iWet: The Intelligent WRF Ensemble Tool

Leveraging deep learning hyperparameter tuning frameworks

Meteorology and Climate - Modeling for Air Quality

UC Davis Conference Center • September 11-13, 2019

Data Assimilation & Inverse Modeling

Derek Jensen, Donald Lucas, Clifford Anderson-Bergman, Sonia Wharton

LLNL-PRES-789577 This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under contract DE-AC52-07NA27344. Lawrence Livermore National Security, LLC

Lawrence Livermore National Laboratory

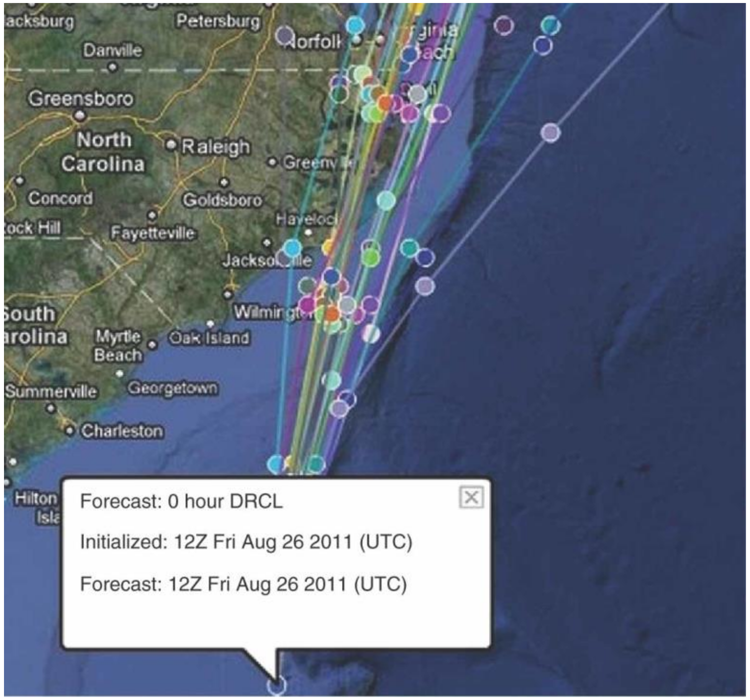
Ensembles can be great

- Ensemble Benefits
  1. Often provide more accurate forecasts
  2. Quantify uncertainty

- Ensembles seek to diagnose error due to
  1. Imperfect initial conditions
  2. Model imperfections

- Types of Ensembles
  - Initial Conditions
  - Boundary Conditions (For local-area models)
  - Observational Data Assimilation
  - Multi-model
  - Multi-physics
  - Perturbed Physics

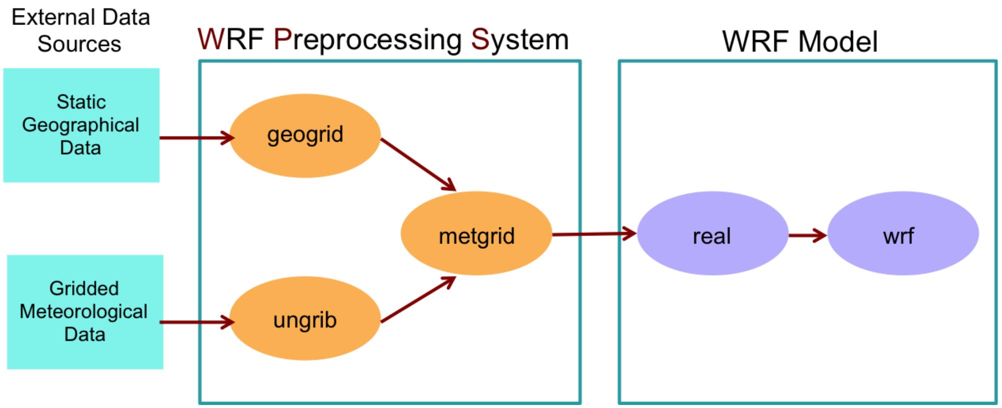
- Ensemble Challenges
  - Easy to design impossibly large ensembles
  - Junk ensemble members artificially inflate uncertainty
  - Unsampled sources of uncertainty create false confidence
  - Expensive to run and post process



