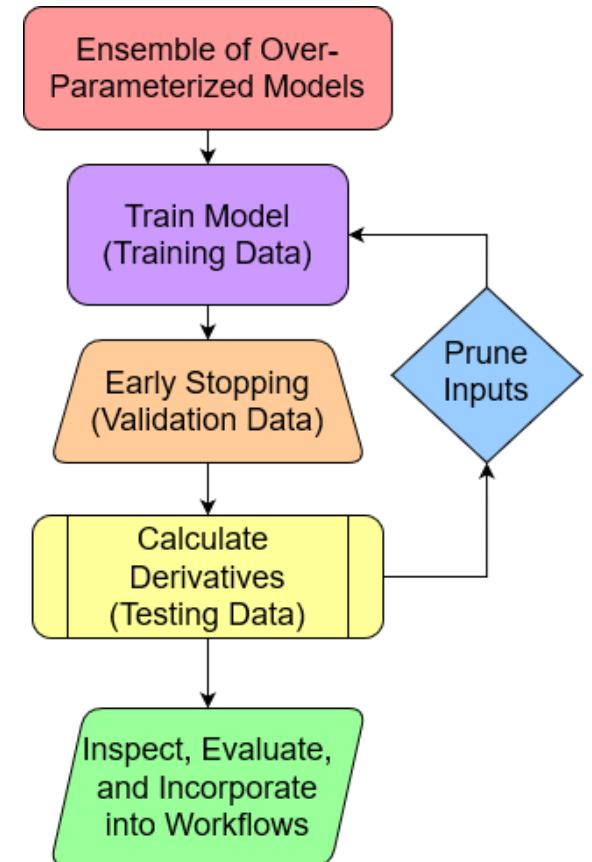
Burns Bog EC station

- Harvested peatland undergoing active restoration
- 8+ years of meteorological & flux ( $\text{CO}_2$  and  $\text{CH}_4$ ) data



# Training Procedures

- Larger ensemble = more robust model
  - $N \leq 10$  for data exploration/pruning
- Three way cross-validation
  - Train/validate/test
- Early Stopping: after  $e$  epochs
  - $e = 2$  for pruning stage

