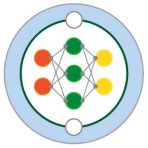
AI/ML Efforts in CESM Using Python

A lesson in pain

Will Chapman, Judith Berner, Jim Edwards

Mar 04, 2024

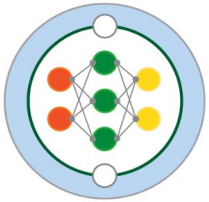


This material is based upon work supported by the National Center for Atmospheric Research, which is a major facility sponsored by the National Science Foundation under Cooperative Agreement No. 1852977.

Background / Context

M²LInES - Multiscale Machine Learning In Coupled Earth System Modeling

M²LInES (pronounced M-square-lines) is an international collaborative project with the goal of improving climate projections, using scientific and interpretable Machine Learning to capture unaccounted physical processes at the air-sea-ice interface.



Two main sources for uncertainty and model error:

1.

Initial Condition Uncertainty

Initial Condition State Variables (U,V,T,Q)

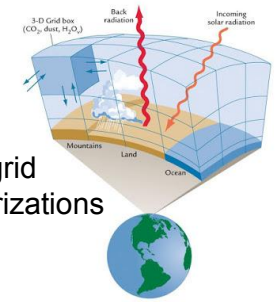
Dynamic Core + Physics Package

the discretized equations of motion

subgrid processes

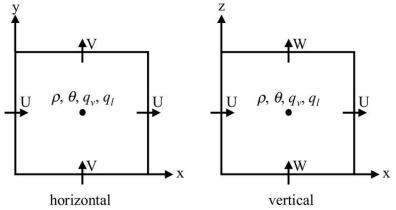
Climate Modeling

2.



Subgrid parameterizations

Numerical Approx.

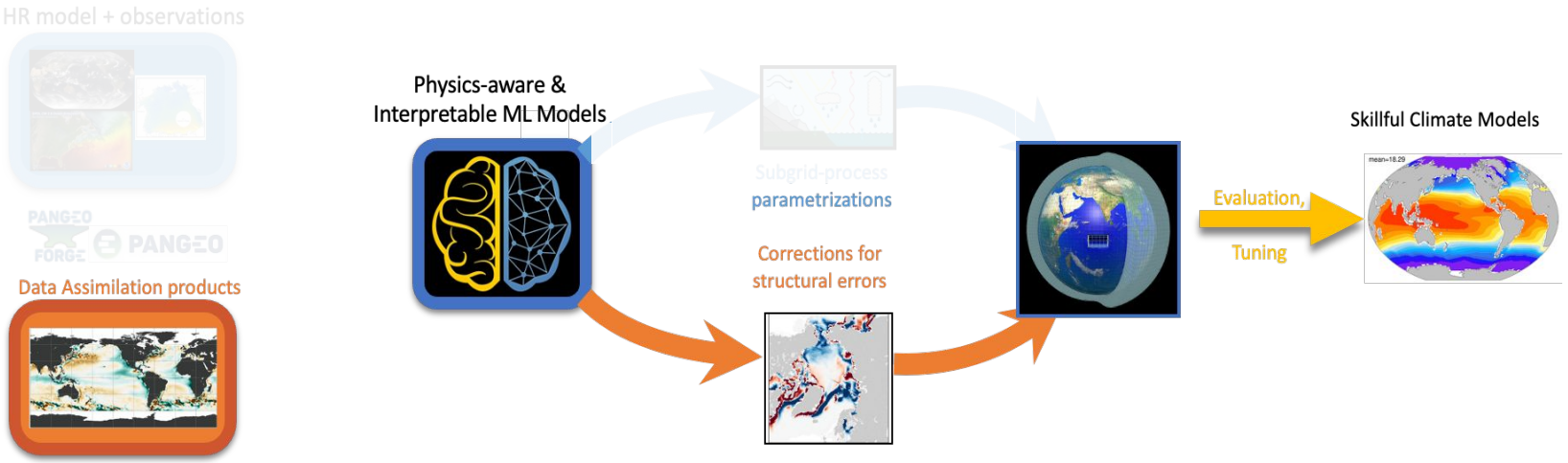


Model Deficiency

How can we help address these errors with machine learning?

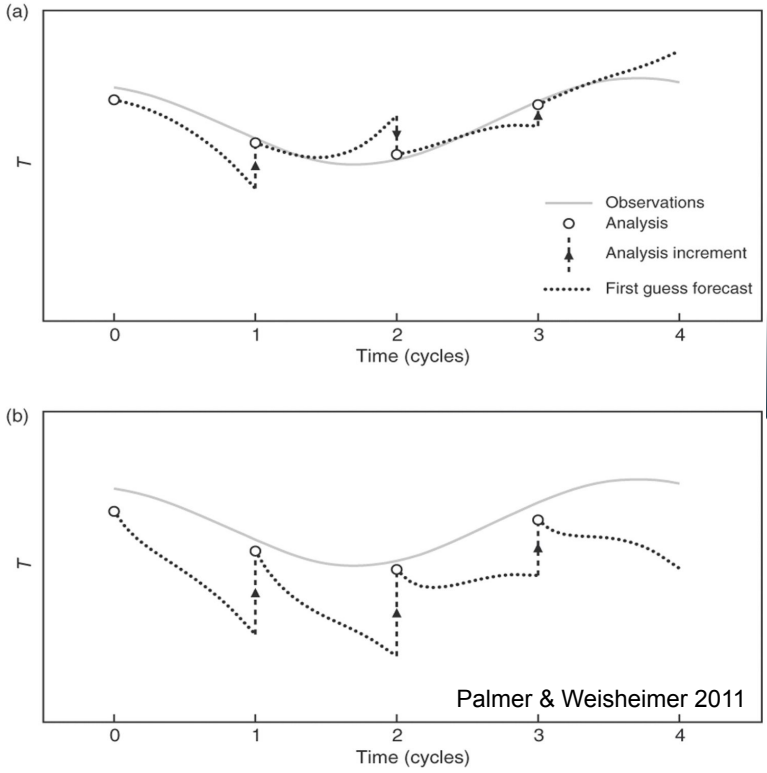
https://m2lines.github.io/

Model Deficiency learning pathways



Slide Courtesy of Laure Zanna

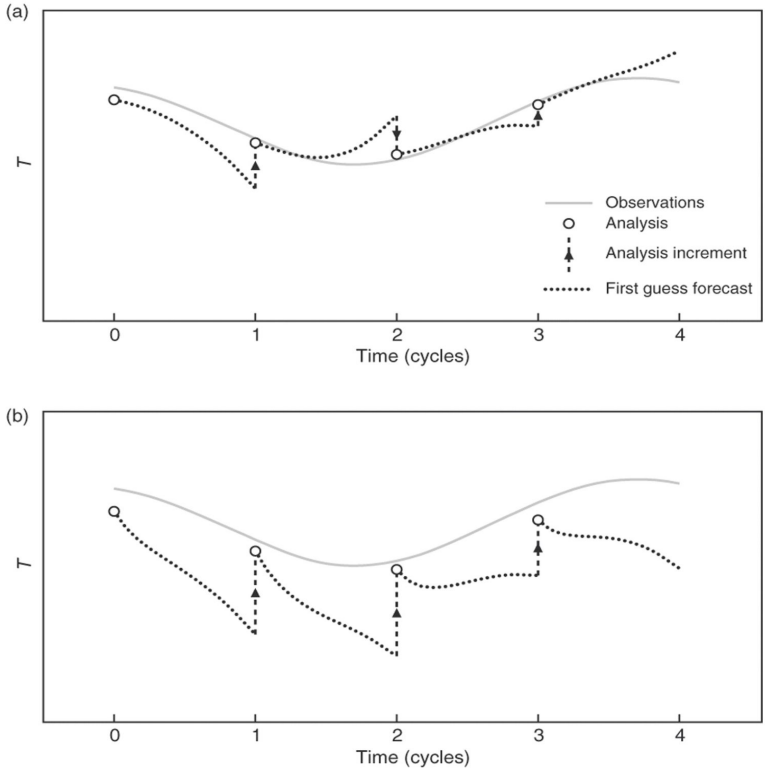
Using Data Assimilation to Identify Structural Model Error



