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	- Model ensembles - 3-way cross validation - Pruning inputs
Black boxes models	- Plot partial derivatives - Feature importance
Computationally expensive	- Tensorflow - GPU processing

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*

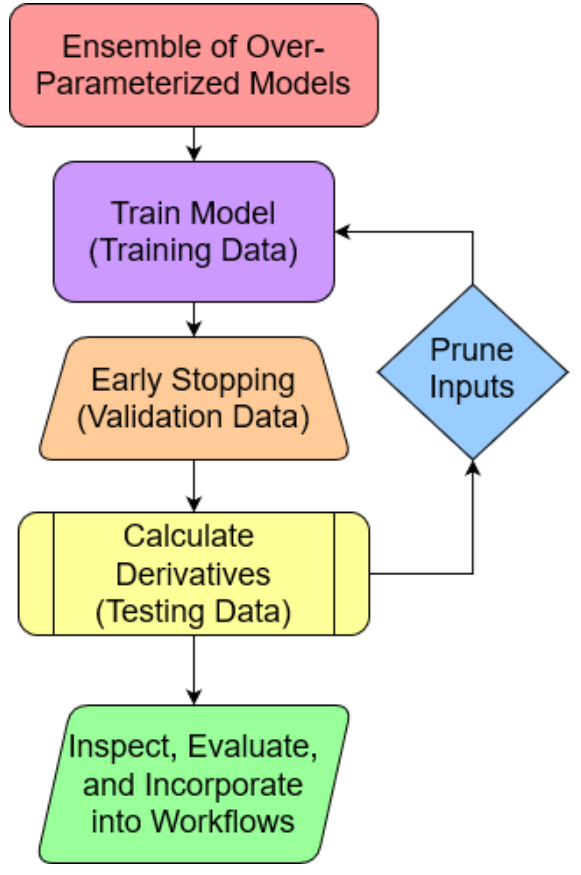
Example Data

Burns Bog EC station

- Harvested peatland undergoing active restoration
- 8+ years of meteorological & flux (CO_2 and CH_4) data

