DeepHyper: Scalable neural architecture search for surrogate modeling and uncertainty quantification

ML4I, Lawrence Livermore National Laboratory,
August 11, 2021.

Romit Maulik
MCS & LCF Divisions - Argonne National Laboratory
Applied Mathematics – Illinois Institute of Technology, Chicago

Outline

In this talk we shall go through

- Neural architecture search (NAS) at scale using DeepHyper.
- Surrogate model discovery for geophysical flows using NAS.
- Comparisons of NAS Surrogates with state-of-the-art forecast models.
- Using NAS for ensemble epistemic uncertainty quantification.
- Some other interesting tidbits.

Motivation for NAS

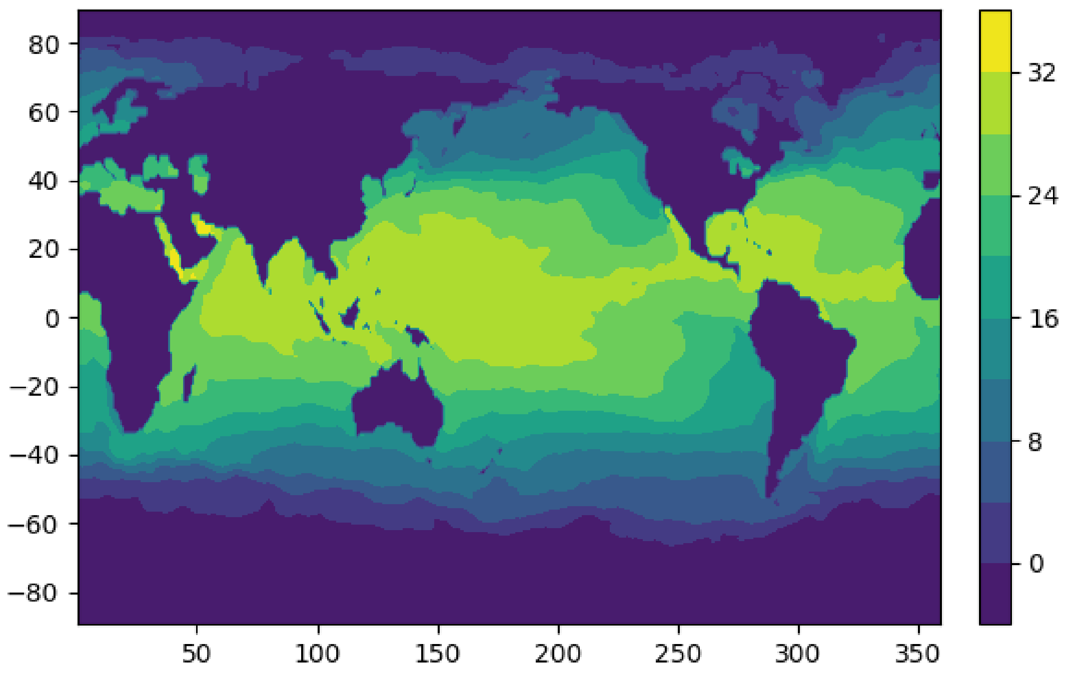
Surrogate models may be used for accelerating geophysical forecasting and downstream tasks such as data assimilation. PDE-based methods suffer from large compute/memory costs.

Two phase development for PDE-free forecasting

- **Surrogate formulation (dimension reduction)**
- **Neural network discovery (at scale).**

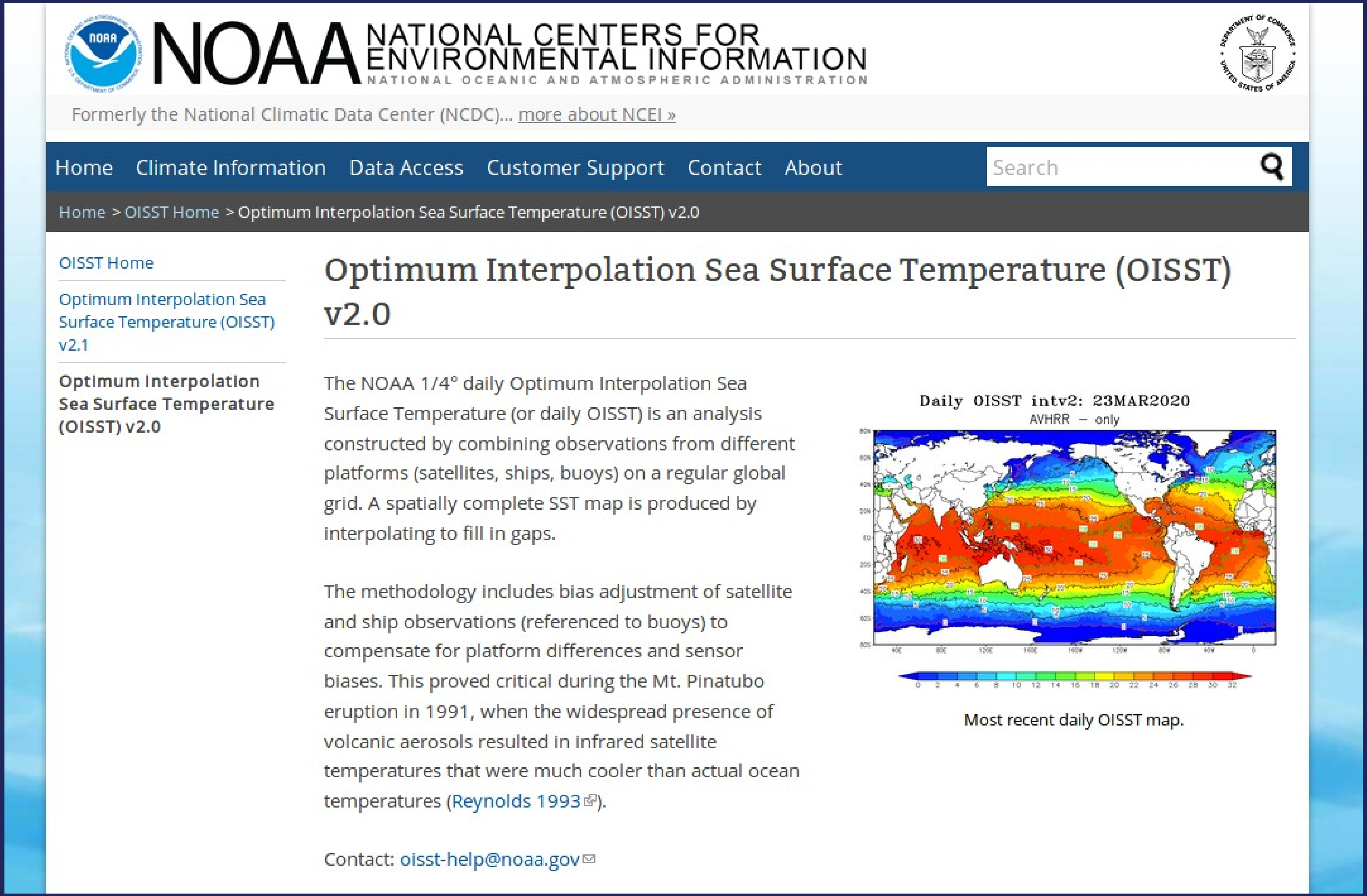
Temperature forecasting:

- **Weekly averaged sea-surface temperature**
- **Applications: forecasting ENSO/MJO phenomena, predicting aquatic migration patterns.**



J. Climate, 20, 5473-5496
ORNL DAAC, https://doi.org/10.3334/ORNLDAAC/1328

Our representative dataset



Originally available daily on 1/4^{o} grid - we down-sample to 1^{o} and average weekly.

Generated from satellites and ship observations.

Periodic dynamics (seasonal) but also full of long term patterns (El Niño)

https://www.ncdc.noaa.gov/oisst/optimum-interpolation-sea-surface-temperature-oisst-v20

The proper orthogonal decomposition

The Swiss-army knife of data analysis in computational physics

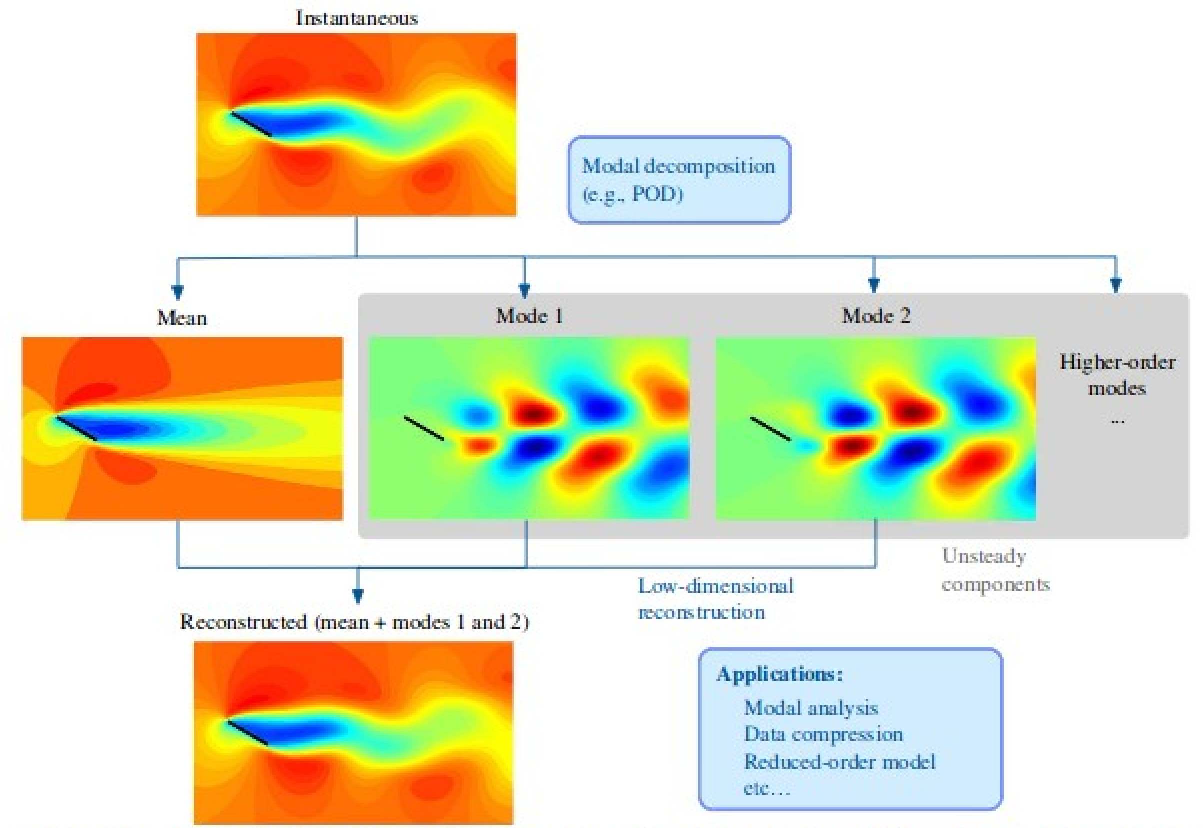


Fig. 1 Modal decomposition of two-dimensional incompressible flow over a flat-plate wing [25,26] ($Re = 100$ and $\alpha = 30$ deg). This example shows complex nonlinear separated flow being well represented by only two POD modes and the mean flowfield. Visualized are the streamwise velocity profiles.

