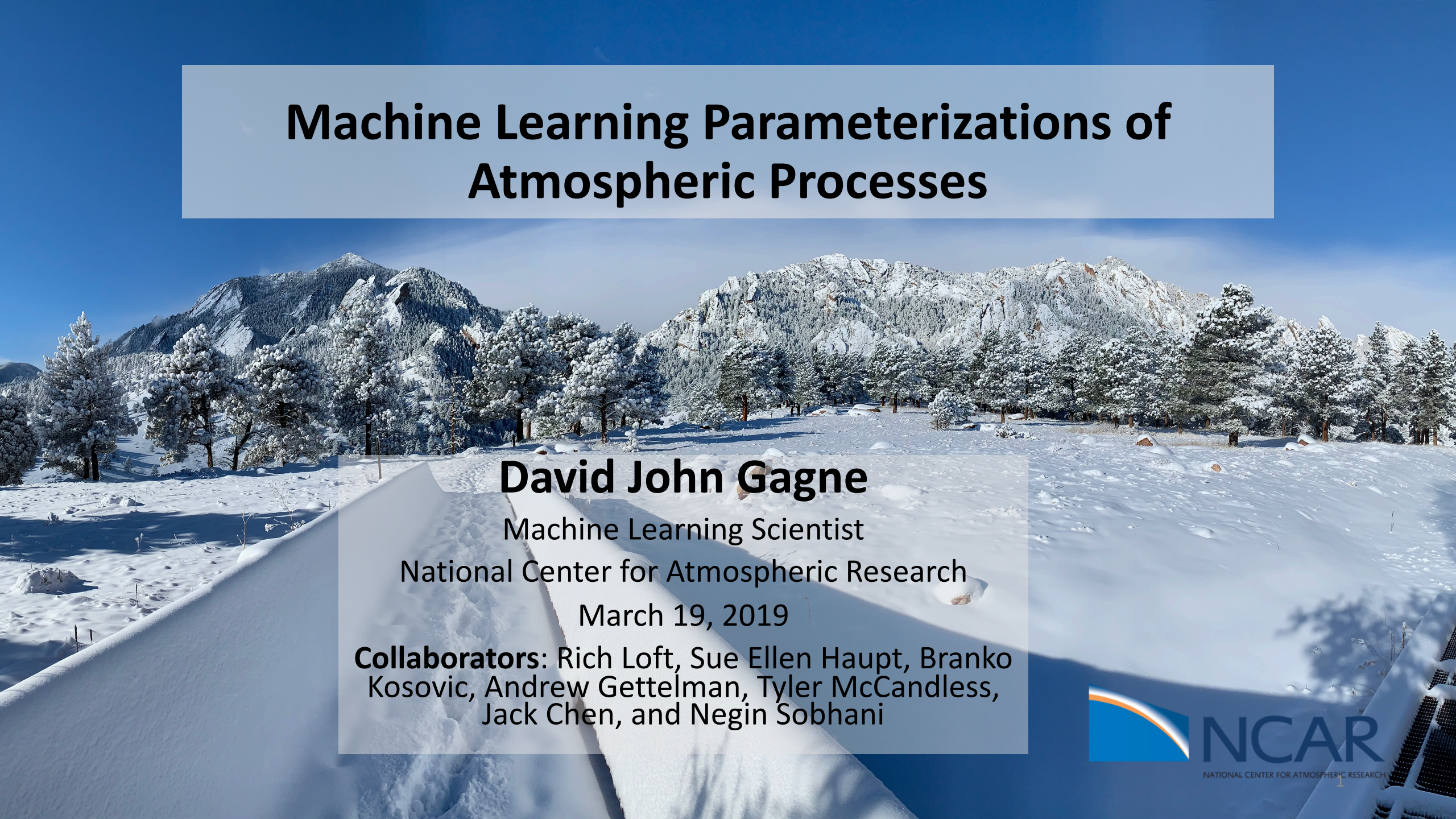
Machine Learning Parameterizations of Atmospheric Processes



David John Gagne

Machine Learning Scientist

National Center for Atmospheric Research

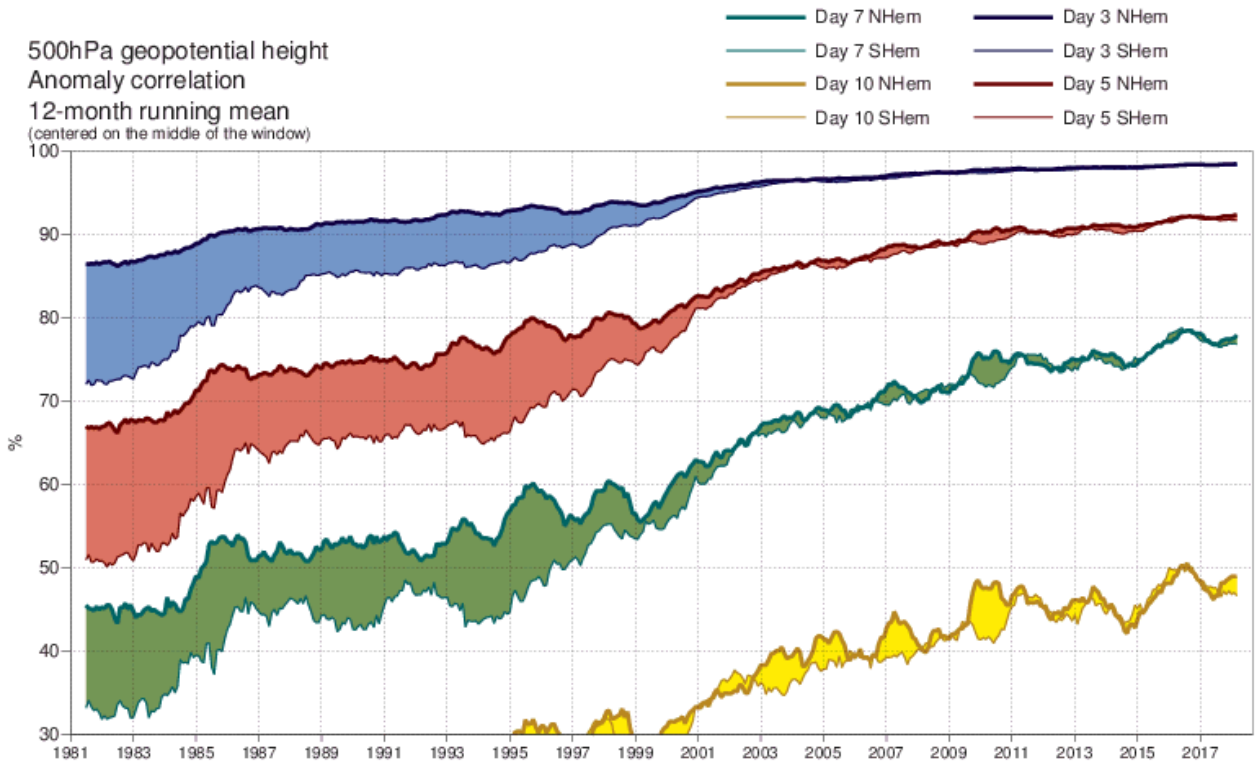
March 19, 2019

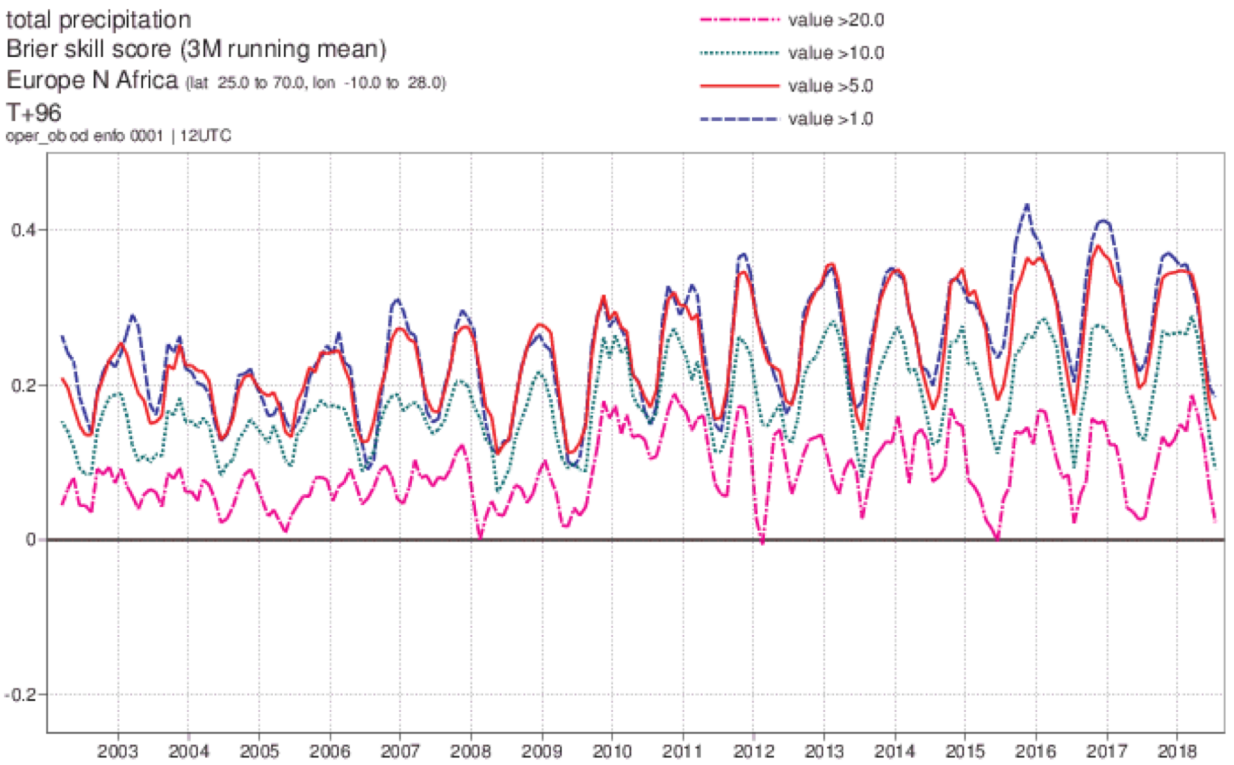
Collaborators: Rich Loft, Sue Ellen Haupt, Branko Kosovic, Andrew Gettelman, Tyler McCandless, Jack Chen, and Negin Sobhani

NCAR
NATIONAL CENTER FOR ATMOSPHERIC RESEARCH

Numerical Modeling Is Ripe for Disruption

ECMWF Prediction Errors

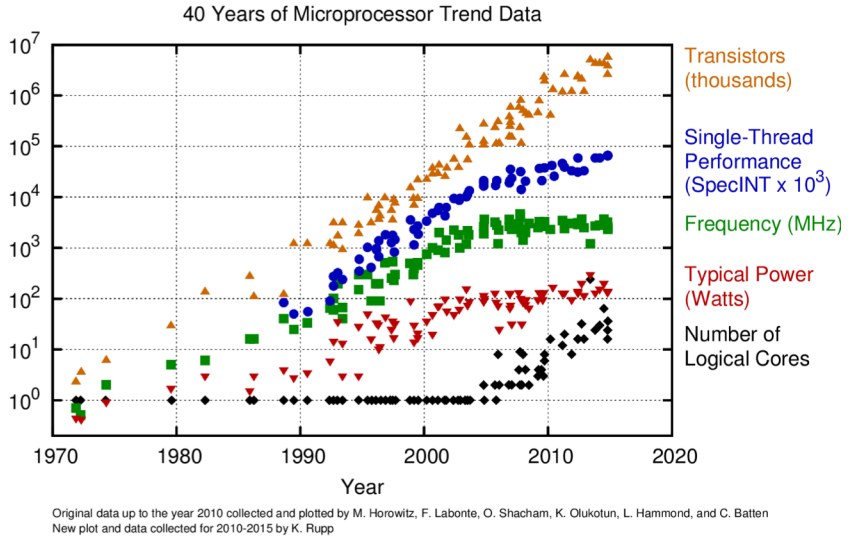




- NWP model predictions have steadily improved thanks to better data assimilation and resolution
- Weather and climate models have grown increasingly complex
- However, the rate of NWP model error improvement is slowing
- Trends in high performance computing systems are making the current development paradigm unsustainable

Processors Hitting Fundamental Limits

- **Processor trends**
  - Transistor size -> nanoscale
  - Hitting power density limits
  - Flat/slowing thread and clock speeds
  - More cores (∝ transistors) per proc.
  - Lower fraction of peak for BW limited code



Source: Karl Rupp

- **Climate model and computing trends not aligned!**
  - Climate applications are state heavy, memory bandwidth intensive, with low arithmetic intensity.
  - Physics code is branchy, hard to vectorize, has divides and load imbalances.
- **Climate Model Benchmark: 3 km GCM at 5 sim. Years/day**
  - Extrapolated 3 km FV3 Simulation: 0.97 years/day on 100,000 cores
  - 17x too slow!

