Accelerating and Explaining Earth System Process Models with Machine Learning

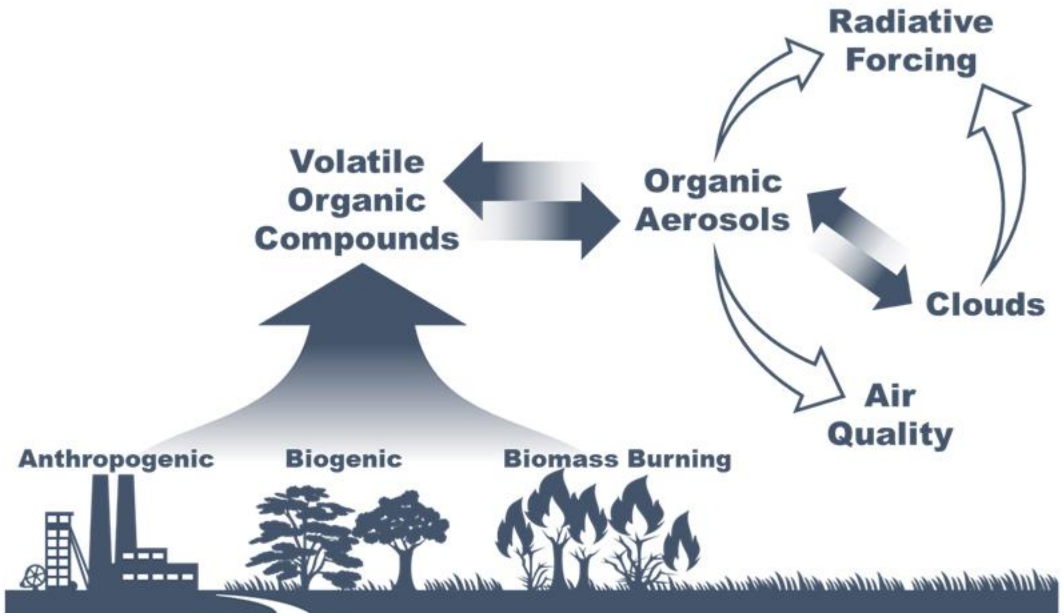
David John Gagne
National Center for Atmospheric Research
Boulder, CO USA

October 5, 2020

This material is based upon work supported by the National Center for Atmospheric Research, which is a major facility sponsored by the National Science Foundation under Cooperative Agreement No. 1852977.

Motivation

- Models of small particles in the atmosphere can model bulk properties or small-scale interactions
- Interaction models produce significantly different results from bulk counterparts but are too computationally expensive to run within weather/climate simulations
- Machine learning emulators trained on limited runs from the complex models can approximate them at a far smaller computational cost.
- **Goals**
  – **Develop machine learning emulators for**



Machine Learning the Warm Rain Process

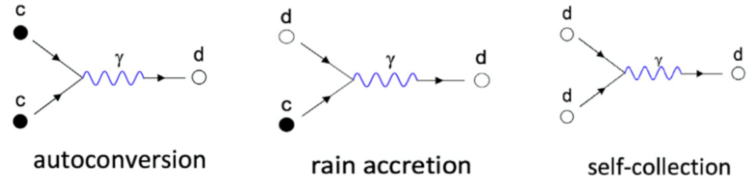
Andrew Gettleman, David John Gagne, Chih-Chieh Chen, Matthew Christensen, Zachary Lebo, Hugh Morrison, Gabrielle Gantos

Available at https://www.essoar.org/doi/abs/10.1002/essoar.10503868.1

Microphysics Emulation: 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.

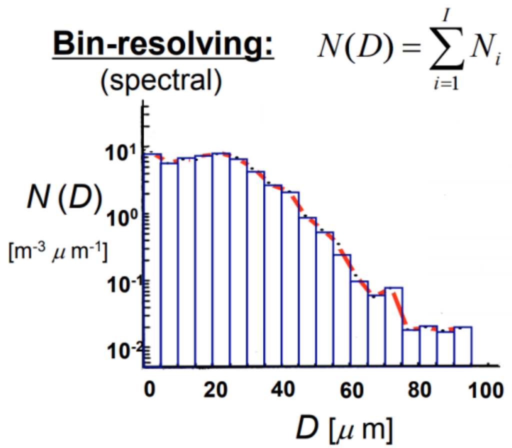


Detailed bin codes are too expensive for large scale models, so empirical functions are used instead.

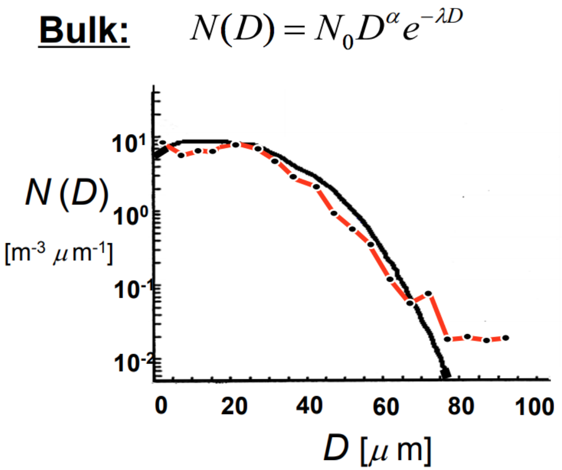
Can a machine learning approach provide a more accurate emulation of precipitation formation processes without a significant increase in computation?

Goal: Put a detailed bin process into a global general circulation model and emulate it using ML techniques.

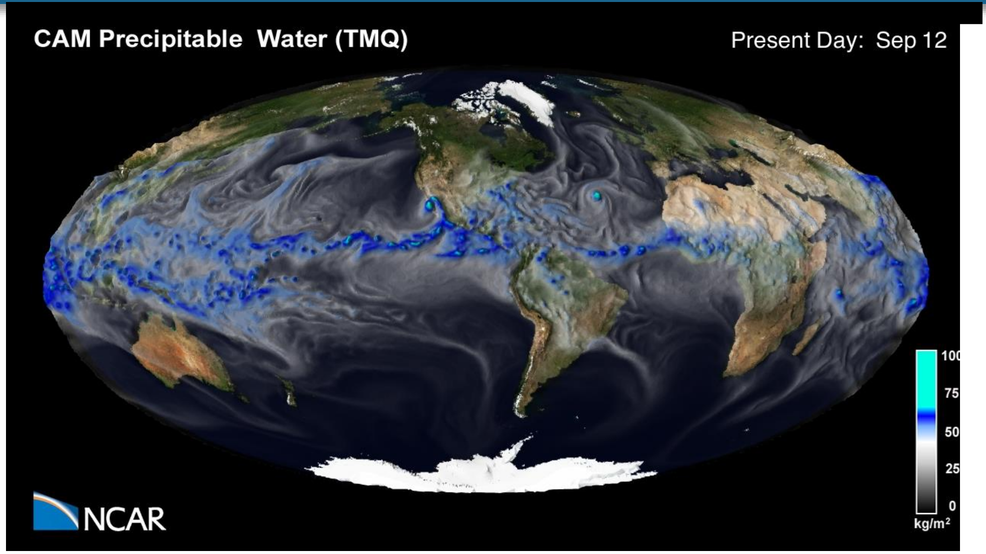
Bin Scheme (Tel Aviv University (TAU) in CAM6): Divide particle sizes into bins and calculate evolution of each bin separately.



Bulk (MG2 in CAM6): Calculate warm rain formation processes with a semi-empirical particle size distribution (PSD) based on exponential fit to LES bin microphysics runs.

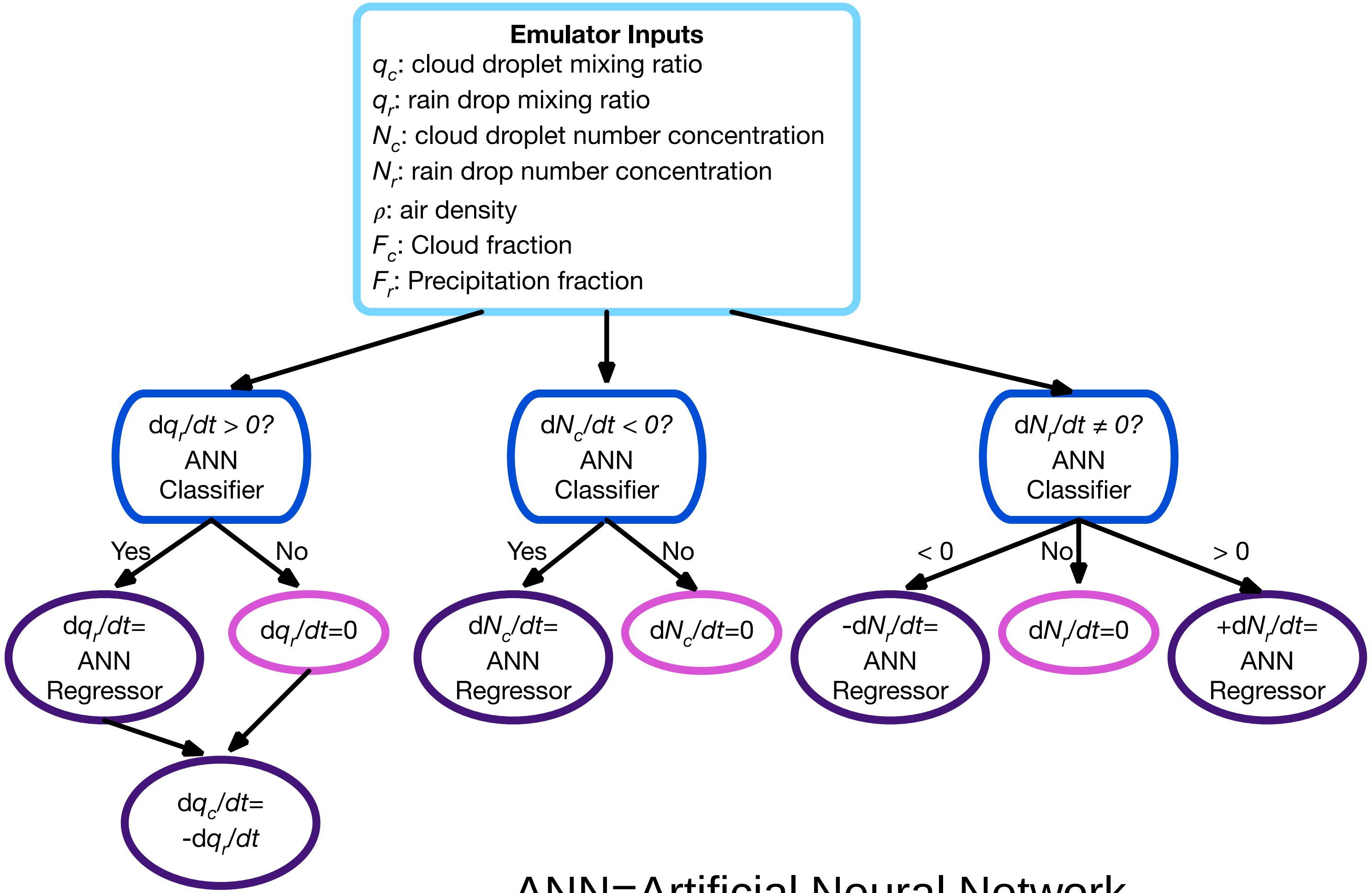


Microphysics Emulator Procedure



Data Generation

1. Run CAM6 for 2 years with fixed forcing from other CESM components
2. Output global microphysics input and output fields every 25 hours
3. Identify grid cells with non-negligible cloud and rain water mixing ratios
4. Save filtered data to csv files
5. Logarithmic transform and normalize input and tendency fields