- **Schematic**: *The analysis increment is correcting the model trajectory systematically to a warmer state.*
- *This is evidence of a systematic model error (assuming unbiased obs)*

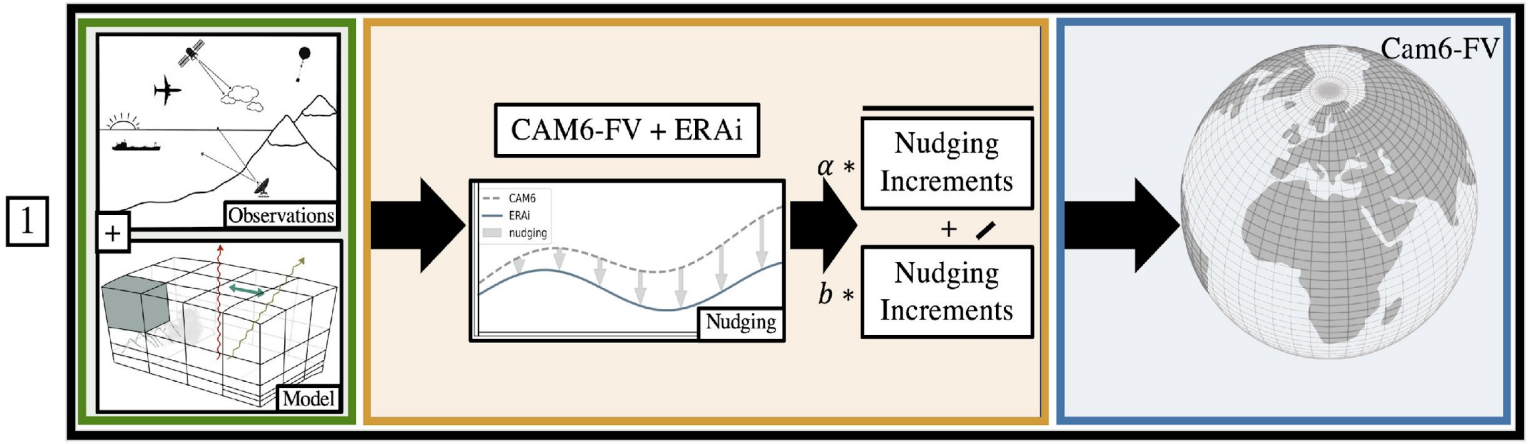
Although the amplification of the effect of [INSERT FORCING] on [INSERT BIASED PROCESS] will occur on ***timescales of decades***, the intrinsic timescale associated with [INSERT BIASED PROCESS] itself is typically on the order of hours. Hence it should in principle be possible to assess whether the anomalously small values of [INSERT PROCESS] are realistic or not, by studying the performance of such models in short-range weather prediction mode.

Using Data Assimilation to Identify Structural Model Error



- **Schematic**: *The analysis increment is correcting the model trajectory systematically (averaged over many initializations) to a warmer state.*
- *This is evidence of a systematic model error (assuming unbiased obs)*
- *Klinker & Sardeshmukh 1997, Rodwell and Palmer 2007.*

Although the amplification of the effect of [INSERT FORCING] on [INSERT BIASED PROCESS] will occur on ***timescales of decades***, the intrinsic timescale associated with [INSERT BIASED PROCESS] itself is typically on the order of hours. Hence it should in principle be possible to assess whether the anomalously small values of [INSERT PROCESS] are realistic or not, by studying the performance of such models in short-range weather prediction mode.



1.1 ERAi Creation (4Dvar) [2010-2019; Dee et al. 2011]

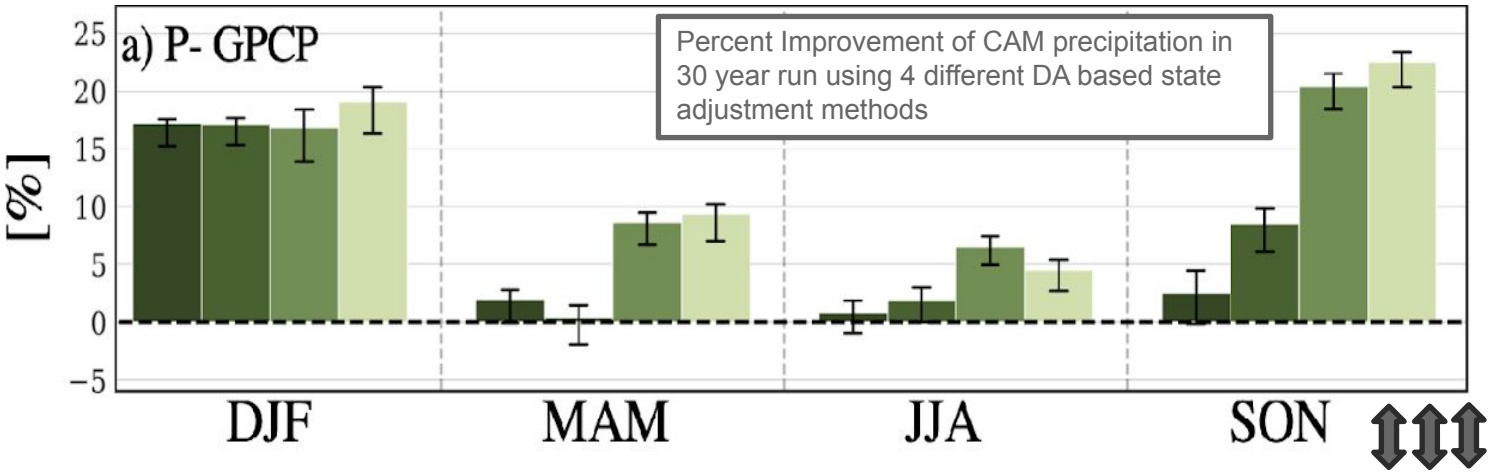
1.2 Nudging Increment Creation [2010-2019]

2.1 DART Increment Creation (EAKF) [2010-2019; Raeder et al. 2021]

1.2 2.2 Online Bias Correction [1982-2010]