Courtesy Rich Loft, NCAR

Paths for Further Improvement

- Code optimization: redesign model codes to run more efficiently
  - Pros: model runs faster, discover potential bugs
  - Cons: extremely labor intensive, less interpretable code
- Heterogeneous Computing: run model components on GPUs, FPGAs, TPUs, etc.
  - Pros: Great speedup for parallel processes, power savings
  - Cons: rewriting code for each framework, data transfers
- **Machine learning emulation**: replace model components with ML model
  - Pros: large speedups, approximate higher complexity fast
  - Cons: data, non-stationarity, interfaces to ML software
- *Note*: NCAR is working on all of these areas

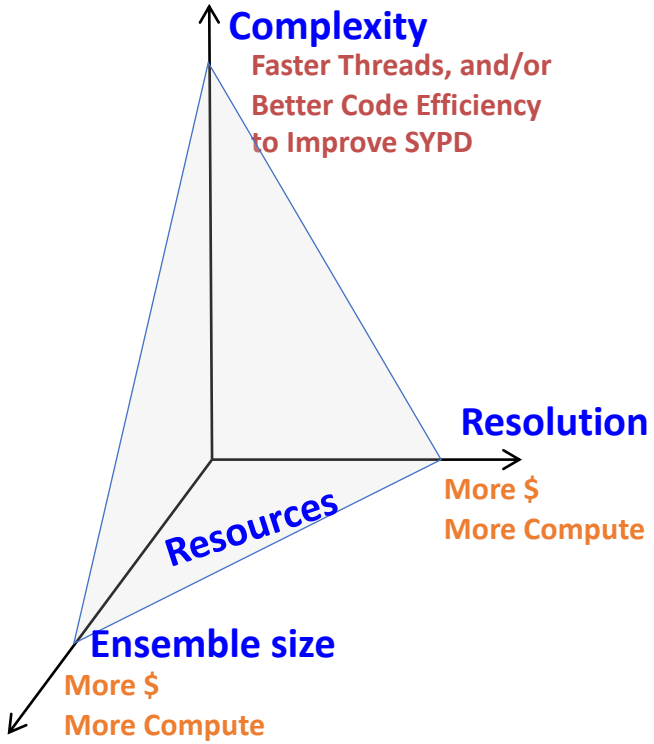
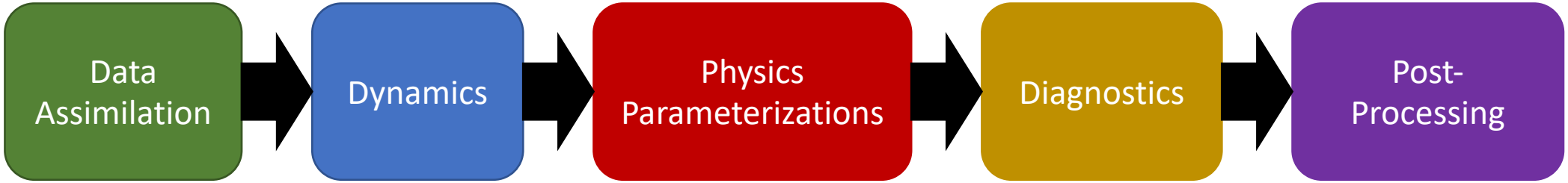
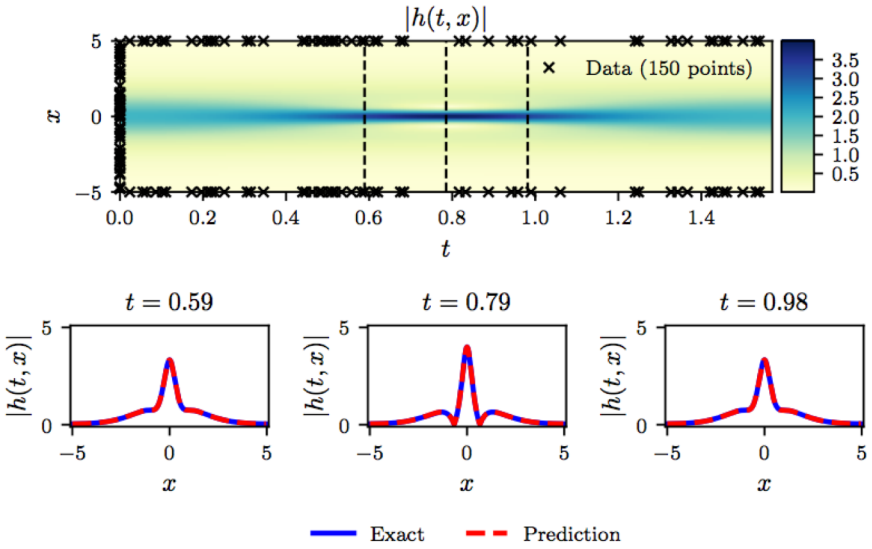


Figure courtesy Rich Loft via Jim Kinter and others

Machine Learning Along the NWP Pipeline

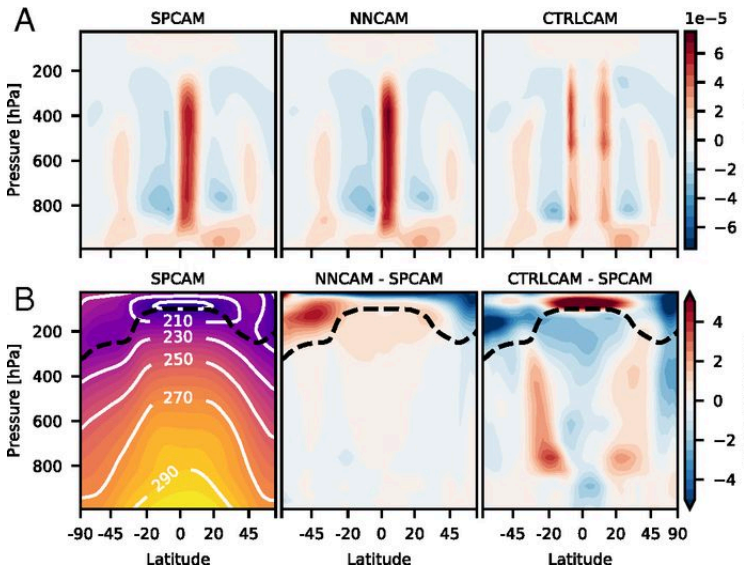


Physics-Informed Neural Network



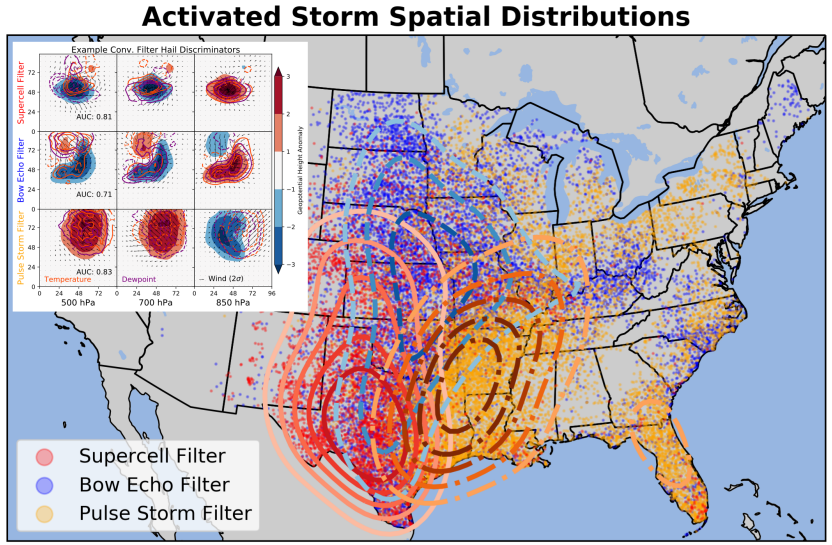
Raissi, M., P. Perdikaris, and G. E. Karniadakis, 2017: Physics Informed Deep Learning (Part I): Data-driven Solutions of Nonlinear Partial Differential Equations, arXiv Preprint. https://arxiv.org/abs/1711.10561

Neural Network Parameterization



S. Rasp, M. S. Pritchard, and P. Gentine, 2018: Deep learning to represent subgrid processes in climate models. PNAS, 39, 9684-9689. https://doi.org/10.1073/pnas.1810286115

Machine Learning Hail Prediction



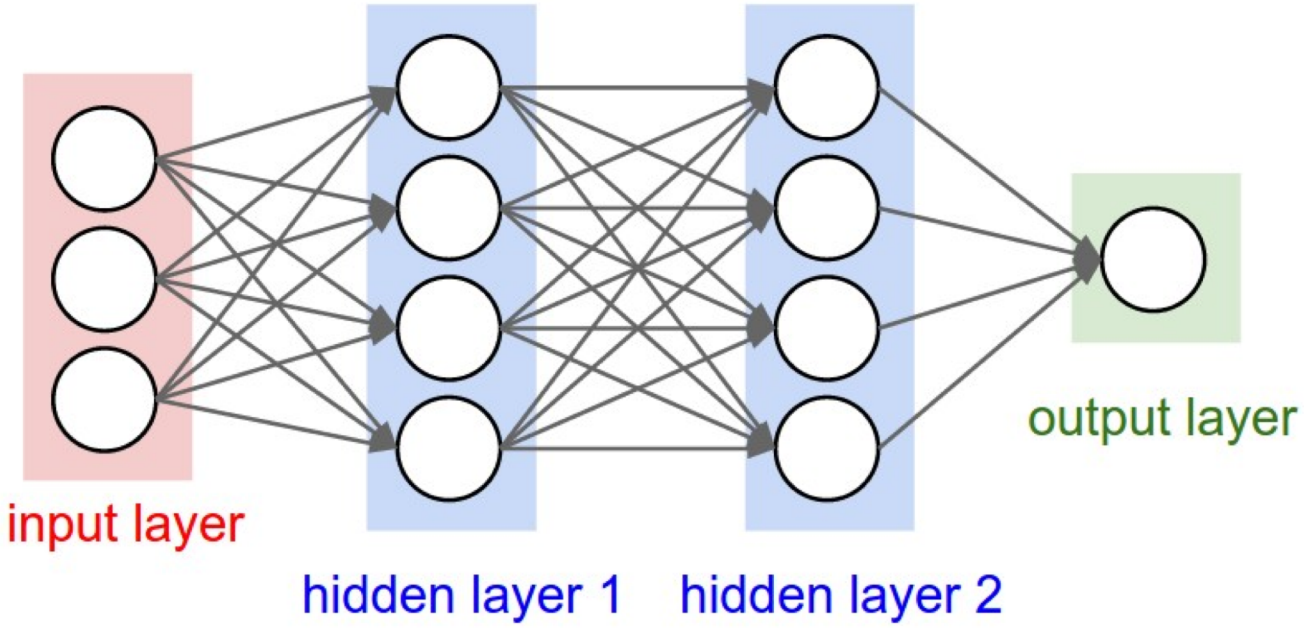
D. J. Gagne, S. E. Haupt, and D. W. Nychka 2019: Interpretable Deep Learning for Spatial Analysis of Severe Hailstorms. Mon. Wea. Rev., Accepted.

How Can ML Parameterizations Help Physical Models?

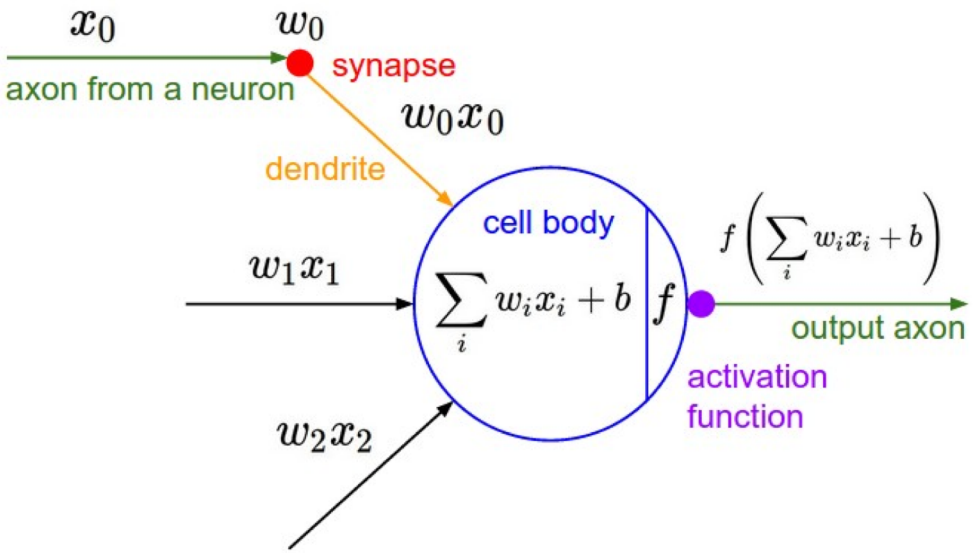
- Emulate an existing parameterization to enable faster computation
- Example: Krasnopolsky et al. (2010) emulating the radiation physics with a neural network
- Emulate a computationally intense but more realistic parameterization
- Example: **Neural network emulating cloud droplet to rain drop growth processes**
- Create a new parameterization from long records of observations
- Example: **Random forest parameterization of energy fluxes between lower atmosphere and land surface**