Example Hurricane Spaghetti Plot
https://doi.org/10.1002/wcc.187

Intelligent WRF Ensemble Tool Wishlist

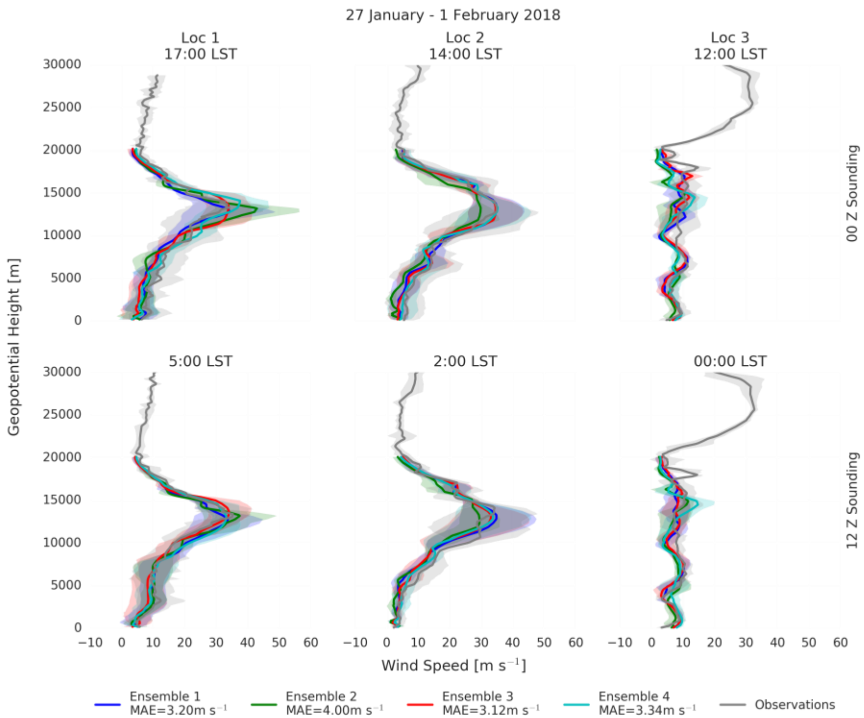
- Automate entire WRF workflow
  - Download met data
  - Generate directory structure
  - Minimize duplicate work
  - Parallel where possible
  - Support restarts
  - Convenient for single runs
- WRF Version agnostic
  - Modifies user-specified namelist templates
- Lightweight
  - Single input deck
- Parameter Sampling
  - Run all combinations
  - Randomly sample subset
  - Intelligently select trials
  - Early stopping for low-performing trials
- Address Ensemble Challenges
  - Handle large, multi-dimensional ensembles
  - Prevent junk ensemble members
  - Sample many sources of uncertainty
  - Easy to run and post process



WRF model workflow

The WRF Ensemble Tool (Wet) is a Good Start

- Automate entire WRF workflow
  - Download met data ✓
  - Generate directory structure ✓
  - Minimize duplicate work ✓
  - Parallel where possible ✓
  - Support restarts ✓
  - Convenient for single runs ✓
- WRF Version agnostic
  - Modifies user-specified namelist templates ✓
- Lightweight
  - Single input deck ✓
- Parameter Sampling
  - Run all combinations ✓
  - Randomly sample subset ✓
  - Intelligently select trials ✗
  - Early stopping for low-performing trials ✗
- Address Ensemble Challenges
  - Handle large, multi-dimensional ensembles !
  - Prevent junk ensemble members !
  - Sample many sources of uncertainty ✓
  - Easy to run ✓ and post process ✗



4-member Wet ensemble validated against radiosonde obs.