ANN=Artificial Neural Network

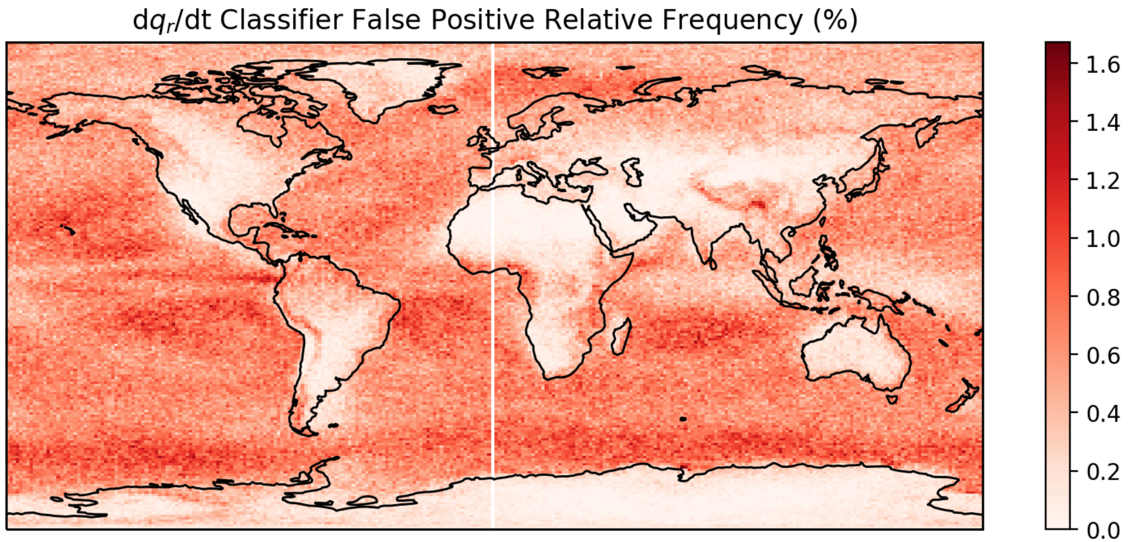
Classifier Results

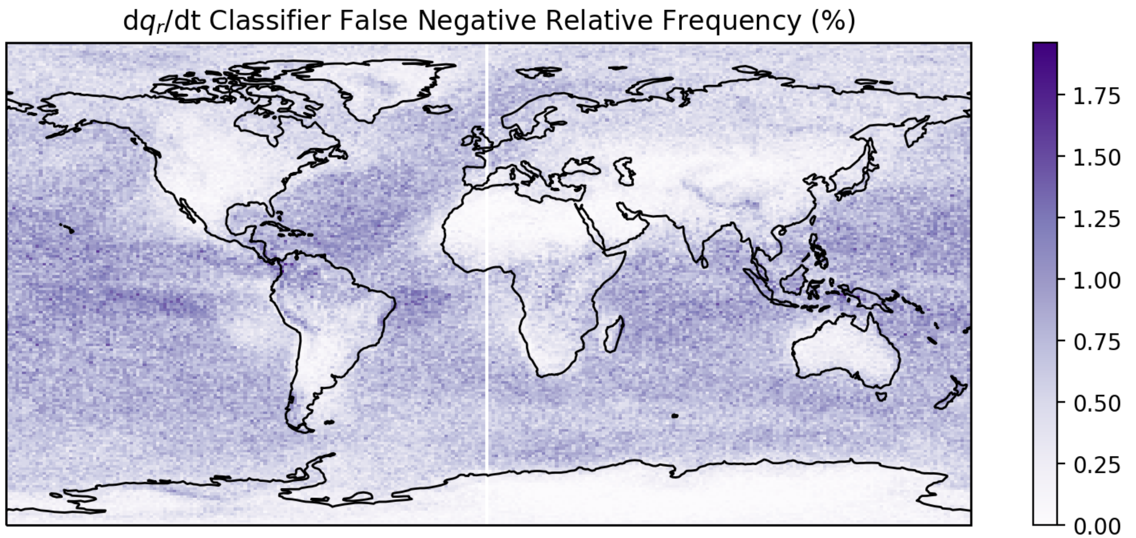
Classifier Results

	TAU QR 1	TAU QR 0	Total
NN QR 1	**41.7%**	0.7%	42.4%
NN QR 0	0.8%	**56.8%**	57.6%
Total	42.5%	57.5%	**98.4%**

	TAU NC 1	TAU NC 0	Total
NN NC 1	**52.9%**	0.5%	53.4%
NN NC 0	0.2%	**46.3%**	46.5%
Total	53.1%	46.8%	**99.3%**

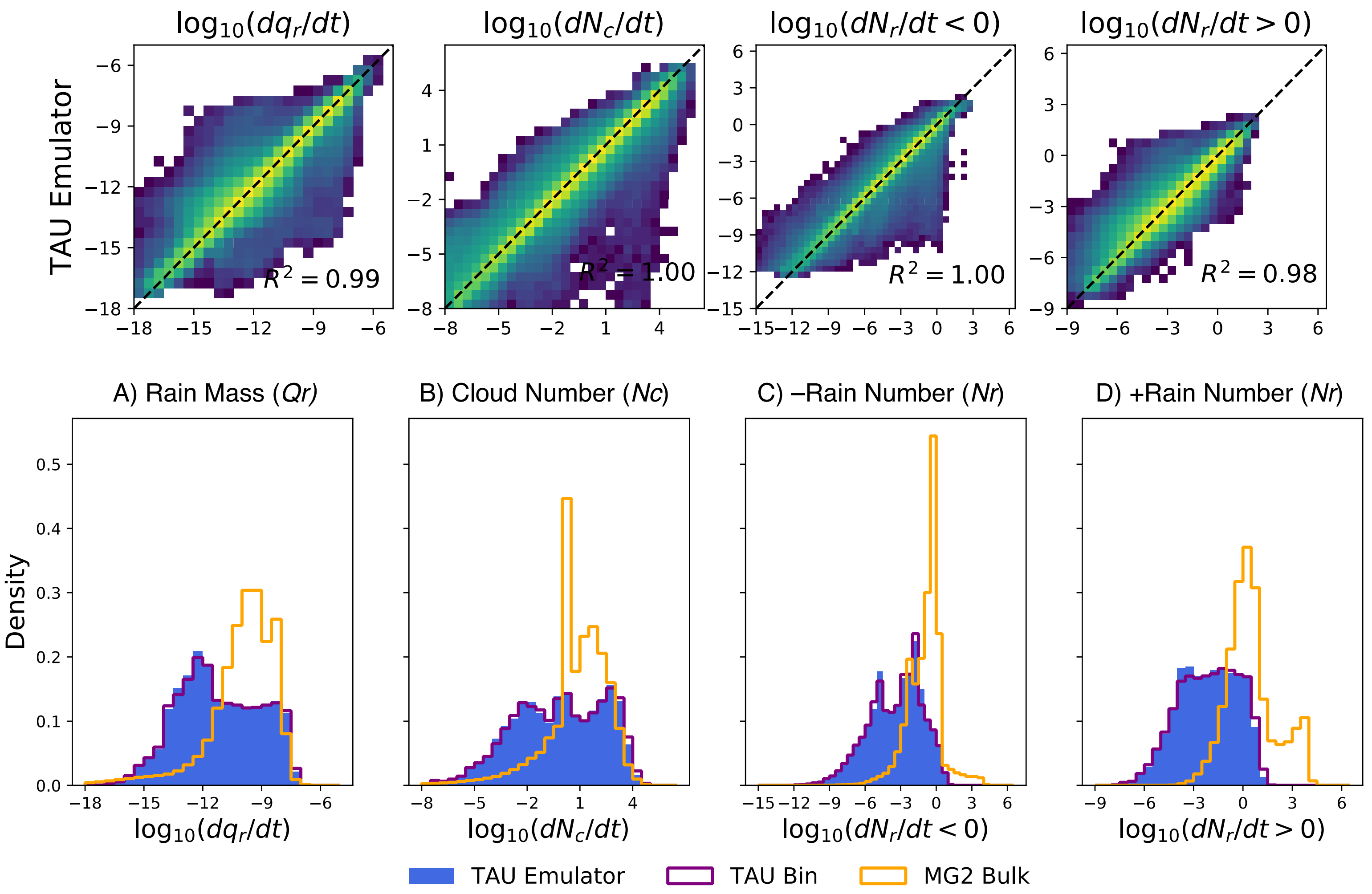
	TAU NR -1	TAU NC 0	TAU NR 1	Total
NN NR -1	**35%**	0.0%	0.4%	35.4%
NN NR 0	0.1%	**43.1%**	0.3%	43.5%
NN NR 1	0.2%	0.5%	**20.4%**	21.1%
Total	35.3%	43.6%	21.1%	**98.5%**





Regressor Results

Output	R^2	MAE	Hellinger
dq_r/dt	0.991	0.095	4.53e-4
dn_c/dt	0.995	0.112	1.49e-3
$dn_r/dt < 0$	0.995	0.081	6.04e-4
$dn_r/dt > 0$	0.978	0.178	1.18e-3