Neural Network Basics

Artificial Neural Network Structure



Perceptron (artificial neuron)



Training Procedure

1. Send batch of training examples through network
2. Calculate prediction error
3. Calculate error gradients back through layers and update weights
4. Repeat over all training examples until errors are satisfactory

Definitions

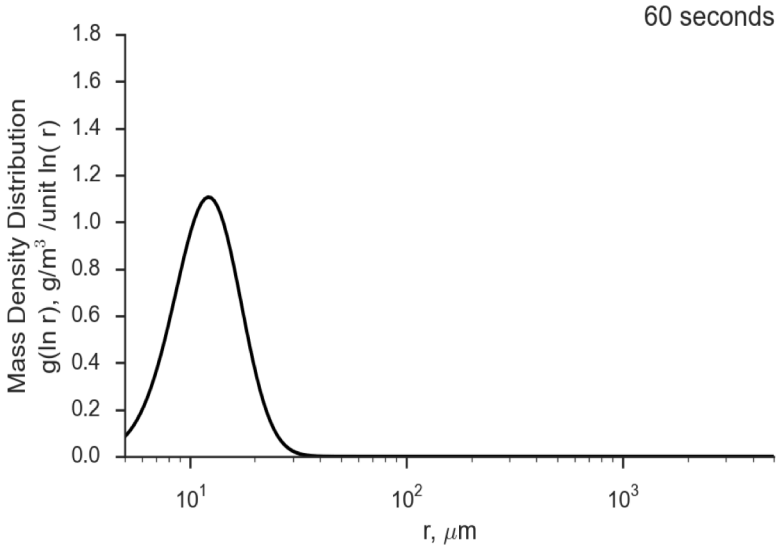
Batch: subset of training examples used to update weights

Epoch: One pass through all examples in training set

Images from http://cs231n.github.io/convolutional-networks/

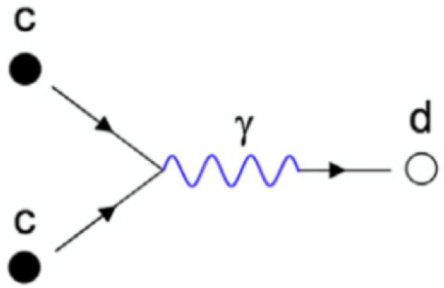
Emulating Cloud Microphysics: Motivation

- Precipitation formation is a critical uncertainty for weather and climate models.
- Different sizes of drops interact to evolve from small cloud drops to large precipitation drops.
- Detailed codes (right) are too expensive for large scale models, so empirical approaches are used.
- Let's emulate one (or more)
- Goal: put a detailed treatment into a global model and emulate it using ML techniques.
- Good test of ML approaches: can they reproduce a complex process, but with simple inputs/outputs?

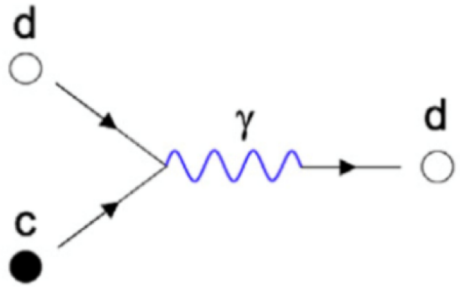


Sd-coal model output animation
Credit: Daniel Rothenberg

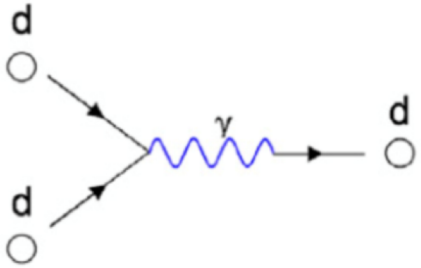
Cloud Droplet to Rain Drop Processes



autoconversion



rain accretion



self-collection

Cloud droplets grow into rain droplets through 3 processes

Autoconversion: cloud droplets stochastically collide in a chain reaction to form rain drops

Rain Accretion: rain drops can collide with other cloud droplets

Self-Collection: rain drops can collide with other raindrops

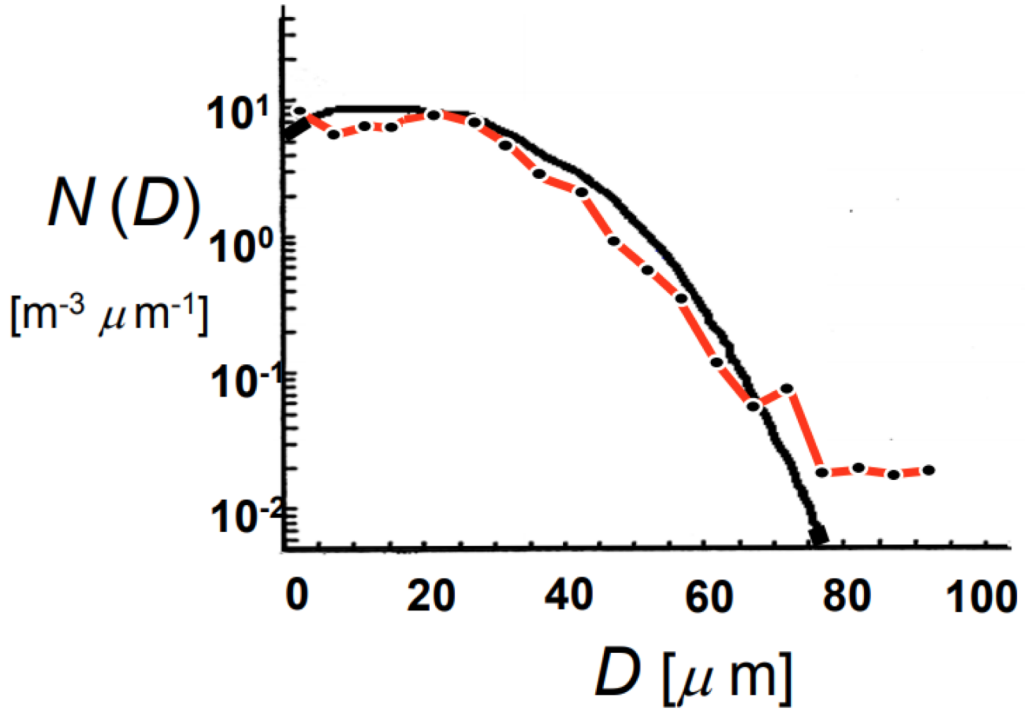
d: rain drop
c: cloud droplet
CCN: cloud condensation nuclei

Microphysics: Bulk vs. Bin Schemes

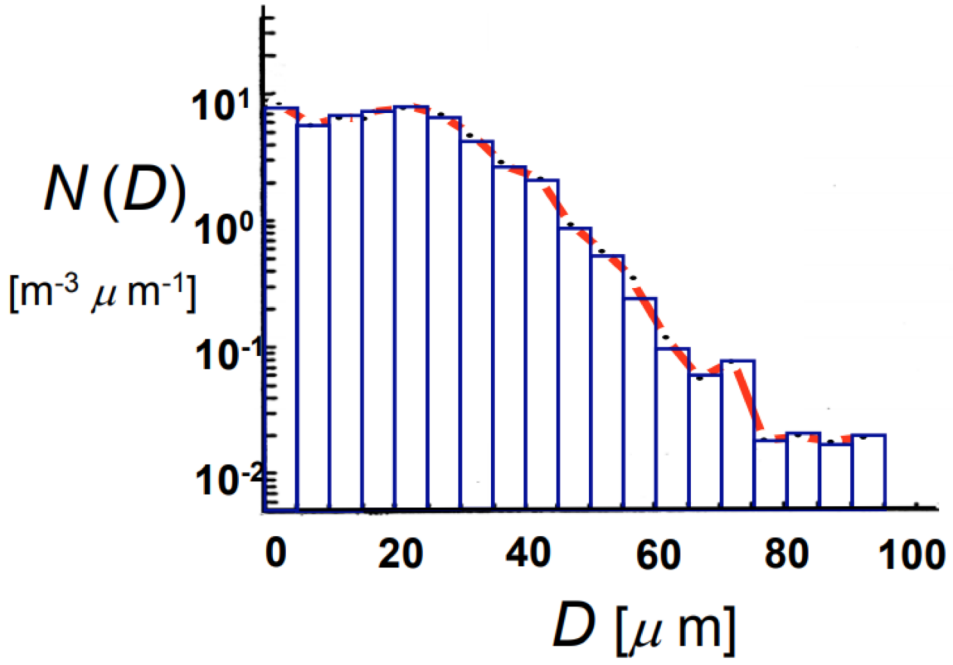
Bulk scheme (MG2 in CAM6):
Calculate with a semi-empirical particle size distribution (PSD). Gamma distribution often used.

Bin Scheme (Tel Aviv University (TAU) in CAM6):
Divide particle sizes into bins and calculate evolution of each bin separately. Better representation of interactions but much more computationally expensive.

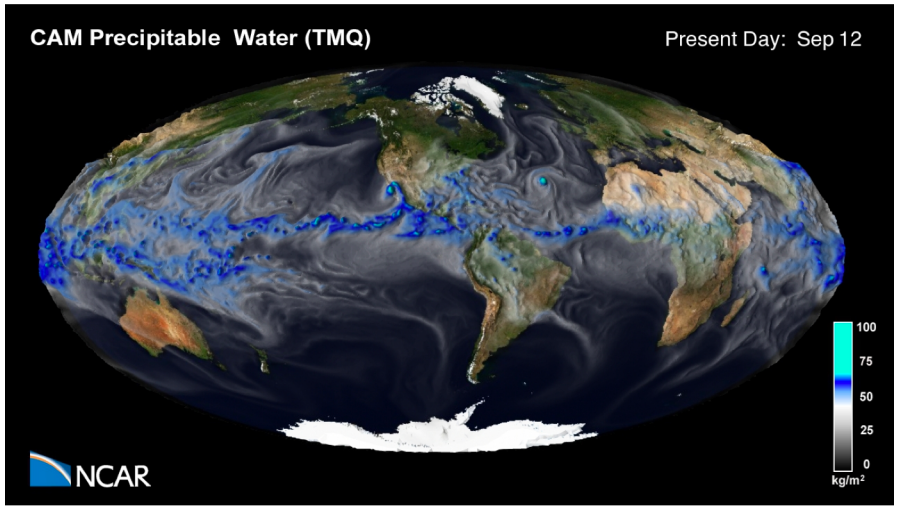
Bulk: $N(D) = N_0 D^{\alpha} e^{-\lambda D}$



