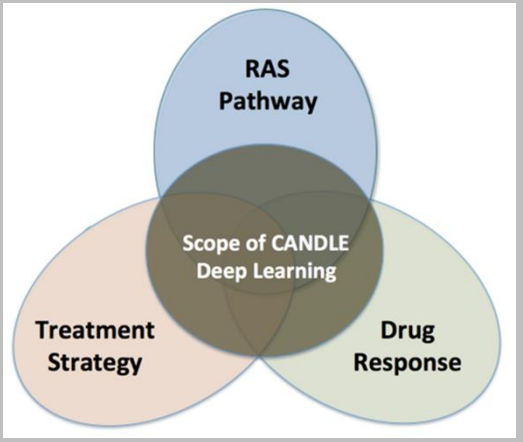
CANCER DEEP LEARNING ENVIRONMENT

Argonne NATIONAL LABORATORY

CANDLE TUTORIAL



JUSTIN M WOZNIAK
Computer Scientist
Data Science & Learning
Argonne National Laboratory

ARVIND RAMANATHAN,
HARRY YOO,
JAMAL MOHD-YUSOF

CANDLE Tutorial @ Exascale Computing Project Annual Meeting
January 14, 2019

CANDLE: BRIEFEST OVERVIEW

CANDLE: CANcer Deep Learning Environment

- Running (mini) cancer application cases from partner “Pilot” projects
  - Collaborate with National Cancer Institute, Frederick National Lab, etc.
- Focus on deep learning approach: TensorFlow, LBANN, et al.
- Focus on cases that require exascale: parallel training, ensembles
- Focus on scalable workflows: using the Swift/T workflow system
- Today’s tutorial will give you an overview of the concepts and tools

SCHEDULE

1. 9:00 a.m. **Justin Wozniak** Set-up and Orientation, Overview of day (30 min)
2. 9:30 a.m. **Arvind Ramanathan** Motivating examples (30 min)
3. 10:30 a.m. **Jamal Mohd-Yusof** CANDLE libs (30 min)
4. 11:00 a.m. **Harry Yoo** Migrating to CANDLE (45 min)
5. 11:45 a.m. **Lunch break** (1hr 45 min)
6. 1:30 p.m. **Justin Wozniak** Hyperparameter Optimization (60 min)
7. 2:30 p.m. **Harry Yoo** Model Ensembles for UQ: Overview and examples (60 min)
8. 3:30 p.m. **Justin Wozniak** Data Parallelism: Overview and examples (60 min)
9. 4:30 p.m. **Study hall**, viz, requirements, feedbacks (60 min)
10. 5:30 p.m. **Scheduled end**

HANDS-ON TUTORIALS

- May be found here:
  - `https://github.com/ECP-CANDLE`
  - Clone the `Tutorials` repo:
    `git@github.com:ECP-CANDLE/Tutorials.git`
  - `$ cd 2019/ECP`
  - See the top-level `README` to get started with the installation

OAK RIDGE National Laboratory

CANDLE Tutorial: CANDLE Overview + Motivating Examples

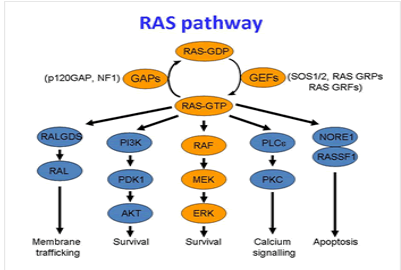
Arvind Ramanathan

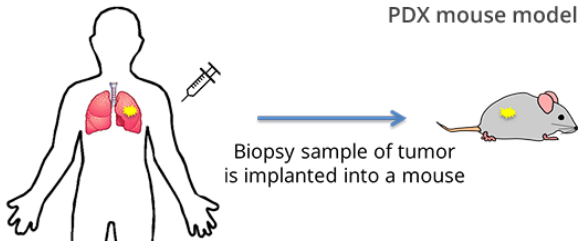
ORNL is managed by UT-Battelle, LLC for the US Department of Energy