POD-bases computed through **method of snapshots**

Method of snapshots finds orthonormal bases which are *ordered* according to variance capture (basically PCA)

Solves for the POD basis through an eigenvalue problem that scales with the number of snapshots

Taira et al., AIAA J., 55, 12, 2017

Long short-term memory neural networks

Specialized neural network architecture for handling data that are correlated in time.

$$G_i = \varphi_S \circ \mathcal{L}_i^{N_c}(a^n)$$

$$G_f = \varphi_S \circ \mathcal{L}_f^{N_c}(a^n)$$

$$G_o = \varphi_S \circ \mathcal{L}_o^{N_c}(a^n)$$

$$s^n = G_f \odot s^{n-1} + G_i \odot \left(\varphi_T \circ \mathcal{L}_z^{N_c}(a^n)\right)$$

$$h^n = G_o \odot \varphi_T\left(s^n\right)$$

$$a^{n+1} = \mathbb{F}(h^n)$$

State flow in time
Allows for non-Markovian assumptions

- → The LSTM is a specialized architecture that allows for forecasting of temporal (non-i.i.d) data
- → The above set of equations is how LSTMs are generally used (1 cell)
- → LSTMs are also occasionally *stacked*

Hochreiter, Sepp, and Jürgen Schmidhuber. "Long short-term memory." Neural computation 9.8 (1997): 1735-1780.

Back to our first problem

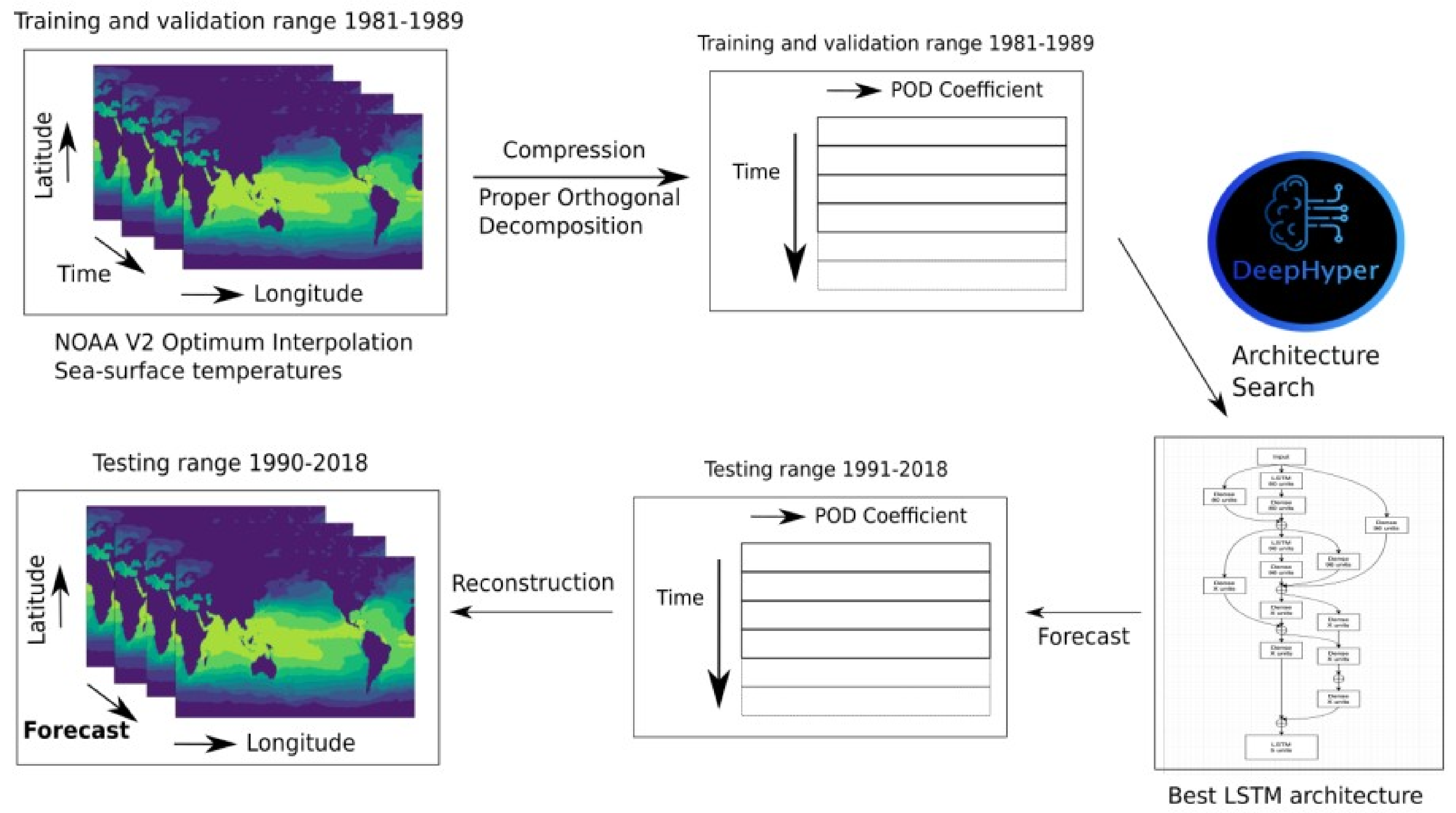


Fig. 1. Our proposed NAS approach for automated POD-LSTM development. Snapshots of spatiotemporally varying training data are compressed by using proper orthogonal decomposition to generate reduced representations that vary with time. These representations (or coefficients) are used to train stacked LSTMs that can forecast on test data. The POD basis vectors obtained from the training data are retained for reconstruction using the forecast coefficients.

The DeepHyper Project

DeepHyper is a scalable hyperparameter and neural architecture search package for leadership class computing systems

Applications: Cancer drug response, geophysical surrogate modeling, neuromorphic computing, nuclear physics.

This talk:

1. **Discover LSTM architectures.**
2. **Discover compression frameworks.**
3. **Use NAS-discovered models for ensemble UQ.**

https://github.com/deephyper/deephyper

Scalable reinforcement-learning-based neural architecture search for cancer deep learning research, SC 2019

Configuring a neural architecture search

How do we define a space of neural networks?

A neural network is represented as a directed acyclic graph with nodes and edges.

Nodes represent possible operations, for example:
1. “Add an identity layer”
2. “Add a layer with 40 neurons”
3. “Add a layer with 60 neurons”
4. “Add a dropout operation”
5. “Add a skip connection to another node”

Nodes can be constant – (i.e., predefined and immutable during the search)

Nodes can be variable – (i.e., the search can tweak these to get better performance)

Each variable node has an upper bound on the number of operations (which may be expressed as a categorical variable). Edges define the flow of the tensor in the graph.

DeepHyper NAS-API

```python
def create_search_space(input_shape=(8,5,),
                        output_shape=(8,5,),
                        num_layers=10,
                        *args, **kwargs):

    arch = KSearchSpace(input_shape, output_shape, regression=True)
    source = prev_input = arch.input_nodes[0]

    # look over skip connections within a range of the 2 previous nodes
    anchor_points = collections.deque([source], maxlen=2)

    for _ in range(num_layers):
        vnode = VariableNode()
        add_lstm_seq_(vnode)

        arch.connect(prev_input, vnode)

        # * Cell output
        cell_output = vnode

        cmerge = ConstantNode()
        cmerge.set_op(AddByProjecting(arch, [cell_output], activation='relu'))
        # cmerge.set_op(Concatenate(arch, [cell_output]))

        for anchor in anchor_points:
            skipco = VariableNode()
            skipco.add_op(Tensor([]))
            skipco.add_op(Connect(arch, anchor))
            arch.connect(skipco, cmerge)

        # ! for next iter
        prev_input = cmerge
        anchor_points.append(prev_input)

        # prev_input = cell_output
    cnode = ConstantNode()
    add_lstm_oplayer_(cnode,5)
    arch.connect(prev_input,cnode)

    return arch
```

Define the shape of our input/output tensors

Define the range of nodes to look for skip connections

Add an LSTM operation

```python
def add_lstm_seq_(node):
    node.add_op(Identity()) # we do not want to create a layer in this case
    #activations = [None, tf.nn.relu, tf.nn.tanh, tf.nn.sigmoid]
    for units in range(16, 97, 16):
        node.add_op(tf.keras.layers.LSTM(units=units, return_sequences=True))
```

Code to project tensors coming from skip connections

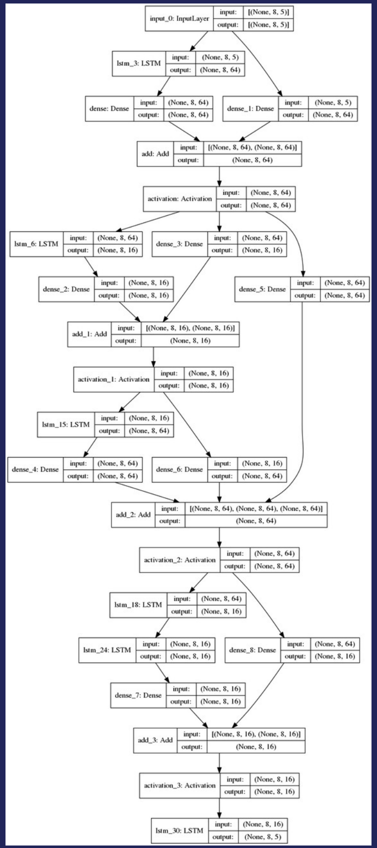
Connect to previous node

The output from the architecture is a constant operation for a consistent last dimension