Training Procedures

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

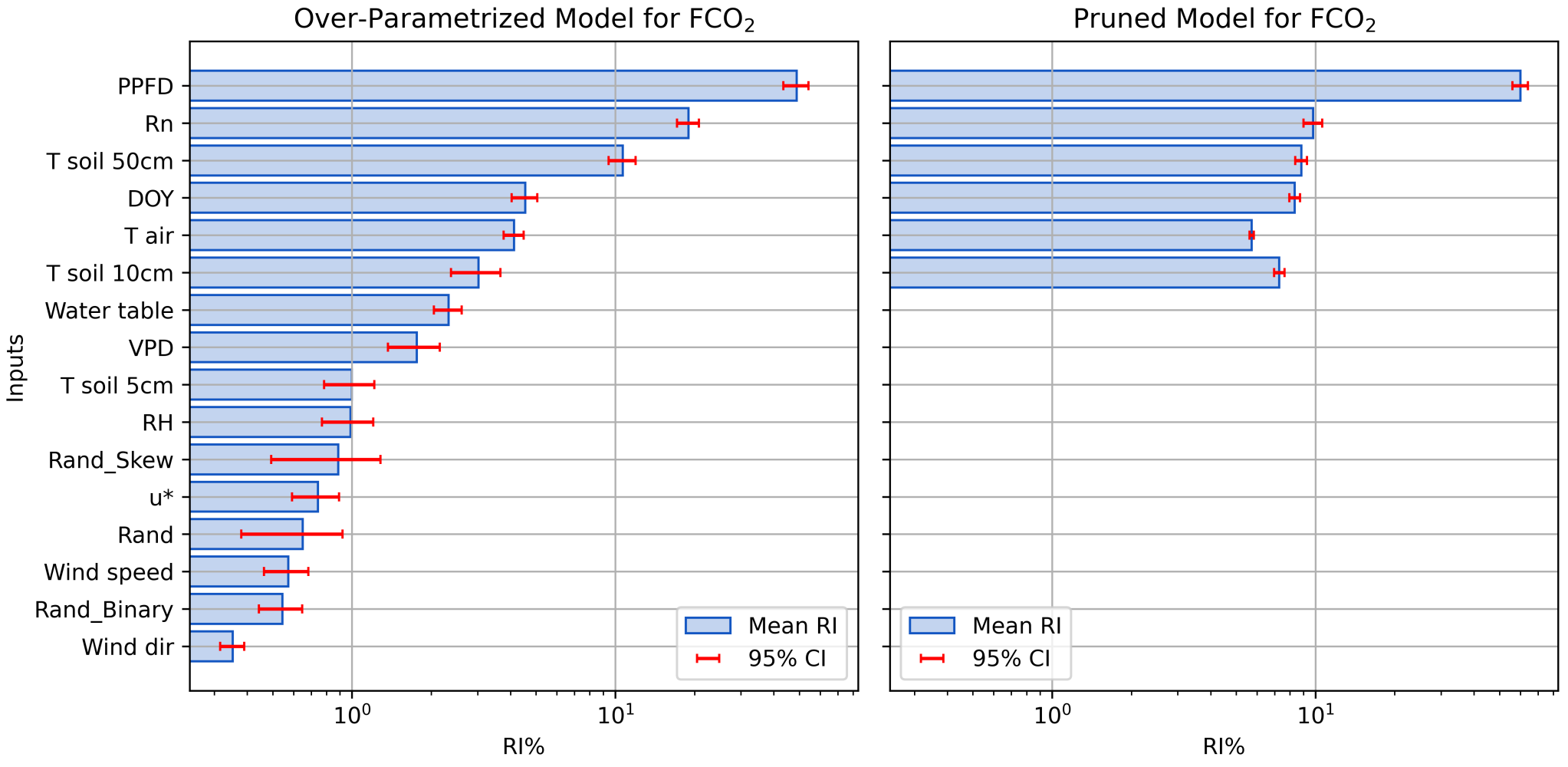


Pruning Inputs

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 $[0\text{-}1]^{.25}$, and a binary integer (0/1)

Before and After Pruning FCO_2



The Final Model

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