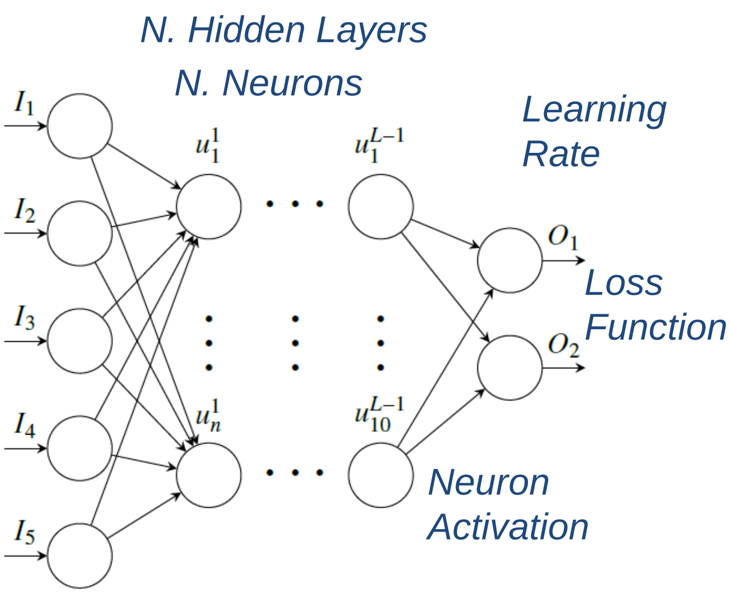
```
WRF_NAMELIST_CHANGES = pd.Series(
    ('time_control', 'run_days'): "0, ",
    ('time_control', 'run_hours'): "12, ",
    ('physics', 'mp_physics'): ['1', '2',],
    ('physics', 'bl_pbl_physics'): ['1', '2'], })
```

Tuning ML Models is Analogous to "Tuning" Ensembles

- ML performance depends on proper hyperparameter tuning

- Common Tunable ML Hyperparameters
  - Hidden Layers and Units
  - Regularization
  - Training Strategy
  - Etc.

- Tuning Methods
  - Manual Search — Ugh
  - Grid/Random Search — Wet
  - Bayesian Optimization — iWet
  - Early Stopping — iWet
  - Reinforcement Learning — I Wish

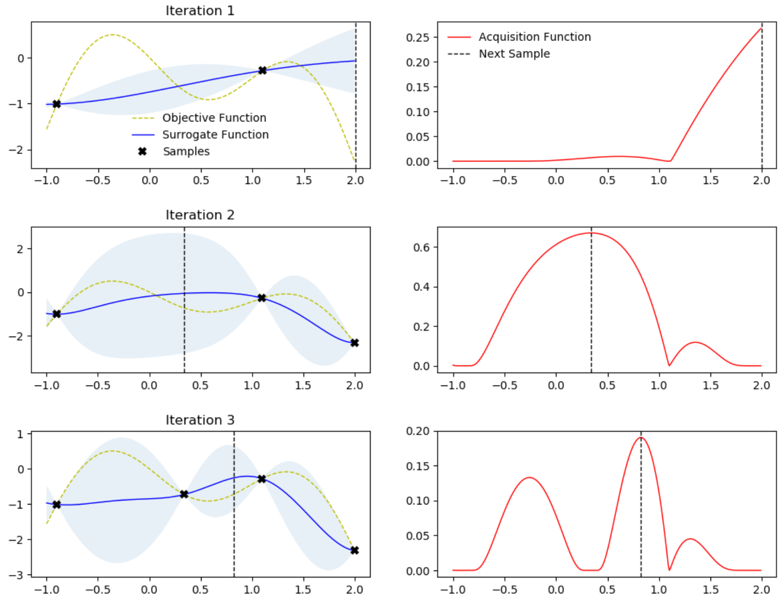


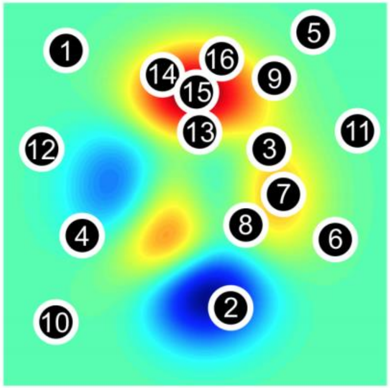
Sample Hyperparameters for a fully-connected DNN that accepts 5 inputs and returns 2 predictions

Intelligent Search – Bayesian Optimization

- Sequential Model-Based Optimization [1]
  - Sequentially fits a probabilistic surrogate model to samples of an unknown objective function
  - An acquisition function chooses the next set of hyperparameters