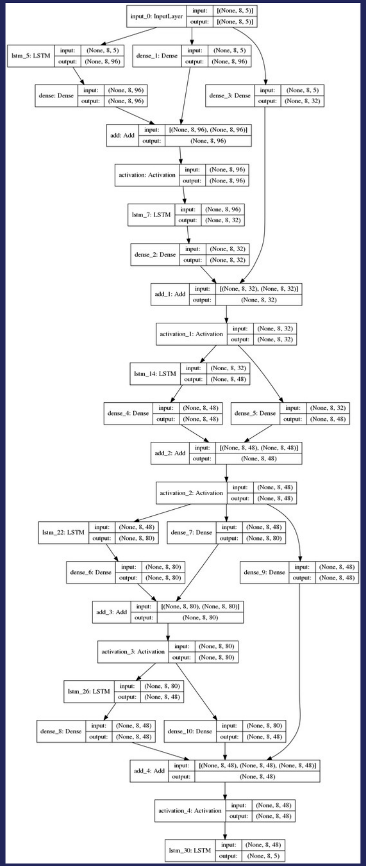
DeepHyper NAS-API

```
search_space = create_search_space(num_layers=5)
ops = [random() for _ in range(search_space.num_nodes)]
search_space.set_ops(ops)
model = search_space.create_model()
model.summary()
plot_model(model, to_file='sampled_neural_network.png', show_shapes=True)
print("The sampled_neural_network.png file has been generated.")
```

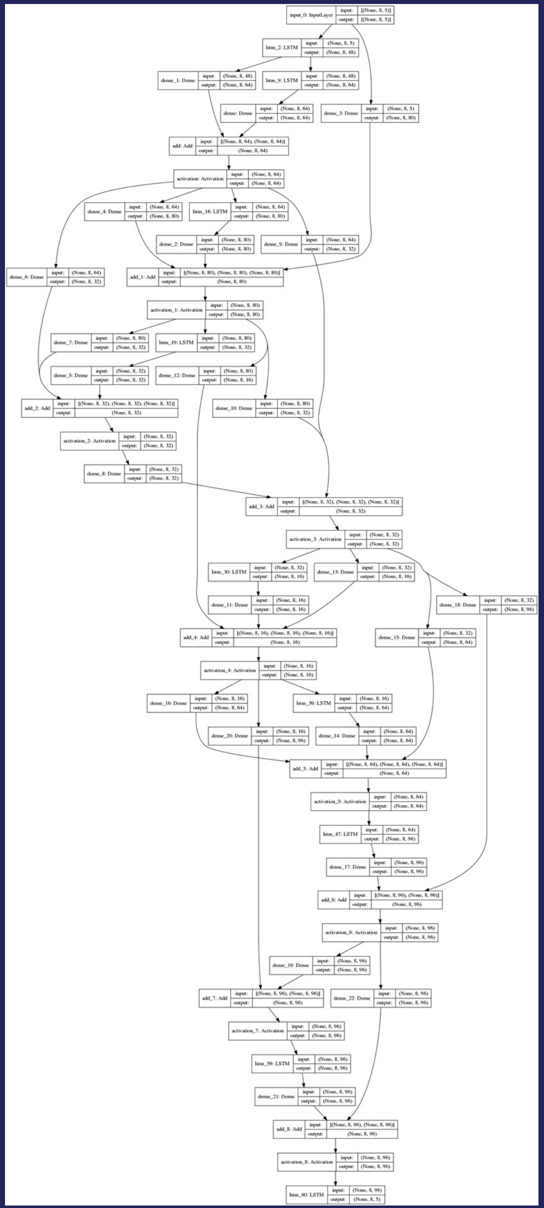
Fun to generate random architectures!



68,152 parameters



172,424 parameters



344,424 Parameters (more skips/layers)

DeepHyper on Theta

1. Multiple compute nodes of Theta can evaluate different architectures (asynchronously*[1])

2. Balsam is used to schedule the different evaluations (integrated into DeepHyper)

3. Two bash commands to fire off a multiple compute node search once load_data and search_space functions are ready.

1. Search strategies may affect this (to be continued)

Argonne NATIONAL LABORATORY

Exploring this search space intelligently

Regularized evolution to explore the search space of possible architectures

Algorithm 1 Aging Evolution

```
population ← empty queue                          ▷ The population.
history ← ∅                                       ▷ Will contain all models.
while |population| < P do                         ▷ Initialize population.
    model.arch ← RANDOMARCHITECTURE()
    model.accuracy ← TRAINANDEVAL(model.arch)
    add model to right of population
    add model to history
end while
while |history| < C do                            ▷ Evolve for C cycles.
    sample ← ∅                                    ▷ Parent candidates.
    while |sample| < S do
        candidate ← random element from population
                            ▷ The element stays in the population.
        add candidate to sample
    end while
    parent ← highest-accuracy model in sample
    child.arch ← MUTATE(parent.arch)
    child.accuracy ← TRAINANDEVAL(child.arch)
    add child to right of population
    add child to history
    remove dead from left of population           ▷ Oldest.
    discard dead
end while
return highest-accuracy model in history
```

Initialize architectures randomly

Evaluate their validation metrics

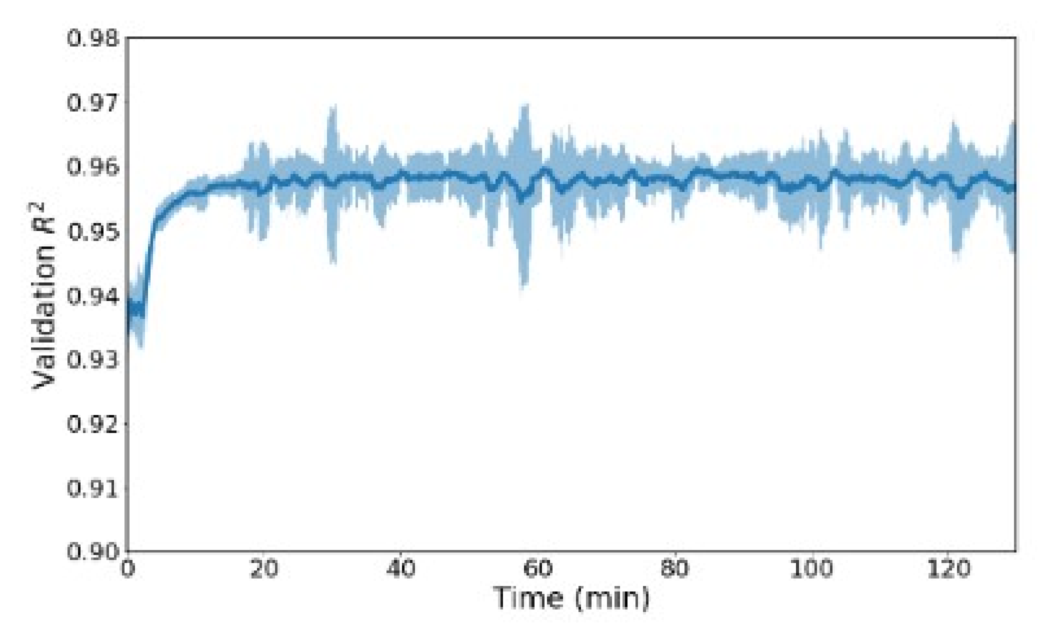
Choose some samples from the total population randomly

***Mutate* the highest accuracy network in the set of samples (perturb one variable node)**

Real, Esteban, et al. "Regularized evolution for image classifier architecture search." Proceedings of the aaai conference on artificial intelligence. Vol. 33. 2019.

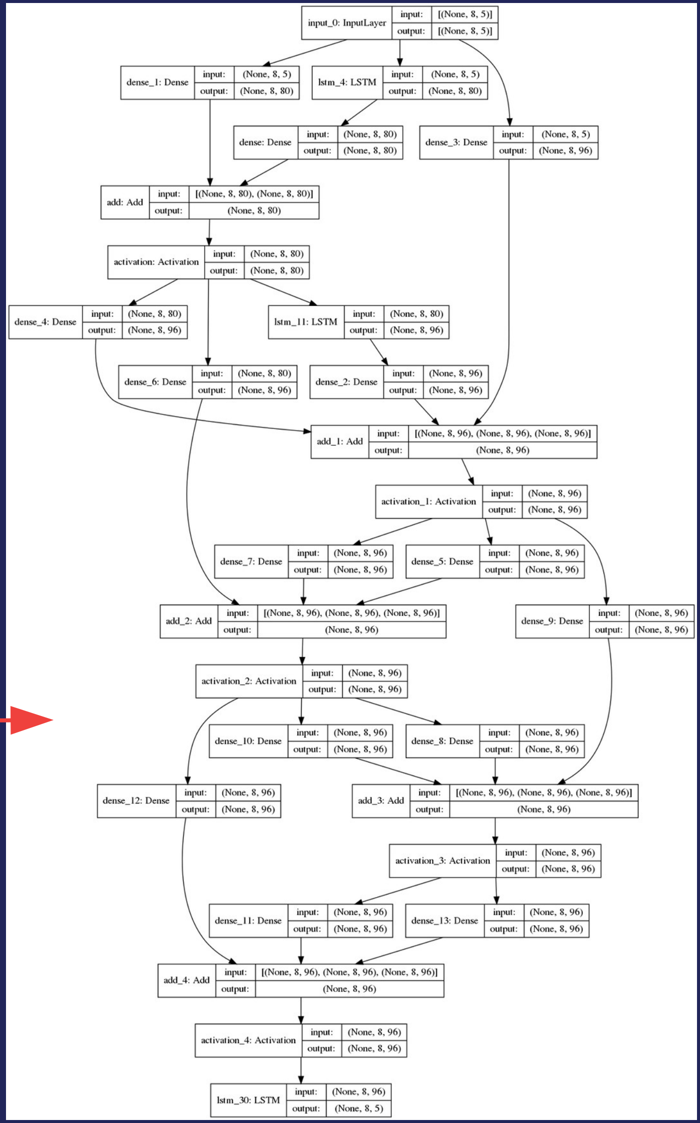
Searching for a surrogate LSTM

1. Experiment run on 128 compute nodes of Theta for 3 hours of wall time
2. Skip-connection look-back window of 2 nodes
3. Training for 20 epochs
4. Post-training for 100 epochs
5. Network with 5 layers



Validation R^2

Best model



Worth the cost?