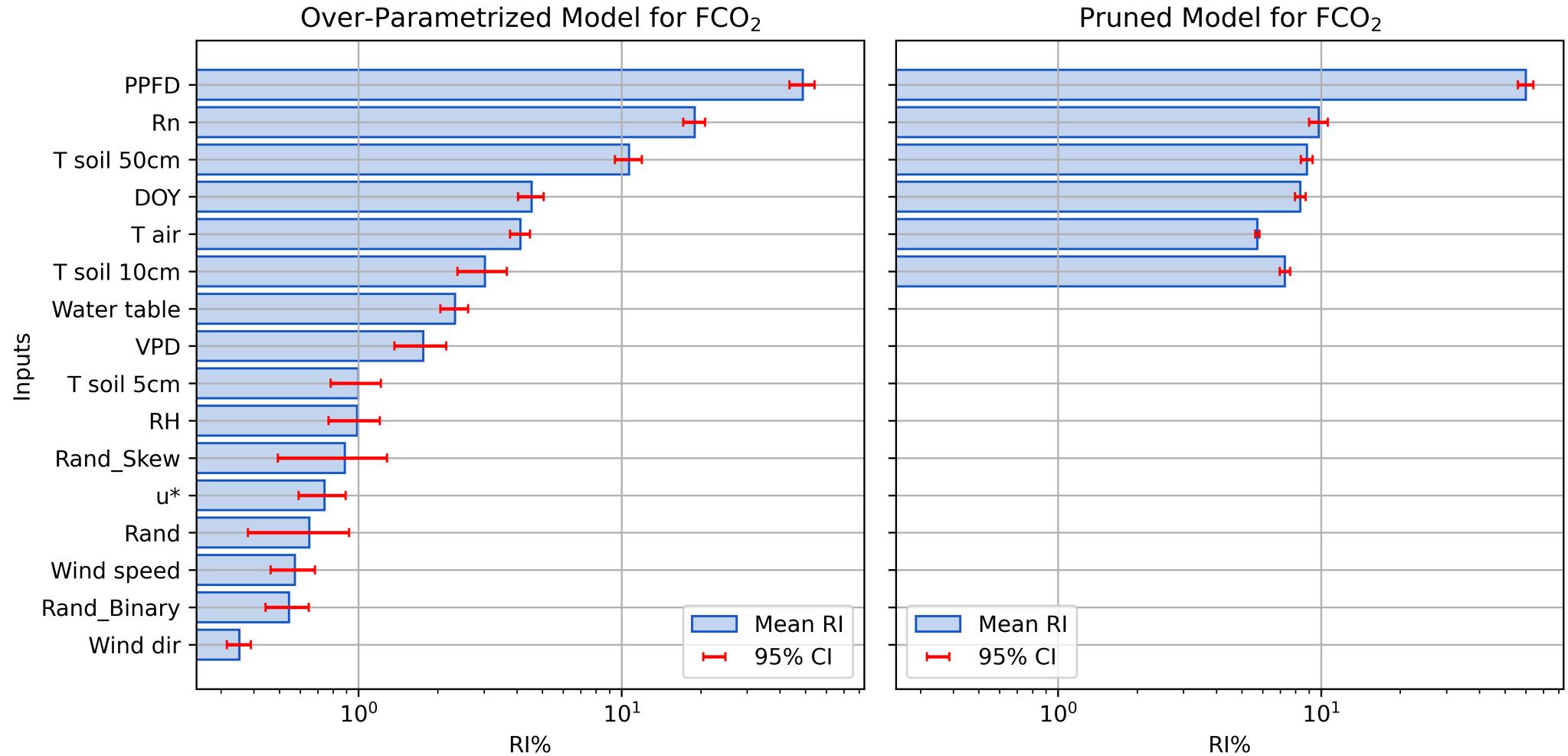


# Pruning Inputs

Calculate partial first derivative of the output with respect to each input over test data domain.

- **Relative Influence (RI) of inputs**
  - Normalized sum of squared derivatives (SSD)
- Iteratively remove inputs with RI below a threshold
  - Use set of random scalars inputs to determine threshold
    - e.g., a float [0-1], a skewed float  $[0-1]^{.25}$ , and a binary integer (0/1)