Aspects of SMBO	ML Tuning	WRF Ensemble
1. Parameter Space	n. layers, dropout, etc.	ICBC, DAS, MP, SP
2. Objective Function	ML Training	Running WRF
3. Surrogate Model	e.g. Gaussian Process	e.g. Gaussian Process
4. Acquisition Function	e.g. Expected Improvement	e.g. Expected Improvement
5. Performance History	✓	✓





(top left) Example of the unknown objective function and surrogate model and (top right) acquisition function [2] (bottom) Example of SMBO convergence [3]

[1] Koehrsen, Will "A Conceptual Explanation of Bayesian Hyperparameter Optimization for Machine Learning" *Toward Data Science*, Jun 24, 2018

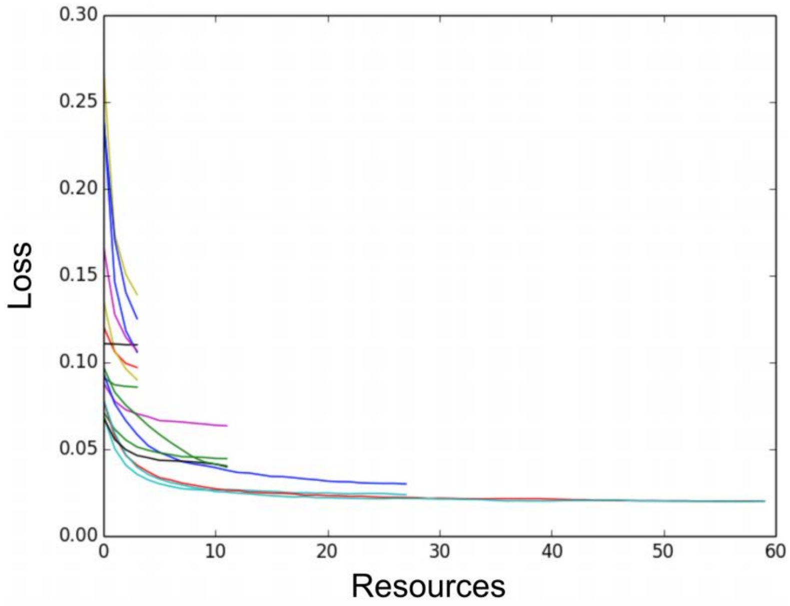
[2] Krasser, Martin "Bayesian Optimization" krasserm.github.io, March 21, 2018

Intelligent Scheduling – Early Stopping

- Hyperband Early Stopping: Focus on hyperparameter *evaluation*, not *selection* to optimize your compute resources [3]

- A bandit-based approach to optimization
  - Online algorithm to maximize return on investment. Which slot machine should the gambler play? [4]

- Li et al. reports that intelligent scheduling beats intelligent search



Example of scheduling methods like early stopping [3]

[3] Li, Lisha, et al. "Hyperband: a novel bandit-based approach to hyperparameter optimization." The Journal of Machine Learning Research 18.1 (2017): 6765-6816.
[4] Davidson-Pilon, Cameron. Bayesian methods for hackers: probabilistic programming and Bayesian inference. Addison-Wesley Professional, 2015.