A comparison with baseline ML methods

Model	NAS-LSTM	Linear	XGBoost	Random Forest	LSTM-40	LSTM-80	LSTM-120	LSTM-200
Training/Validation	**0.985**	0.801	0.966	0.823	0.916	0.931	0.9223	0.902
Testing	**0.876**	0.172	-0.056	0.002	0.742	0.734	0.746	0.739

We compare the performance of the LSTM obtained by DeepHyper against some baseline time-series forecasting methods.

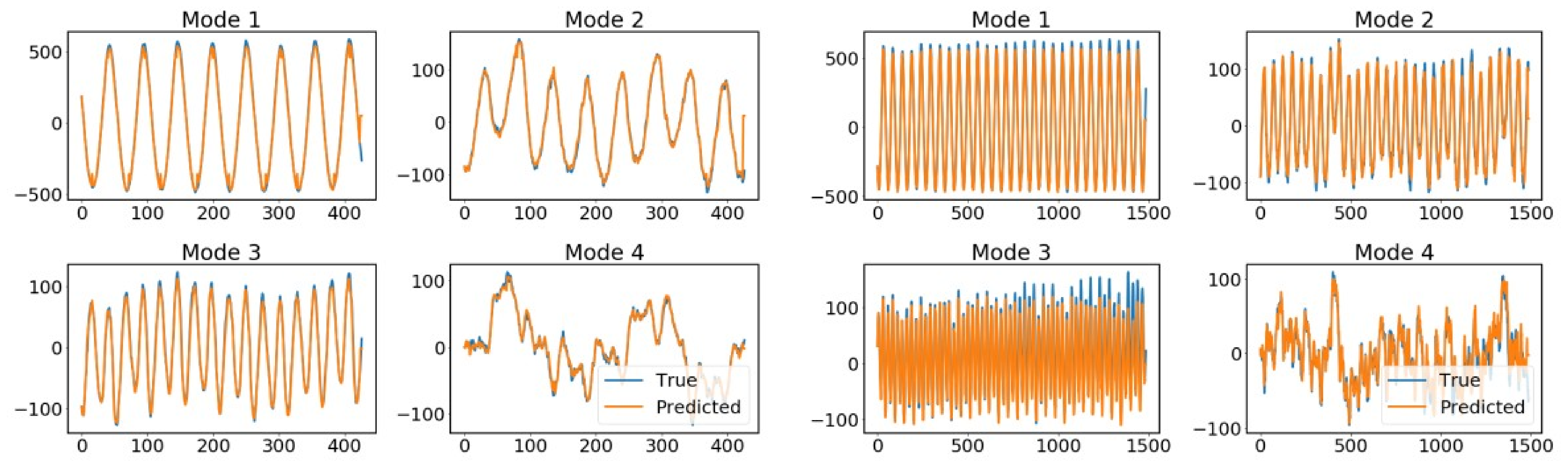
Linear/XGBoost/Random-forest methods are utilized within a general non-autoregressive time-series prediction framework without exogeneous inputs.

Xie, Jinyu, and Qian Wang. "Benchmark Machine Learning Approaches with Classical Time Series Approaches on the Blood Glucose Level Prediction Challenge." KHD@ IJCAI. 2018.

Science assessments

Window-in and window-out predictions (8 week windows). No feedback of outputs as inputs.

How well does the architecture accomplish our predictive task?



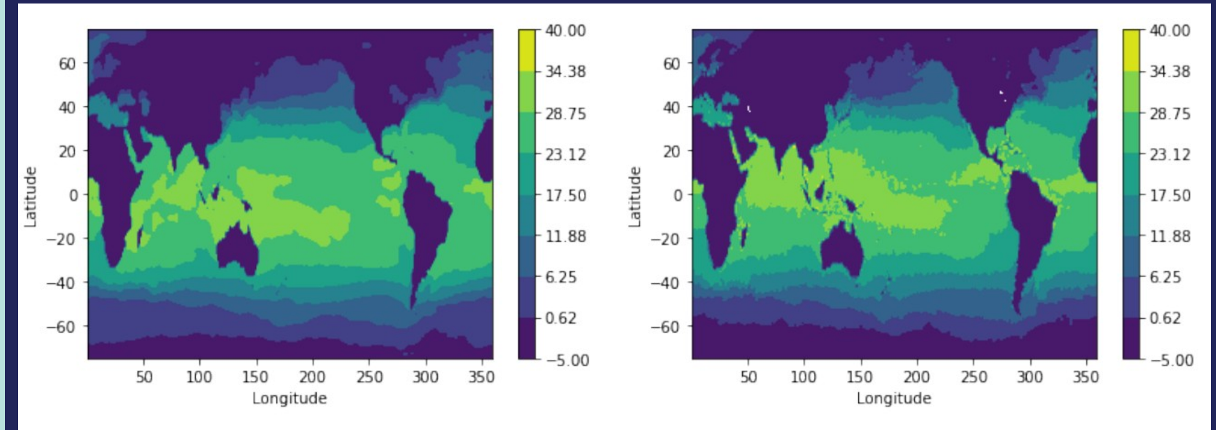
(a) NOAA-SST Training data forecast

(b) NOAA-SST Testing data forecast

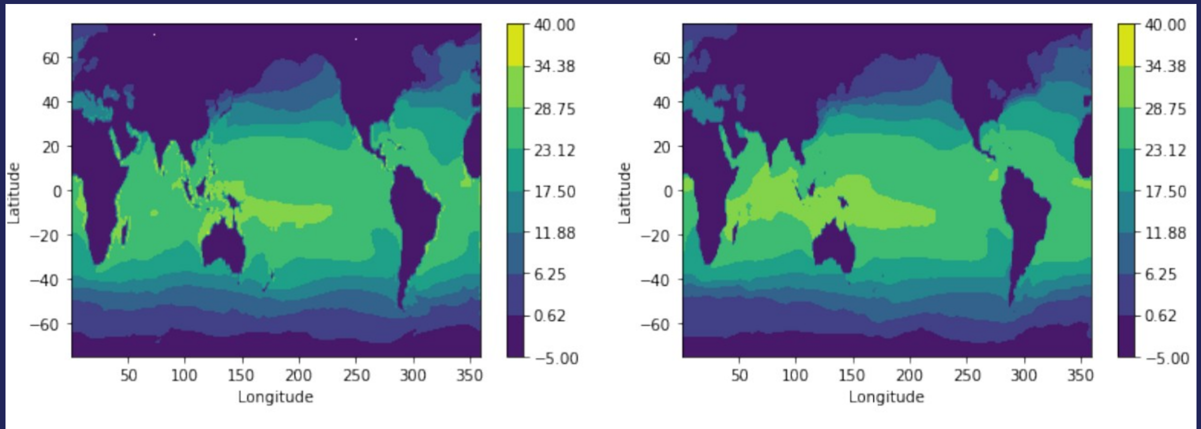
Forecasts can be seen to diverge as we get closer to 2018.

Science assessments

How well does the architecture accomplish our predictive task?



(a) NOAA (Truth)   (b) HYCOM



(c) CESM   (d) POD-LSTM

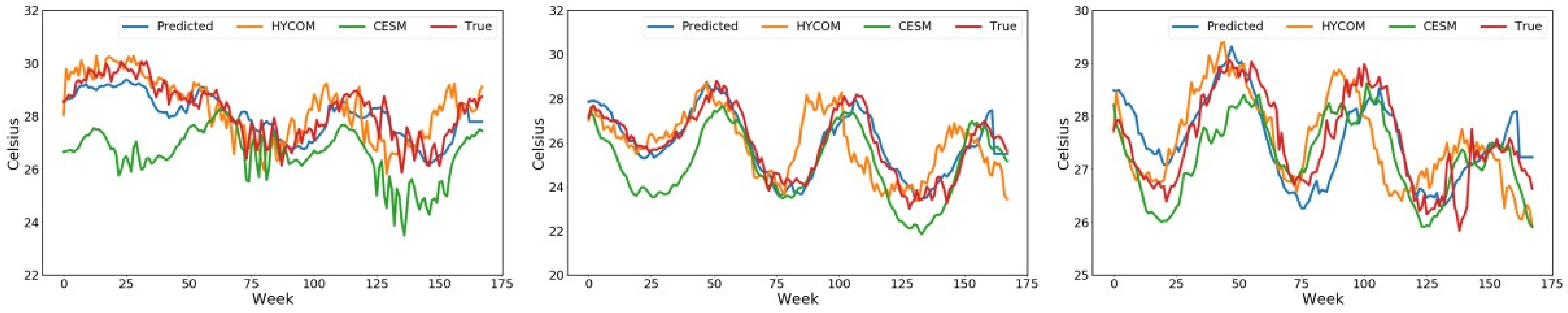
HYCOM run using US Navy DoD Supercomputing Resource Center (daily). 800 core-hours/day of forecast on a Cray XC40.

CESM (for a 1920-2100) forecast required 17 million core-hours on Yellowstone (NCAR HPC Resource) per member of ensemble (30 members)

Everything looks pretty *OK* in the eyeball norm.

Science assessments

How well does the architecture accomplish our predictive task?



(a) -5 ° latitude, 210 ° longitude (b) +5 ° latitude, 250 ° longitude (c) +10 ° latitude, 230 ° longitude