# Before and After Pruning FCO<sub>2</sub>



# The Final Model

Once pruning is complete, re-train the final production level model, excluding the random scalars

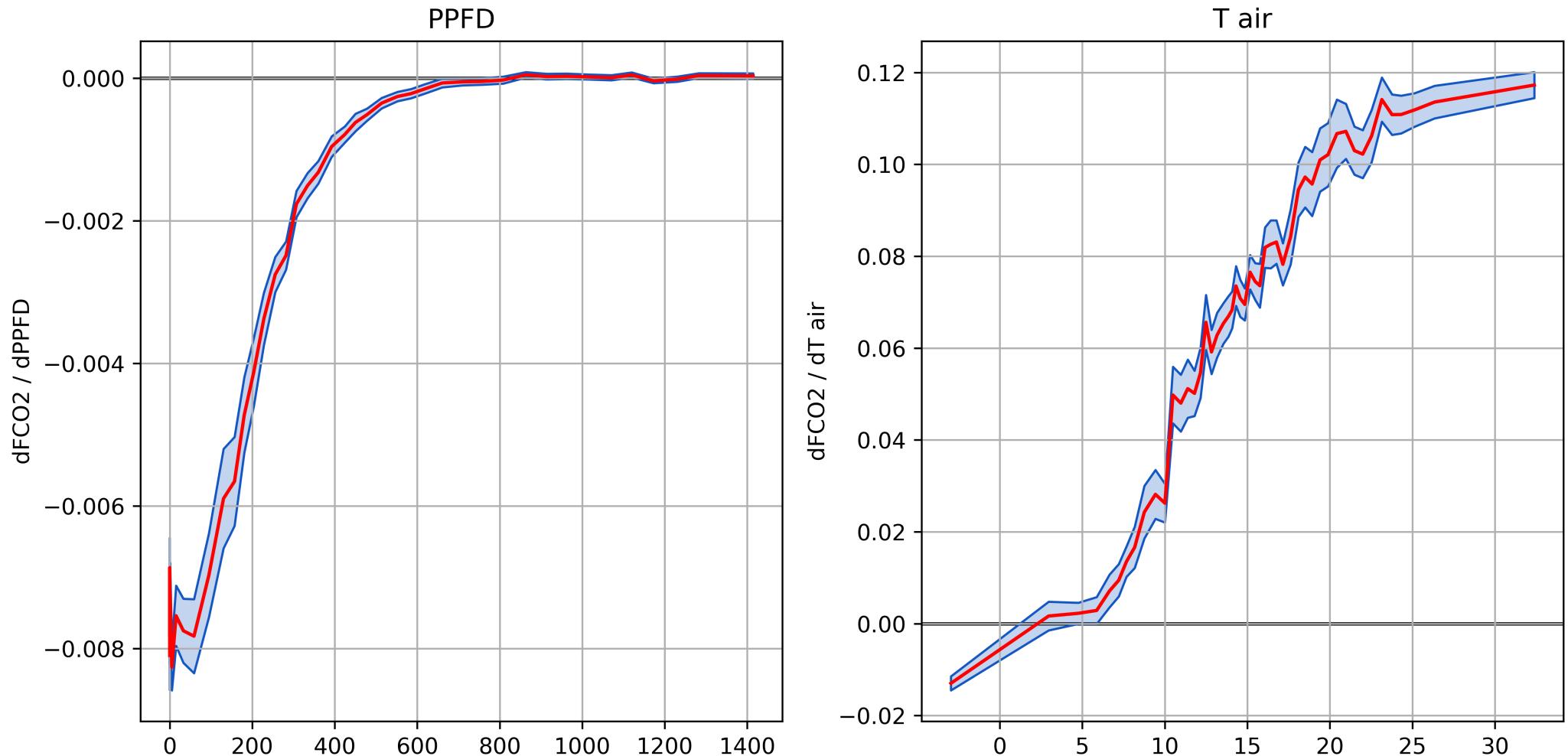
- Increase the ensemble size (e.g.,  $N = 30$ )
  - Could increase early stopping criteria (e.g.,  $e = 10$ )
  - Larger  $e$  drastically increases training
- Plot the model derivatives as a final check
  - If derivatives look implausible
    - Adjust inputs/parameters and try again