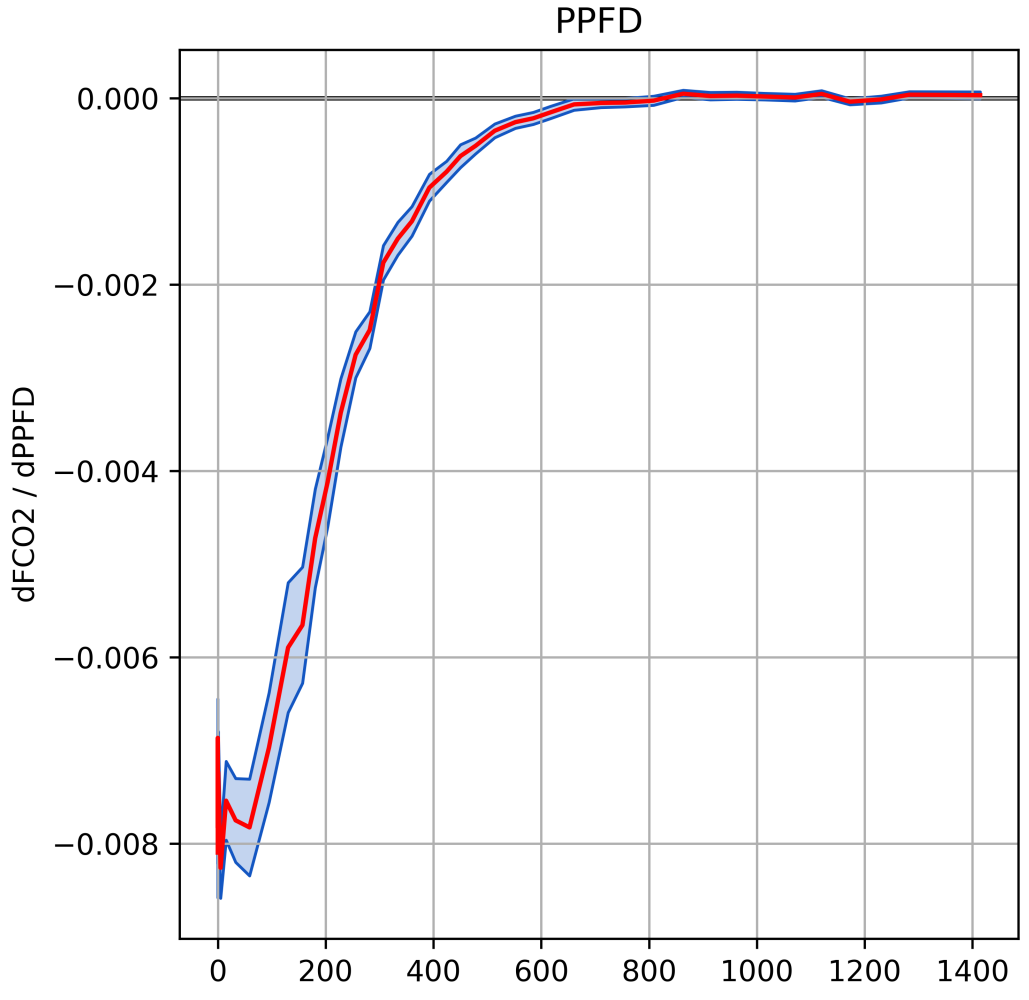
- Increase the ensemble size (e.g., N =30)
  - Could increase early stopping criteria (e.g., e = 10)
  - Larger e drastically increases training
- Plot the model derivatives as a final check
  - If derivatives look implausible
    - Adjust inputs/parameters and try again