Full Fortran based framework

CNN Running in CESM2.1

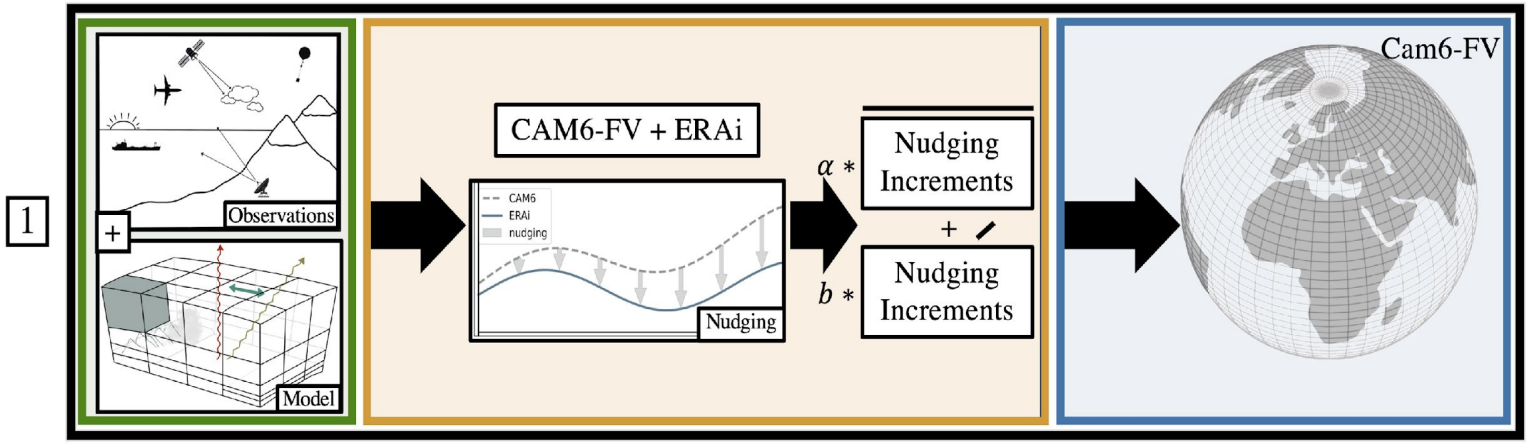


Quarterly Journal of the Royal Meteorological Society
RMetS

RESEARCH ARTICLE

Deterministic and Stochastic Tendency Adjustments Derived from Data Assimilation and Nudging

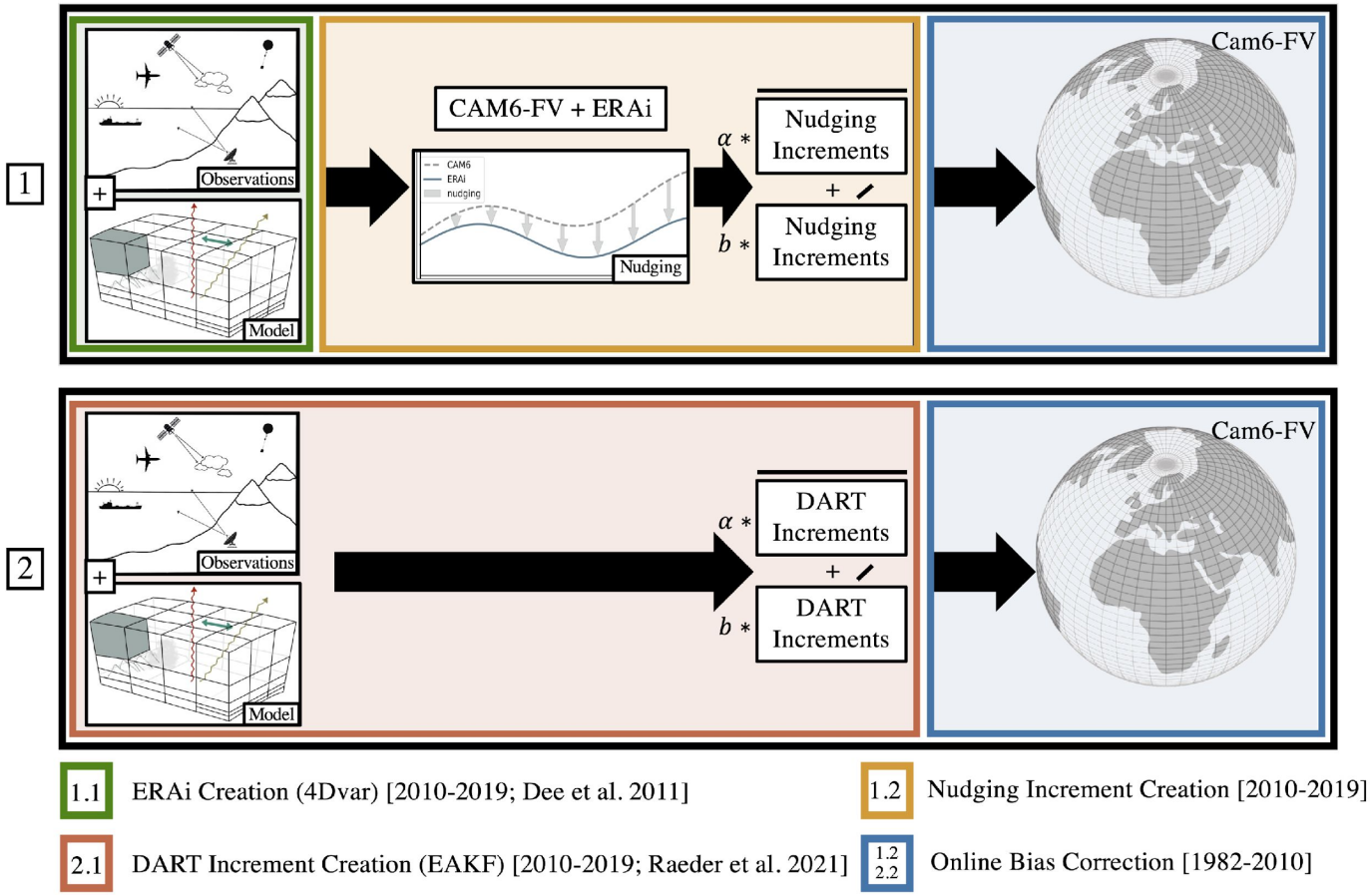
William E. Chapman PhD, Judith Berner PhD



- Nudging increments and DA increments pick up the same general features of systematic model bias, particularly in the tropics at lower model levels
- Overall, we find a positive impact of an online model-error representation based on re-inserting DA increments and nudging on the climatological bias.
- The addition of a stochastic tendency reduced this bias and created an more accurate representation of major modes of variability when compared to observations.