Bin-resolving: (spectral) $N(D) = \sum_{i=1}^{I} N_i$

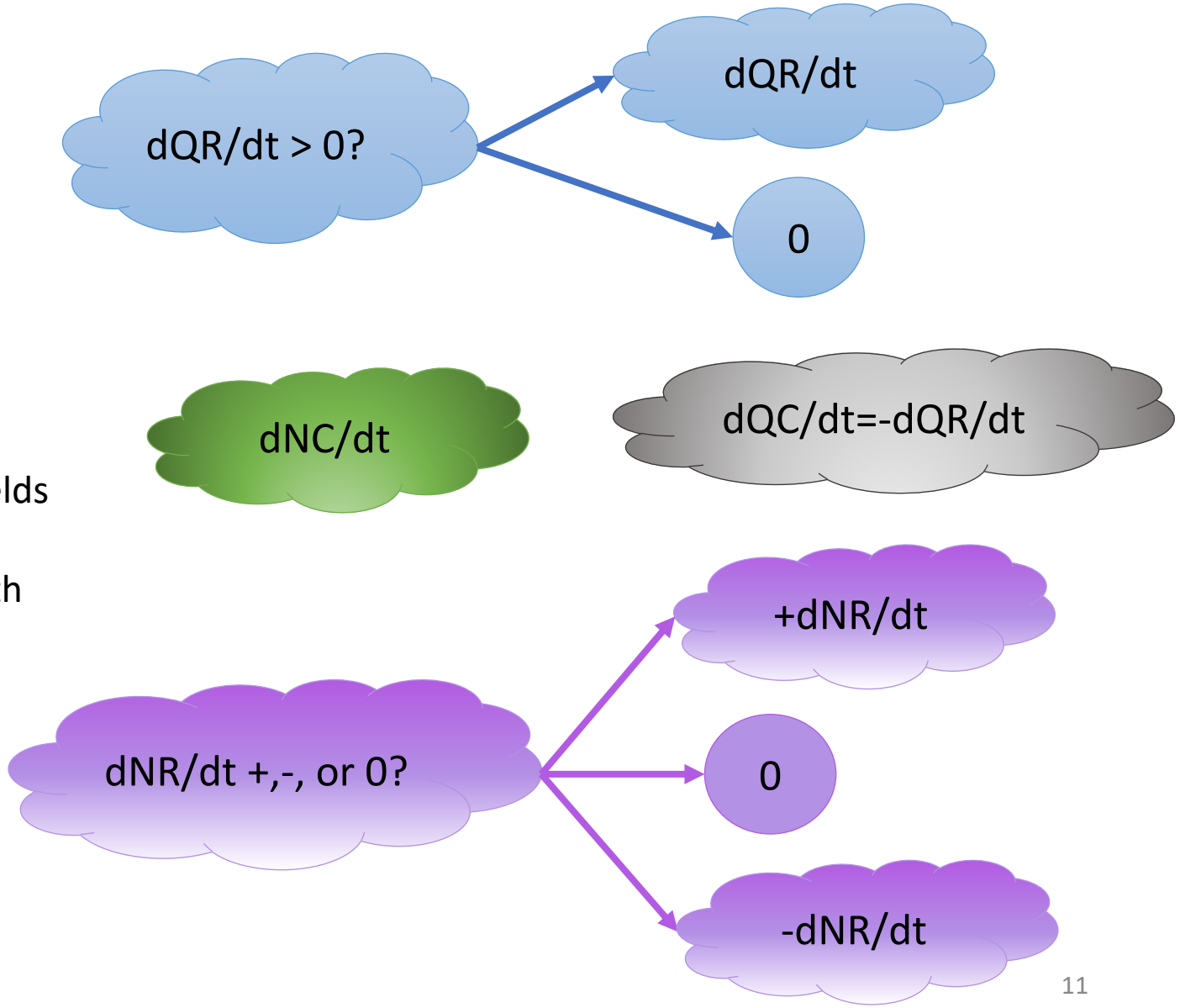


Microphysics Emulator Procedure

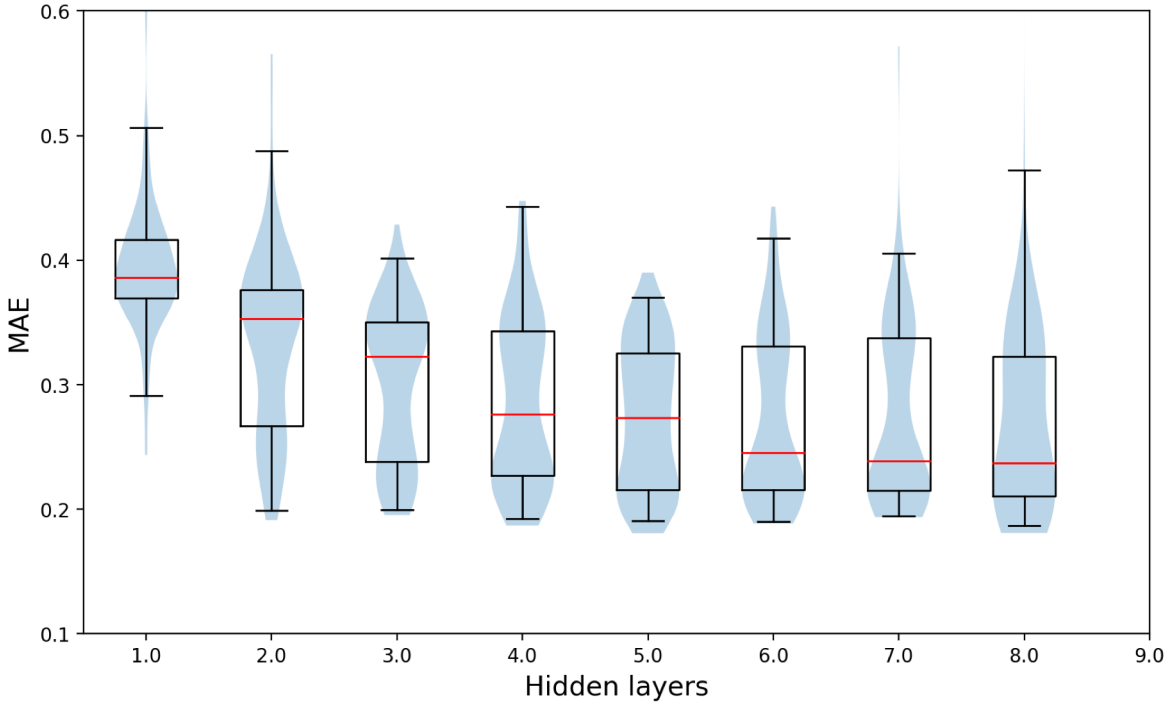


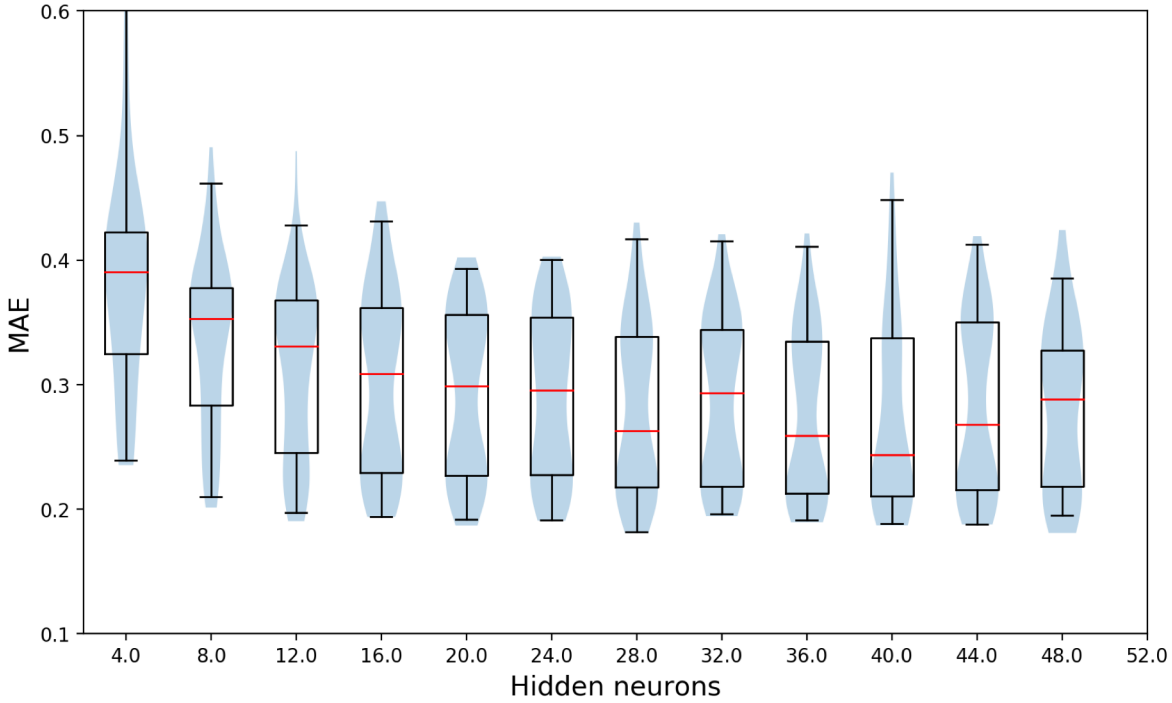
1. Run CESM2/CAM6 for 2 years
2. Output global microphysics input and output fields every 123 hours
3. Filter and subsample data to find grid points with realistic amounts of cloud water
4. Inputs:

Cloud water mixing ratio (QC)
Cloud water number concentration (NC)
Rain water mixing ratio (QR)
Rain water number concentration (NR)
Air density
Temperature



Random Hyperparameter Search Validation

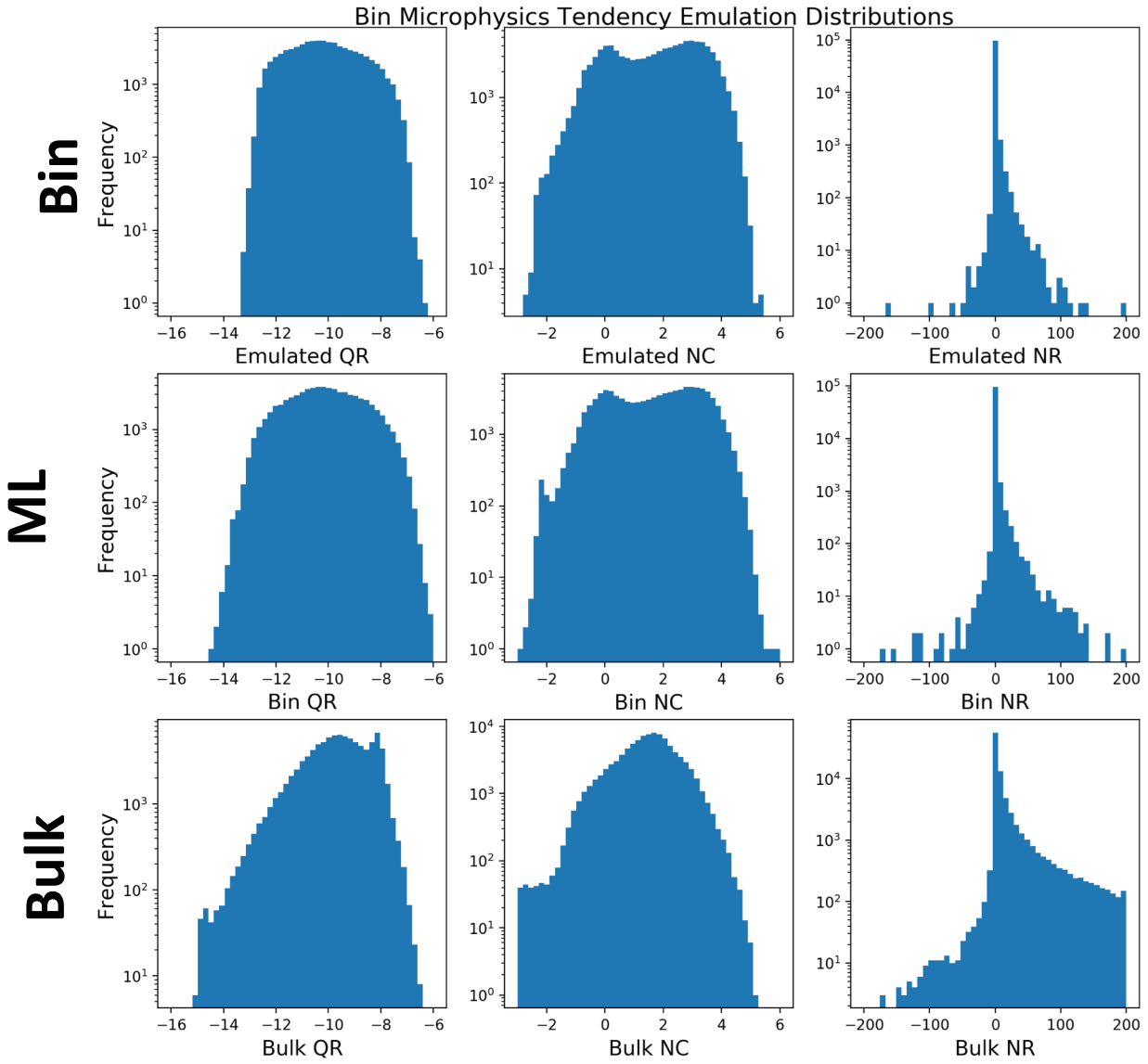


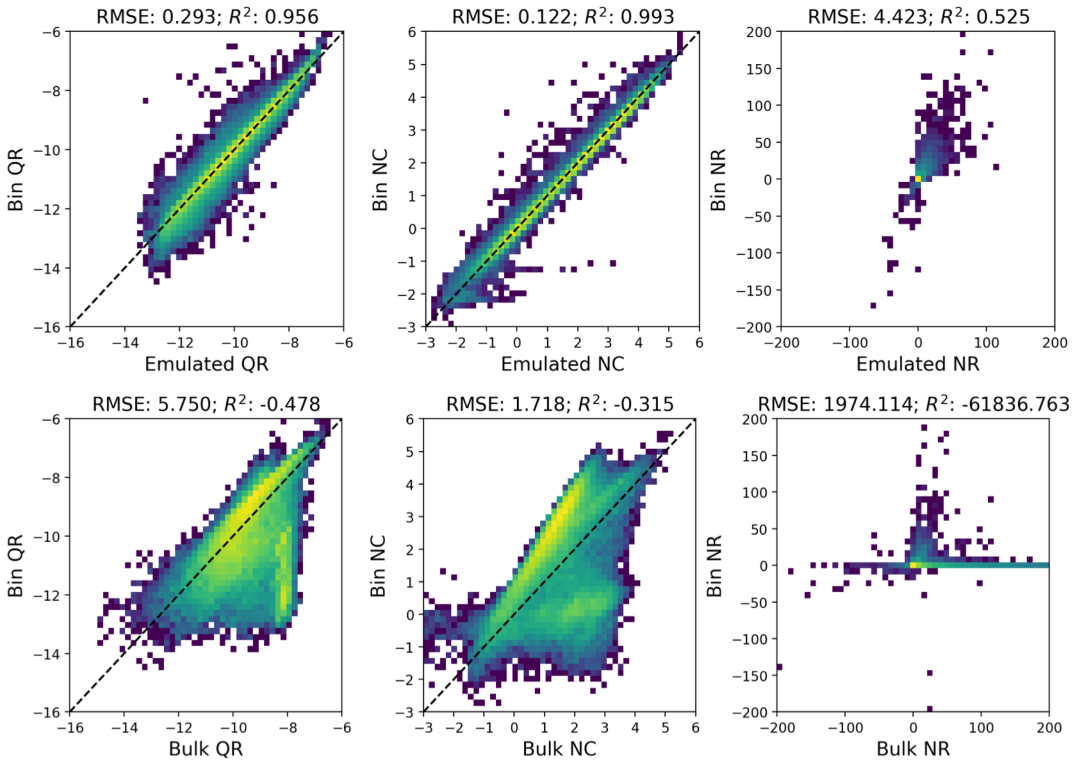


- Trained 1000 dense neural networks with random hyperparameter samples
- Trend of marginal hyperparameter distribution reveals influence