# Plotting Derivatives

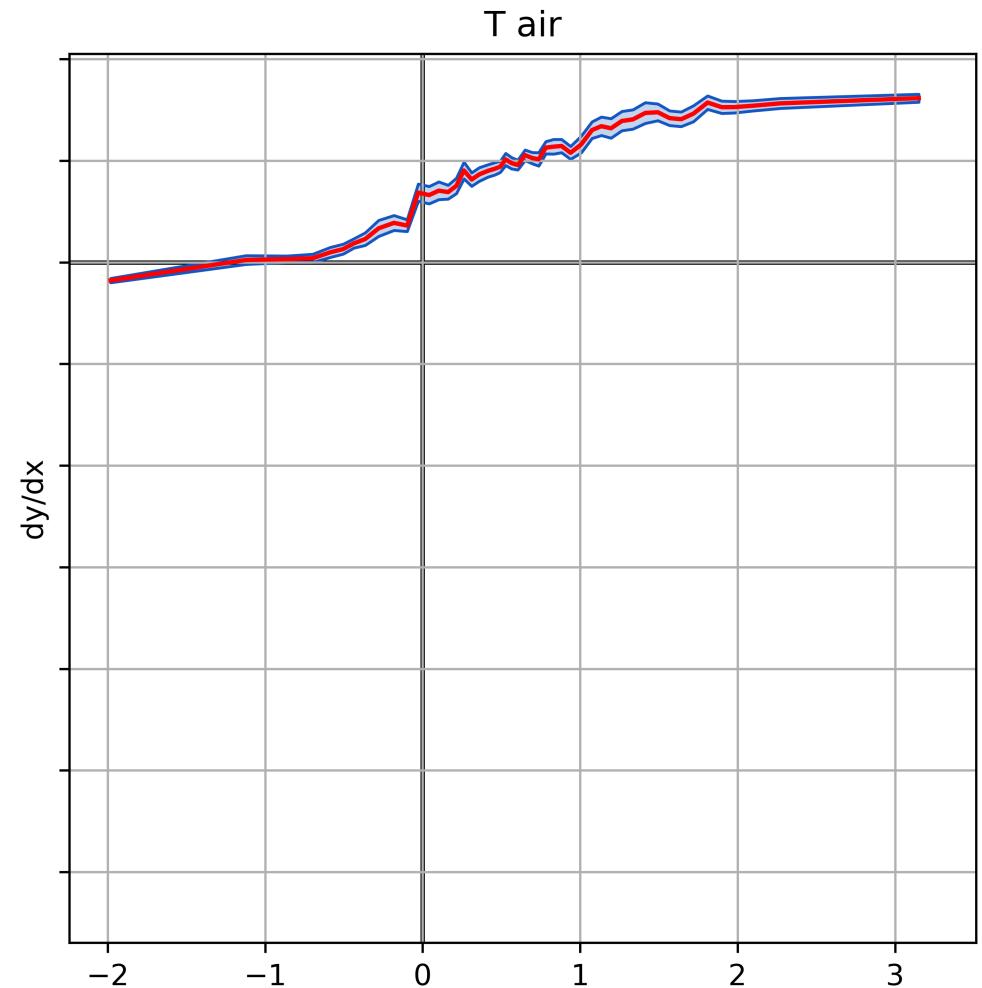
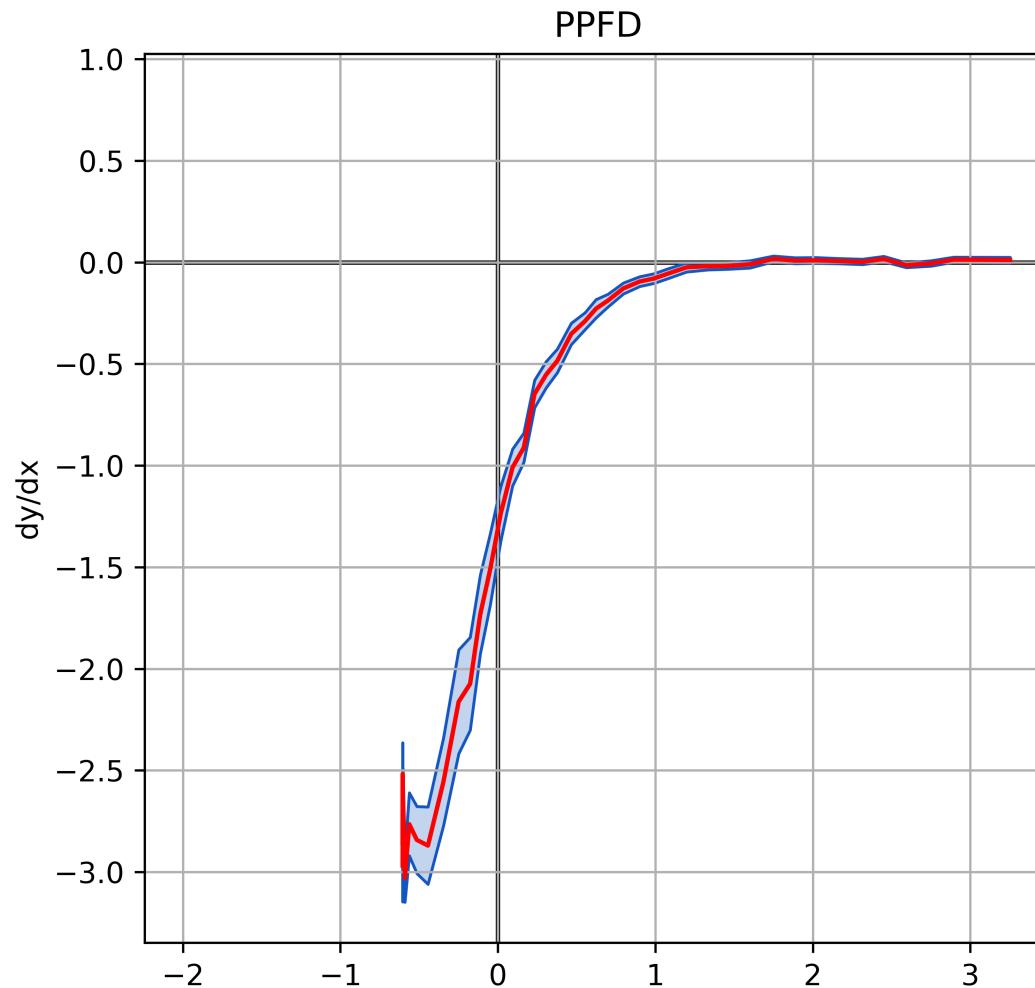
Helps ensure model responses are physically plausible

- An essential step and key advantage of NN models
- Raw derivatives show true feature responses
- Normalized derivatives scaled by input variance
  - Relative input effects on common scale
  - What the model “sees”
- 95% confidence intervals around derivatives indicate modeled confidence in relationships