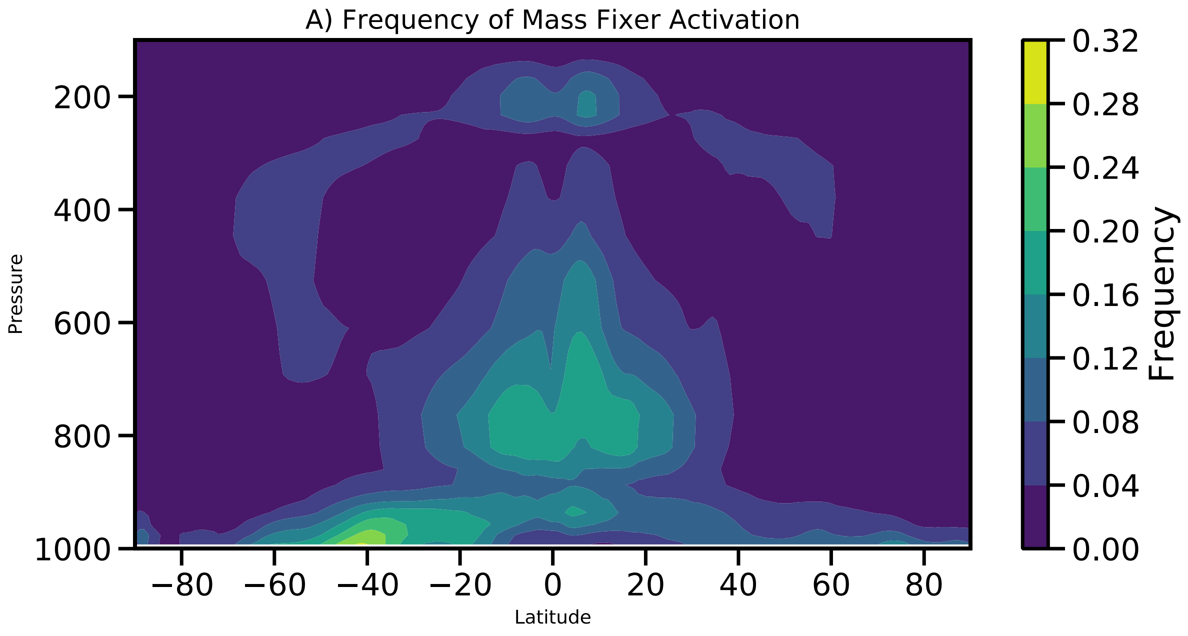


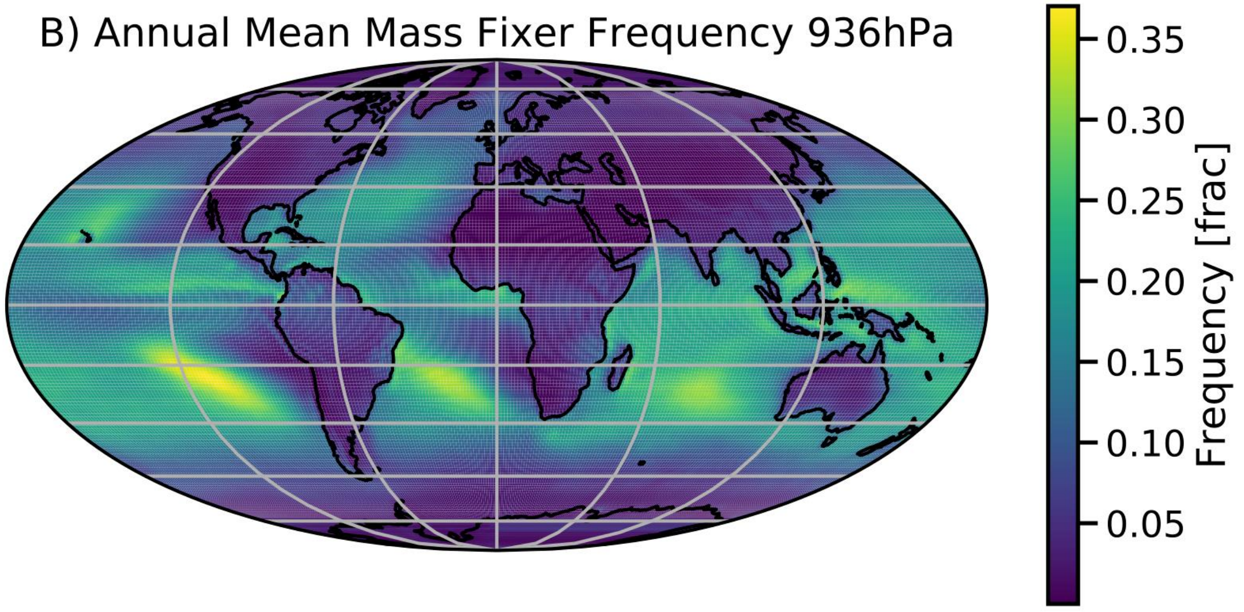
Simulations

- CAM6: Control
- TAU or TAU-bin: Stochastic Collection Kernel
- TAU-ML: Machine learning Emulator for TAU code

- For each, global 0.9°x1.25° simulation, 9 years, 1st year high frequency instantaneous output
  - Base (2000 Climatology)
  - Pre-Industrial (1850) aerosols. (For aerosol cloud interactions)
  - SST+4K (For Cloud Feedbacks)

Mass Fixer for ML Emulator

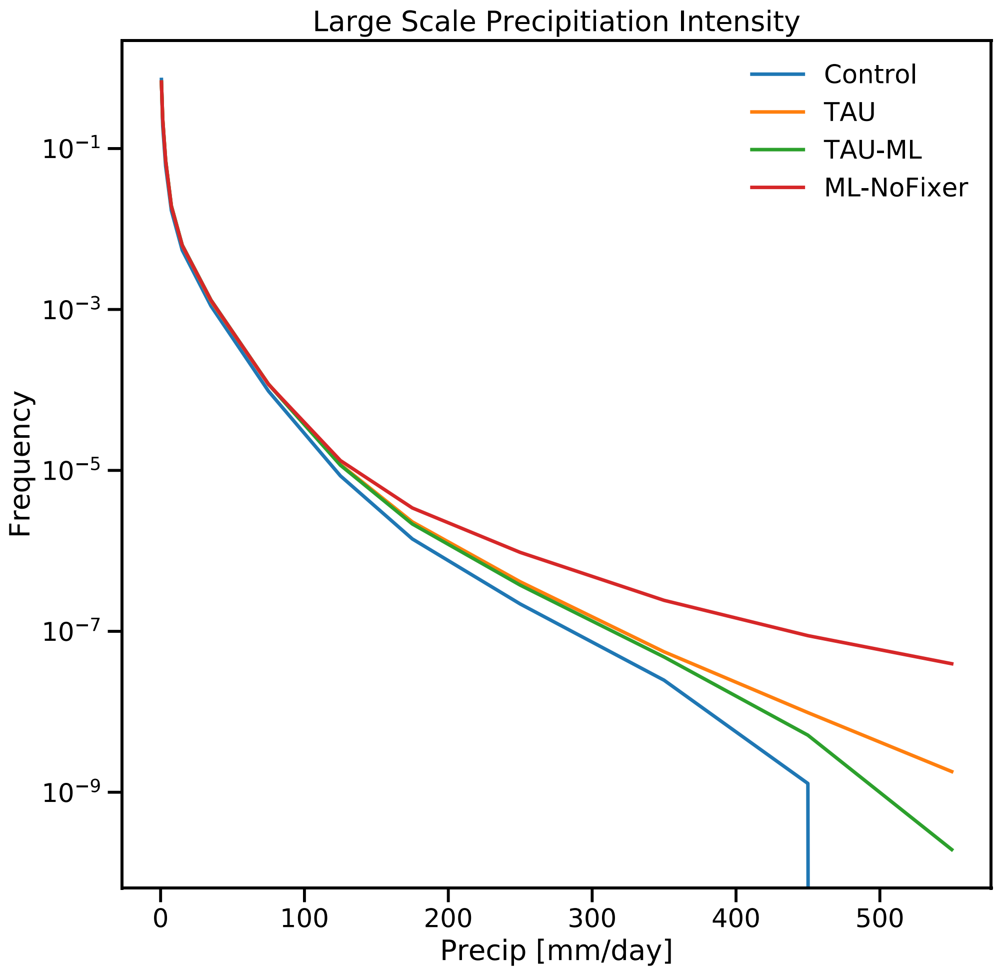


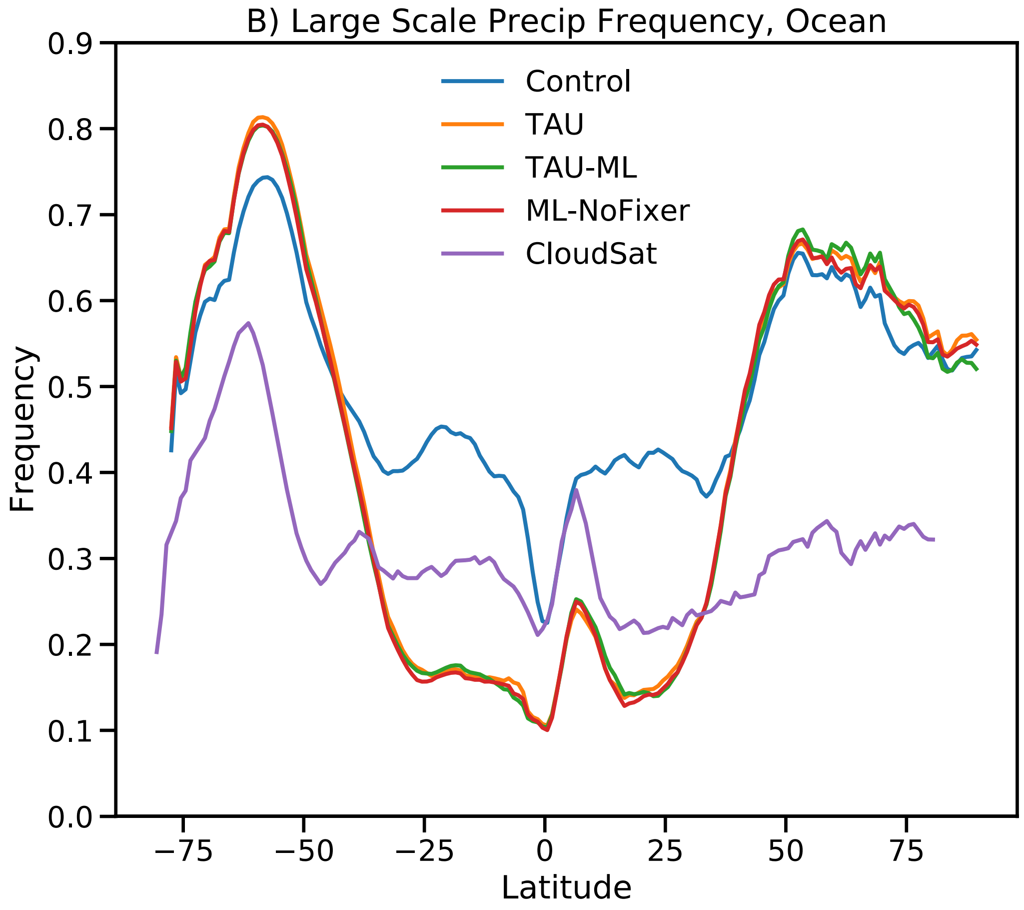


How often does mass fixer kick in and where?

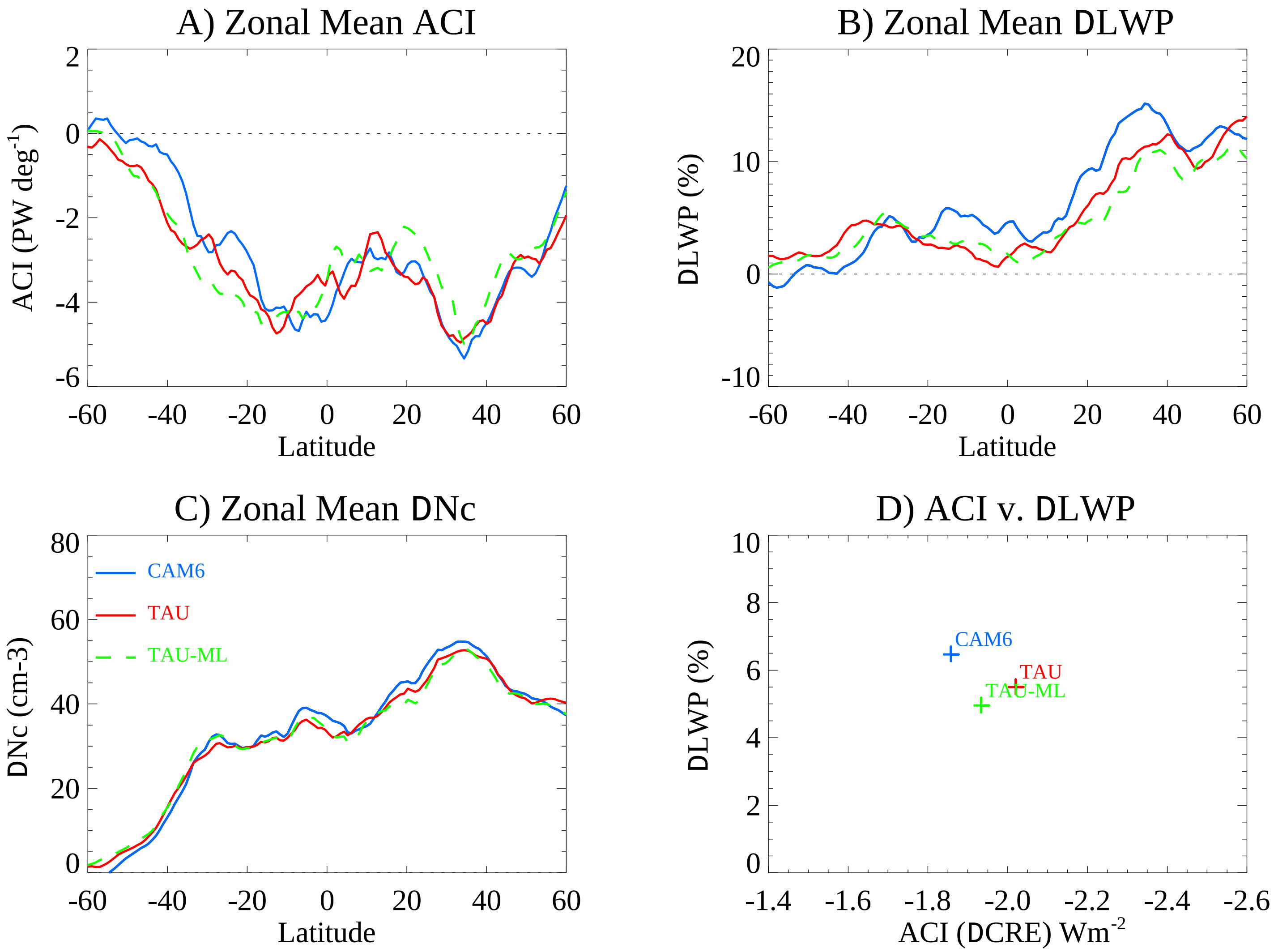
- Low altitudes and tropical high altitudes (cirrus)
- Low altitude (below is 936hPa), mostly in sub-tropical strato-cumulus regions, edge of stratus regions. Mostly SH.
- Also a tropical peak at 800hPa

Precipitation Feedbacks





Cloud Feedbacks



- ACI are similar between control and TAU code.
- Slightly lower LWP change, but forcing is similar, a bit higher in S. Hemisphere.
- Emulator reproduces TAU results.