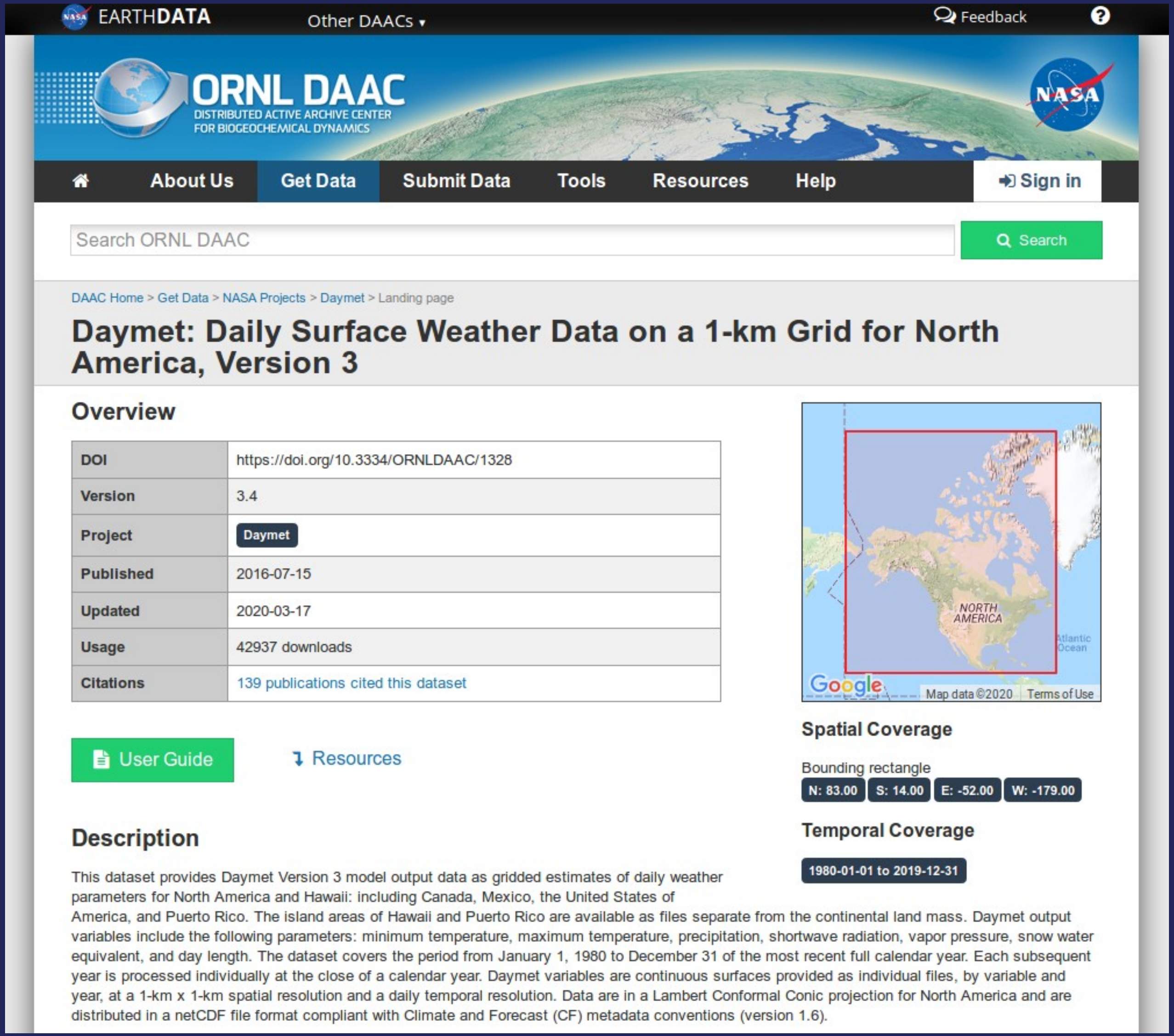
TABLE I

RMSE BREAKDOWN (IN CELSIUS) FOR DIFFERENT FORECAST TECHNIQUES COMPARED AGAINST THE NAS-POD-LSTM FORECASTS BETWEEN APRIL 5, 2015, AND JUNE 24, 2018, IN THE EASTERN PACIFIC REGION (BETWEEN -10 TO +10 DEGREES LATITUDE AND 200 TO 250 DEGREES LONGITUDE). THE PROPOSED EMULATOR MATCHES THE ACCURACY OF THE PROCESS-BASED MODELS FOR THIS PARTICULAR METRIC AND ASSESSMENT.

	RMSE (°Celsius)							
	Week 1	Week 2	Week 3	Week 4	Week 5	Week 6	Week 7	Week 8
Predicted	0.62	0.63	0.64	0.66	0.63	0.66	0.69	0.65
CESM	1.88	1.87	1.83	1.85	1.86	1.87	1.86	1.83
HYCOM	0.99	0.99	1.03	1.04	1.02	1.05	1.03	1.05

Recurrent neural architecture search for geophysical emulation, SC 2020.

NASA DayMet – Daily maximum temperature



Daytime maximum temperature. Originally available on 1 km^2 grid for North America.

Coarsened to $10km^2$ – To be used for testing architecture (not trained framework!)

Generated by a mix of remote sensing, experimental measurement and numerical simulations.

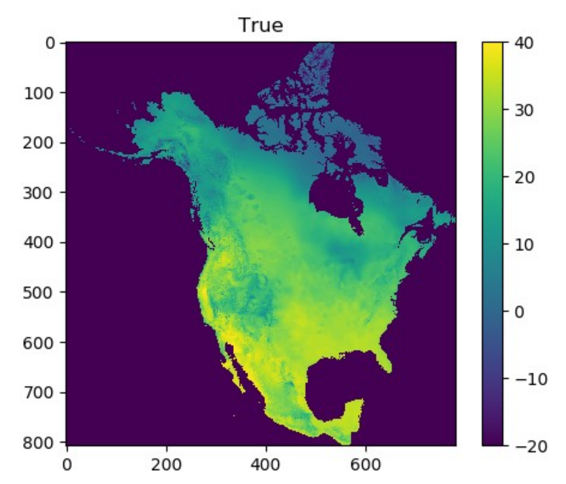
87 GB of data per year – 2015,2016,2017.

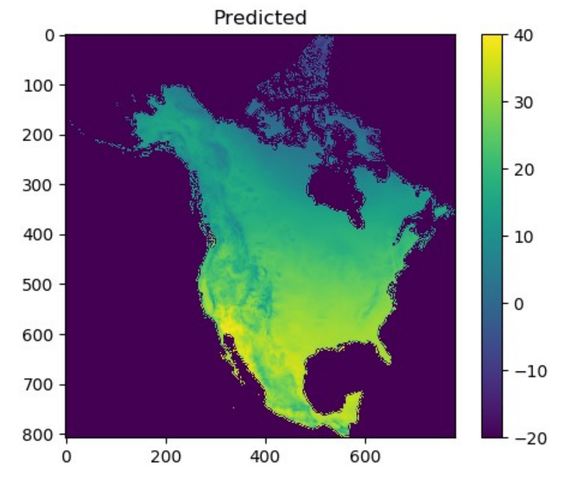
Also have precipitation/daylight (looking into that for future work)

https://daac.ornl.gov/cgi-bin/dsviewer.pl?ds_id=1328

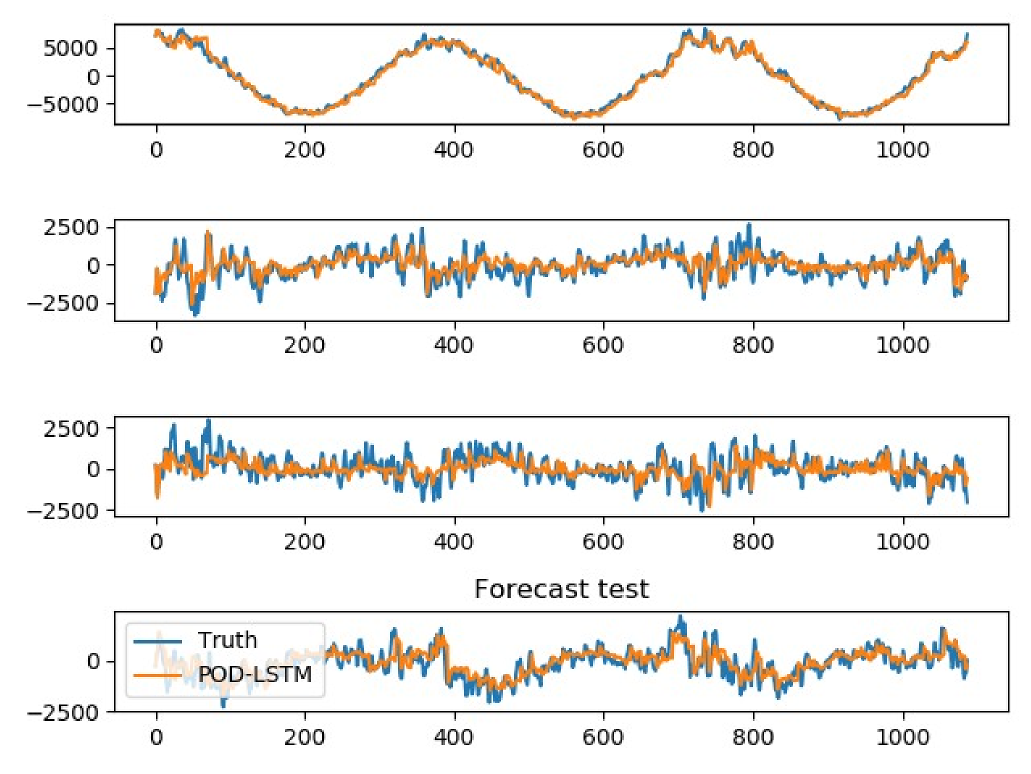
Science assessments

Using the same architecture on a different data set (with retraining)



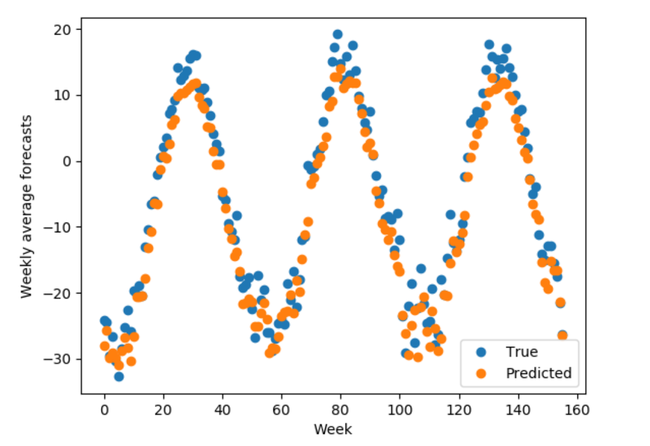


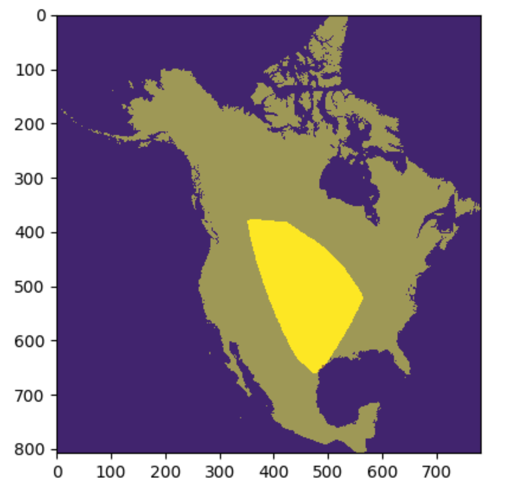
POD Coefficients



ORNL DayMET dataset (8000x8000) per day for 40 years (temperature, daylight, rainfall)