# Partial Derivatives of FCO<sub>2</sub>



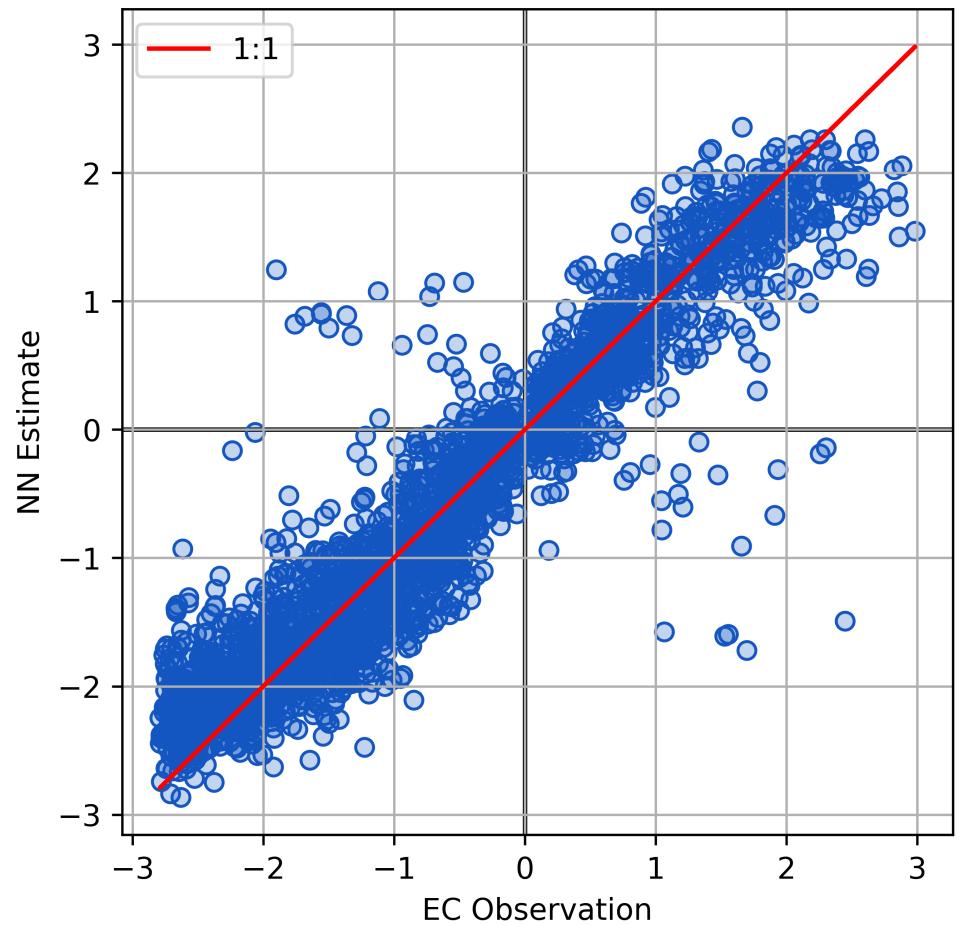
# Normalized Derivatives of FCO<sub>2</sub>