Partial Dependence Plots

Goal: understand average sensitivities of input fields while accounting for nonlinear interactions within model

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

Temperature	Dewpoint	Pressure
280	10	986
280	14	1014
280	2	992
280	25	1025
280	6	950

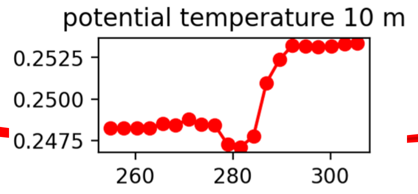
2. Feed fixed data through model

Machine Learning or Physical Model

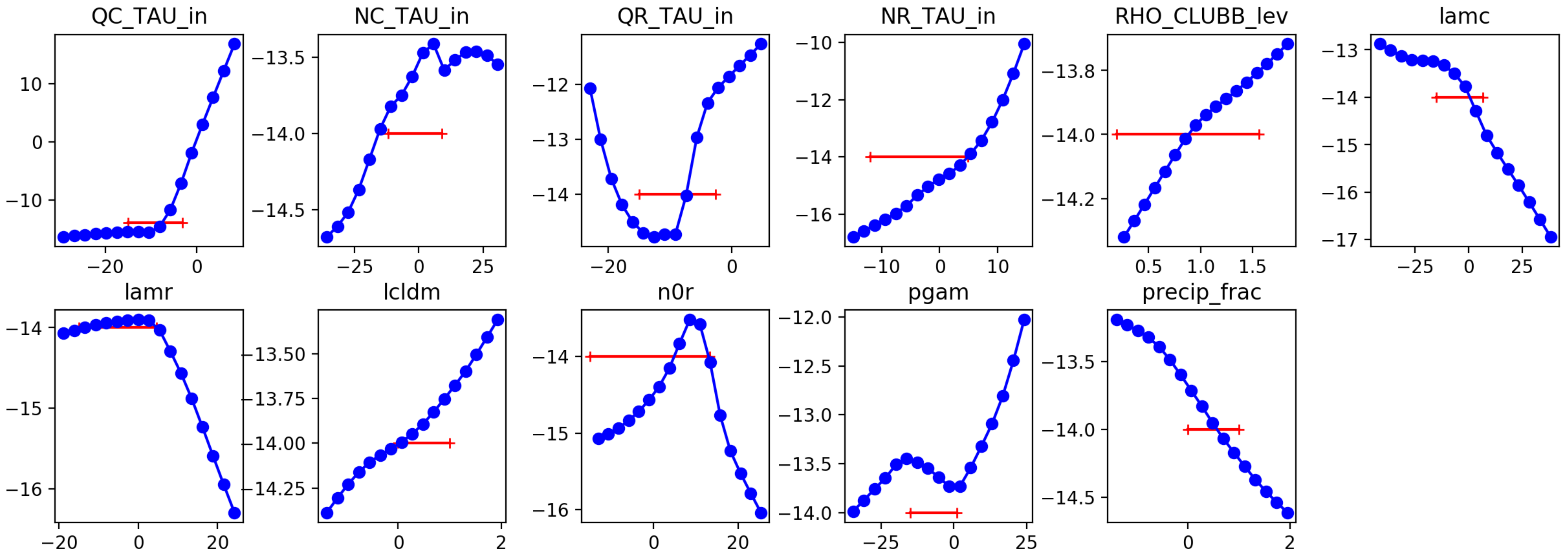
3. Calculate mean prediction for fixed value

Mean Prediction

4. Repeat process for range of input values



Microphysics Emulator Partial Dependence (Expanded Range)



Outside the range of the training data (red) the neural network extrapolates mostly linearly

Microphysics Summary and Challenges

- Neural network emulator set largely replicates the behavior of the TAU bin microphysics warm rain processes
- Successfully ran in CAM6 in training climate
- Both tendencies and feedbacks from emulator closely match original scheme

Challenges

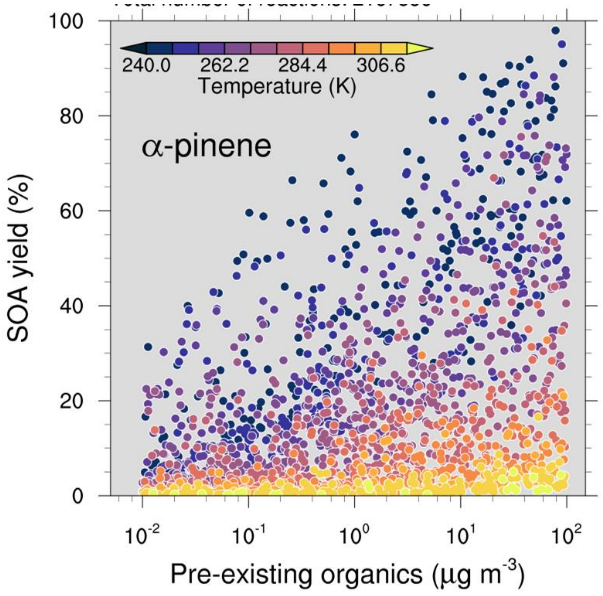
- Running in future and pre-industrial climates results in more calls to mass fixer
- Linear extrapolation behavior may not be appropriate for certain variables. How to constrain?
- Training superdroplet scheme emulator

Machine Learning Emulation of the GECKO-A Chemistry Model

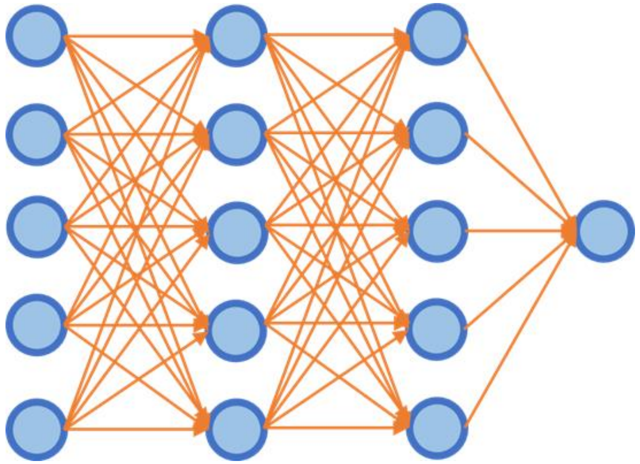
David John Gagne, Charlie Becker, John Schreck, Keely Lawrence, Siyuan Wang, Alma Hodzic

GECKO-A Challenge: Build An Emulator For 3-D Models?

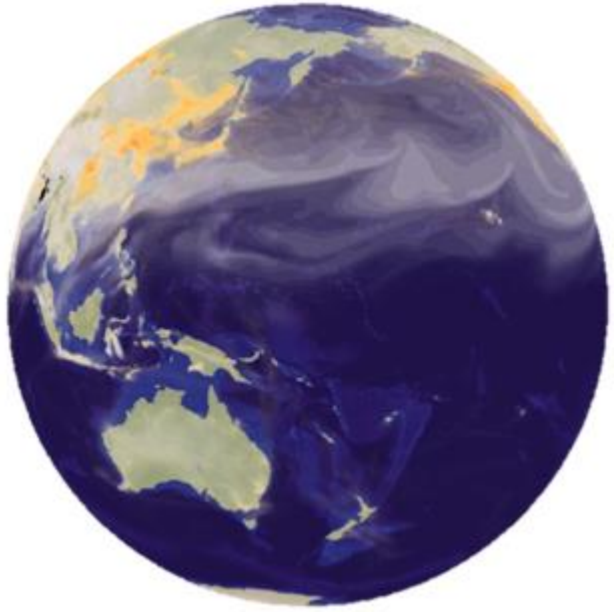
GECKO-A Training Library



Machine-Learning Emulator

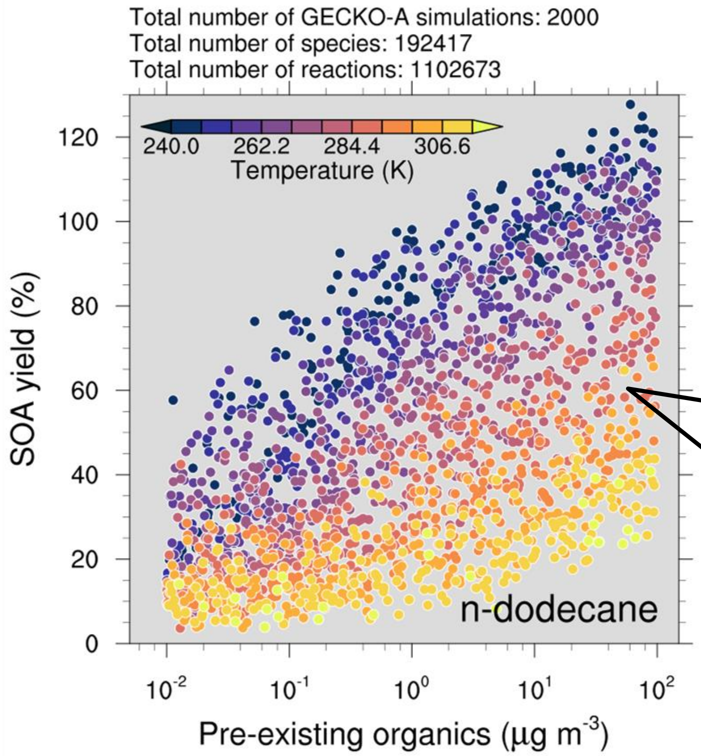


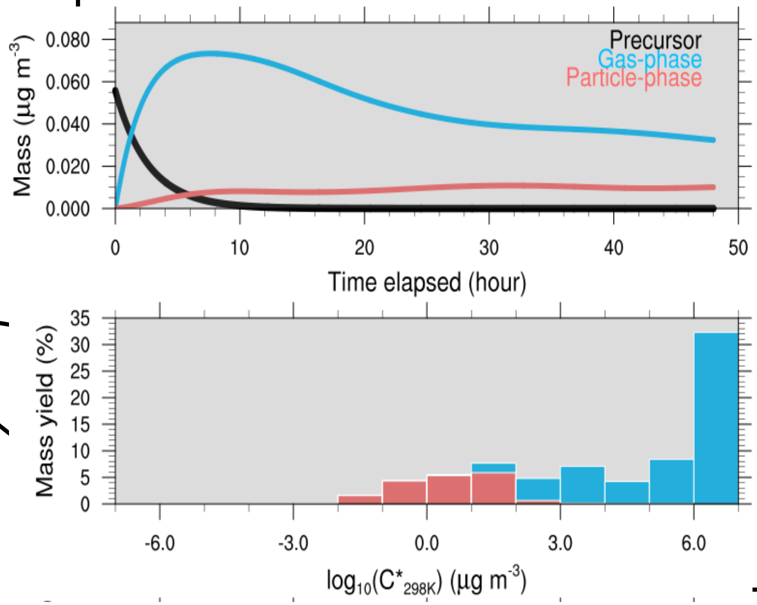
3-D Models

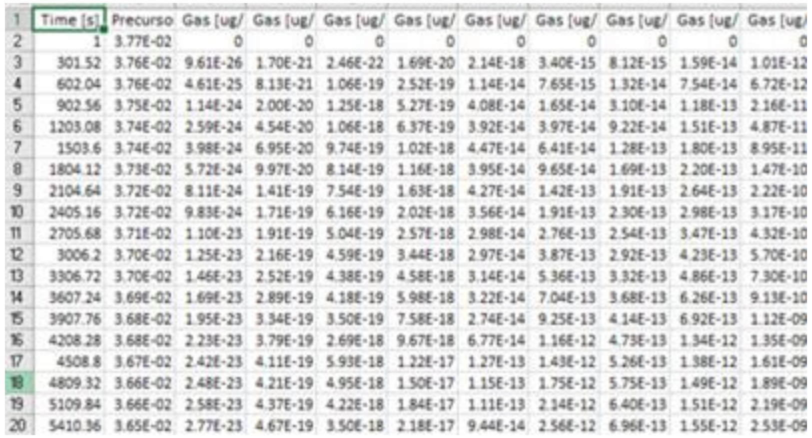


- Many inspiring applications out there: machine-learning emulators using explicit/process-level models, and implementing the trained emulators into large-scale models. Such explicit/process-level models are otherwise too expensive for large-scale models.
- The goal of this project is to train the machine-learning emulator using the “library” generated by the hyper-explicit chemical mechanism, GECKO-A.