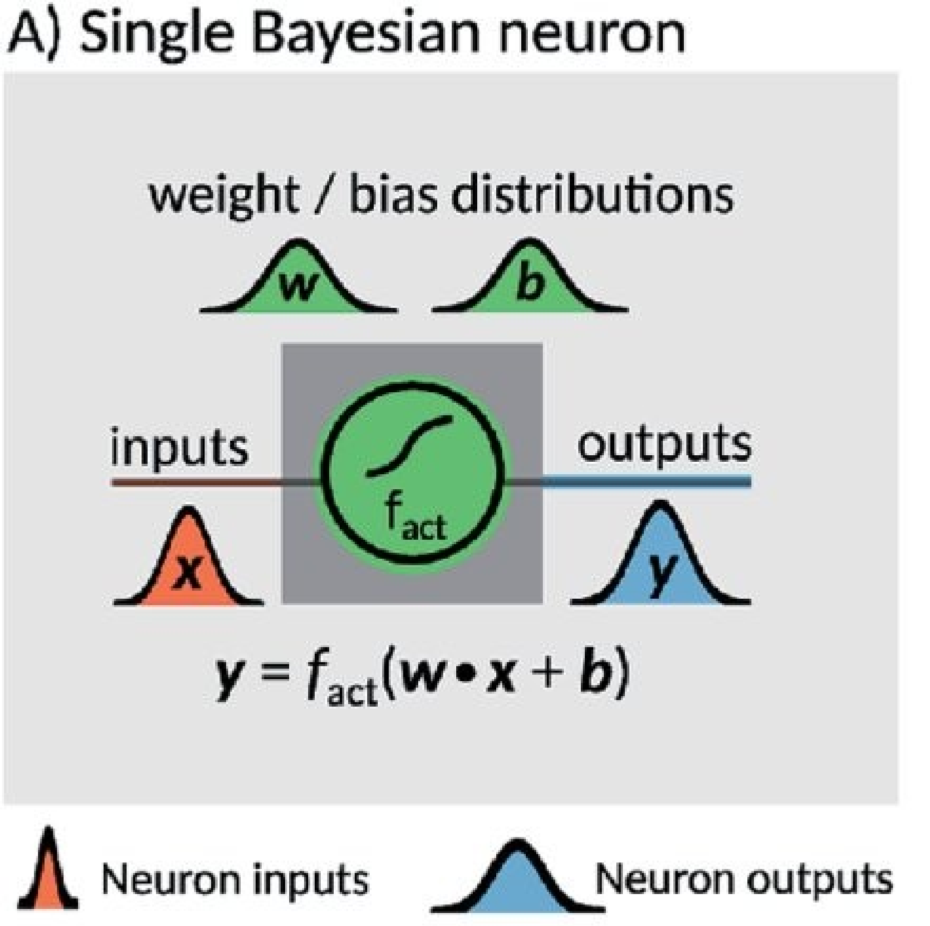
Weekly average predictions 2016-2018

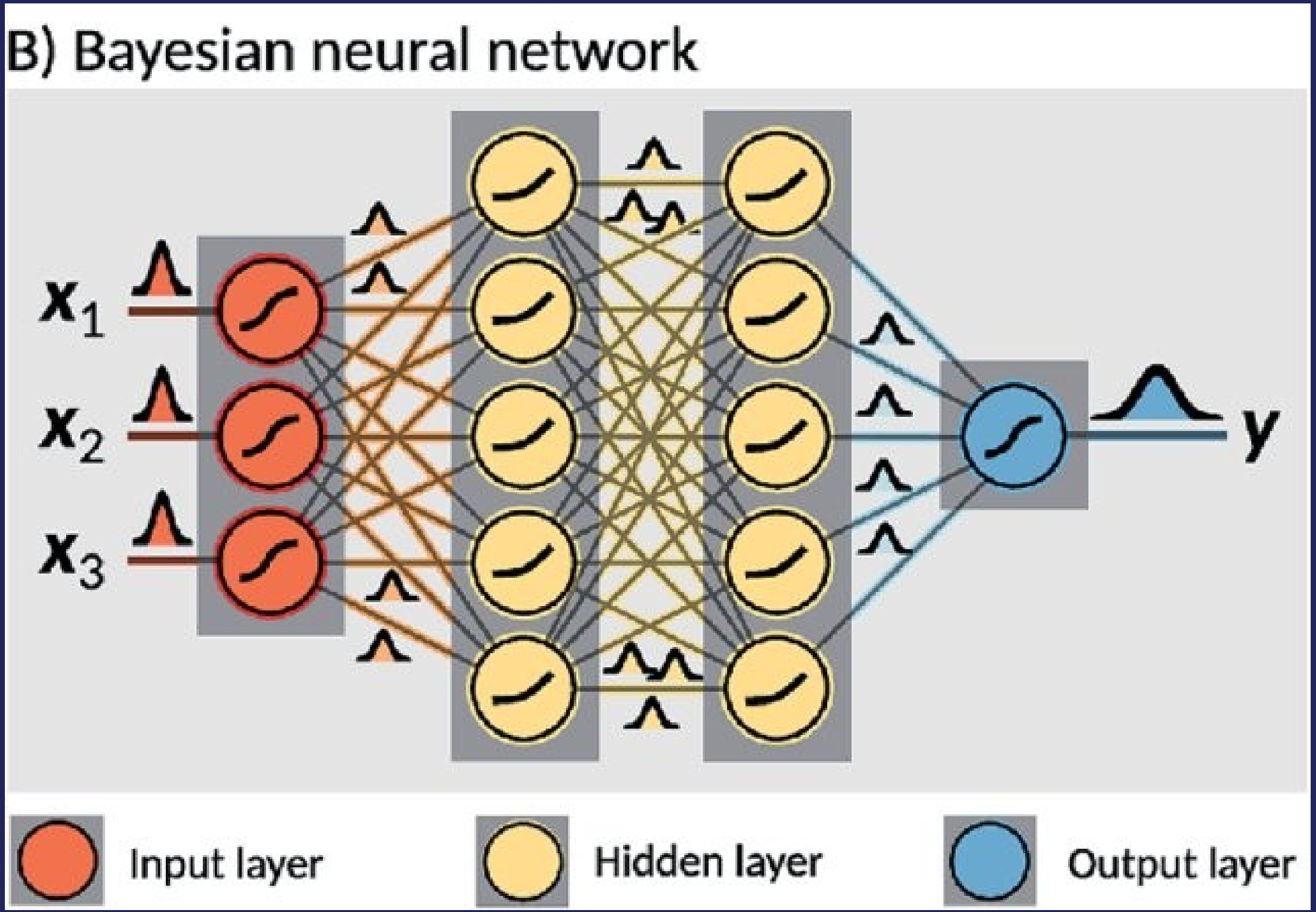




With D. Skinner, B. Lusch, R. Kotamarthi, V. Rao

NAS UQ – Primer: Bayesian neural networks





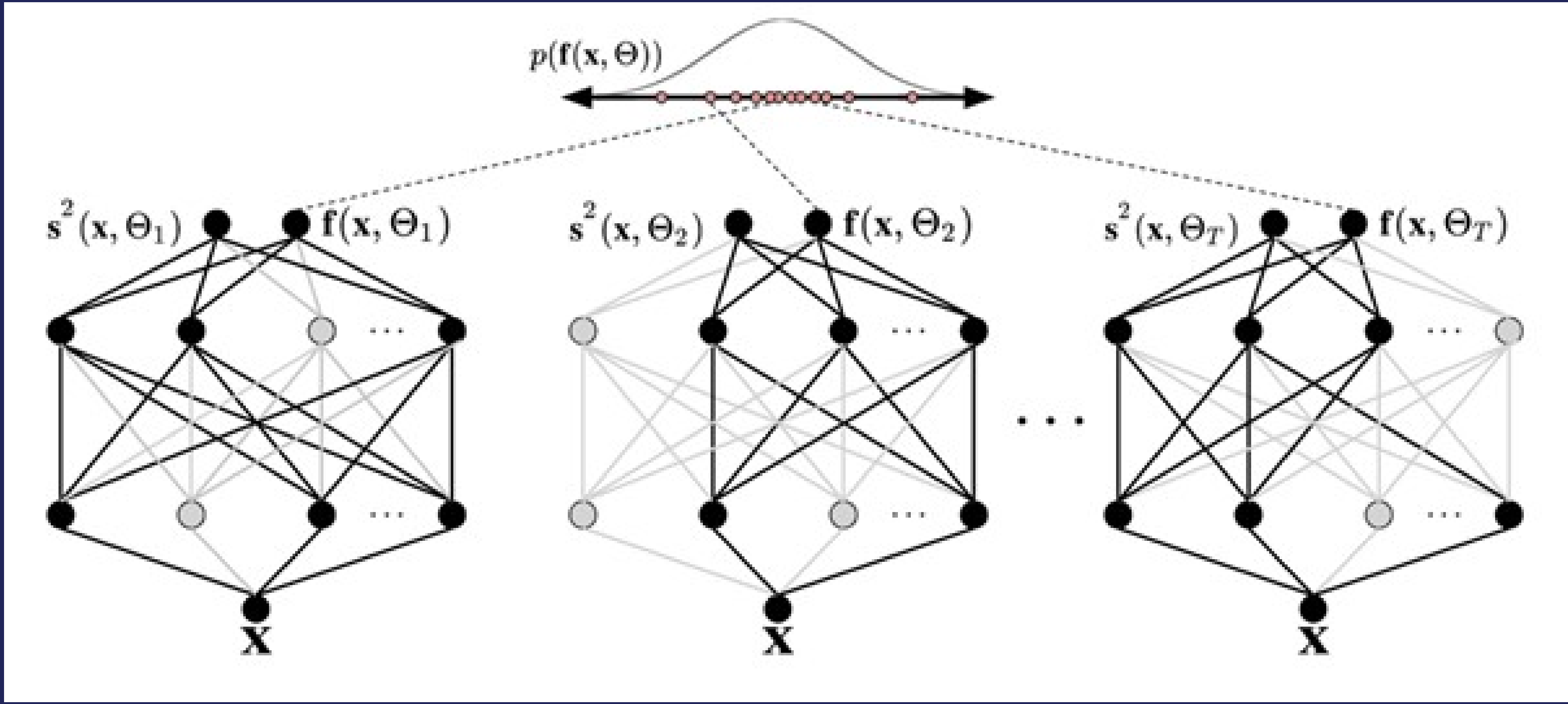
Exploration of posterior (for example with HMC) is infeasible so variational inference with the KL-divergence distance is used, assuming each weight is unimodal Gaussian (so mean and variance are parameters)

Hernández-Lobato, José Miguel, and Ryan Adams, ICML. PMLR, 2015.