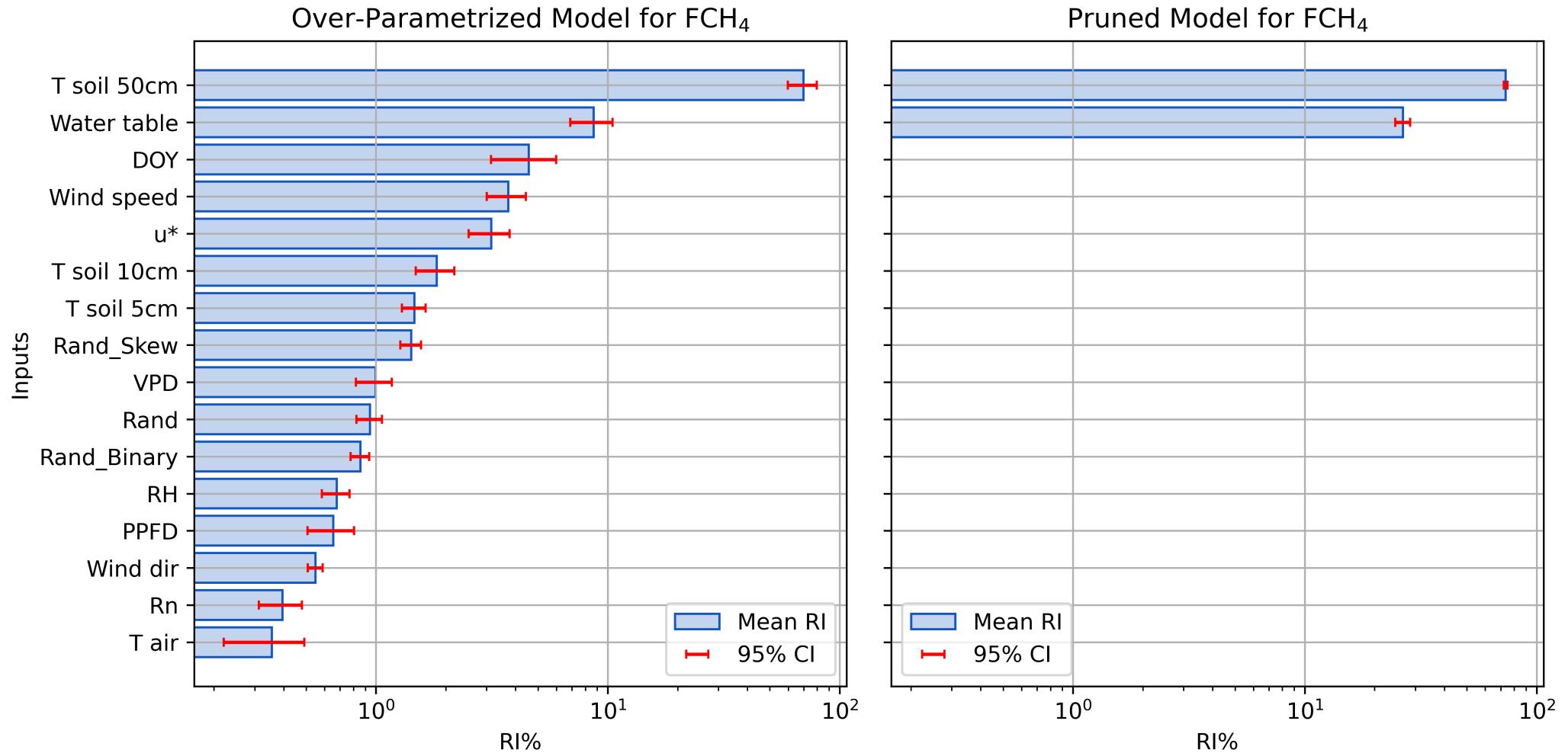
# Model Performance FCO<sub>2</sub>

Plot the model outputs and validation metrics calculated with the test data.