Goal: Build Emulator to Predict the Total Organic Aerosol







	Time [s]	Precurso	Gas [ug/	Gas [ug/	Gas [ug/	Gas [ug/	Gas [ug/	Gas [ug/	Gas [ug/	Gas [ug/	Gas [ug/
2	1	3.77E-02	0	0	0	0	0	0	0	0	0
3	301.52	3.76E-02	9.61E-26	1.70E-21	2.46E-22	1.69E-20	2.14E-18	3.40E-15	8.12E-15	1.59E-14	1.01E-12
4	602.04	3.76E-02	4.61E-25	8.13E-21	1.06E-19	2.52E-19	1.14E-14	7.65E-15	1.32E-14	7.54E-14	6.72E-12
5	902.56	3.75E-02	1.14E-24	2.00E-20	1.25E-18	5.27E-19	4.08E-14	1.65E-14	3.10E-14	1.18E-13	2.16E-11
6	1203.08	3.74E-02	2.59E-24	4.54E-20	1.06E-18	6.37E-19	3.92E-14	3.97E-14	9.22E-14	1.51E-13	4.87E-11
7	1503.6	3.74E-02	3.98E-24	6.95E-20	9.74E-19	1.02E-18	4.47E-14	6.41E-14	1.28E-13	1.80E-13	8.95E-11
8	1804.12	3.73E-02	5.72E-24	9.97E-20	8.14E-19	1.16E-18	3.95E-14	9.65E-14	1.69E-13	2.20E-13	1.47E-10
9	2104.64	3.72E-02	8.11E-24	1.41E-19	7.54E-19	1.63E-18	4.27E-14	1.42E-13	1.91E-13	2.64E-13	2.22E-10
10	2405.16	3.72E-02	9.83E-24	1.71E-19	6.16E-19	2.02E-18	3.56E-14	1.91E-13	2.30E-13	2.98E-13	3.17E-10
11	2705.68	3.71E-02	1.10E-23	1.91E-19	5.04E-19	2.57E-18	2.98E-14	2.76E-13	2.54E-13	3.47E-13	4.32E-10
12	3006.2	3.70E-02	1.25E-23	2.16E-19	4.59E-19	3.44E-18	2.97E-14	3.87E-13	2.92E-13	4.23E-13	5.70E-10
13	3306.72	3.70E-02	1.46E-23	2.52E-19	4.38E-19	4.58E-18	3.14E-14	5.36E-13	3.32E-13	4.86E-13	7.30E-10
14	3607.24	3.69E-02	1.69E-23	2.89E-19	4.18E-19	5.98E-18	3.22E-14	7.04E-13	3.68E-13	6.26E-13	9.13E-10
15	3907.76	3.68E-02	1.95E-23	3.34E-19	3.50E-19	7.58E-18	2.74E-14	9.25E-13	4.14E-13	6.92E-13	1.12E-09
16	4208.28	3.68E-02	2.23E-23	3.79E-19	2.69E-18	9.67E-18	6.77E-14	1.16E-12	4.73E-13	1.34E-12	1.35E-09
17	4508.8	3.67E-02	2.42E-23	4.11E-19	5.93E-18	1.22E-17	1.27E-13	1.43E-12	5.26E-13	1.38E-12	1.61E-09
18	4809.32	3.66E-02	2.48E-23	4.21E-19	4.95E-18	1.50E-17	1.15E-13	1.75E-12	5.75E-13	1.49E-12	1.89E-09
19	5109.84	3.66E-02	2.58E-23	4.37E-19	4.22E-18	1.84E-17	1.11E-13	2.14E-12	6.40E-13	1.51E-12	2.19E-09
20	5410.36	3.65E-02	2.77E-23	4.67E-19	3.50E-18	2.18E-17	9.44E-14	2.56E-12	6.96E-13	1.55E-12	2.53E-09

Demo: what the data looks like

GECKO-A Library:

- 2000 GECKO-A simulations: in each run, we run GECKO-A under certain condition for 5 days
- 2000 input files (csv).
- Each file contains: (i) mass of precursors; (ii) mass of products in the gas-phase; and (iii) mass of products in the particle-phase. All (i)-(iii) as a function of time.

GECKO Data

Metadata

Metadata	Units	Label
Number Experiments	2000	id
Total Timesteps	1440	Time
Timestep Delta	300 seconds	-

Potential Input Variables

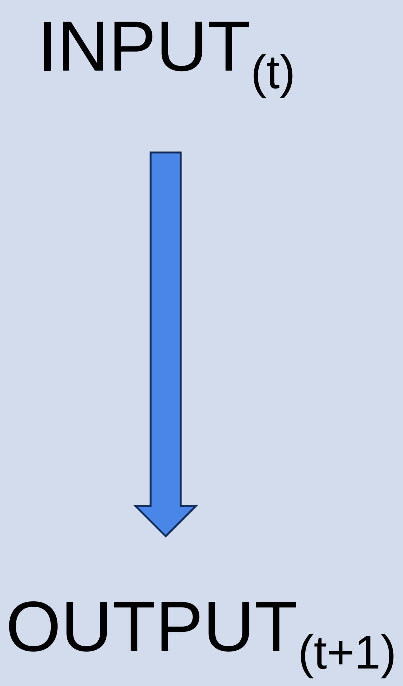
Variable Name	Units	Type
Precursor	ug/m3	Varies
Gas	ug/m3	Varies
Aerosol	ug/m3	Varies
Temperature	K	Static
Solar Zenith Angle	degree	Static
Pre-existing Aersols	ug/m3	Static
o3	ppb	Static
nox	ppb	Static
oh	10^6 molec/cm3	Static

Potential Output Variables

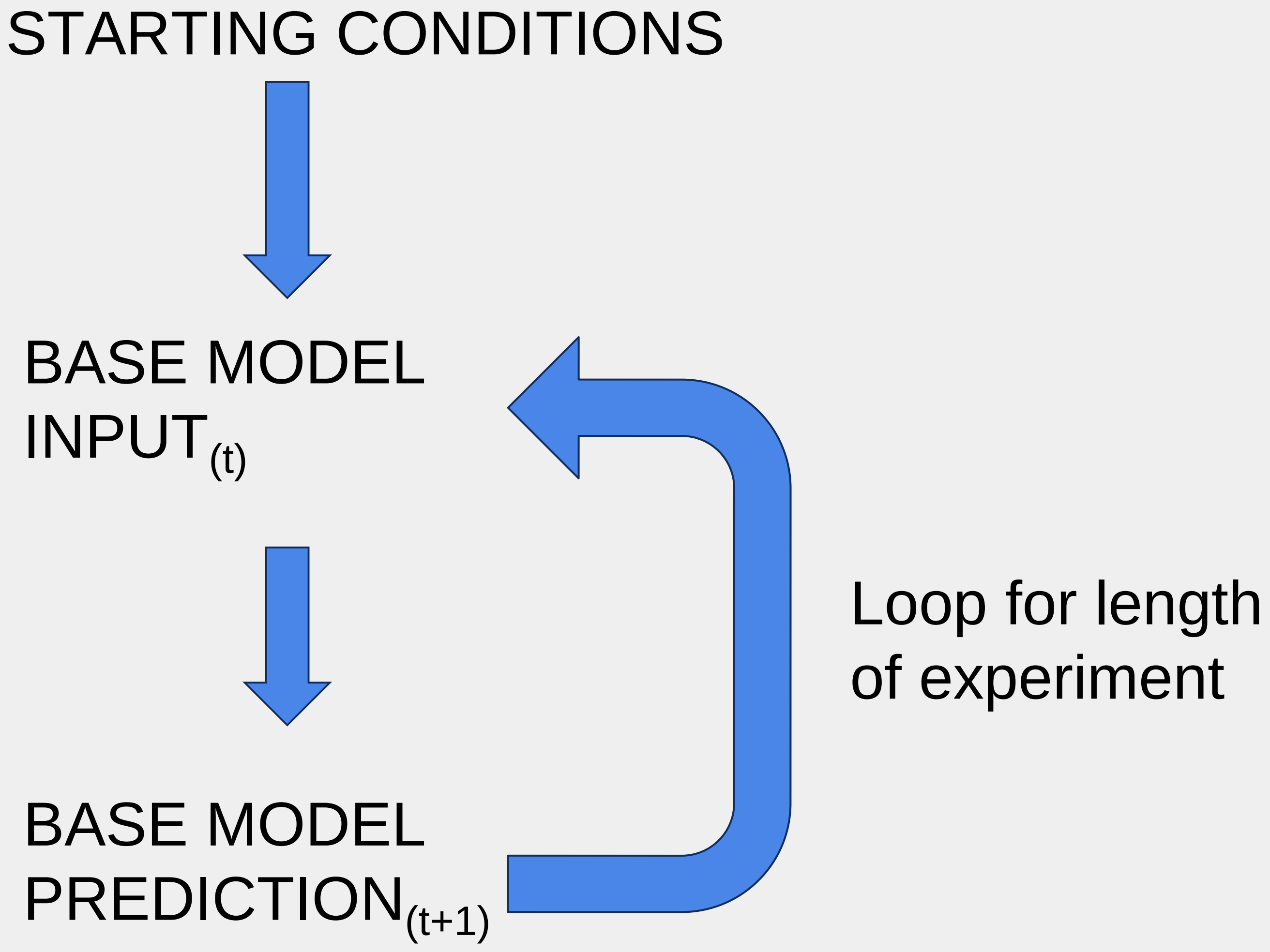
Variable Name	Units	Type
Precursor (at t+1)	ug/m3	Varies
Gas (at t+1)	ug/m3	Varies
Aerosol (at t+1)	ug/m3	Varies

- Use fixed environmental conditions and concentration of precursor, gas and aerosol for a given precursor type
- Generated data for toluene, dodecane, and alpha-pinene