Image credit: Hase et al., Chemical Science 10(8), 2019

Argonne NATIONAL LABORATORY

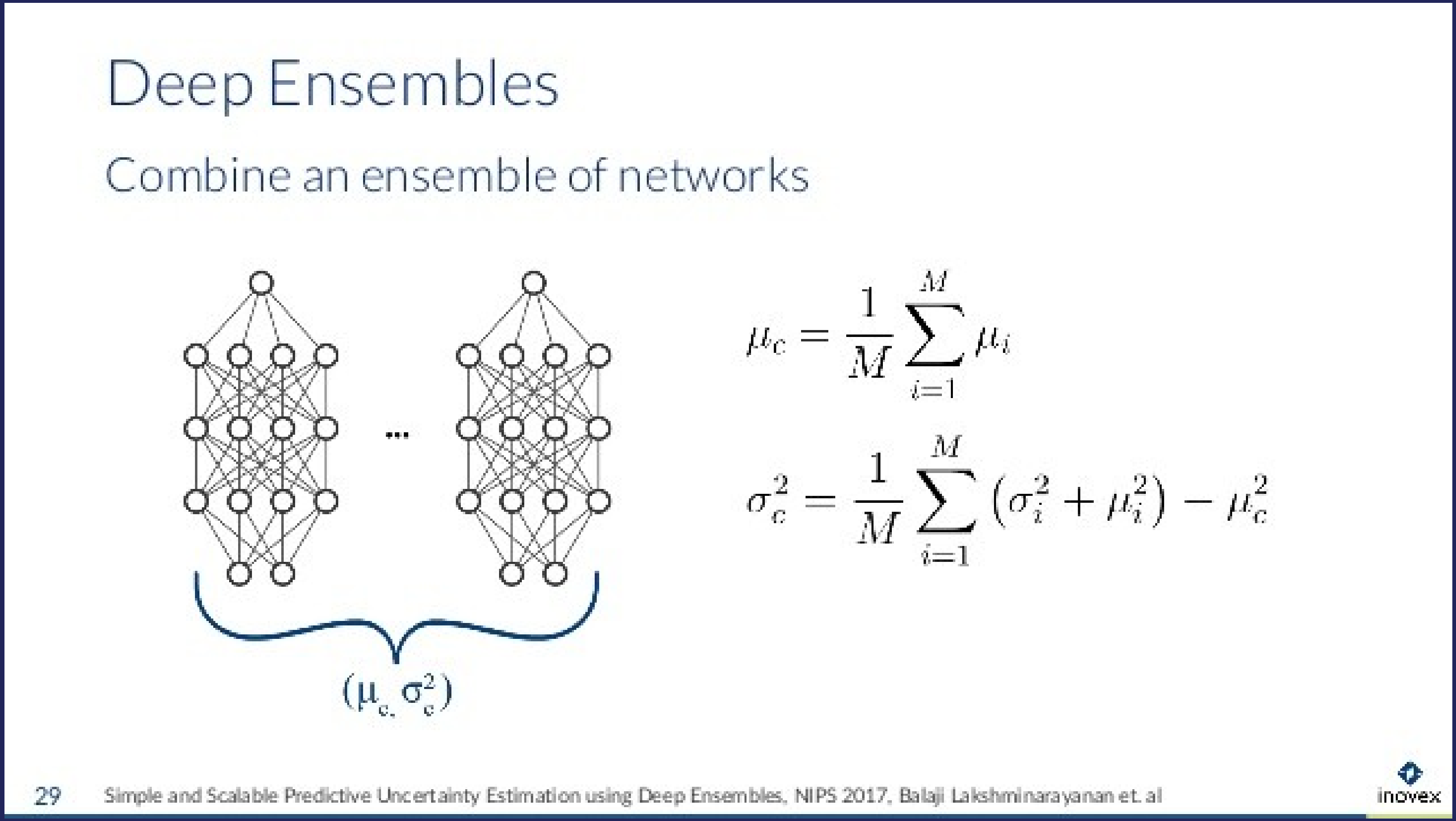
NAS UQ – Primer: Monte Carlo dropout



An approximate Monte-Carlo sampling of the posterior can be performed, easily, by randomly switching off neurons during multiple inferences.

Srivastava et al., JMLR, 15 (1), 1929-1958

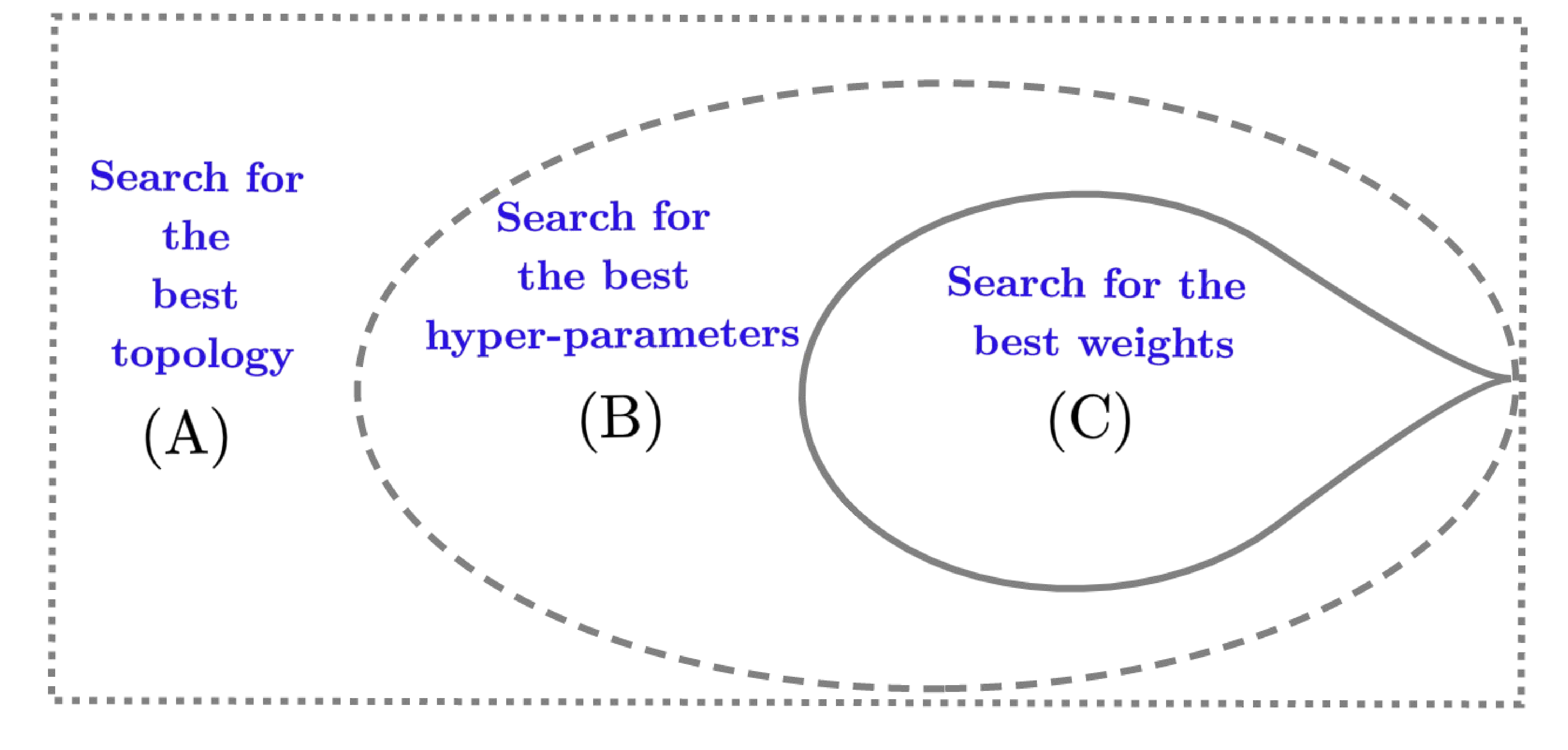
NAS UQ – Primer: Deep ensembles



Several models trained from different initializations and each model is a ‘sample’ in hypothesis space. **Apparently outperforms Monte-Carlo dropout and probabilistic backpropagation.**

Lakshminarayanan B, Pritzel A, Blundell C. NeurIPS 2017 Dec 4 (pp. 6405-6416).

Deep ensembles based UQ with DeepHyper (AutoDEUQ)



With Romain Egele, Krishnan Raghavan, Bethany Lusch, Prasanna Balaprakash

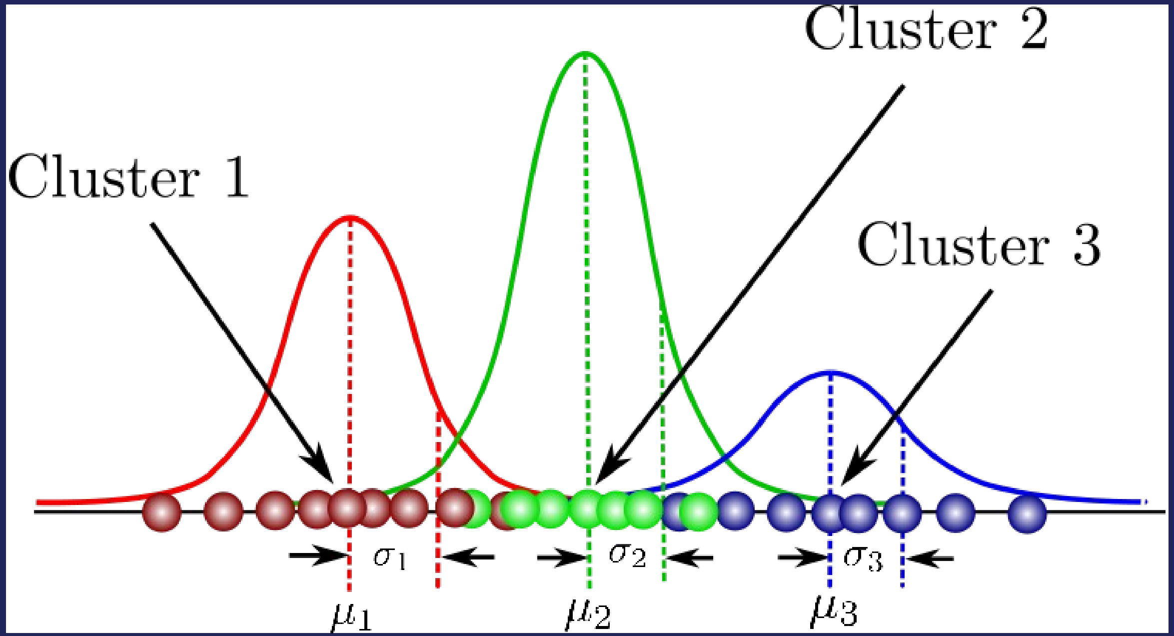
AutoDEUQ algorithm (joint HPS and NAS)