Ray Tune Scales Intelligent Search & Scheduling

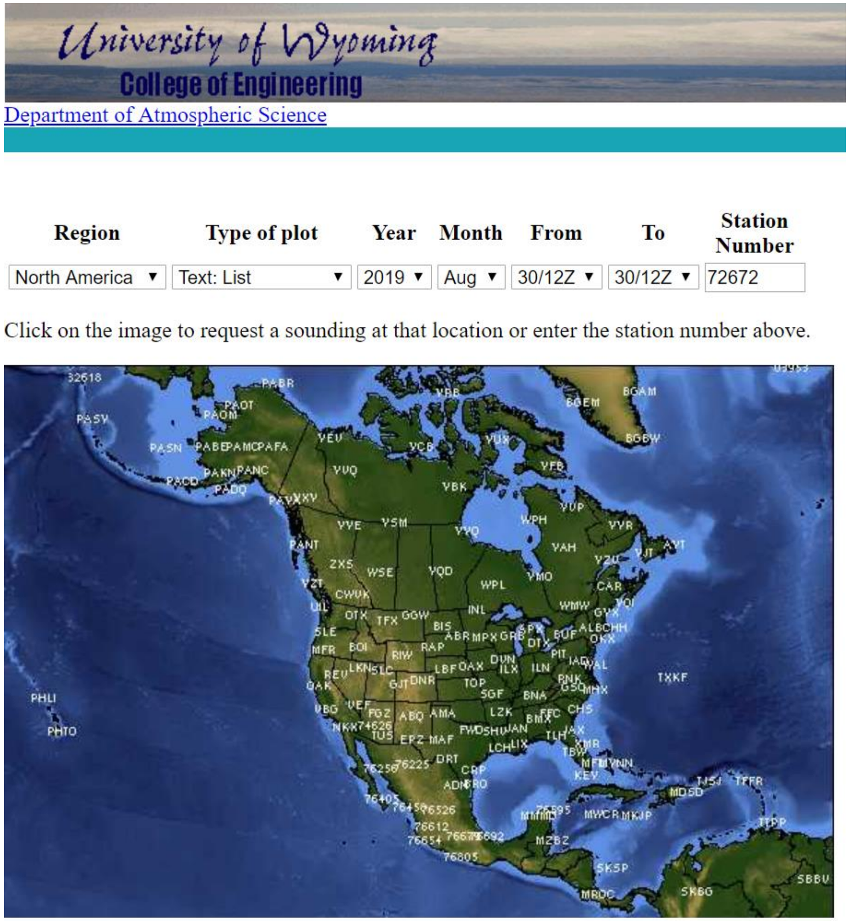
- Ray is a fast and simple framework for building and running distributed applications [5]
  - Multi node parallelization
  - Graceful error handling
  - Efficiently handles large objects
  - Easy to implement with a single Python decorator `@ray.remote`
  - Includes several ML libraries

- Tune is a Ray library for hyperparameter tuning at any scale [6]
  - Pair a Search algorithm with a scheduler
  - Search Algorithms:
    - Random Search
    - HyperOpt*
    - Nevergrad
    - Scikit-Optimize
  - Schedulers
    - Population Based Training
    - Hyperband*
    - Median Stopping Rule

[5] Moritz et al., "Ray: A Distributed Framework for Emerging AI Applications" arXiv:1712.05889v2
[6] Liaw et al., "Tune: A Research Platform for Distributed Model Selection and Training" arXiv preprint arXiv:1807.05118, 2018

What is an appropriate tuning reward??

- An execution of a WRF trial and subsequent reward/cost calculation constitute an evaluation of the objective function to be maximized/minimized
- The user needs to define an appropriate cost/reward
- One Idea: Utilize U. Wyoming's Weather Web API
- iWet will automatically download specified radiosonde sites and compute mean-absolute error

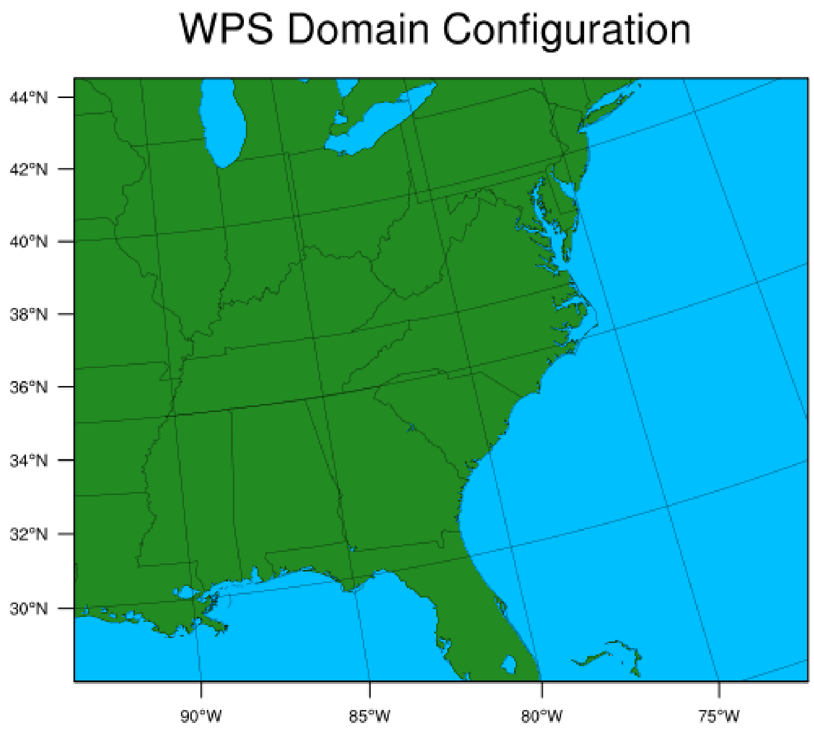



```
# http://weather.uwyo.edu/upperair/sounding.html
#                   (name,   region, id)
sonde_sites = [('BNA', '72493', 'naconf'),
               ('BMX', '72230', 'naconf'),
               ('JAX', '72206', 'naconf')]
```