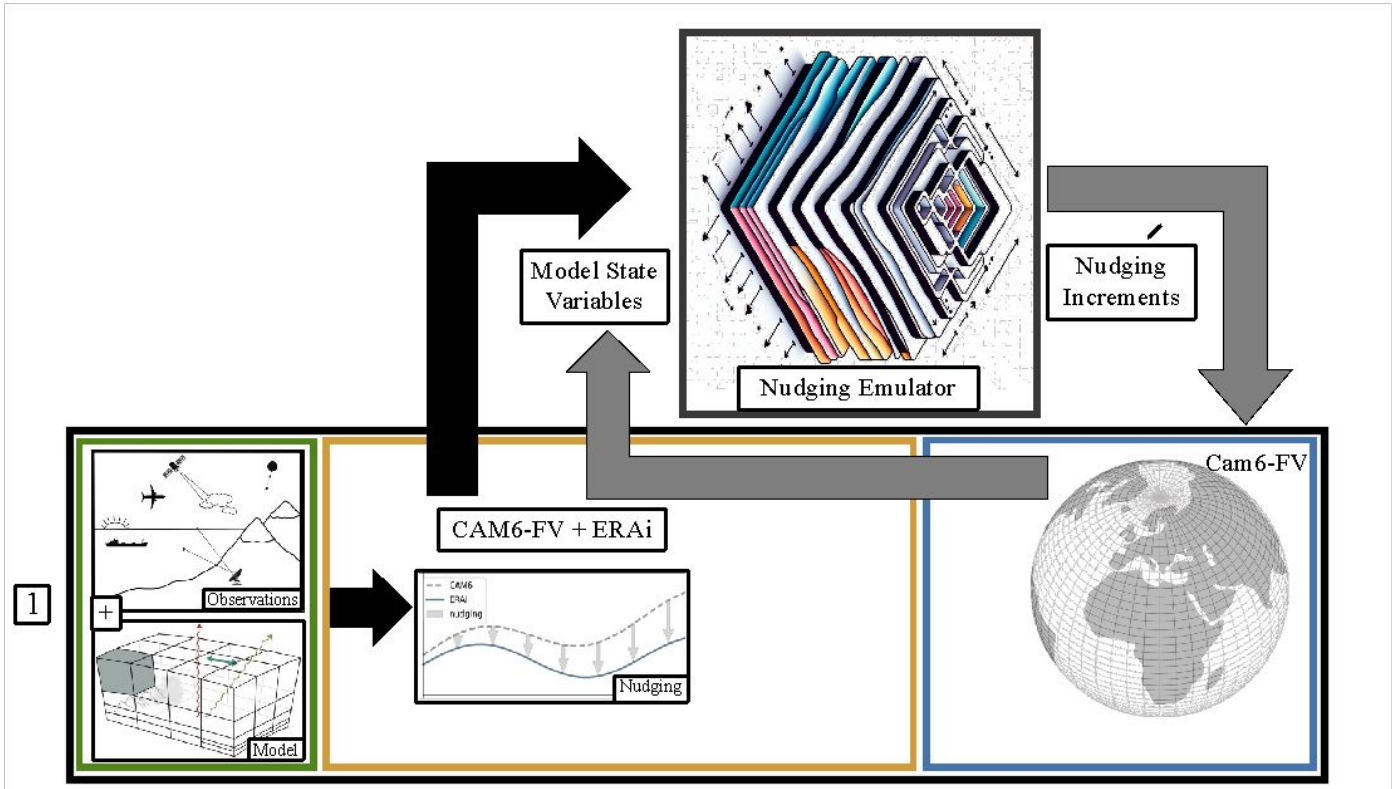
Machine Learning Emulation

Can this process be replaced with a Machine Learning pipeline to create a “state-dependent” correction?



Machine Learning Emulation

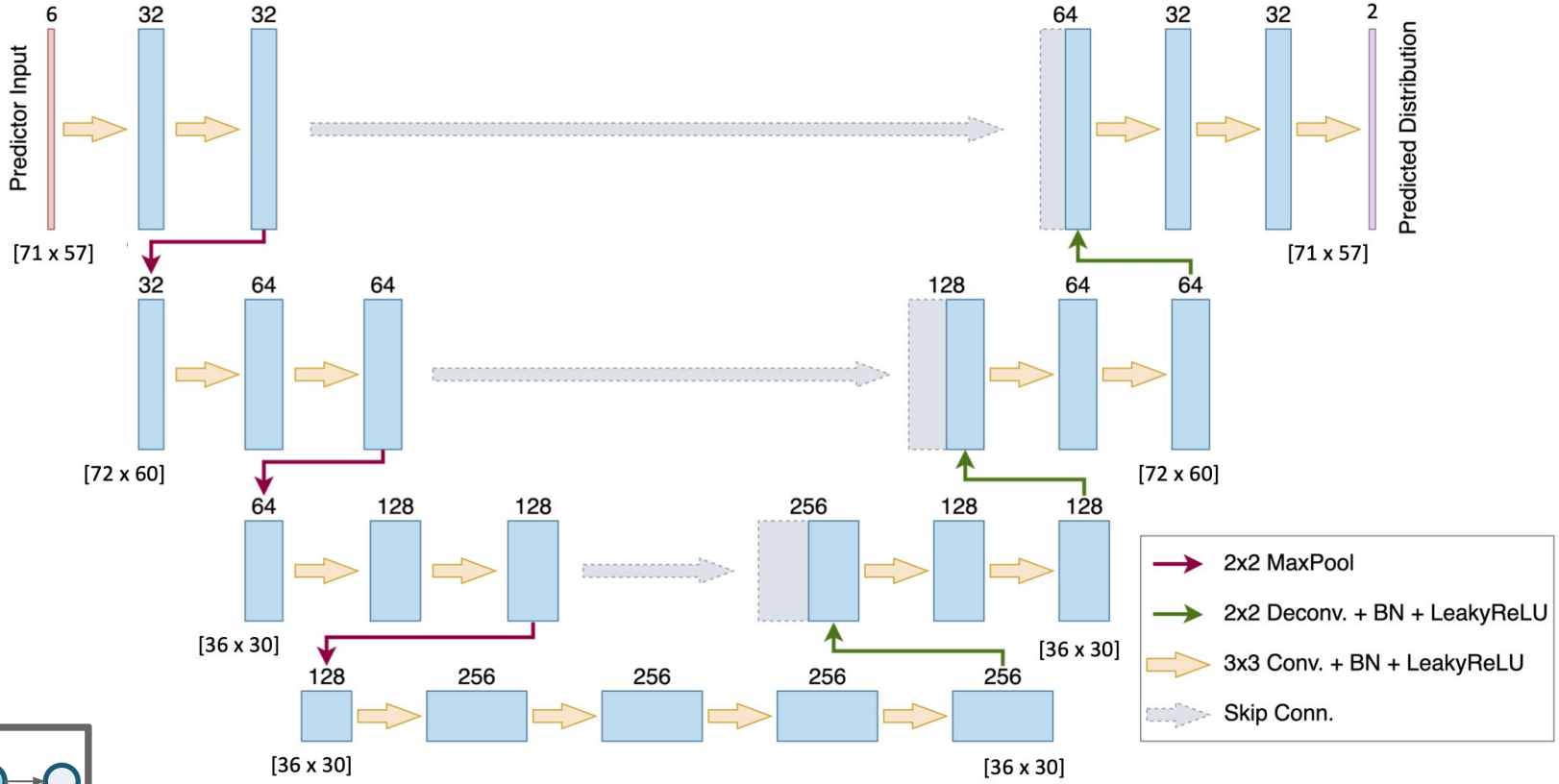
Can this process be replaced with a Machine Learning pipeline to create a "state-dependent" correction?

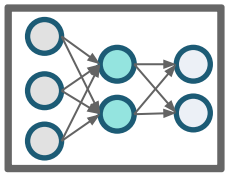


UNET – D.A. Error Emulator

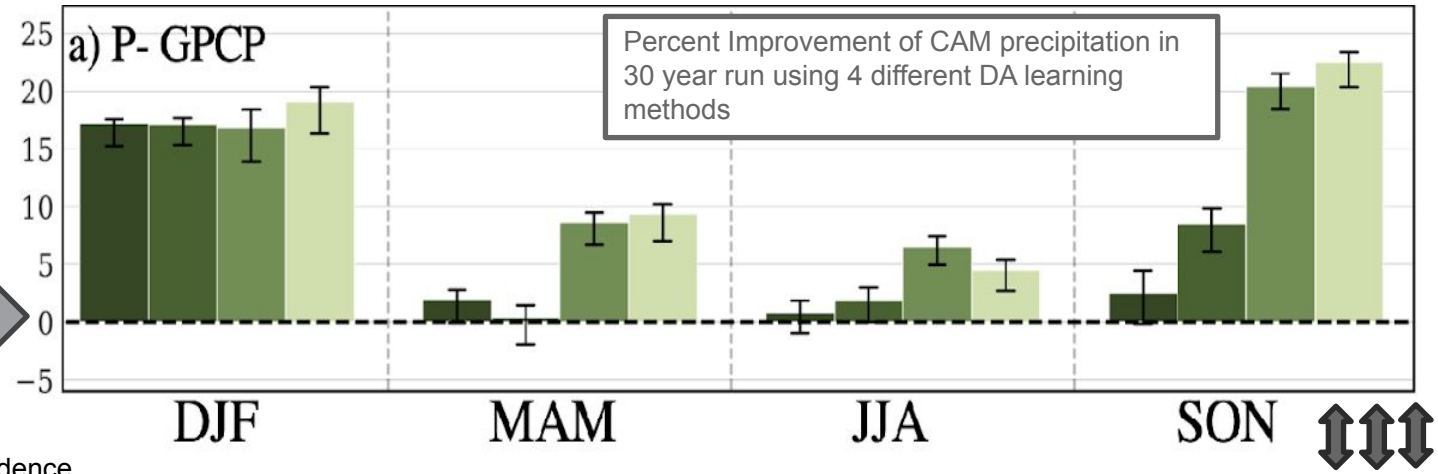
INPUT: Model State + 5 predictor variables at each model level