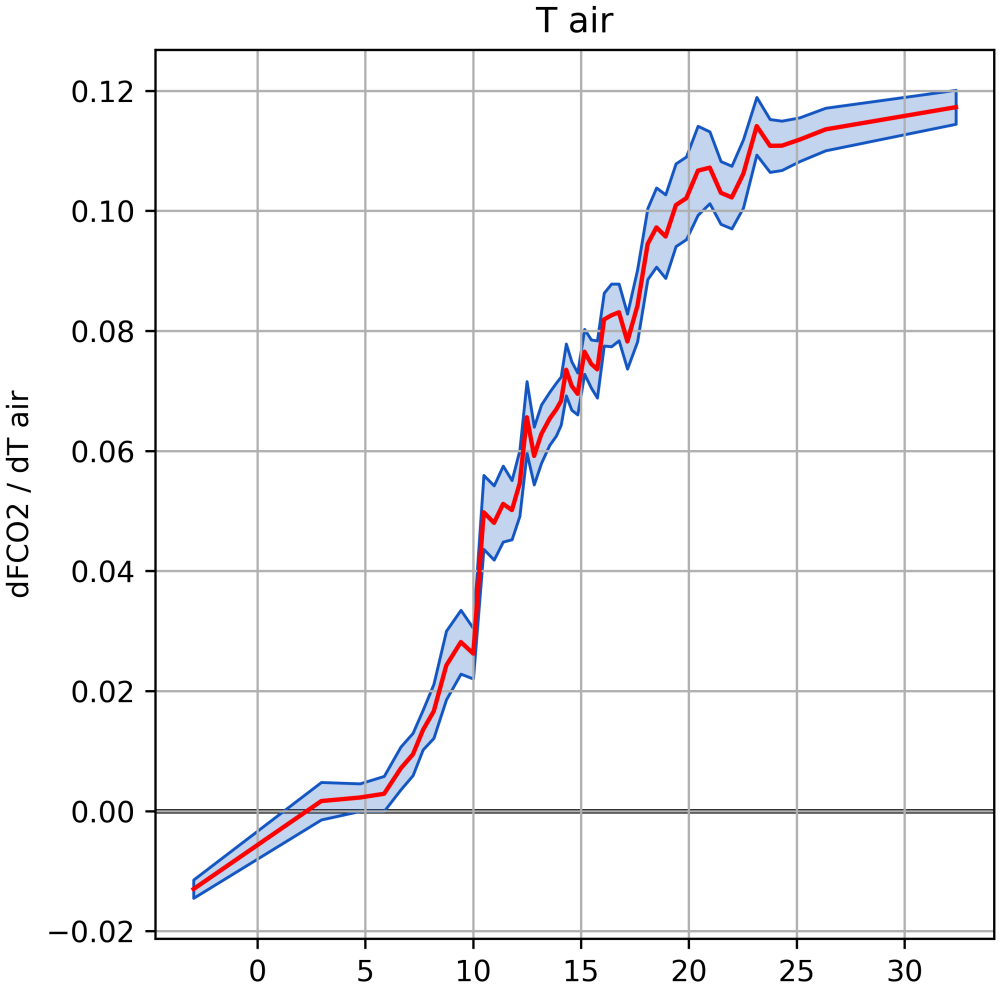
Plotting Derivatives

Helps ensure model responses are physically plausible

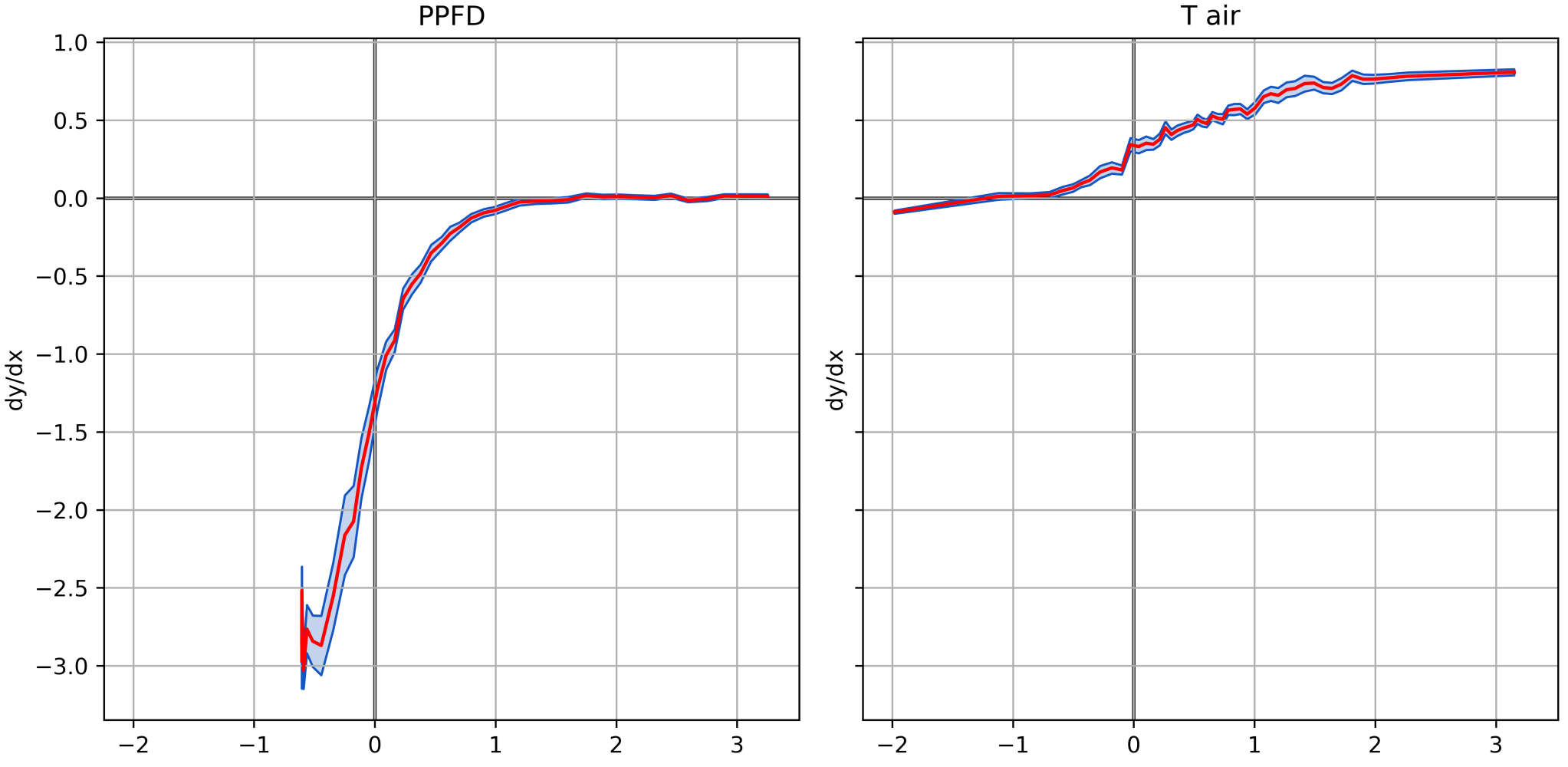
- An **essential step** and **key advantage** of NN models
- Raw derivatives show true feature responses
- Normalized derivatives scaled by input variance
  - Relative input effects on common scale
  - What the model “sees”
- 95% confidence intervals around derivatives indicate modeled confidence in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