A very hasty "Hello, World!"

- Default WRF 4.0 namelist settings
- Ensemble Parameters:

Parameter	Values
Met	ECMWF, GFS
Surface Physics	YSU, MYJ, QNSE
Micro Physics	Options 0 – 4

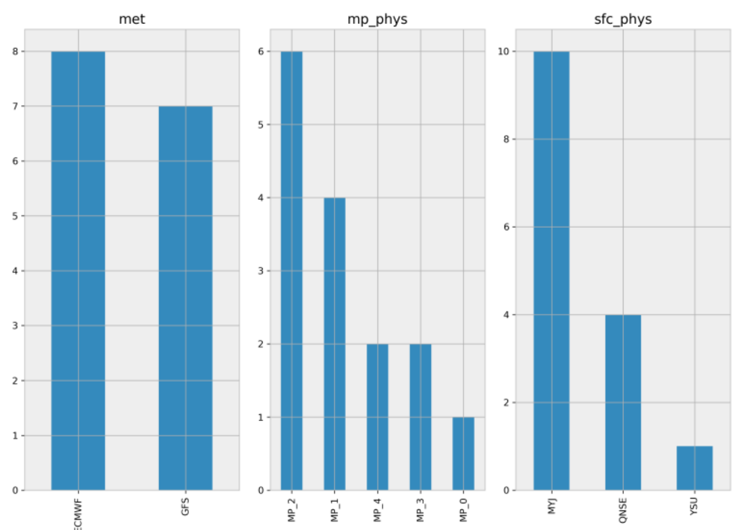


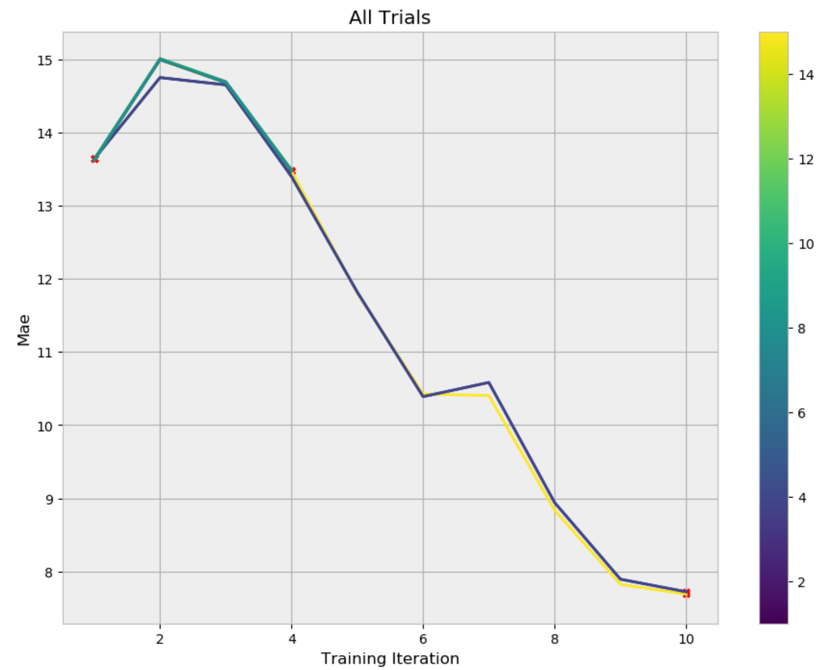
- $2 * 3 * 5 = 30$ Possible Combinations
- 15 Trials
- Allocate 1 CPU per WRF Run
- Calculate reward every 12 hours of model time