- Median performance levels off after 6 layers
- 20 neurons per layer appear to be sufficient to represent patterns in data

Microphysics Results





- Neural network microphysics emulates distribution and exact values of bin microphysics more closely than bulk microphysics

Partial Dependence Plots

1. Set all instances for one variable in a dataset to a single value

Temperature	Dewpoint	Pressure
15	10	986
15	14	1014
15	2	992
15	25	1025
15	6	950

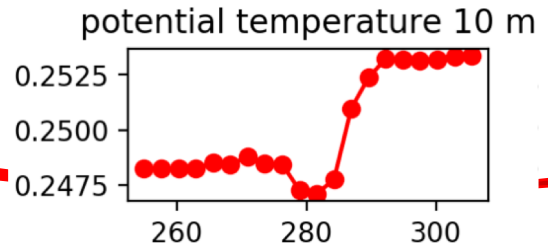
2. Feed fixed data through model

Machine Learning or Physical Model

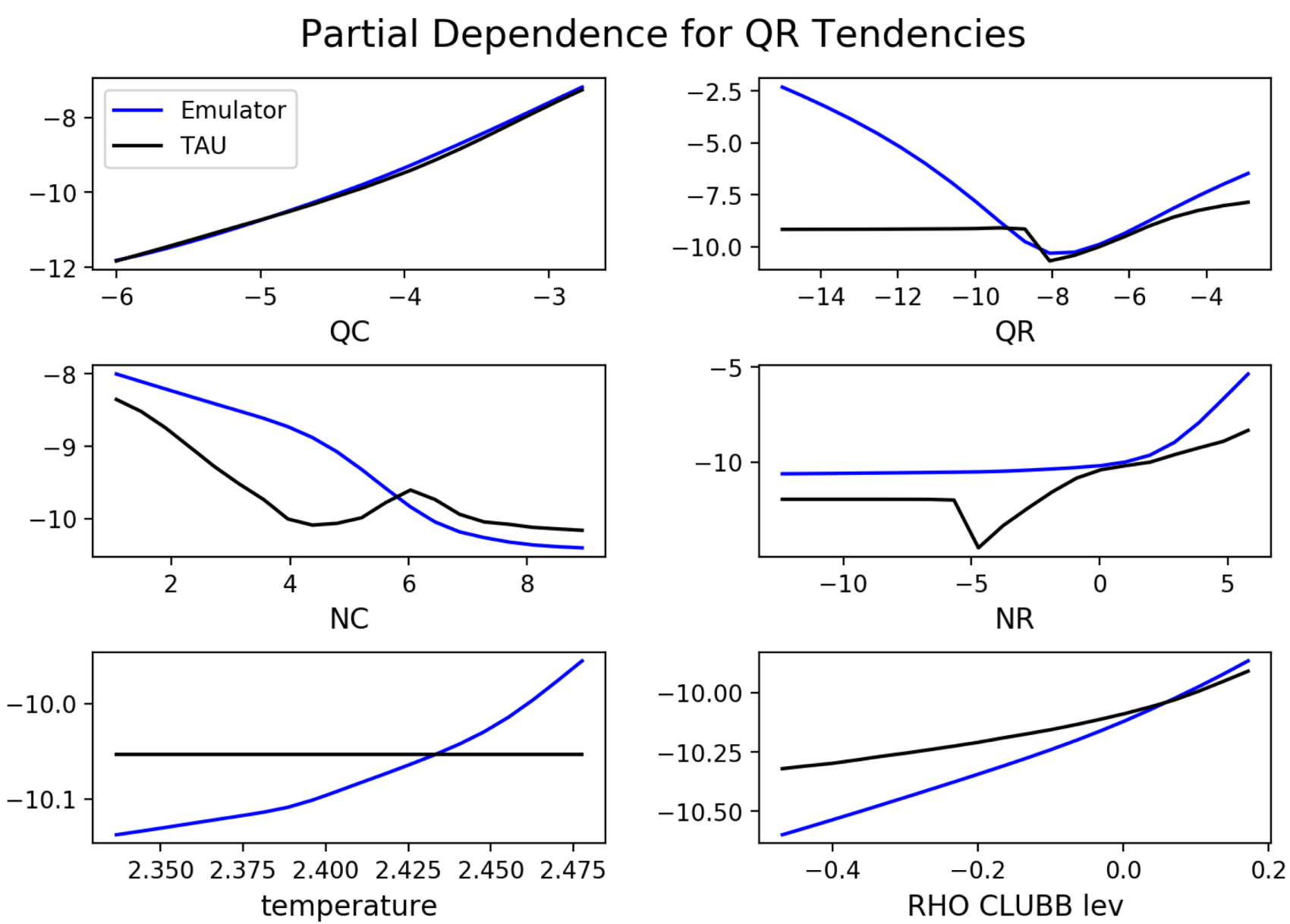
3. Calculate mean prediction for fixed value

Mean Prediction

4. Repeat process for range of input values

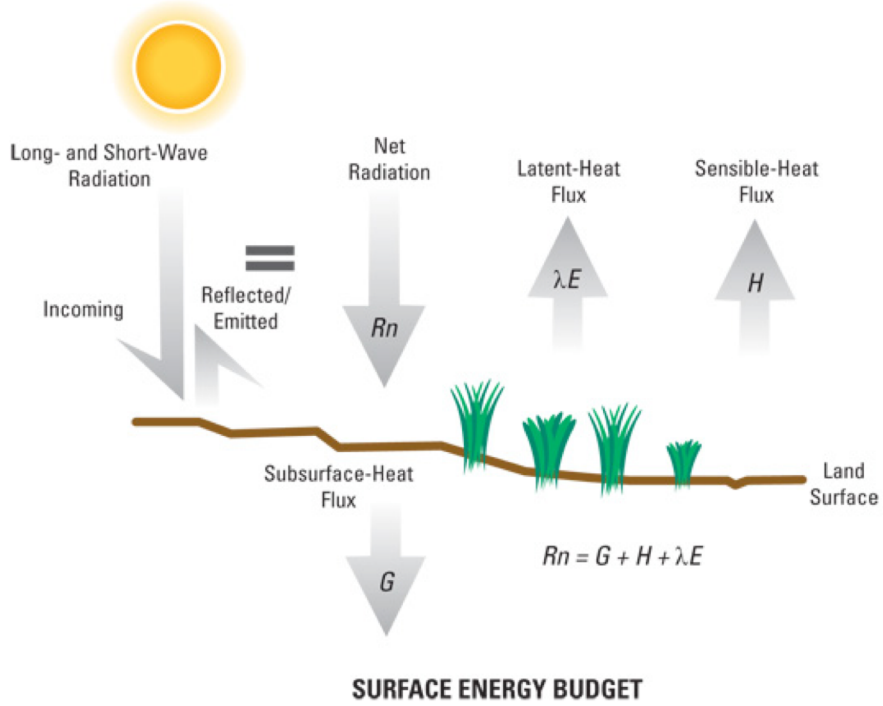


Microphysics Emulator Results

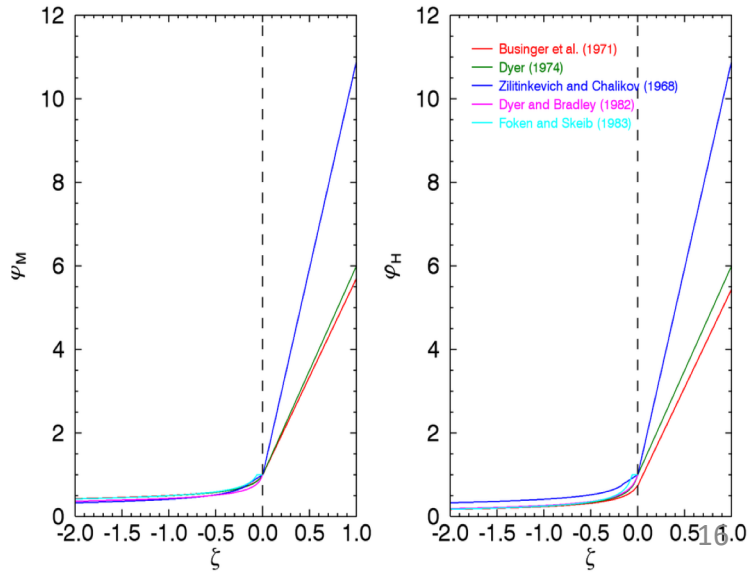


Surface Layer Parameterizations

- The energy transfer (flux) from the atmosphere to the land surface is modeled by the surface layer parameterization.
- The flux depends on gradients in wind speed, temperature and moisture between the lower-atmosphere and air just above the surface
- In atmospheric models Monin-Obukhov similarity theory is used to determine surface fluxes and stresses.
- Stability functions Φ_M (momentum) and Φ_H (heat) are determined experimentally.
- Stability functions come from field studies under nearly ideal atmospheric flow conditions characterized by horizontally homogeneous, flat terrain and stationarity. **However, the stability functions show a large amount of variation.**



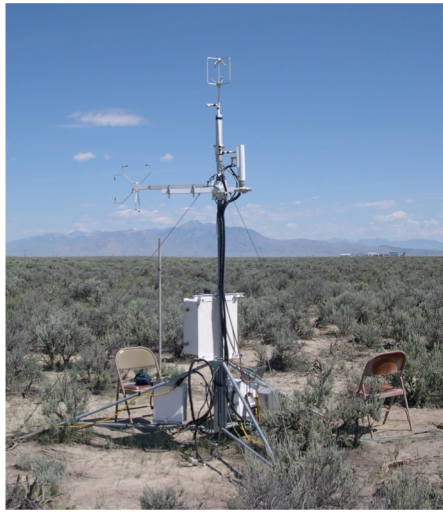
https://nevada.usgs.gov/et/measured.htm



Surface Layer Methods

- Regression is commonly used to estimate stability functions and thus relationship between surface stresses and fluxes and wind and temperature profiles.
- Instead, we use machine learning algorithms to develop models relating surface stresses and fluxes to wind and temperature profiles.
- Most of the previous field studies used to determine stability functions were process studies of episodic nature - a few months in length.
- To develop machine learning models we need long observational records.
- We have therefore selected two data sets that provide multiyear records:
  - KNMI-mast at Cabauw (Netherlands), 213 m tower, 2003 - 2017,
  - FDR tower near Scoville, Idaho – measurements from 2015 – 2017
- Fit random forest to each site to predict friction velocity and sensible heat flux

Cabauw



Idaho

Input and Output Variables

Common Variables	Heights (Idaho/Cabauw)
Potential Temperature (K)	2m, 10m, 15m/20m, 40/45m
Solar Radiation (w m-2)	Surface
Wind Speed (m/s)	10m, 15m/20m, 40/45m
Bulk Richardson number	10 m- 0 m
Pressure (hPa)	Surface
Relative Humidity (%)	2 m, 10 m
Obukhov Length (m)	
Moisture Availability (%)	5cm/3cm
Skin Temperature (K)	0 m
Solar Zenith Angle (degrees)	