OUTPUT: D.A. Tendencies

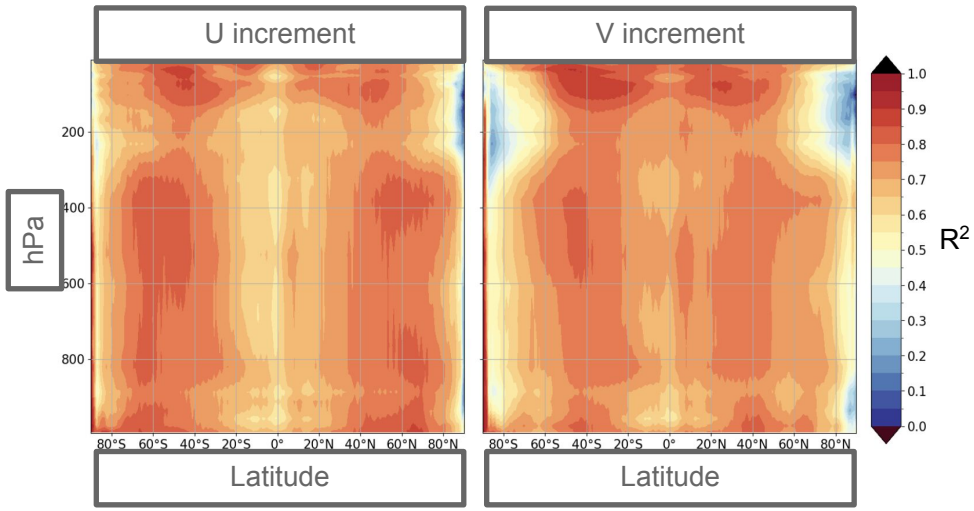




CNN Running in CESM2.1



State Dependence CNN



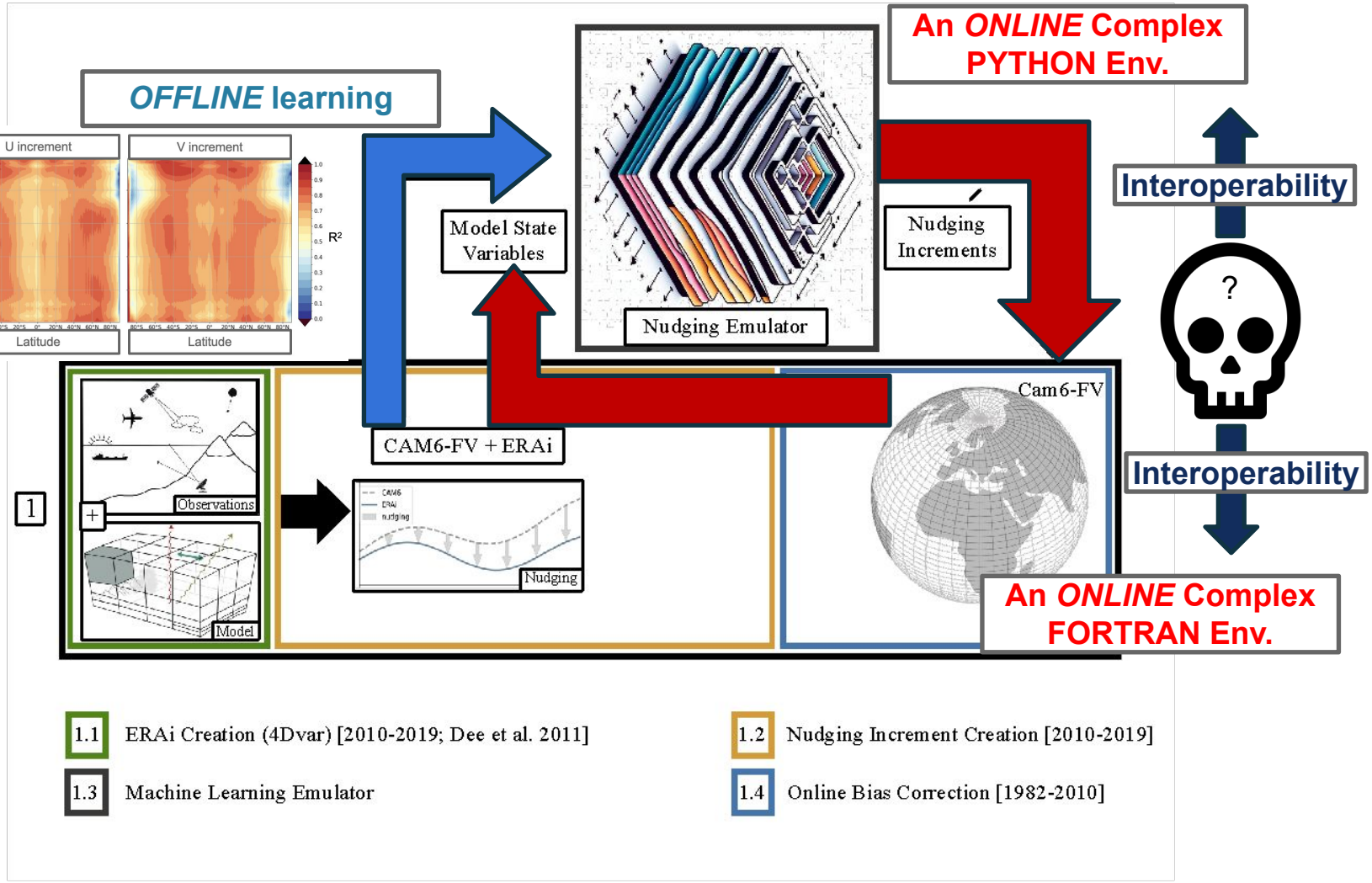
Quarterly Journal of the Royal Meteorological Society
RMetS
RESEARCH ARTICLE

Deterministic and Stochastic Tendency Adjustments Derived from Data Assimilation and Nudging

William E. Chapman PhD, Judith Berner PhD

Machine Learning Emulation

Can this process be replaced with a Machine Learning pipeline to create a "state-dependent" correction?



Forpy

Forpy: A library for Fortran-Python interoperability.

