# An opinion: ML for radiation

The first applications of ML in atmospheric modeling were emulators for radiation. This is a terrible idea.

Radiation packages combine empirical bits with analytic approximations to well-known governing equations

Using ML to solve the radiative transfer equation is like using ML for advection

So: use ML to speed the empirical bits, correct for errors in formulation ("physical regularization")… but don't emulate the radiative transfer solver

# ML as interpolator

In the last year I've worked with two different grad students building neural networks emulators for the table look-ups used by the "gas optics" in RRTMGP. (See Menno Veerman's talk in #D4S1BR#3 Infrastructure)

It's computationally cheap to compute training data but takes care to span the input space. Both studies started with sparse samples and in-filled.

Physical understanding was key to generalizability (training data arrives as vertical integral; generalization requires fitting specific absorption)

Interesting discussions of pragmatic issues including hyper parameter tuning:

Menno Veerman et al: https://doi.org/10.1098/rsta.2020.0095
(preprint at https://arxiv.org/abs/2005.02265)

Peter Ukkon et al: https://doi.org/10.1029/2020MS002226 (coming soon)

# ML for gas optics++

The resulting networks are small, simple, and accurate. They are faster because computational intensity and
highly-optimized routines

We could provide ML-accelerated gas optics in a week if we had a robust way to efficiently batch-evaluate neural networks from within Fortran and/or C++

# ML for gas optics- -

"Faster" means ~2x and relies on bespoke implementations of neural network evaluators with ugly GPU implementations.

Sam Silva gave a shout-out to UCI's more general FKB, but this relies on the speed of a Fortran intrinsic for efficiency (ineffective especially cross-architecture). It also implements a small subset of possible architectures.

Maybe we shouldn't be reinventing the wheel? Could we build Fortran hooks to existing (efficient, GPU-ified) frameworks (Pytorch, Keras)? Or will the overhead be too large?

These aren't idle questions - I'm working with Nvidia, Menno, Mike and others to try strategies. But DOE/E3SM/….  could travel this path too.

Robust infrastructure to efficiently evaluate ML models from within Fortran and C++ would get gobbled up.