Output equations

$$\tau = \rho u_*^2$$

$$H = -\rho c_p u_* \theta *$$

$$LH = L_e \rho u_* q_*$$

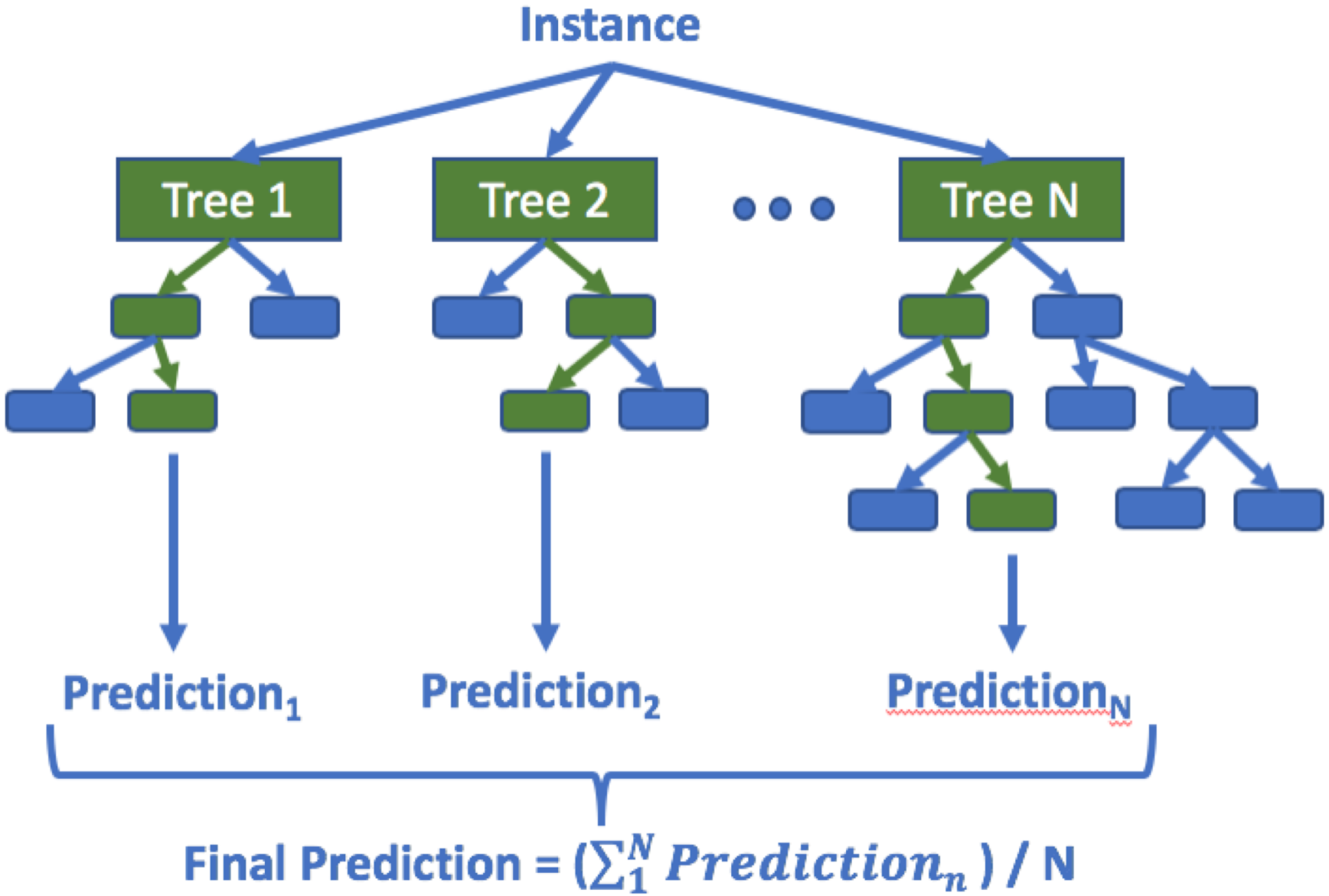
Predictands

u*=Friction velocity

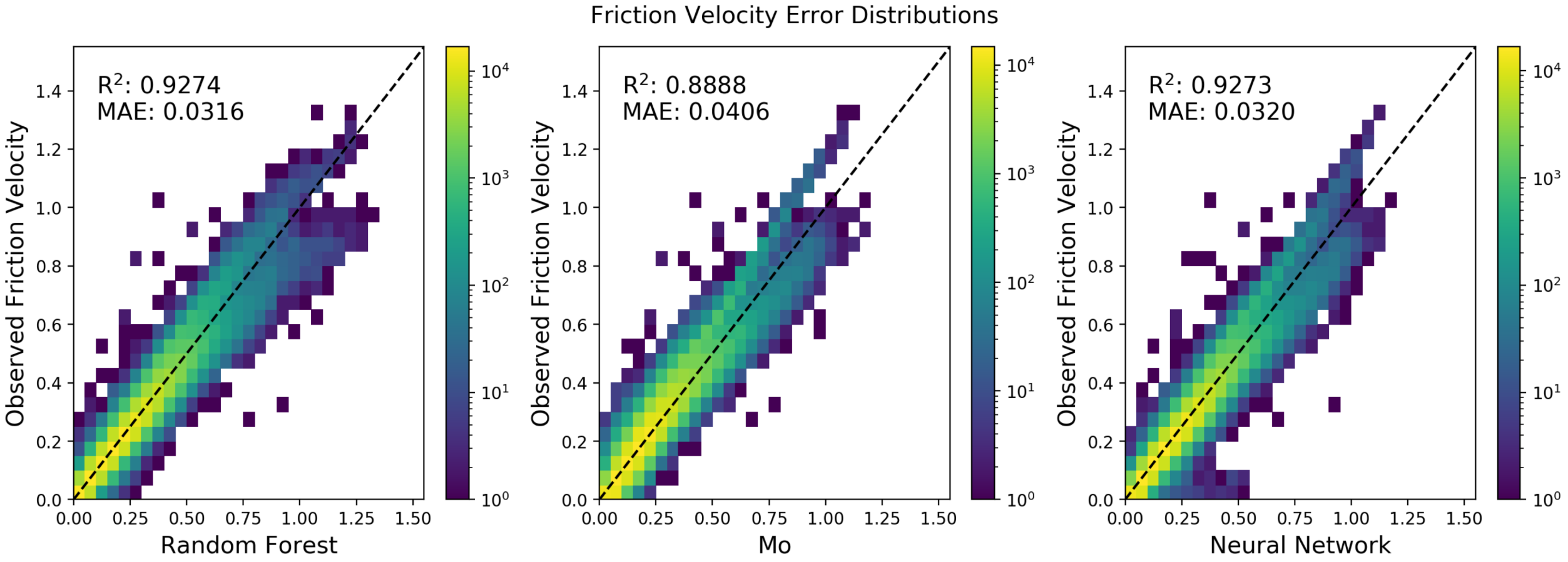
θ*=Temperature scale

q*=Moisture scale

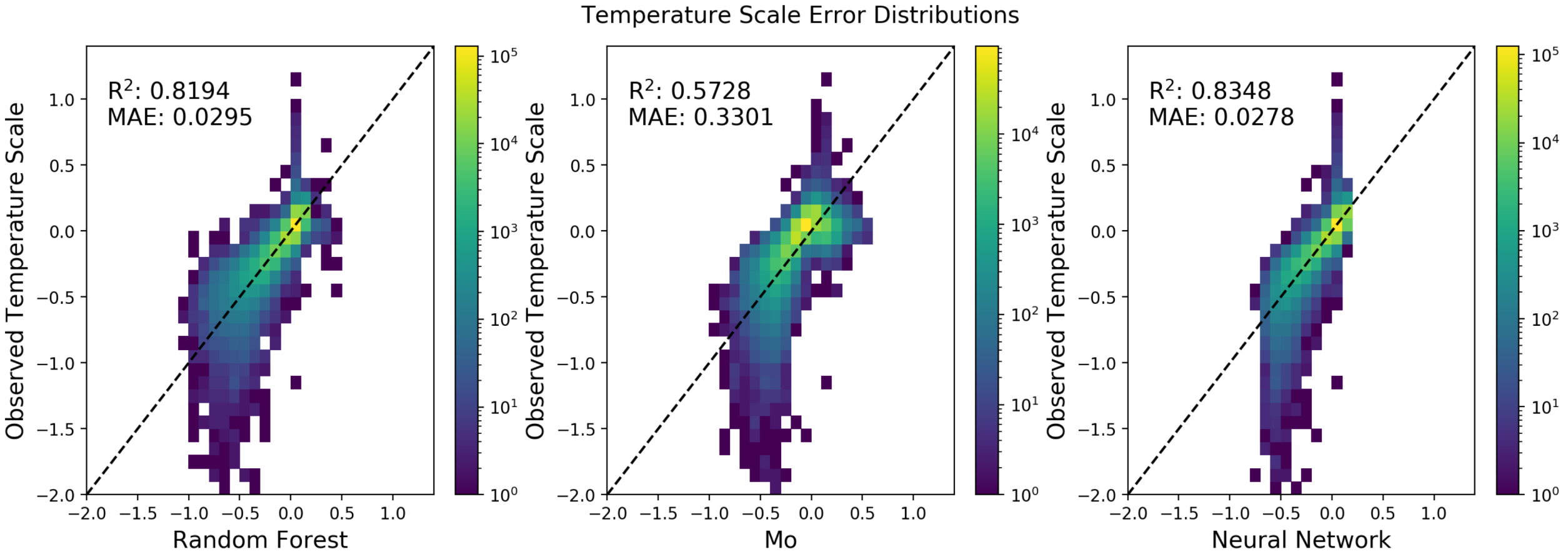
ML Model: Random Forest



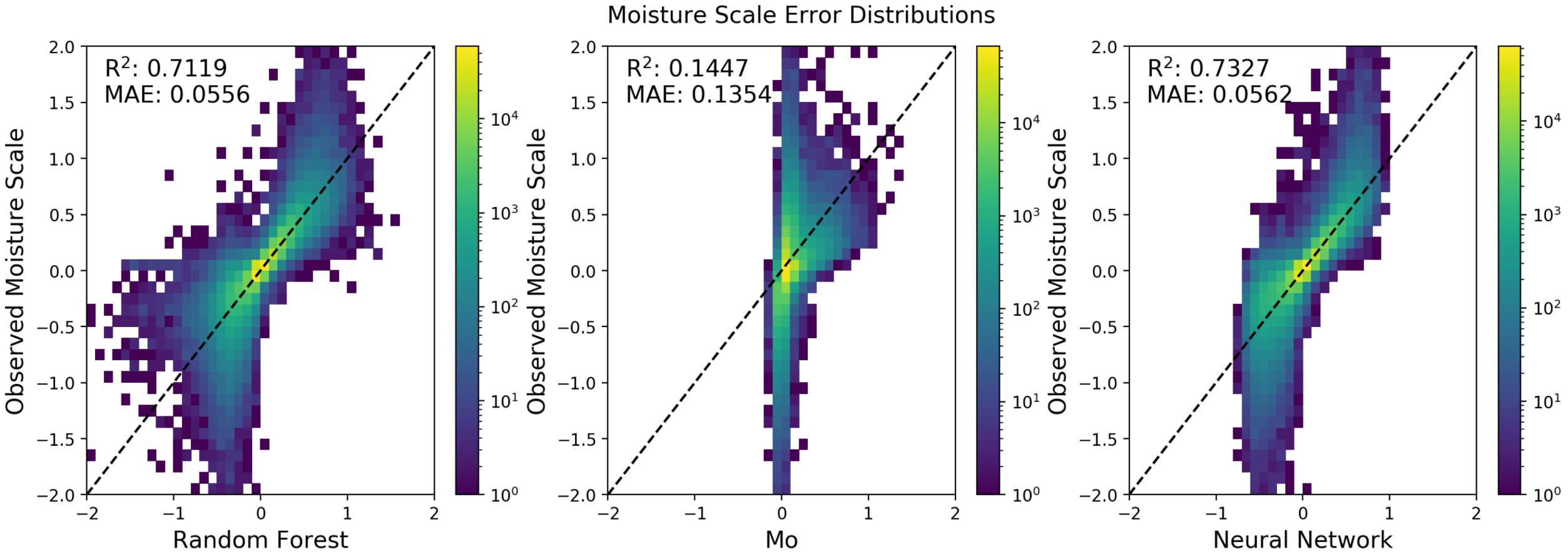
Surface Layer Results: Friction Velocity