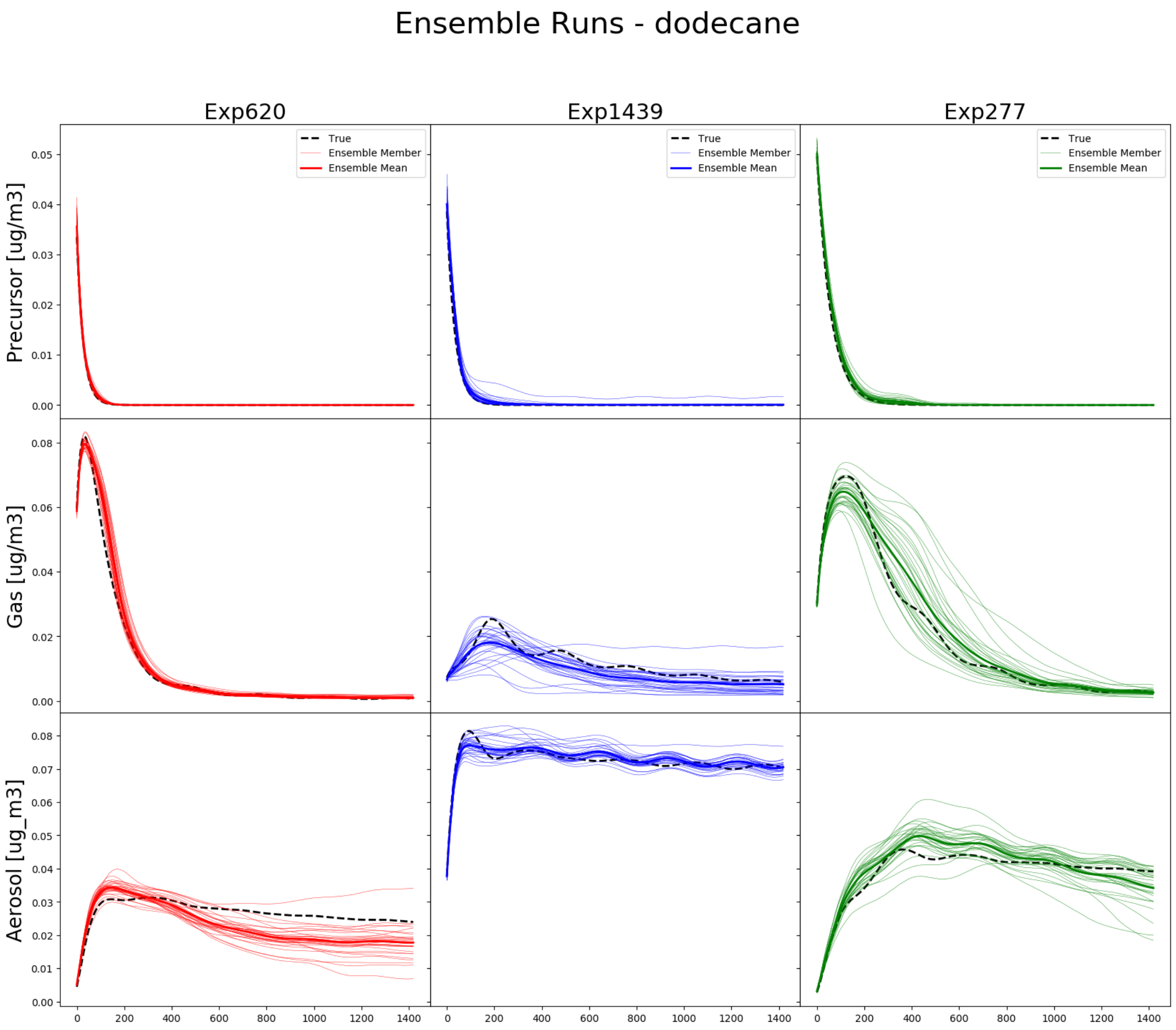
Base Model



Box Emulator Model



MACHINE LEARNING EMULATION OF GECKO-A
Static Environmental Conditions



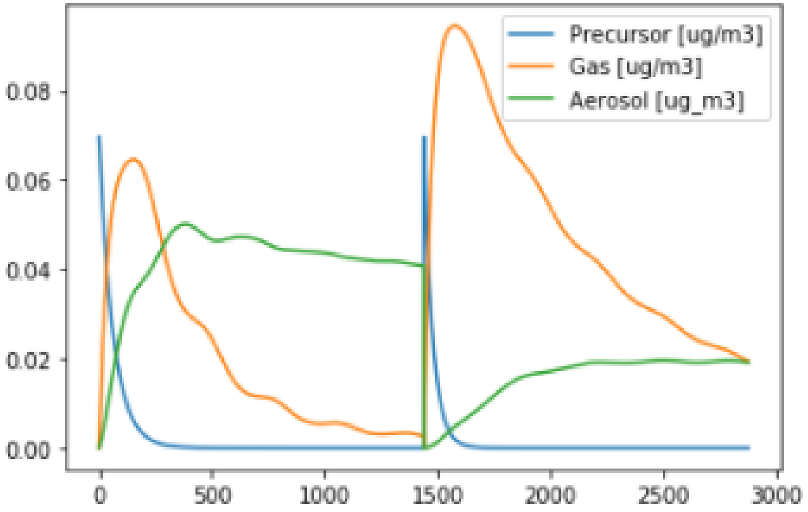
LSTM FRAMEWORK:

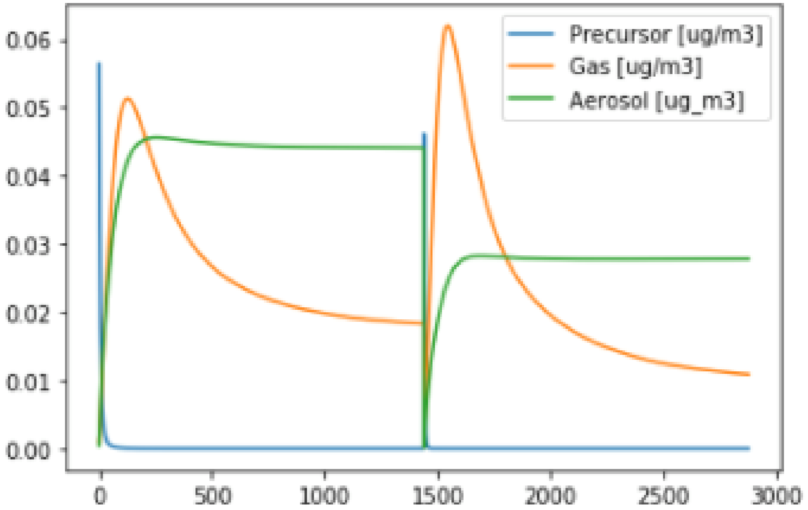
- Recurrent neural network combined with 1D convolutions through time (depends on previous 20 time steps to predict single future time step)
- Trained on 1600, 5-day Experiments (300s time steps) - validated on 200 experiments

CHALLENGES:

- Recurrent networks tend to prevent runaway error propagation but have major challenges incorporating them into a 3D transport model

Single Timestep Model Runs





SINGLE TIMESTEP FRAMEWORK:

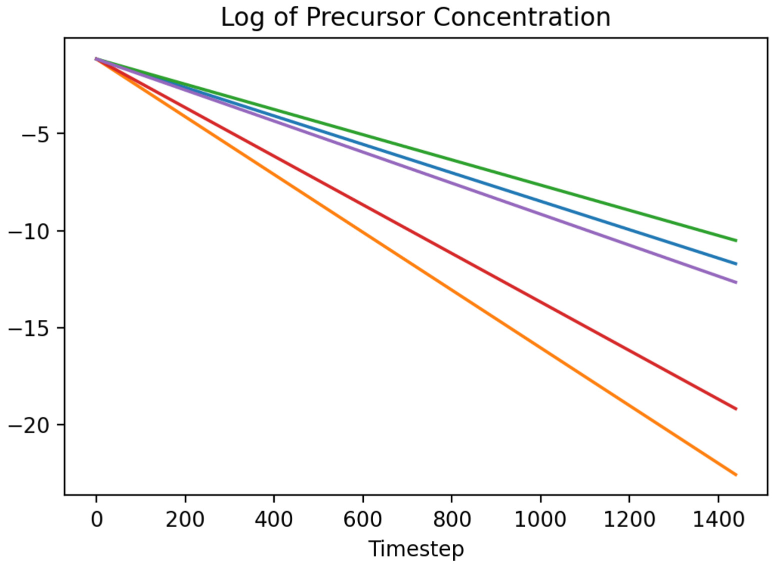
- Fully-connected single-layer neural network with SELU activation function
- Trained on single timestep input

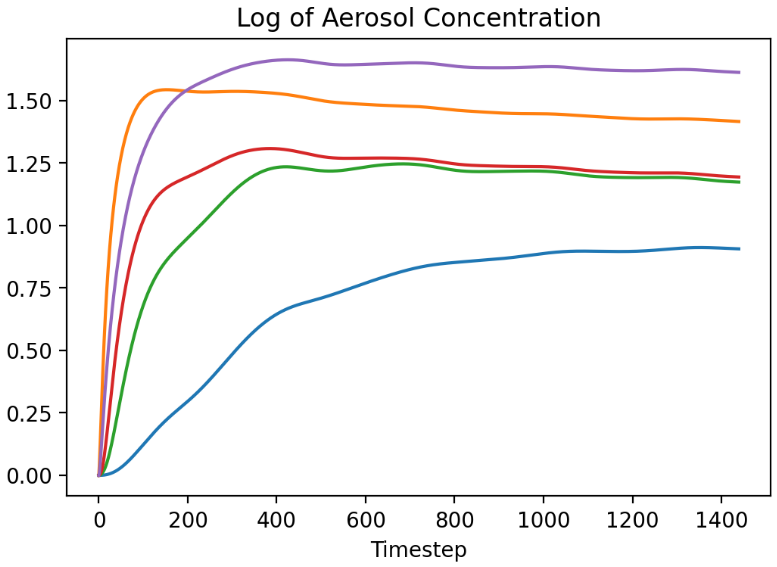
CHALLENGES:

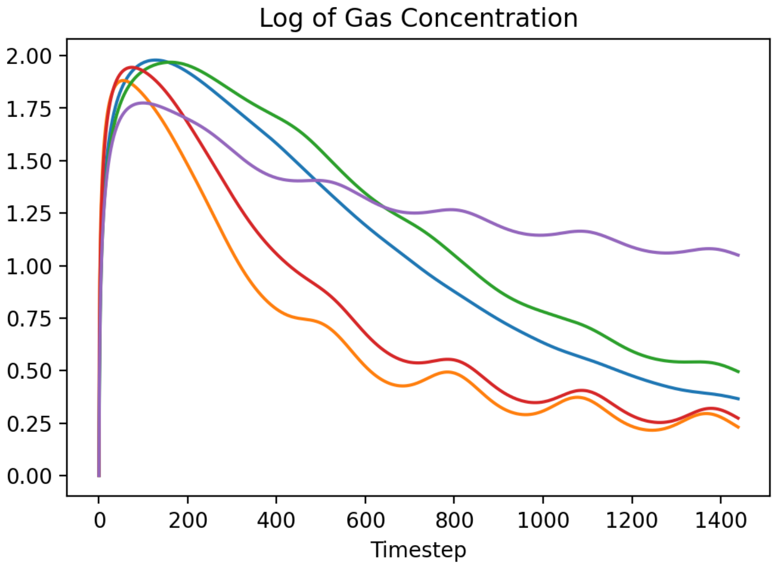
- Machine learning captures early changes fine but struggles with later parts of run
- Performs well in offline tests but not in emulator box model

GECKO: Challenges

- Large magnitude difference in both absolute values and tendencies between early and later parts of each simulation
- Machine learning optimization biased toward adjusting initial values more due to larger errors
- Precursor decreases exponentially but gas and aerosol have more complex variability pattern