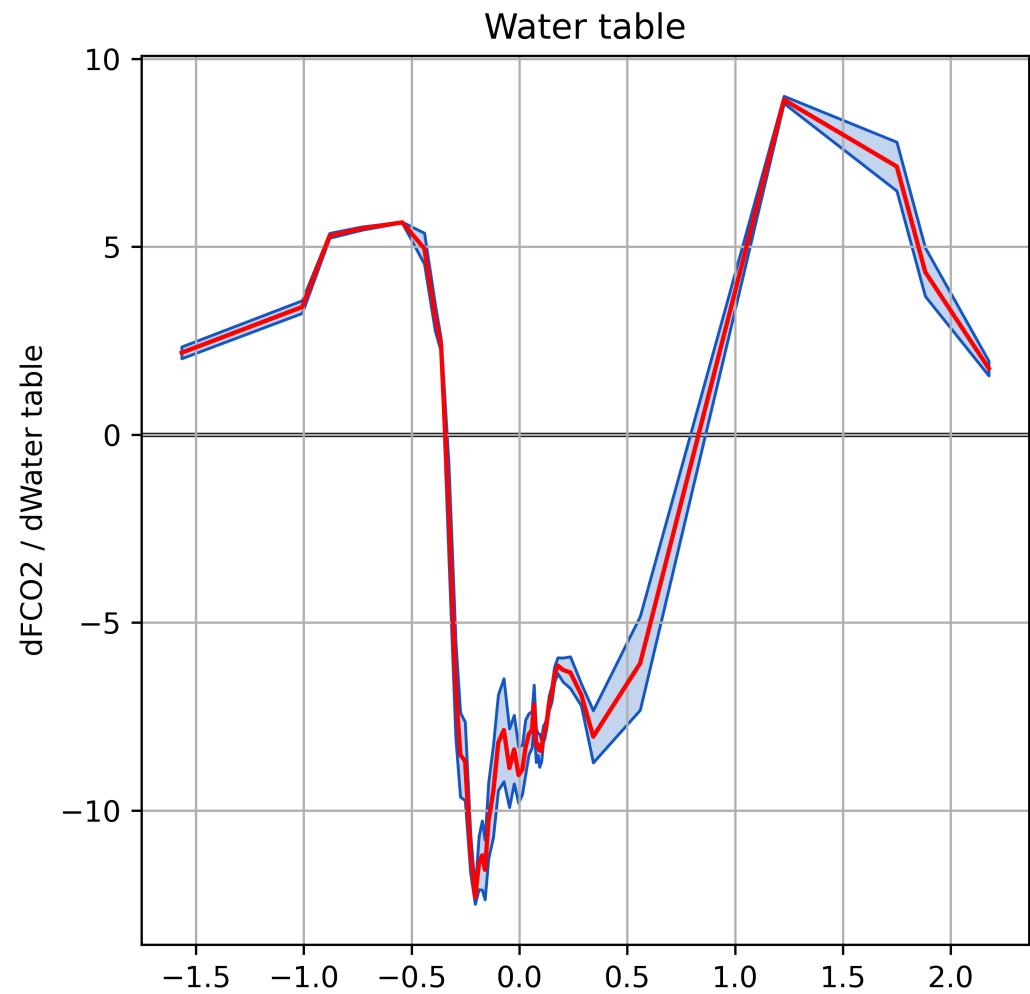
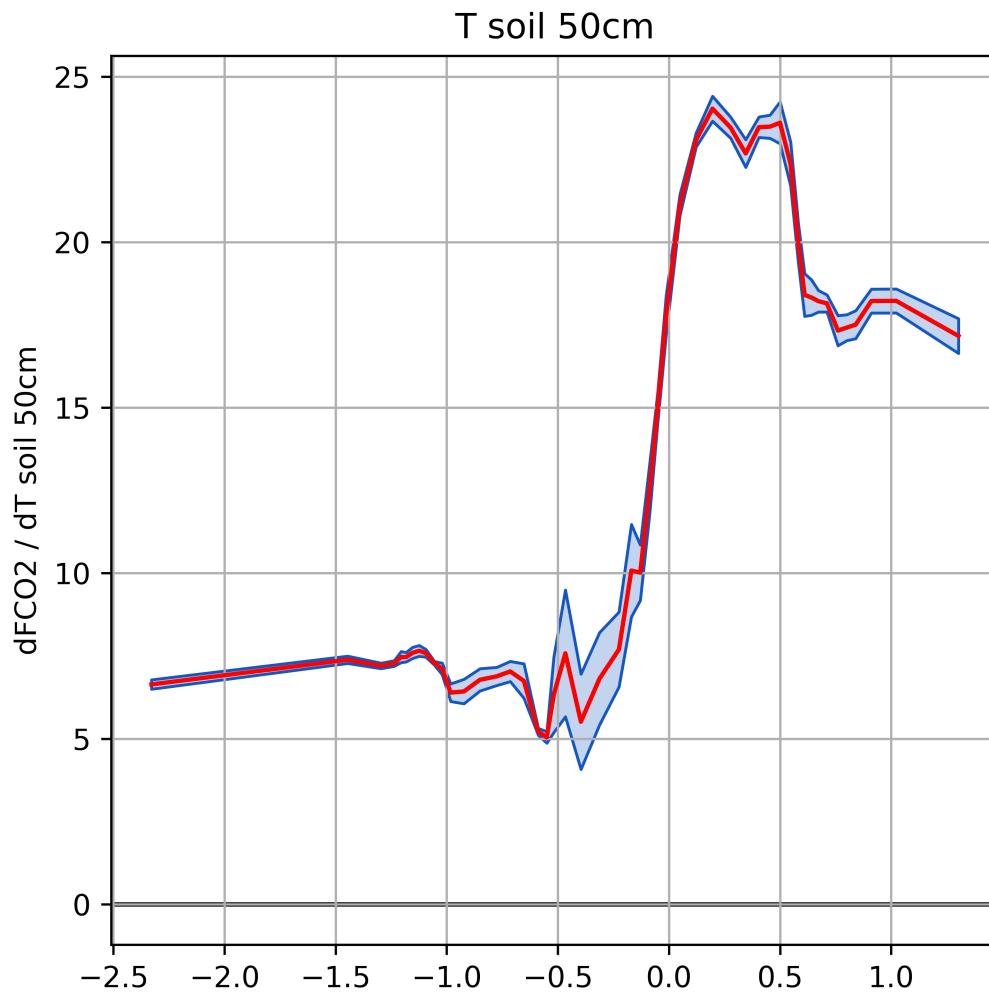
Metric	Score
RMSE	$0.45 \mu\text{mol m}^{-2} \text{s}^{-1}$
r <sup>2</sup>	0.89