Partial Derivatives of FCO_2





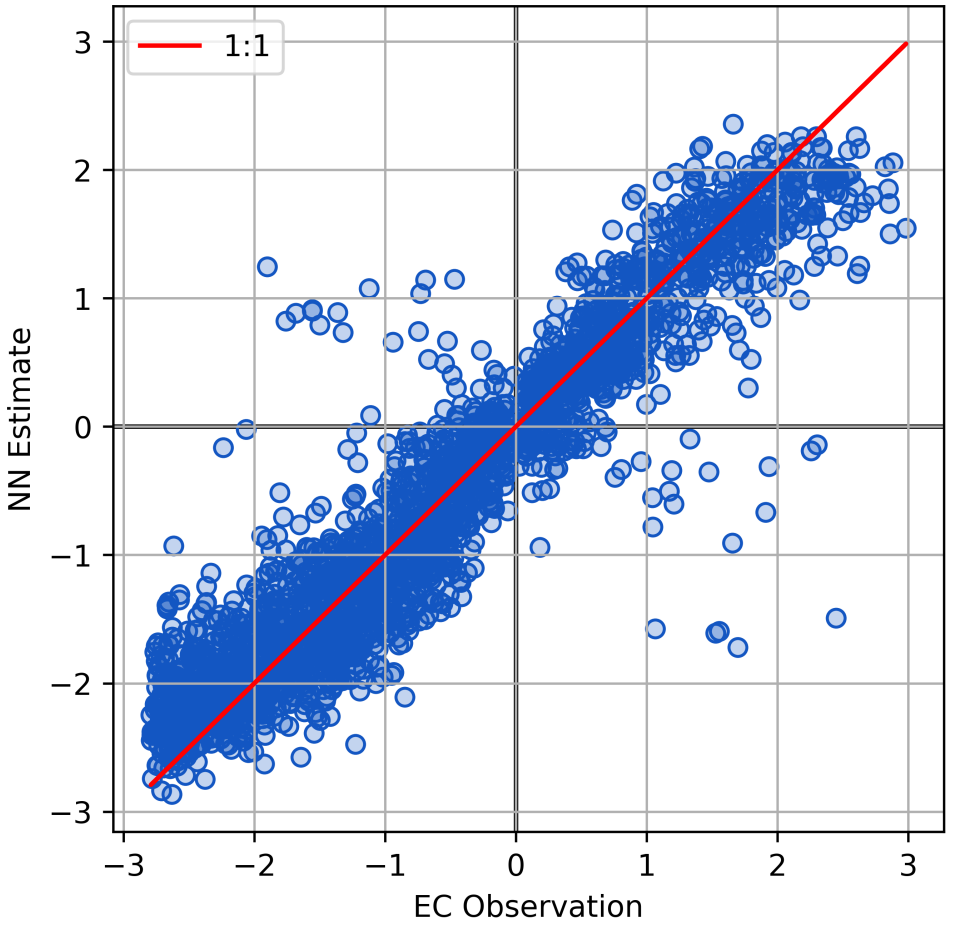
Normalized Derivatives of FCO_2



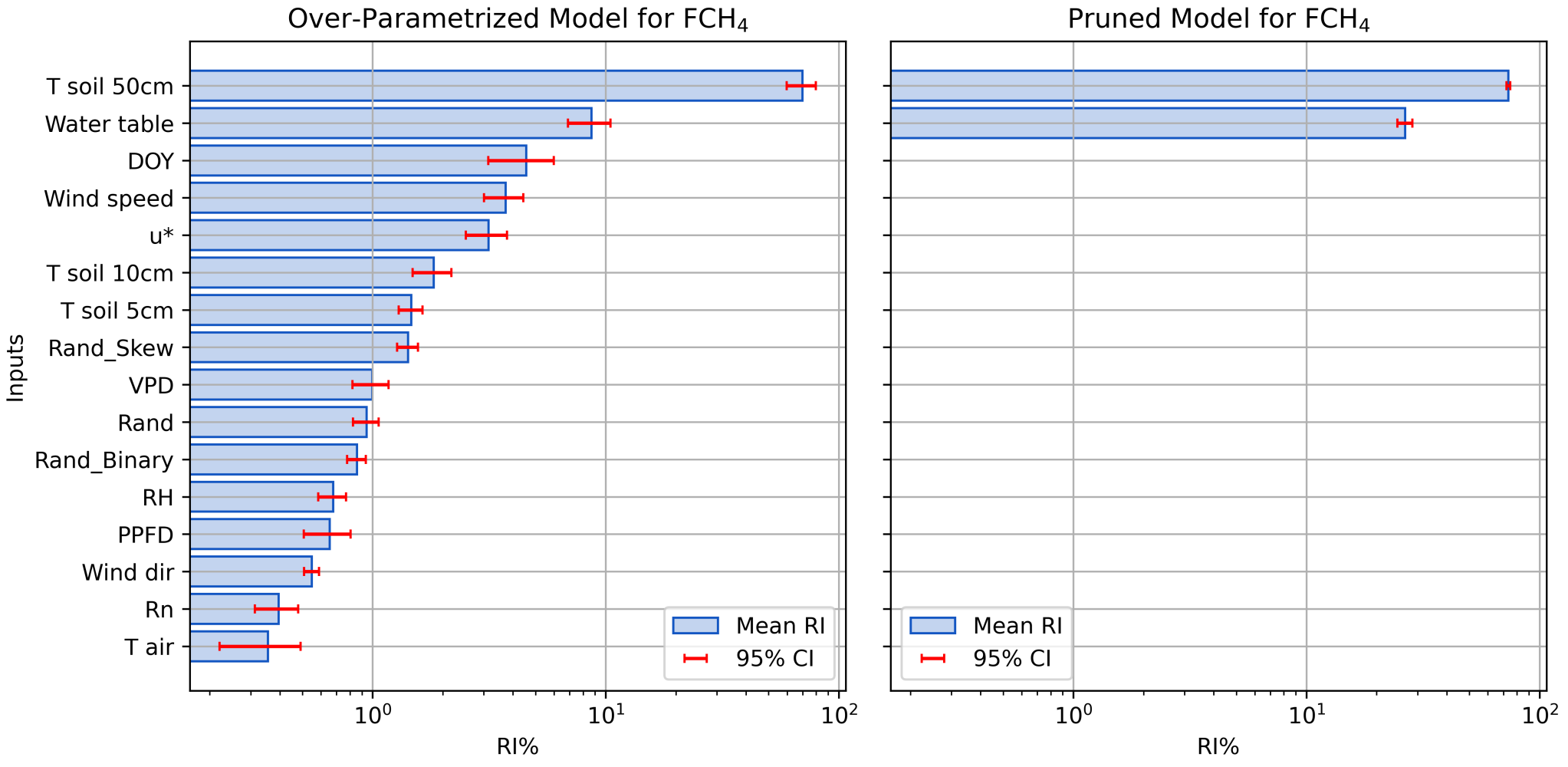
Model Performance FCO_2