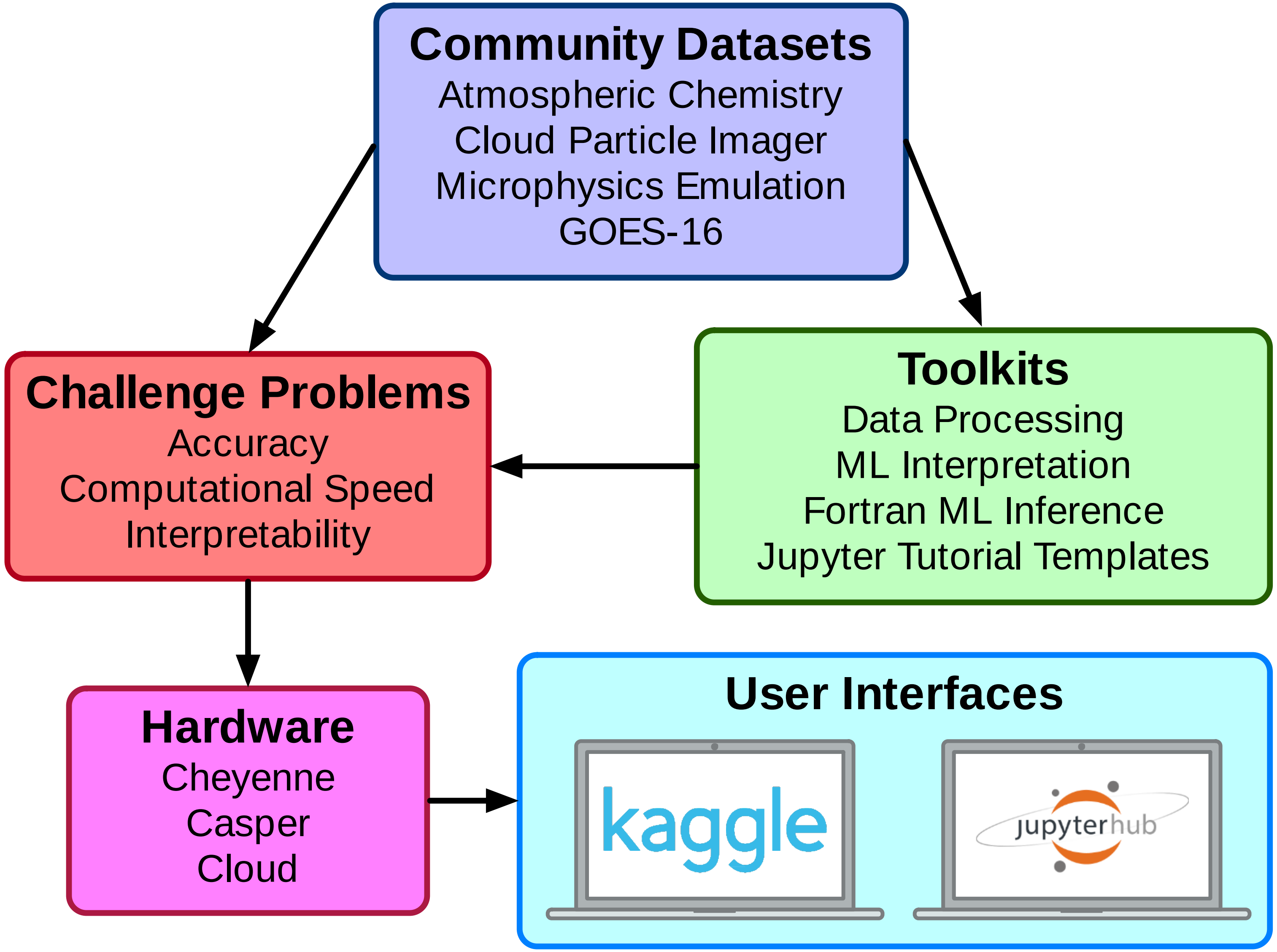






AI for Earth System Science Hackathon Motivation

- Interest in AI/ML for weather and climate problems is growing rapidly
- Earth System Science practitioners need help getting started with ML
- Not enough trained ML-ESS experts to work with everyone
- **Solution:** Host a summer school and hackathon!
- Invite AI-ESS experts to lecture about different aspects of AI and ESS
- Create training materials and domain-focused challenge problems



Hackathon Goals

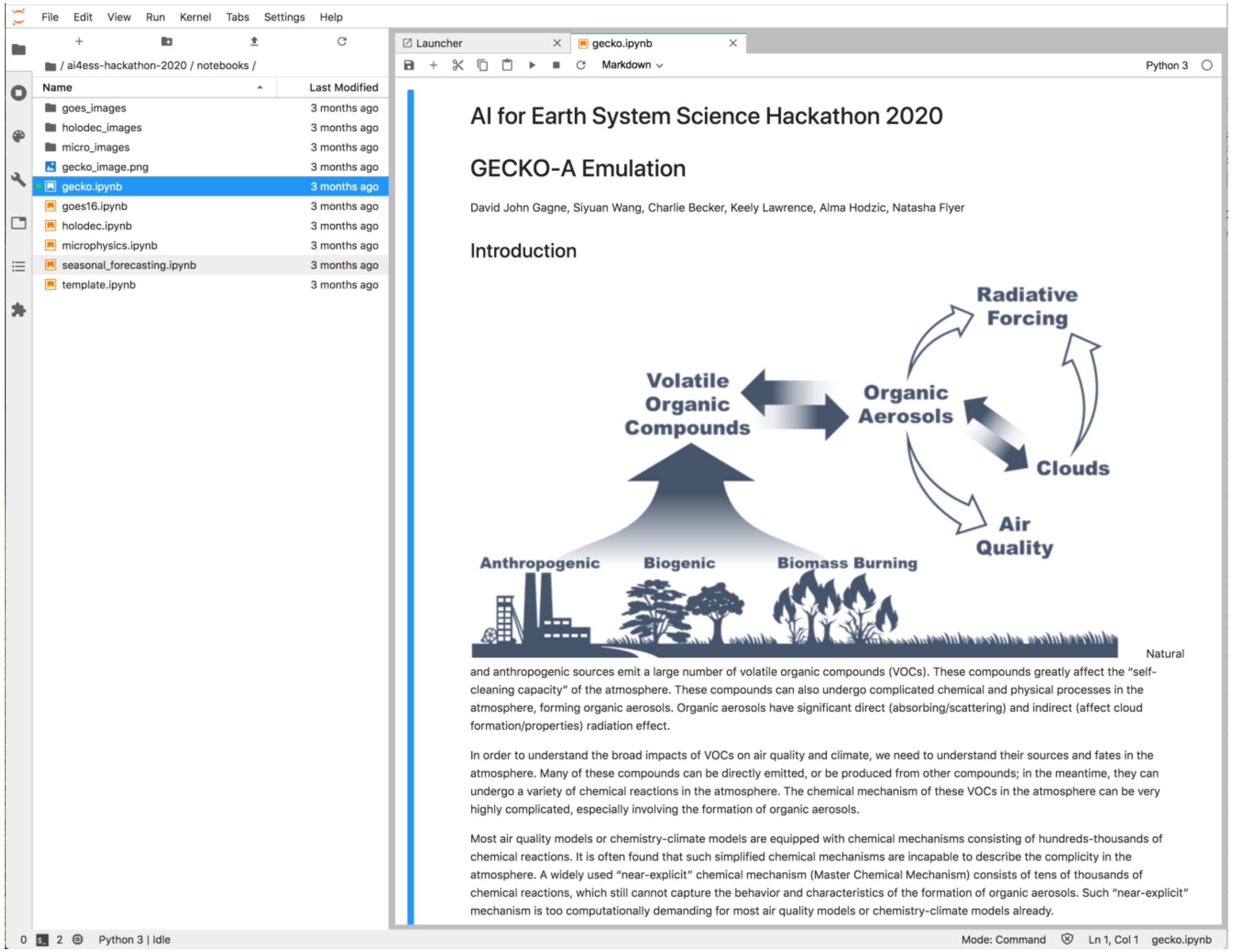
- Give participants machine learning experience with realistic ESS data and problems
- Provide them with sufficient computational resources to train more complex models
- Work collaboratively with a new team with diverse backgrounds
- Originally planned to be in person at Mesa Lab, but COVID happened
- Virtual hackathon allowed greatly expanded participation (80->200)

User Interface

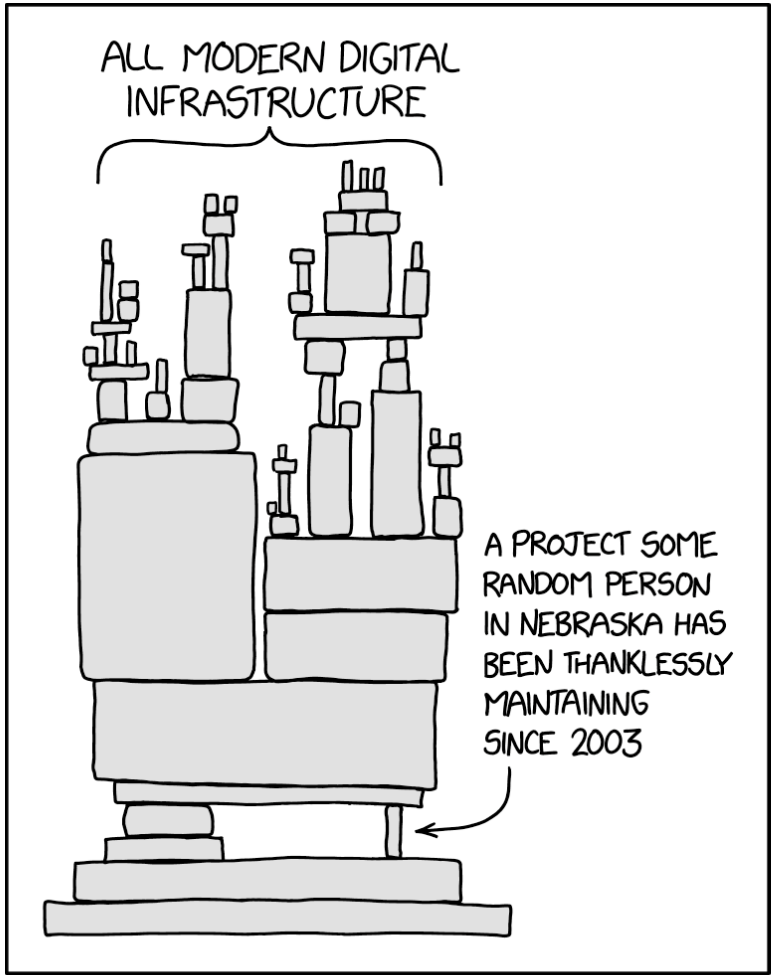
- Jupyterlab in web browser after logging in with Google credentials.
- Jupyterlab is preinstalled with full python data science environment and access to a GPU.
- User can save data in virtual machine that persists over lifetime of hackathon.
- Users can also run challenge problems through Google Colab notebooks

Team Setup

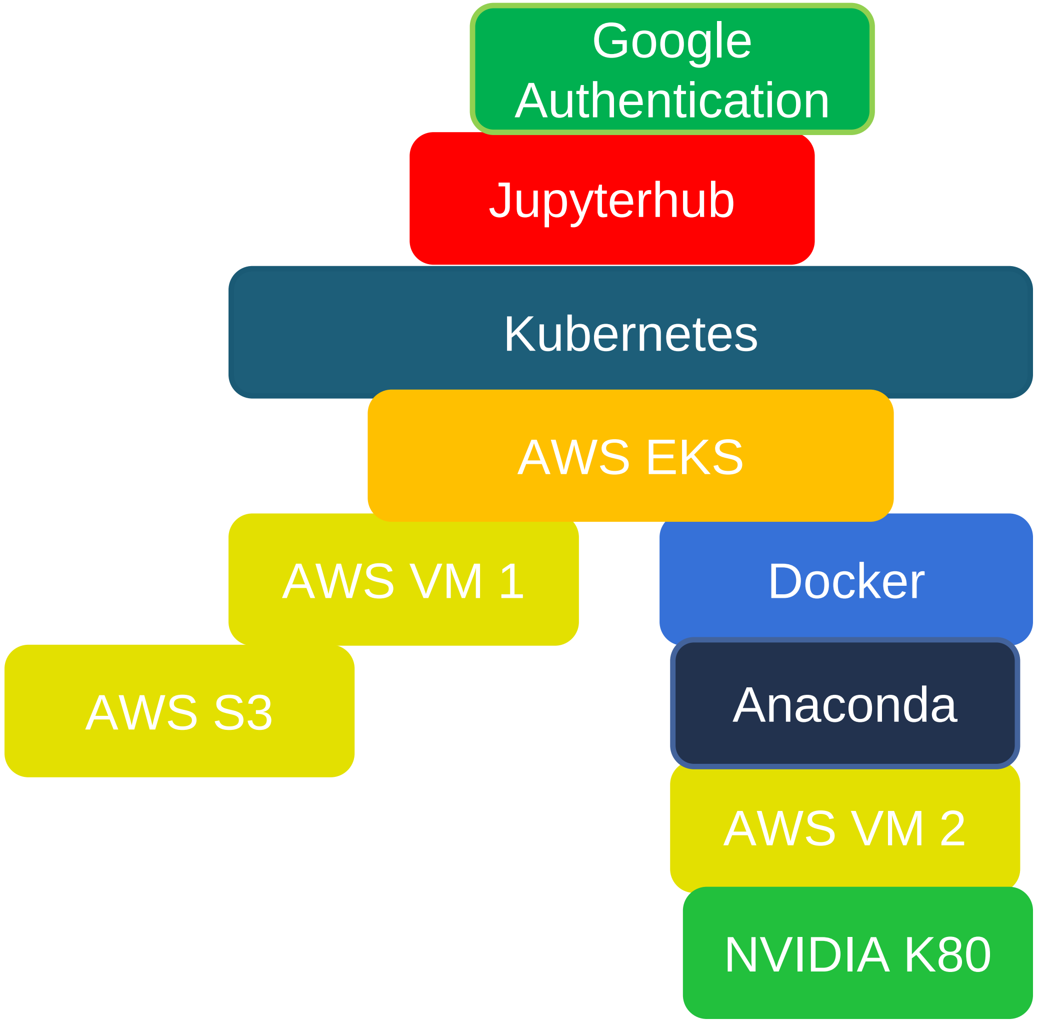
- Users communicate with other team members through Slack
- Teams of 5 were assigned randomly from the registration info.
- Scheduled hackathon period from 2 to 6 PM Mountain Time each day