Surface Layer Results: Friction Velocity



Surface Layer Results: Friction Velocity



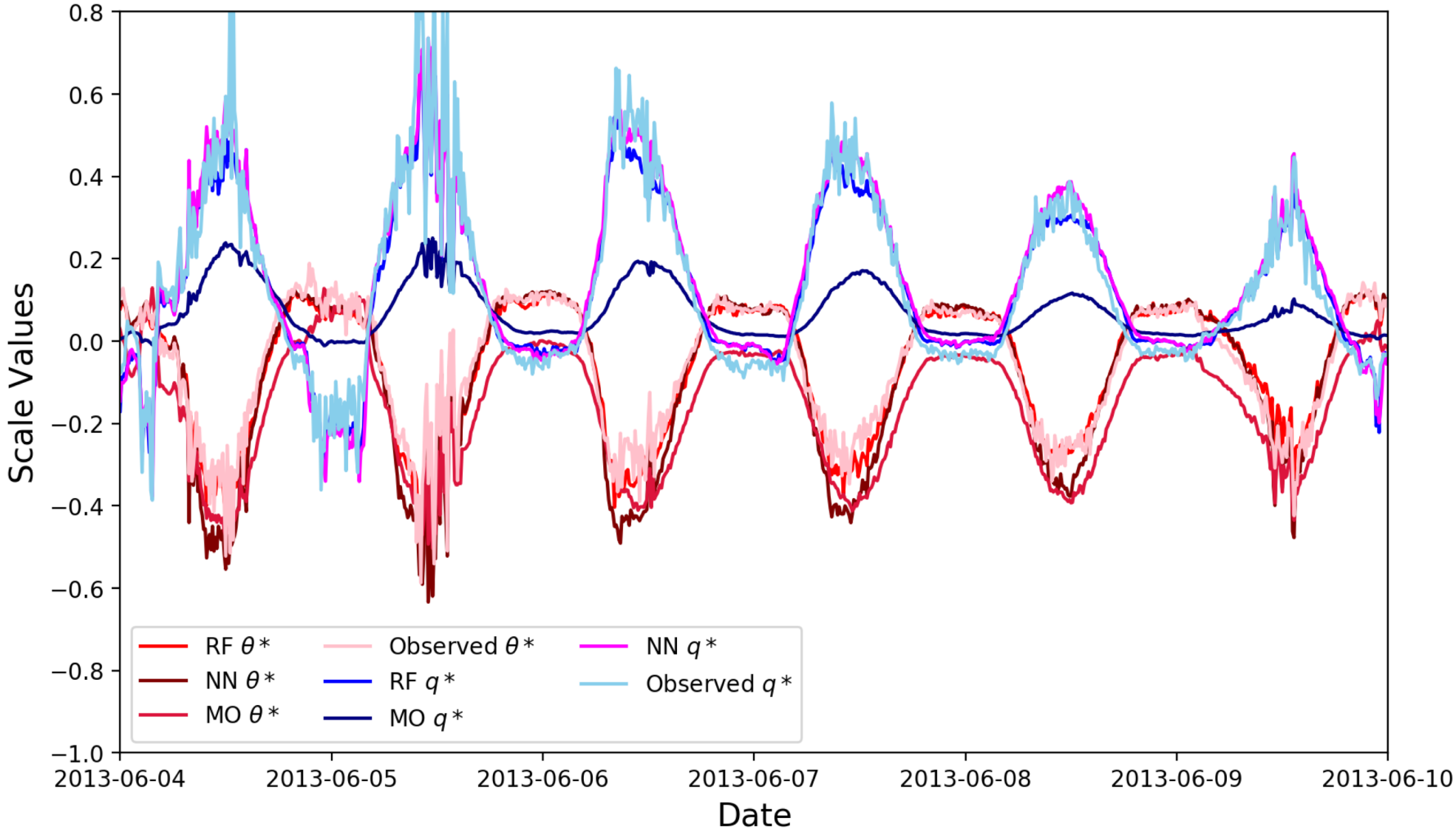
Cross-Training ML Models

	R^2			MAE		
Idaho Test Dataset	**Friction Velocity**	**Temperature Scale**	**Moisture Scale**	**Friction Velocity**	**Temperature Scale**	**Moisture Scale**
MO Similarity	0.85	0.42		0.077	0.203	
RF Trained on Idaho	0.91	0.80	0.41	0.047	0.079	0.023
RF Trained on Cabauw	0.88	0.76	0.22	0.094	0.139	0.284

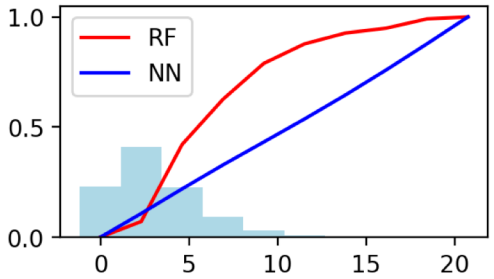
	R^2			MAE		
Cabauw Test Dataset	**Friction Velocity**	**Temperature Scale**	**Moisture Scale**	**Friction Velocity**	**Temperature Scale**	**Moisture Scale**
MO Similarity	0.90	0.44		0.115	0.062	
RF Trained on Cabauw	0.93	0.82	0.73	0.031	0.030	0.055
RF Trained on Idaho	0.90	0.77	0.49	0.074	0.049	0.112

Results Courtesy Tyler McCandless

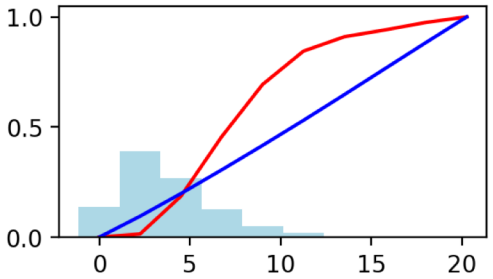
Temperature and Moisture Scale Diurnal Cycles



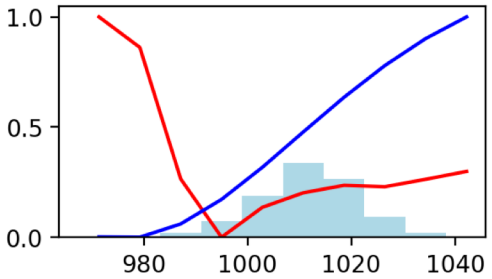
Friction Velocity Partial Dependence



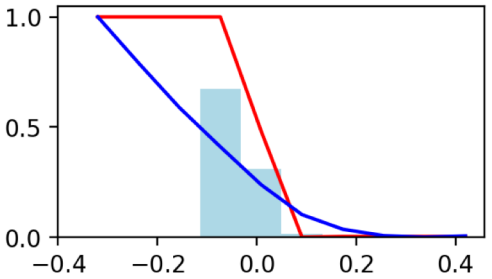
wind speed 10 m m s-1



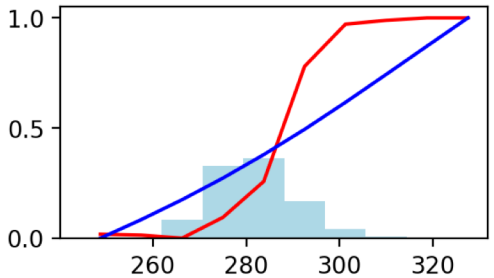
wind speed 20 m m s-1



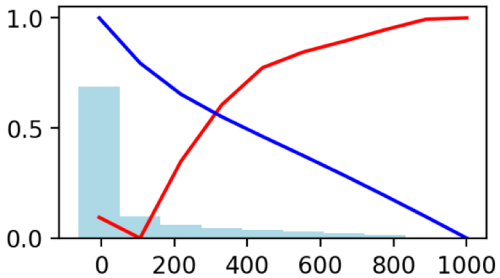
pressure 2 m hPa



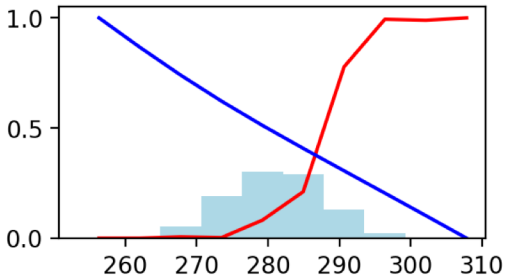
temperature change 15 m K m-1



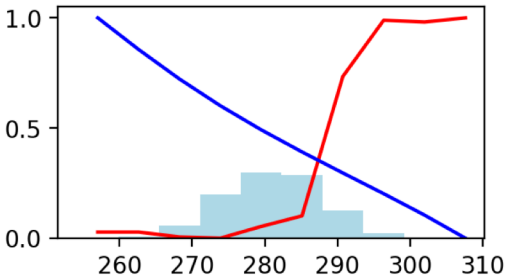
skin virtual potential temperature 0 m K