Algorithm 1: AgE

```
inputs : P: population size, S: sample size, W: workers
output: highest-accuracy model in history
/* Initialization
population ← create_queue(P) // Alloc empty Q of size P
for i ← 1 to W do
 | model.h_a ← random_point(H_a)
 | submit_evaluation(model) // Nonblocking
end
/* Main loop
while not done do
 | // Query results
 | results ← get_finished_evaluations ()
 | if |results| > 0 then
 |  | population.push(results) // Aging population
 |  | // Generate architecture configs
 |  | for i ← 1 to |results| do
 |  |  | if |population| = P then
 |  |  |  | sample ← random_sample(population,S)
 |  |  |  | parent ← select_parent(sample)
 |  |  |  | child.h_a ← mutate(parent.h_a)
 |  |  | else
 |  |  |  | child.h_a ← random_point(H_a)
 |  |  | end
 |  |  | submit_evaluation(child) // Nonblocking
 |  | end
 | end
end
```

Modify evolutionary search to also identify combinations of hyperparameters with architectures

Key idea: Ensembles of models to account for epistemic uncertainty and probabilistic output layer to handle aleatoric uncertainty



For handling complex likelihoods in regression – need to account for probabilistic layers in the output

ML Regression benchmarks

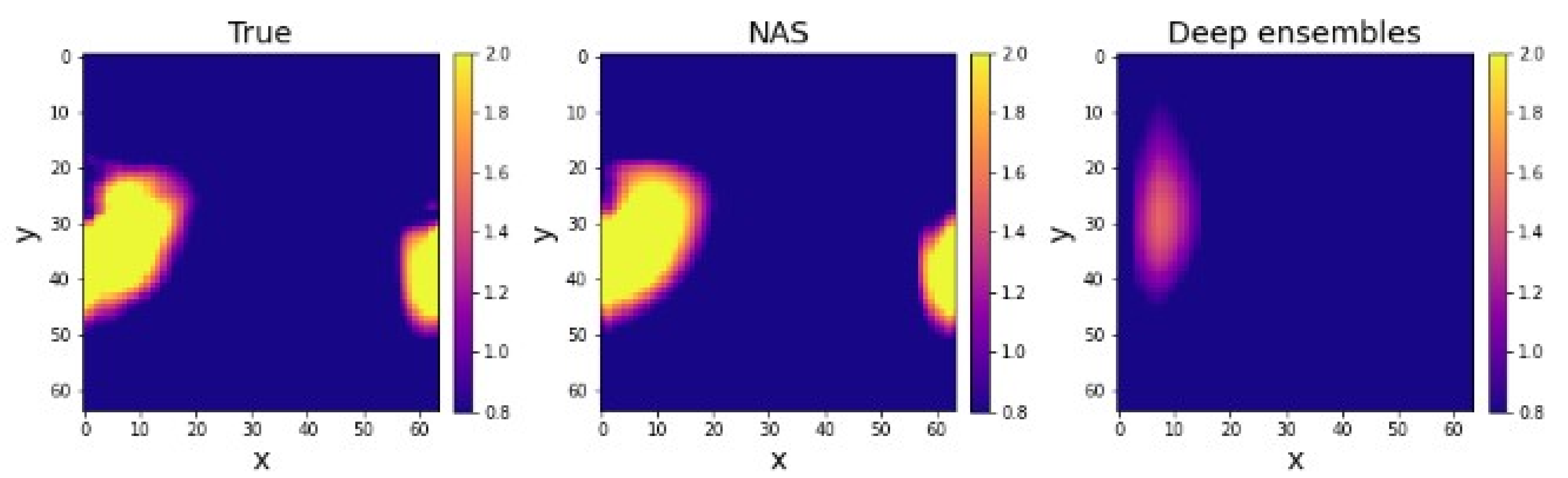
Dataset	NLL						RMSE					
	PBP	MC-Dropout	Deep Ensemble	Hyper Ensemble	AutoDEUQ Greedy	AutoDEUQ Top-K	PBP	MC-Dropout	Deep Ensemble	Hyper Ensemble	AutoDEUQ Greedy	AutoDEUQ Top-K
boston	3,01	2,97	3,28	**2,87**	3,03	2,93	2,57	2,46	2,41	**2,15**	2,93	2,41
concrete	5,67	5,23	6,03	4,7	4,33	**4,18**	3,16	3,04	3,06	4,09	3,02	**2,84**
energy	1,8	1,66	2,09	1,72	0,41	**0,4**	2,04	1,99	1,38	0,9	0,68	**0,62**
kin8nm	0,1	0,1	0,09	0,26	**0,06**	**0,06**	-0,9	-0,95	-1,2	6,89	-1,37	**-1,39**
navalpropulsion	0,01	0,01	**0**	0,01	**0**	**0**	-3,73	-3,8	-5,63	-3,03	**-8,23**	-8,12
powerplant	4,12	4,02	4,11	4,38	**3,42**	3,45	2,84	2,8	2,79	5,24	**2,63**	2,64
protein	4,73	4,36	4,71	5,09	**3,58**	3,61	2,97	2,89	2,83	21,12	**2,45**	2,48
wine	0,64	0,62	0,64	0,73	0,62	**0,61**	0,97	0,93	0,94	1,92	0,94	**0,91**
yacht	1,02	1,11	1,58	1,86	**0,68**	0,7	1,63	1,55	1,18	0,48	0,13	**0,12**
yearprediction	8,88	8,85	8,89	16,84	**7,9**	7,97	3,6	3,59	3,35	7,44	**3,22**	**3,22**

Table 1: Regression benchmark on 10 datasets. Scalar values indicate the mean score of a maximum of 10 repeated experiments.

Output likelihood

$$-\log p_\theta\left(y_n \mid \mathbf{x}_n\right) = \frac{\log \sigma_\theta^2(\mathbf{x})}{2} + \frac{\left(y - \mu_\theta(\mathbf{x})\right)^2}{2\sigma_\theta^2(\mathbf{x})} + \text{constant}\ ,$$

Autoencoder search (epistemic only)



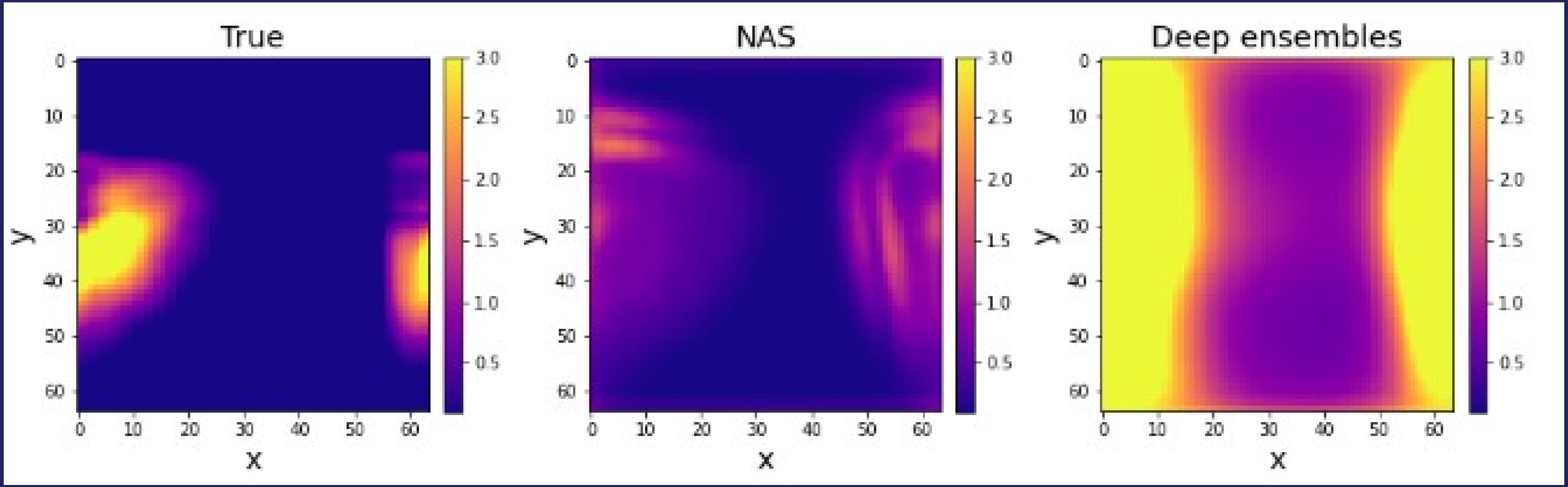
Reconstructions

The best 100 architectures from a set of 10 neural architecture searches (128 nodes each, 3 hours of walltime = 3840 node hours) may be used to perform ensemble UQ.

NAS based UQ superior for science data? Digging underway!

Method	Test MAE	Test MSE	Test NLL
Baseline	0.330	0.388	N/A
Deep ensembles	0.426	0.678	-0.242
Weight averaging	0.200	0.346	-1.576
Dropout	0.400	0.610	7.146
Deephyper	**0.111**	**0.072**	**-1.782**

Autoencoder search (epistemic only)



The best 100 architectures from a set of 10 neural architecture searches (128 nodes each, 3 hours of walltime = 3840 node hours) may be used to perform ensemble UQ.

NAS based UQ superior for science data? Digging underway!

Standard deviations

Method	Test MAE	Test MSE	Test NLL
Baseline	0.330	0.388	N/A
Deep ensembles	0.426	0.678	-0.242
Weight averaging	0.200	0.346	-1.576
Dropout	0.400	0.610	7.146
Deephyper	**0.111**	**0.072**	**-1.782**

Acknowledgements

Scalable Data-Efficient Learning for Scientific Domains
U.S. DOE 2018 Early Career Award
Funded by DOE-ASCR
(2018—Present)

Argonne Leadership Computing Facility
(2018—Present)

SLIK-D: Scalable Machine Learning Infrastructures for Knowledge Discovery
CELS LDRD Program (2016–2018)

ALCF – Margaret Butler Postdoctoral Fellowship

Thank you!