# Before and After Pruning FCH<sub>4</sub>



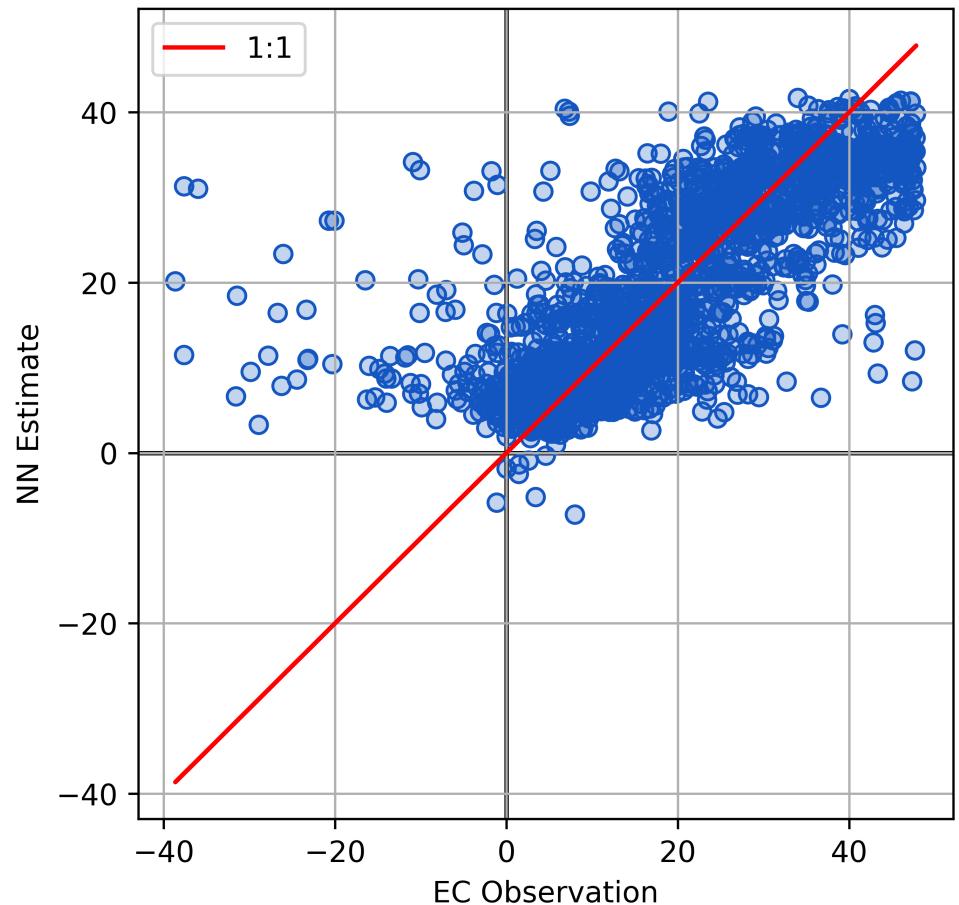
# Normalized Derivatives of FCH<sub>4</sub>



# Model Performance FCH<sub>4</sub>

Plot the model outputs and validation metrics calculated with the test data.

Metric	Score
RMSE	8.58 \$
	mol\$ $m^{-2} s^{-1}$
r <sup>2</sup>	0.63



# Next Steps

- Custom NN architecture: Separating input layers may allow us partition fluxes.
  - e.g.,  $\text{FCO}_2$  into GPP and ER
- Flux footprints: map response to spatial heterogeneity
- Upscaling: in space and time
- $u^*$  filtering: partial derivatives could identify  $u^*$  thresholds
- Compare to process based models (e.g., CLASSIC)

# Thank You

Questions?