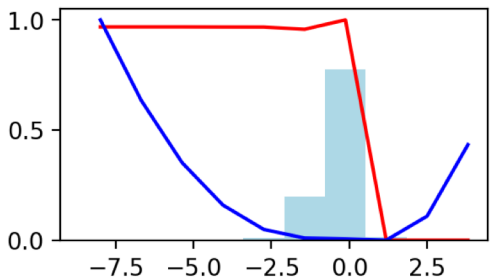
global horizontal irradiance 0 m W m-2



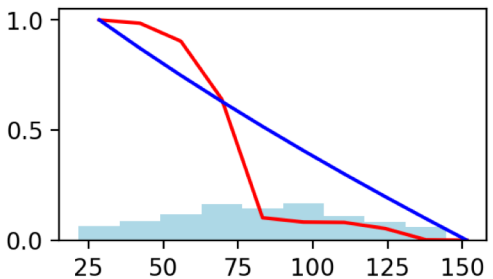
virtual potential temperature 10 m K



virtual potential temperature 20 m K

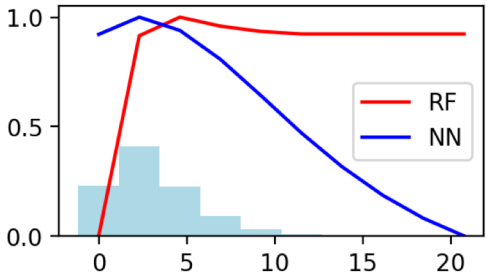


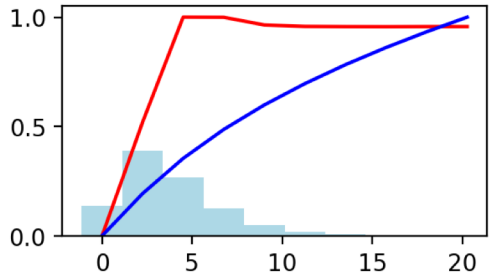
bulk richardson 10 m

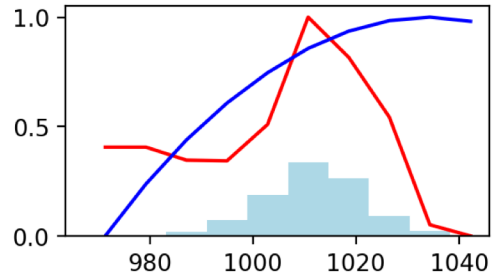


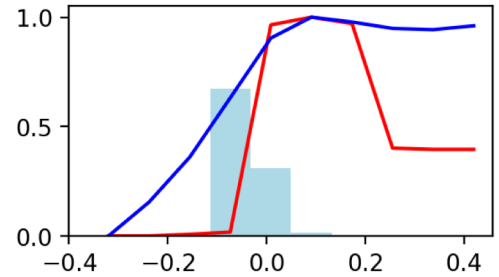
zenith 0 m degrees

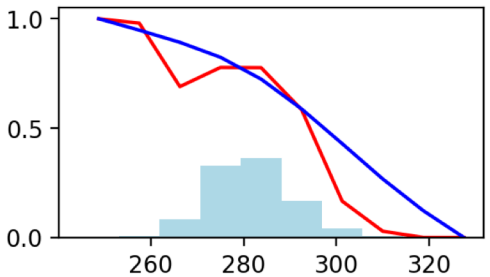
Temperature Scale Partial Dependence

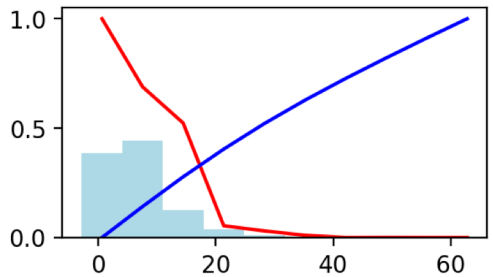


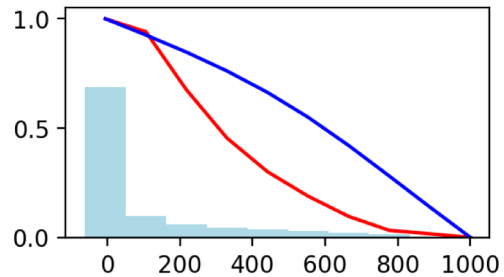


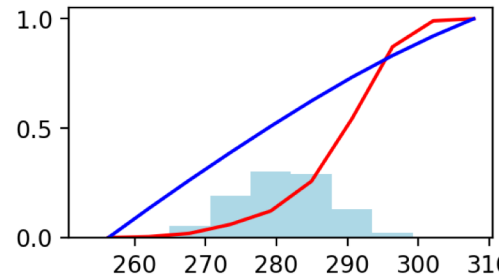


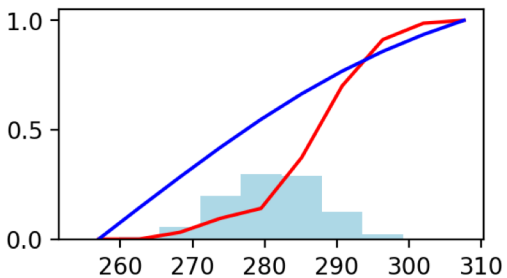


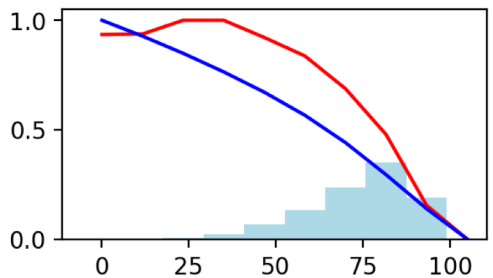


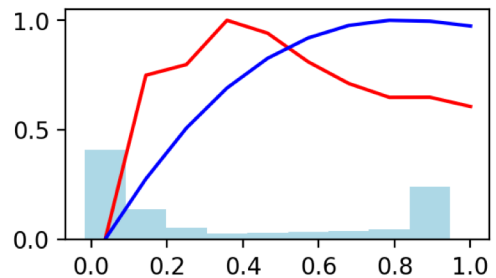


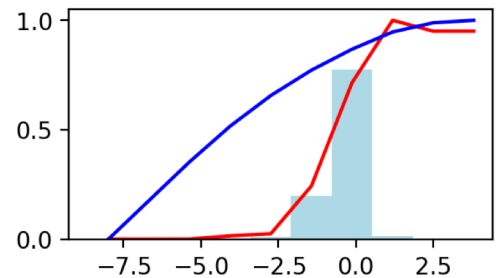


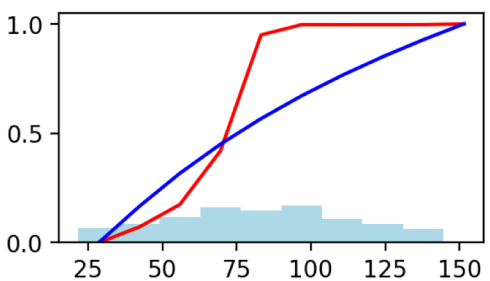




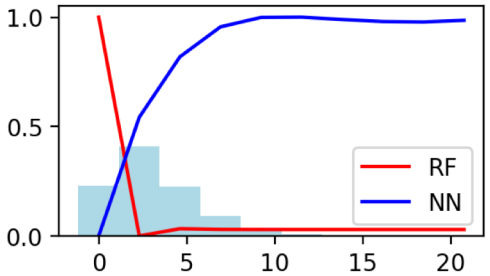




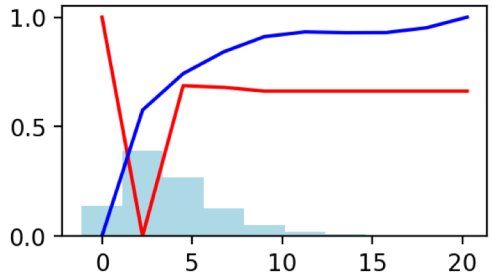


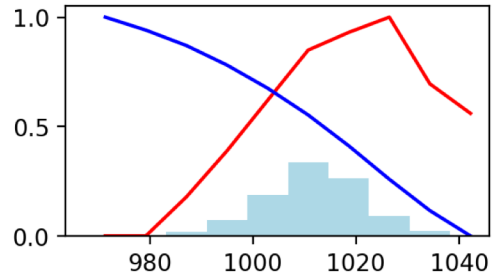


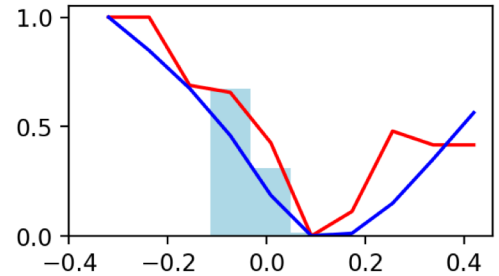
Moisture Scale Partial Dependence

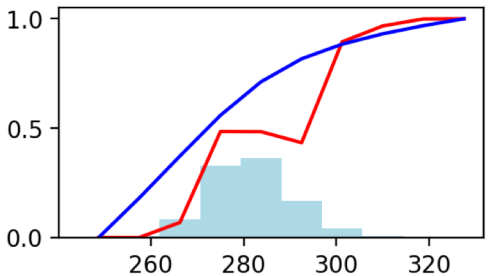


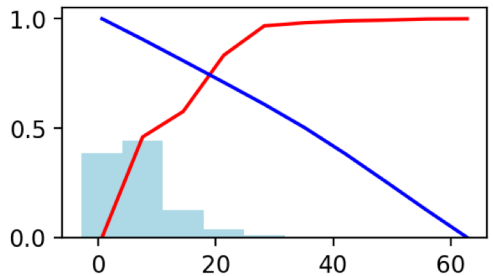
wind speed 10 m m s-1

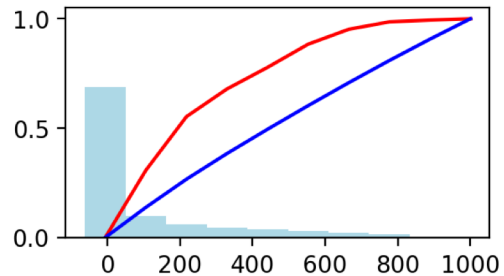

wind speed 20 m m s-1

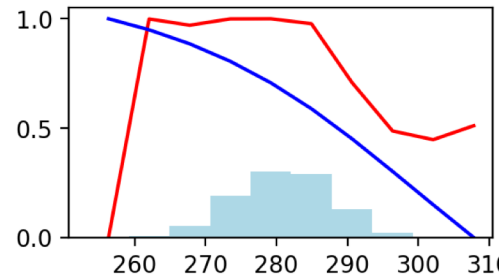

pressure 2 m hPa

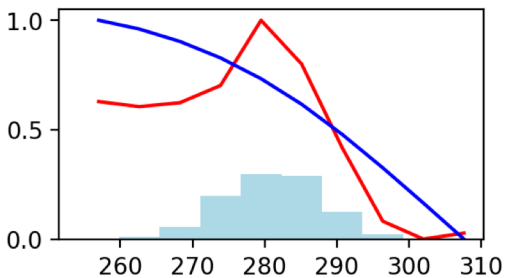

temperature change 15 m K m-1

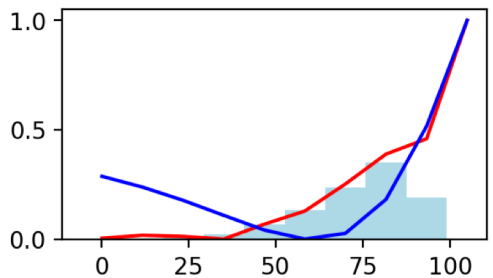

skin virtual potential temperature 0 m K

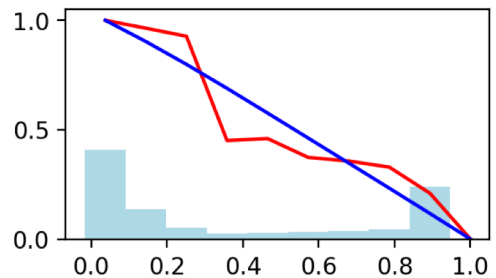

skin saturation mixing ratio 0 m g kg-1

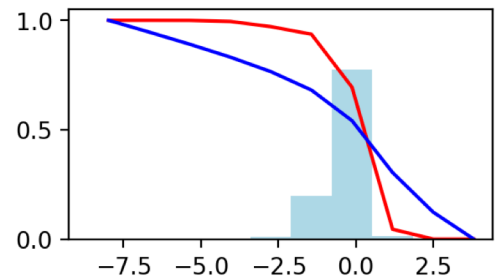

global horizontal irradiance 0 m W m-2

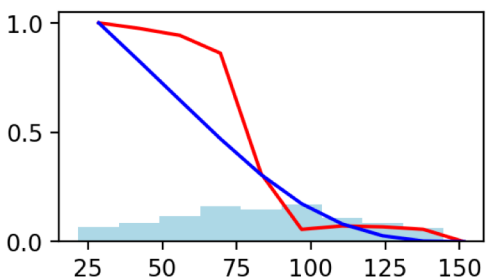

virtual potential temperature 10 m K

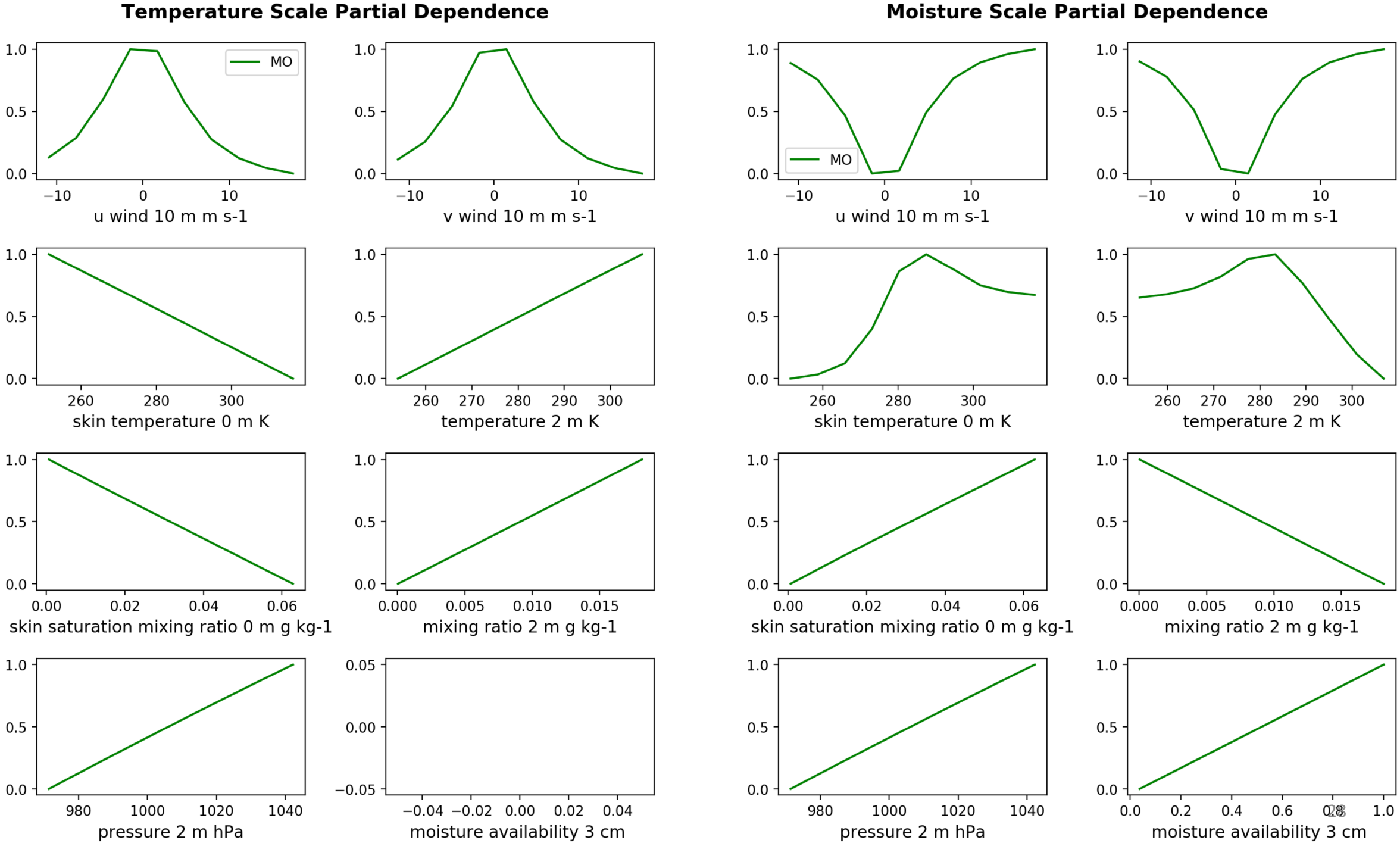

virtual potential temperature 20 m K


relative humidity 10 m %


moisture availability 3 cm


bulk richardson 10 m


zenith 0 m degrees


Temperature Scale Partial Dependence
Moisture Scale Partial Dependence
MO
u wind 10 m m s-1
v wind 10 m m s-1
skin temperature 0 m K
temperature 2 m K
skin saturation mixing ratio 0 m g kg-1
mixing ratio 2 m g kg-1
pressure 2 m hPa
moisture availability 3 cm

ML Integration with NWP Models

- Problem: Atmospheric models are written in Fortran, but the ML model codes are written in Python
- Solution: Fortran neural network and random forest inference subroutines!
- Subroutine Contents
  - Calculate derived input variables
  - Feed inputs into ML models
  - Calculate diagnostics from ML output
- Advantages
  - No outside library dependencies
  - ML models can be switched out easily when more data are available
- Disadvantage
  - More limited ML functionality/optimization compared with community ML models

```
type decision_tree
  integer :: nodes
  integer, allocatable :: node(:)
  integer, allocatable :: feature(:)
  real(kind=8), allocatable :: threshold(:)
  real(kind=8), allocatable :: tvalue(:)
  integer, allocatable :: children_left(:)
  integer, allocatable :: children_right(:)
  real(kind=8), allocatable :: impurity(:)
end type decision_tree
```

Role of GPUs in ML Parameterization

- Most operational numerical weather and climate models run entirely on the CPU on CPU-only supercomputers
- ML models running on these systems have to be able to operate efficiently on the CPU
- However, new atmospheric modeling frameworks can be accelerated with GPUs (MPAS) or run entirely on the GPU (FastEddy)
- ML parameterizations can utilize GPU integration to train and run more complex ML models within the NWP model

Summary

- Machine learning models applied to all parts of the NWP model pipeline can help improve prediction accuracy or computational speed
- A set of neural networks can closely emulate bin microphysical processes and generally match their sensitivities
- Random forests and neural networks can estimate surface layer fluxes better than the current Monin-Obukhov Similarity Theory Approach

Acknowledgements

Collaborators: Sue Ellen Haupt, Rich Loft, Andrew Gettelman, Jack Chen, Negin Sobhani, Tyler McCandless, Branko Kosovic, The National Center for Atmospheric Research is sponsored by the National Science Foundation.

Contact Information

Email: dgagne@ucar.edu

Twitter: @DJGagneDos

Github: djgagne