Trivial example but iWet appears to work

- The top-5 Trials

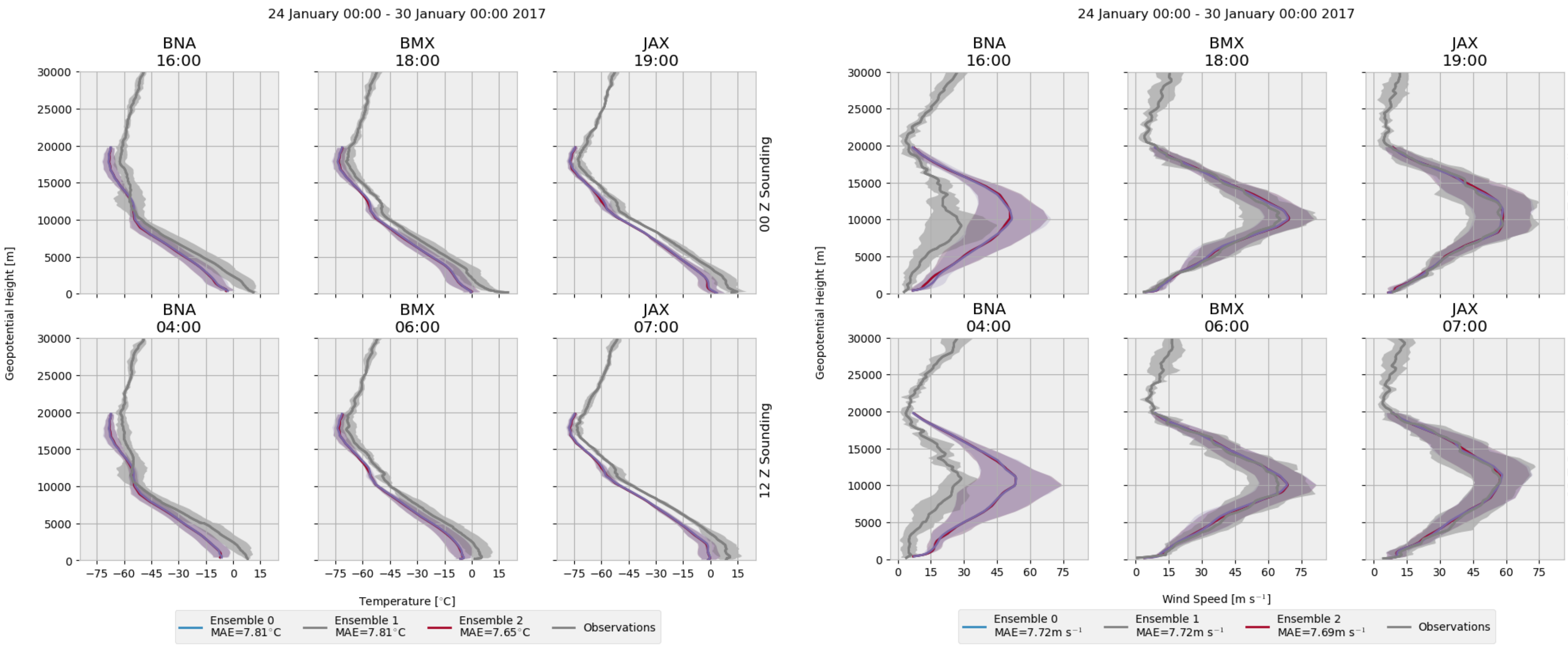
MAE	met	mp_phys	sfc_phys	run	n_it	best_cycle
7.687223	ECMWF	MP_1	MYJ	15	10	10
7.721463	GFS	MP_1	MYJ	3	10	10
13.474602	ECMWF	MP_1	QNSE	6	4	4
13.475293	ECMWF	MP_2	MYJ	10	4	4
13.475676	ECMWF	MP_2	QNSE	13	4	4





(left) Bayesian sampling (top) MAE as a function of training iterations – each iteration is 12 hours of forecast time

No member does particularly well



(left) Temperature validation (right) Wind speed validation

Conclusions

- Wet automates the entire WRF workflow and can brute force ensembles
- By utilizing libraries developed for neural network tuning, we made Wet intelligent
- There is still some cleaning and scaling to address but we hope to release iWet soon
- We are in the process of developing more interesting studies