Hackathon Cloud Infrastructure

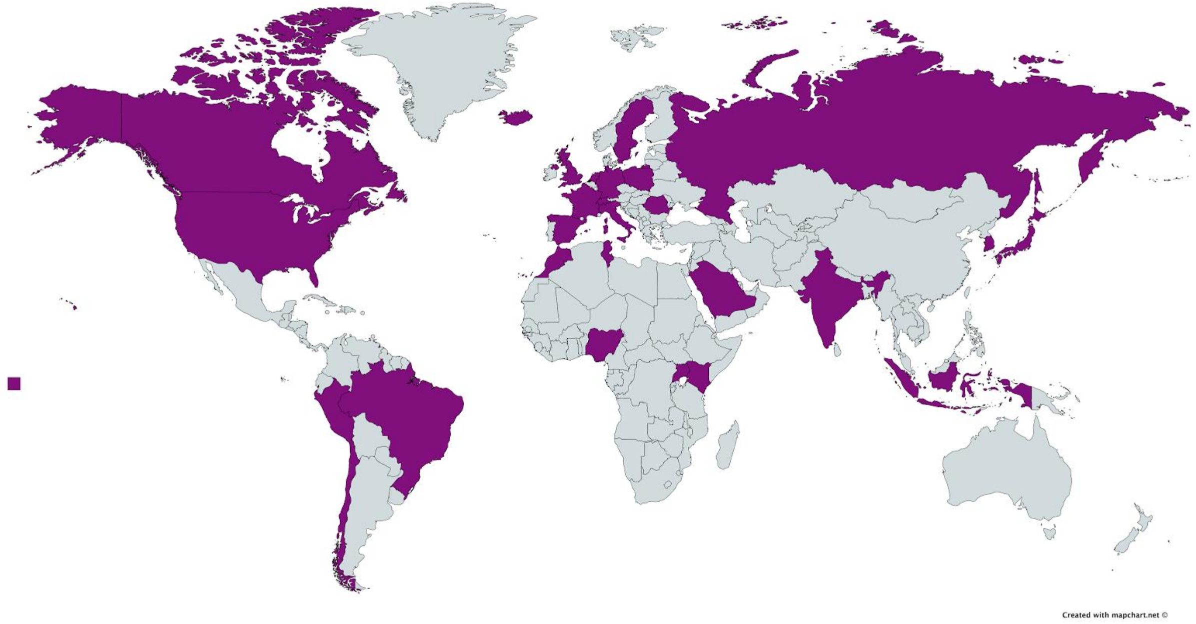


From xkcd.com



Hackathon Participants

- Over 250 initial participants from 5 continents
- ~150 participate throughout the week
- 72 teams
- Over 33K Slack Messages
- ~$36K in AWS for Earth compute credits used



Administration Challenges

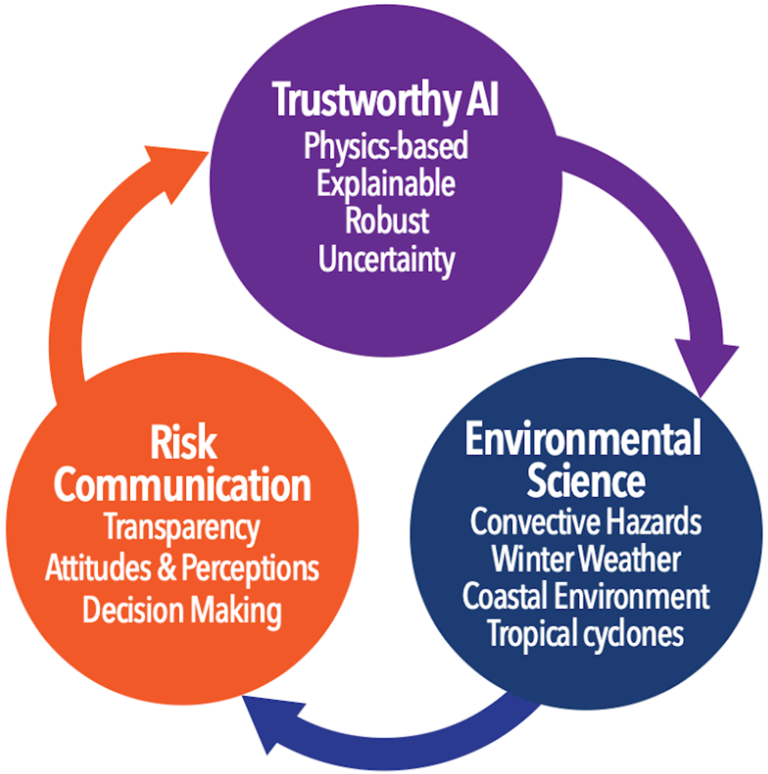
- Setting up Jupyterhub and Kubernetes on AWS
  - Lots of trial and error in the last week
  - Could not get autoscaling to work
- Code and VM bugs/failures
  - Users discovered coding bugs and needed more libraries/extensions in Python environment
  - Using more RAM than available crashes the VM instead of going to swap
- Slack communication
  - Too many notifications turned on by default for admins
  - Receiving questions through mix of official channels and PMs
  - Could have used more people helping answer team questions
- Team management
  - People dropped out throughout the week
  - Some people requested transfers to different teams/problems

Lessons Learned

- Have fewer, more robustly tested and documented challenge problems
- Create challenge problems targeted at different experience levels
- Have synchronous work periods targeted at different time zones
- Satellite admin sites to support different time zones
- Provide clearer guidance up front about tasks, goals, best practices, and expectations
- Provide regular feedback on team submissions
- Charge a registration fee to incentivize participation

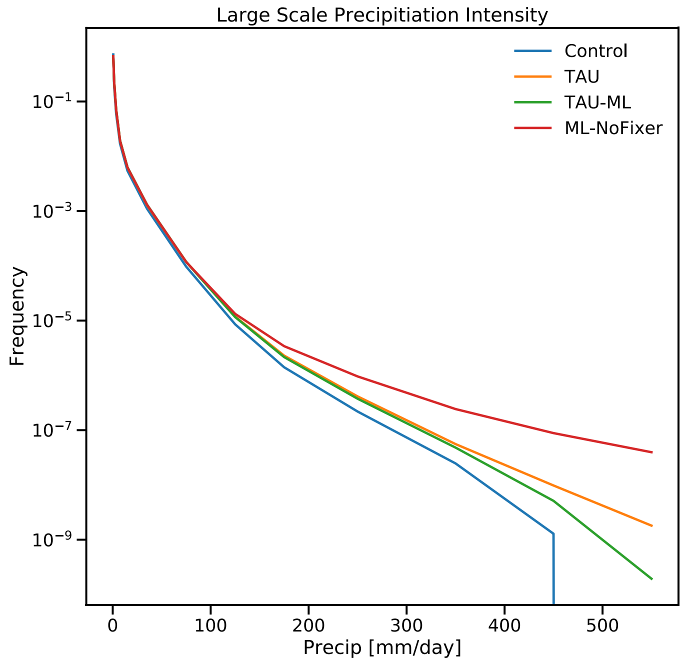
Hackathon notebooks:
https://github.com/NCAR/ai4ess-hackathon-2020

NSF AI Institute for Research on Trustworthy AI in Weather, Climate, and Coastal Oceanography

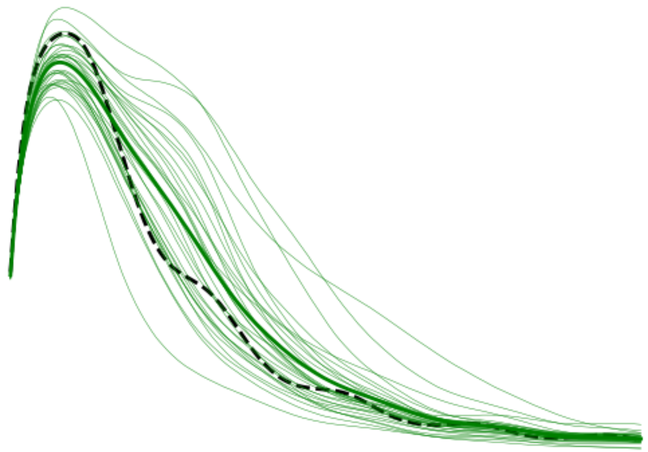


@ai2enviro https://www.ai2es.org

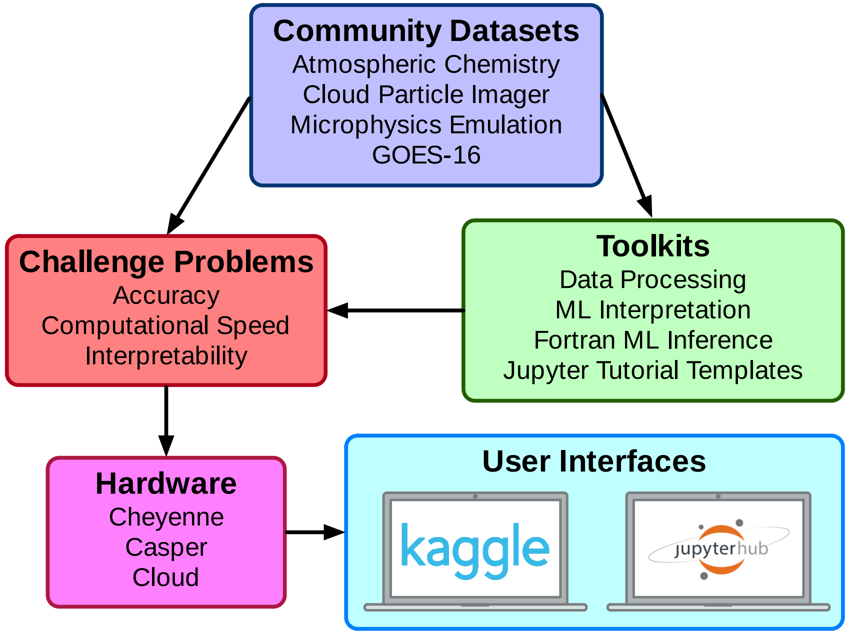
Summary



Microphysics Emulation



GECKO Emulation



AI4ESS Hackathon

Contact Me
Email: dgagne@ucar.edu
Twitter: @DJGagneDos
Github: djgagne

AI4ESS Presentations, Notebooks and Data Links at ai4ess.org