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

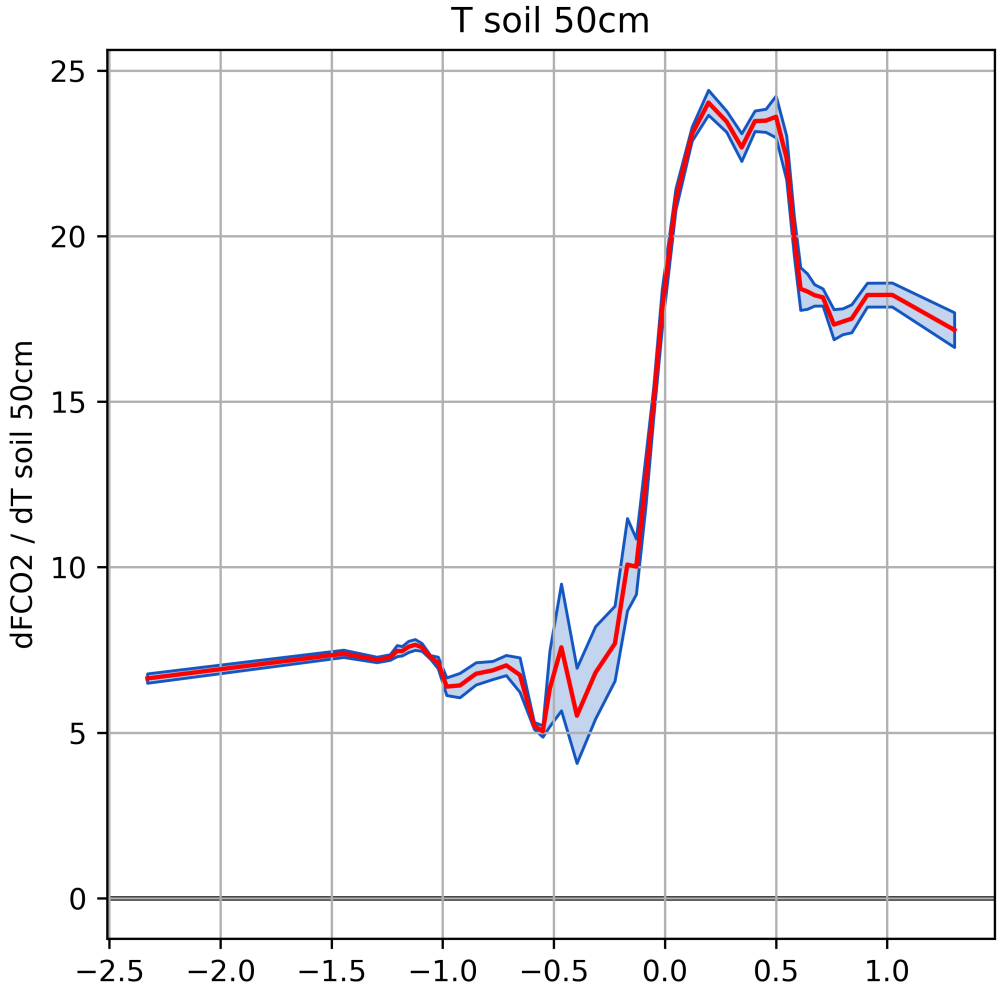
Metric	Score
RMSE	0.45 $\mu mol\, m^{-2}\, s^{-1}$
r^2	0.89