Forpy allows you to use Python features in Fortran ("embedding Python in Fortran")

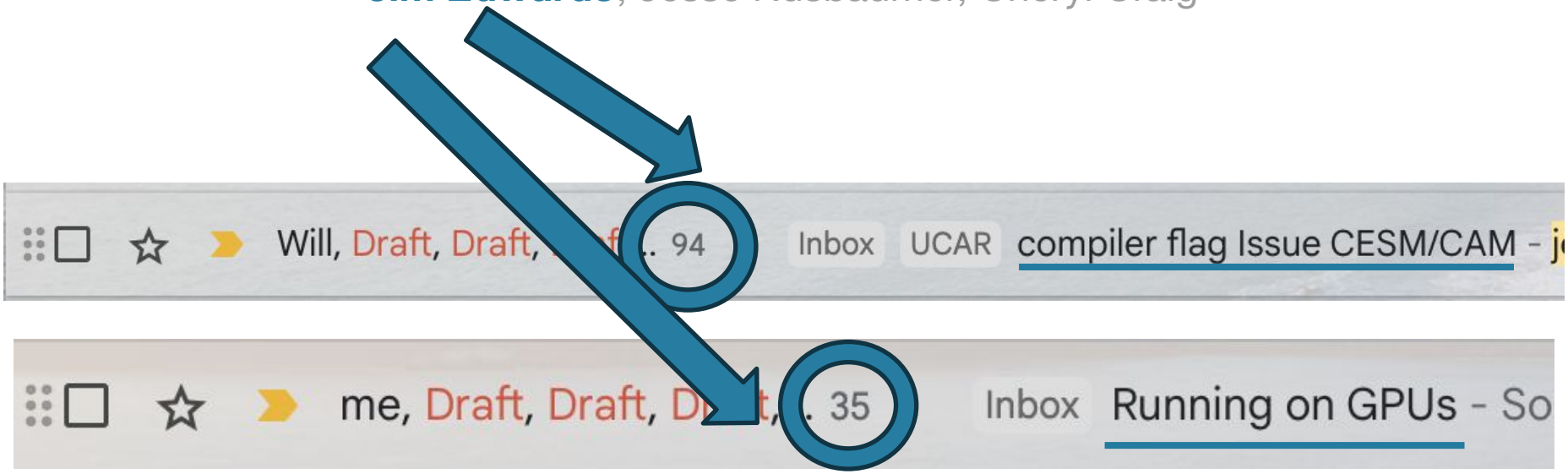
It provides datastructures such as list, dict, tuple and interoperability of arrays using numpy. With forpy you can even import Python modules in Fortran. Simply use your own or third-party Python modules for tasks that you can easily do in Python. For example: plot with matplotlib or use scientific functions from scipy or numpy.

Forpy also works to other way around: You can write Python modules entirely in Fortran (extending Python with Fortran - "Fortran in Python").

https://github.com/ylikx/forpy

Big Shout Out:

Jim Edwards, Jesse Nusbaumer, Cheryl Craig



What needed to happen?

1. Create a ***Machine learning*** Python CONDA environment
2. Place FORPY toolbox so it is discoverable by CESM (externals)
3. ***Iterate 1 million times*** in the CIME build to add compiler flags in the appropriate locations.
4. Make it so CESM can find the CONDA env. and the python scripts (CIME)
4. Write Fortran handshaking script

```
if (masterproc) write(iulog,*)'... placed ndarray ...'
ierror = call_py(return_value,pymodule,"DAMLcnn_run", args)
if (ierror/=0) then; call err_print; endif
```

What needed to happen?

1. Create a ***Machine learning*** Python CONDA environment
2. Place FORPY toolbox so it is discoverable by CESM (externals)
3. ***Iterate 1 million times*** in the CIME build to add compiler flags in the appropriate locations.
4. Make it so CESM can find the CONDA env. and the python scripts (CIME)
4. Write Fortran handshaking script
5. Write Python Handshaking script
6. Party Hard

```
def DAMLcnn_run(*args):
```

The final form of the externals are updated here for a simple working example:

https://github.com/WillyChap/CESM/tree/forpy_cesm
https://github.com/WillyChap/CAM/tree/forpy_cam_rel60
https://github.com/jedwards4b/cime/tree/forpy_cime

Hiccups / Caveats

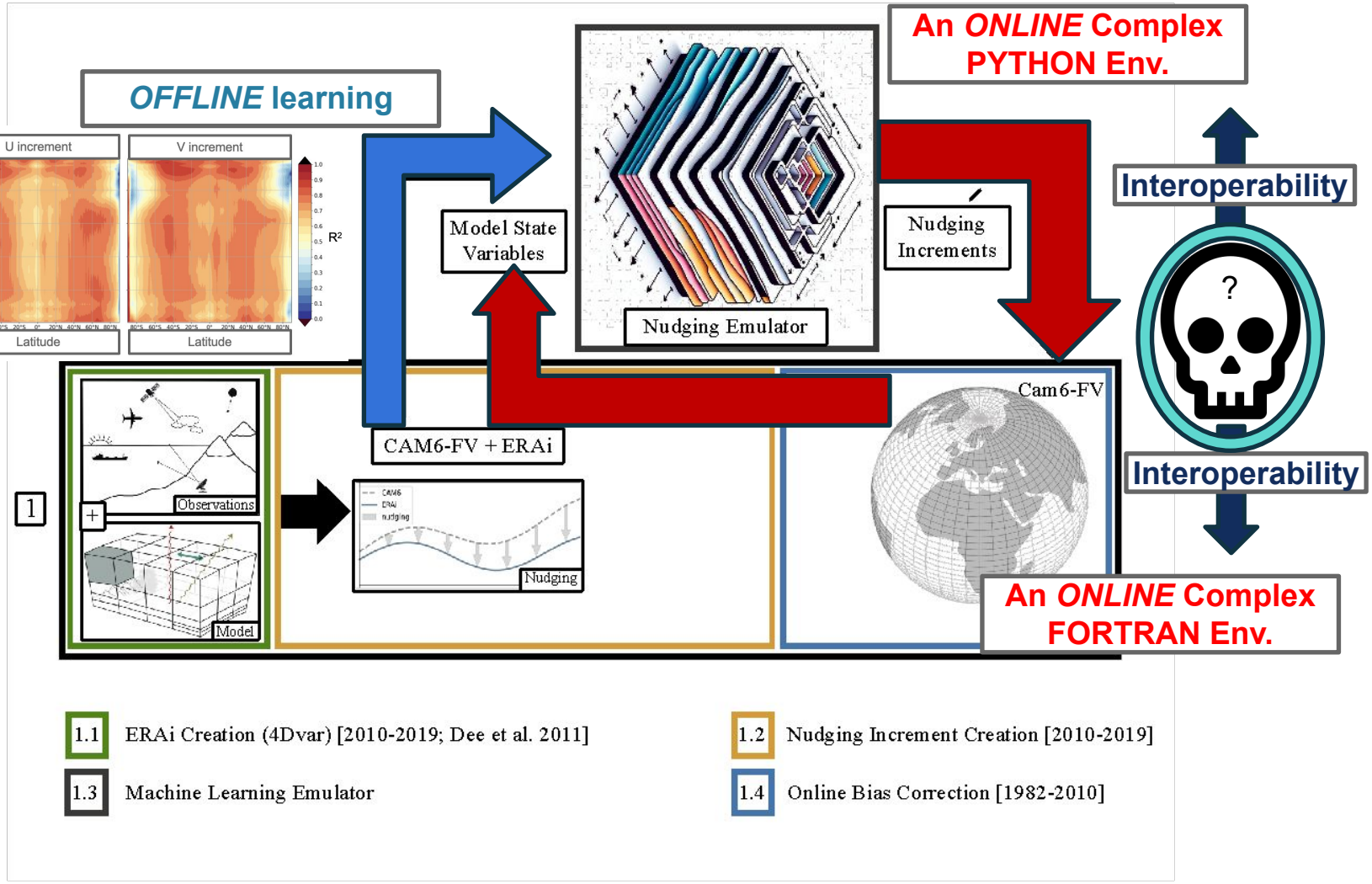
1. Too many small hiccups to name
2. The tuple that is passed in is initially empty? `def DAMLcnn_run(*args):`
3. a.t.m. python must be between 3.8 and 3.10 (***for the build***)
4. No NETCDF libraries in the python conda ENV (***for the build***)
5. import numpy causes floating point overflow (IEEEE_exceptions wrapper)

XML Options/Other stuff in Case:

./xmlchange USE_CONDA=TRUE
export PYTHONPATH=`pwd`/SourceMods/src.cam/:$PYTHONPATH

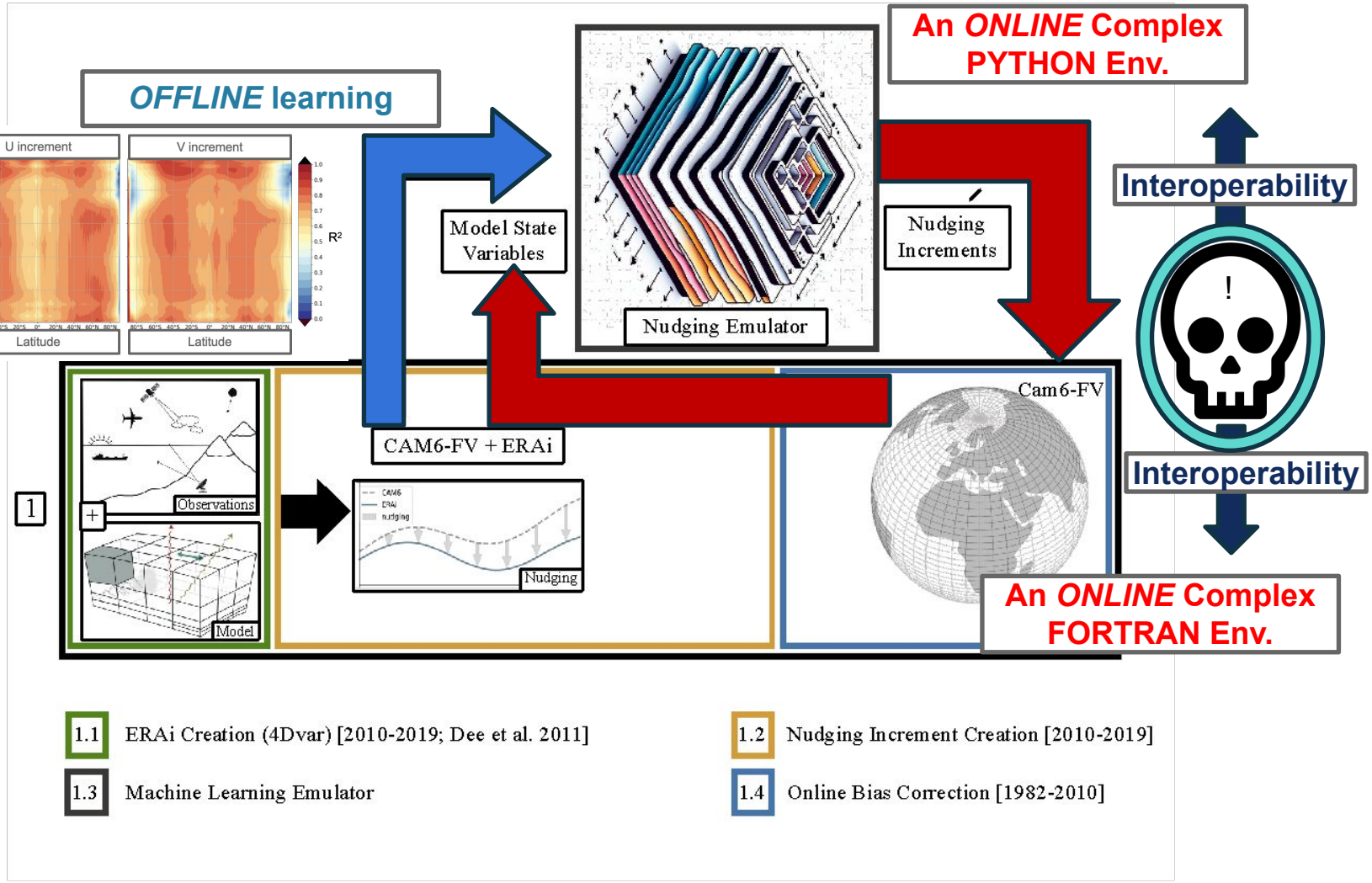
Machine Learning Emulation

Can this process be replaced with a Machine Learning pipeline to create a “state-dependent” correction?



Machine Learning Emulation

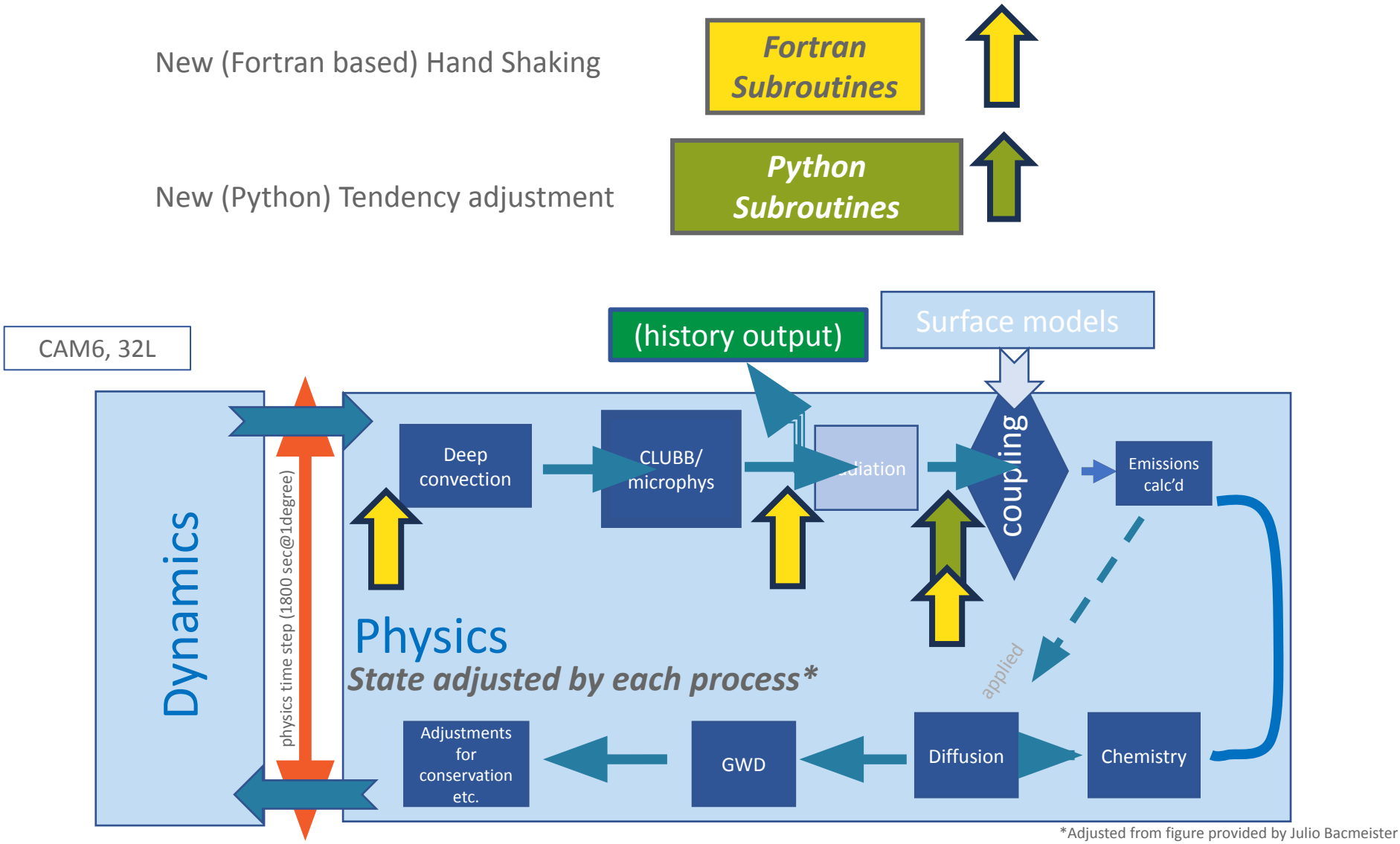
Can this process be replaced with a Machine Learning pipeline to create a “state-dependent” correction?