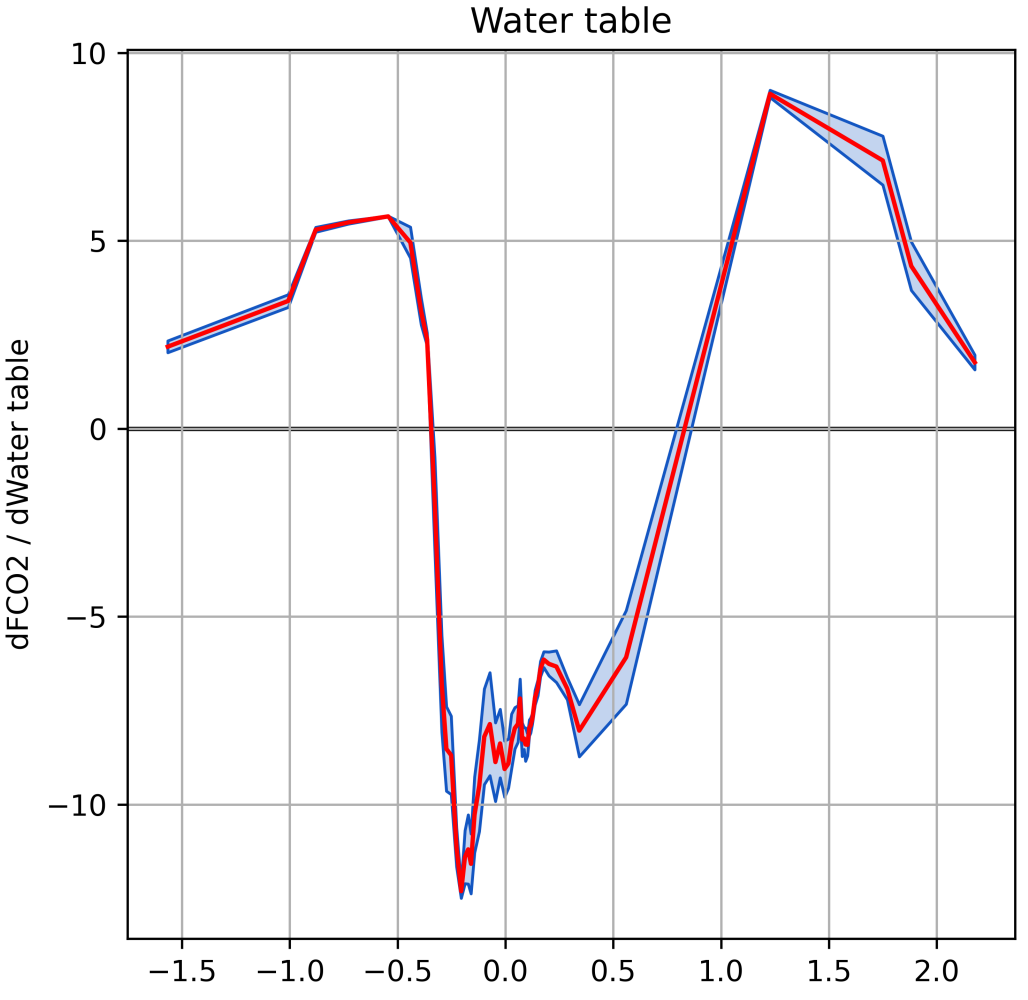


Before and After Pruning FCH_4



Normalized Derivatives of FCH_4

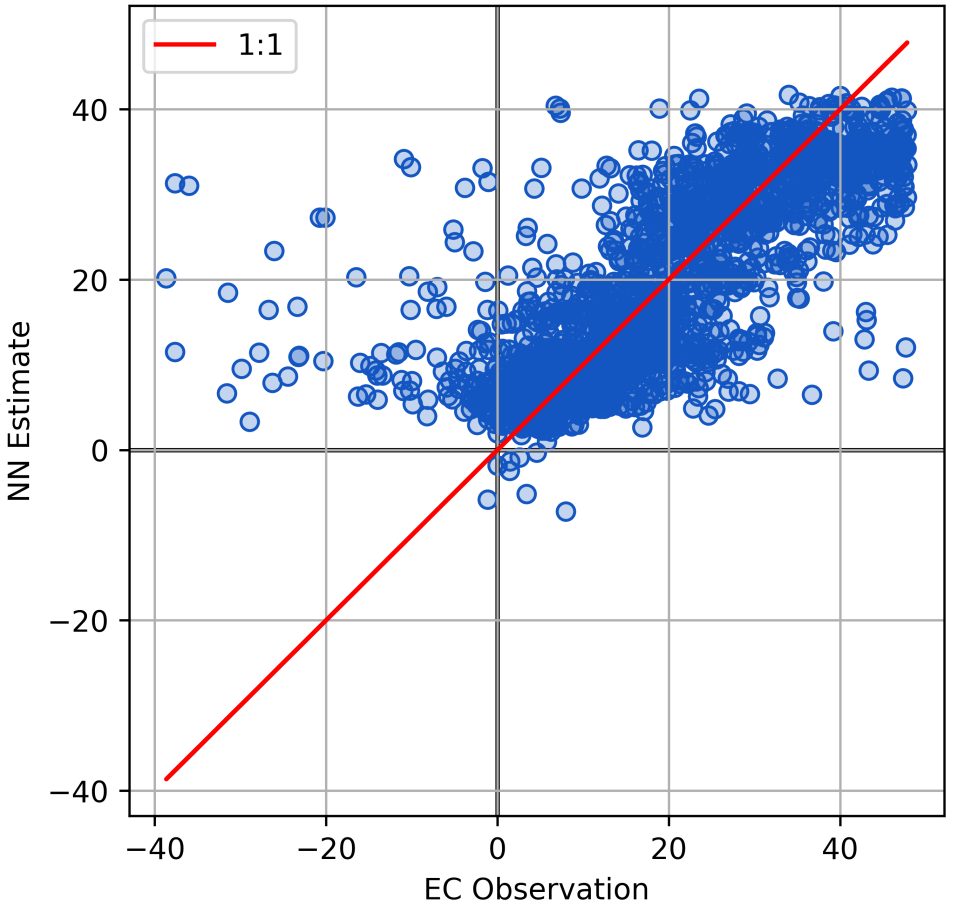




Model Performance FCH_4

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

Metric	Score
RMSE	8.58 $\mu mol\, m^{-2}\, s^{-1}$
r^2	0.63



Next Steps

- Custom NN architecture: Separating input layers may allow us partition fluxes.
  - e.g., FCO_2 into GPP and ER
- Flux footprints: map response to spatial heterogenity
- Upscaling: in space and time
- u* filtering: partial derivatives could identify u* thresholds
- Compare to process based models (e.g., CLASSIC)

Thank You

Questions?