New Error Parameterization:

New (Fortran based) Hand Shaking

New (Python) Tendency adjustment



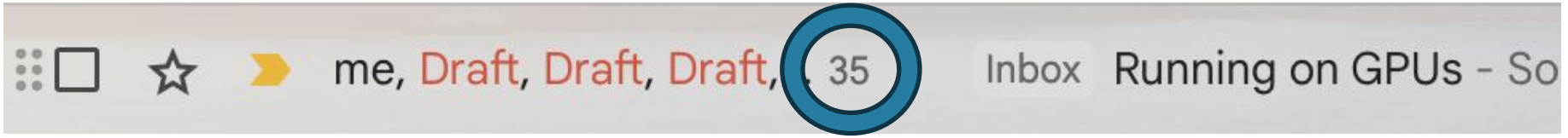
*Adjusted from figure provided by Julio Bacmeister

What needed to happen?

1. Derecho-GPU machine definition
https://github.com/jedwards4b/cime/tree/forpy_cime
./create_newcase --case ./example_case ***--mach derecho-gpu***

2. Enable Python / CUDA to discover GPUs
   - pytorch / CUDA conda build;

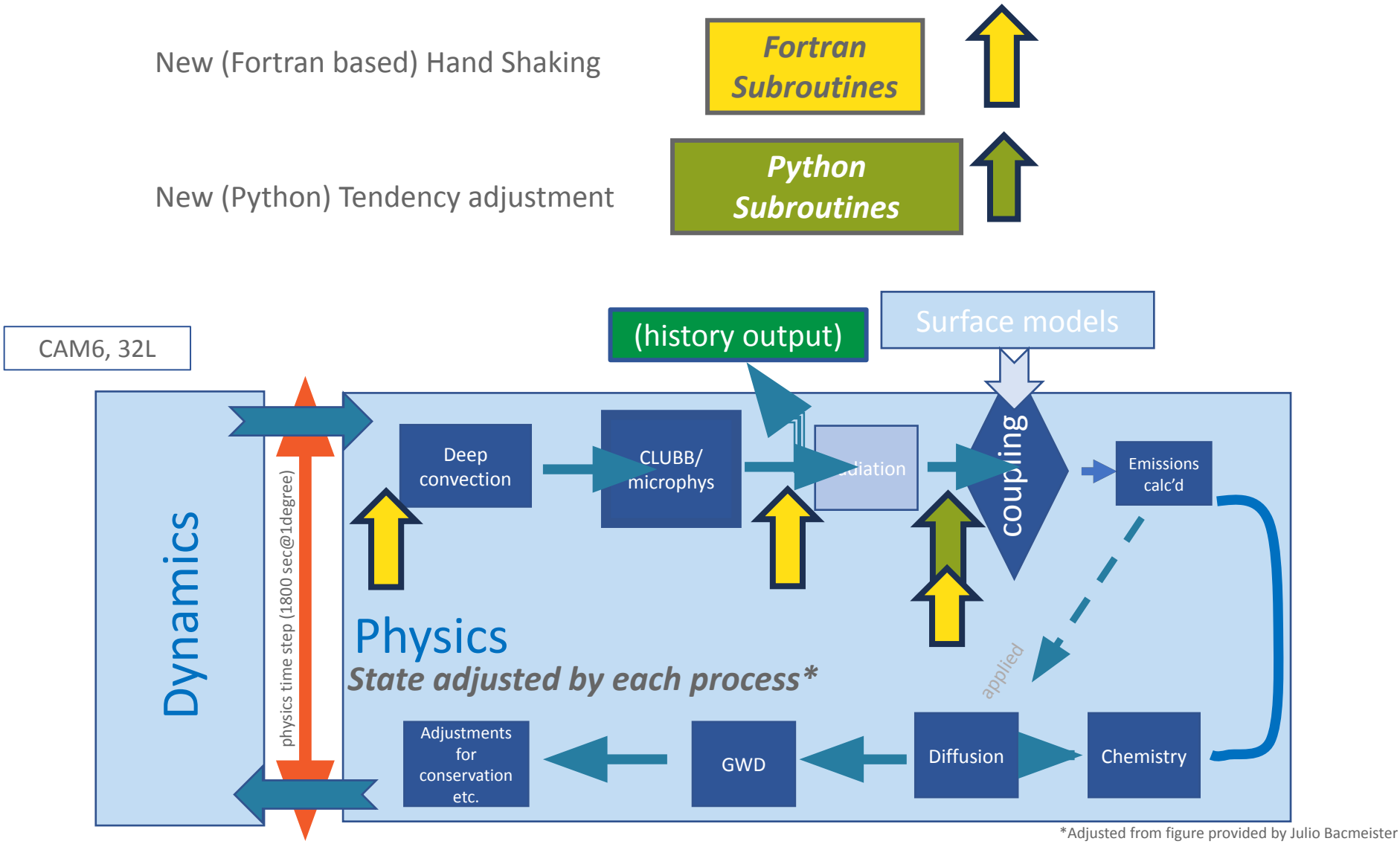
```
def set_gpu(gpu_id):
    os.environ["CUDA_DEVICE_ORDER"] = "PCI_BUS_ID"
    os.environ["CUDA_VISIBLE_DEVICES"] = str(gpu_id)

if gpu_id >= 0:
    device = "cuda"
    set_gpu(gpu_id)
    print('device available :', torch.cuda.is_available())
    print('device count: ', torch.cuda.device_count())
    print('current device: ',torch.cuda.current_device())
    print('device name: ',torch.cuda.get_device_name())
    device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')
```



New Error Parameterization:

New (Fortran based) Hand Shaking

New (Python) Tendency adjustment



*Adjusted from figure provided by Julio Bacmeister

Final Thoughts:

- We have online state-dependent, spatially aware, error tendency corrections to the U,V,T fields in CAM6 derived from the D.A. system.
  - Results are pending.
- ML is generally finicky … we need to be able to rapidly develop and test new architectures, which is preventative when you are manually writing FORTRAN based ML code.
- We can have GPU enabled machine learning CESM-wide (I have already set this up in another model component and it seems to work well).
- I am creating some documentation so that folks know exactly how to use this tool.
- Jim Edwards (and team) is creating a release so it can go into the main branch of CESM2.1.5
- My ”best practices” are growing… but far from complete.

- **The Good:** this doesn’t have to be used just for Machine learning, and can enable other online adjustments via python utility.
- **The Bad:** We still need to test how this scales given the problem space …. Other, less flexible, solutions are out there; There are VERY specific orders that things have to be done; CONDA build env is specific (but just the build)

Questions?
I have limited answers to the interesting stuff….