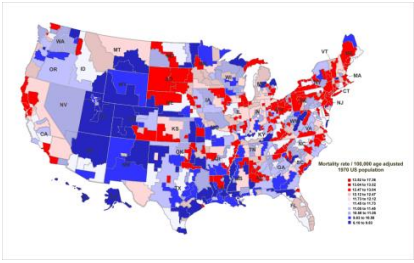
U.S. DEPARTMENT OF ENERGY

NCI-DOE Partnership: Extending the Frontiers of Precision Oncology (Three Pilots)

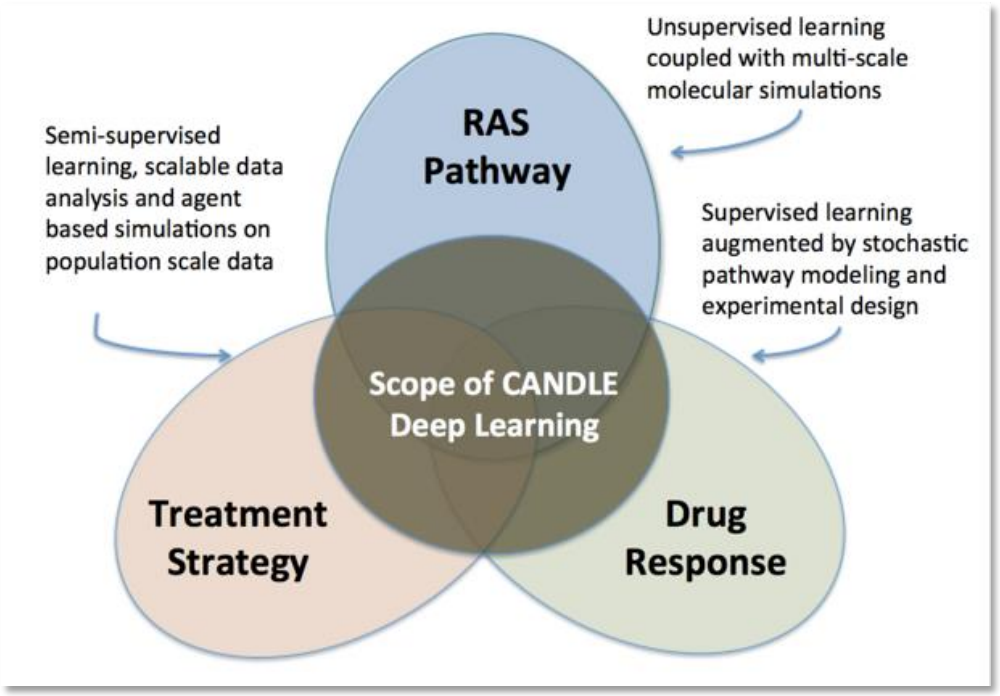
- Cancer Biology
  - **Molecular Scale Modeling of RAS Pathways**
  - Unsupervised Learning and Mechanistic models
  - Mechanism Understanding and Drug Targets
- Pre-clinical Models
  - **Cellular Scale PDX and Cell Lines**
  - ML, Experimental Design, Hybrid Models
  - Prediction of Drug Response
- Cancer Surveillance
  - **Population Scale Analysis**
  - Natural Language and Machine Learning
  - Agent Based Modeling of Cancer Patient Trajectories







ECP-CANDLE: CANcer Distributed Learning Environment



CANDLE Approach

Develop an exscale deep learning environment for cancer

Build on open source deep learning frameworks

Optimize for CORAL and Exascale platforms

Support all three pilot project needs for deep dearning

Collaborate with DOE computing centers, HPC vendors and ECP co-design and software technology projects

NVIDIA IBM CRAY intel AMD

ECP EXASCALE COMPUTING PROJECT

3

Learning Objectives / Outline

- Differences between traditional machine learning (ML) and deep learning (DL) techniques (15 minutes):
  - A practitioner's perspective on ML and DL/AI
- CANDLE Challenge problems and descriptions (25 minutes):
  - What does a CANDLE workflow entail?
  - Example problems and approach
  - Some initial code runs with real datasets
- Capabilities and Ongoing Research (10 minutes):
  - Large-scale data and model parallelism
  - Distributed Hyperparameter optimization

OAK RIDGE National Laboratory

Traditional ML vs. DL

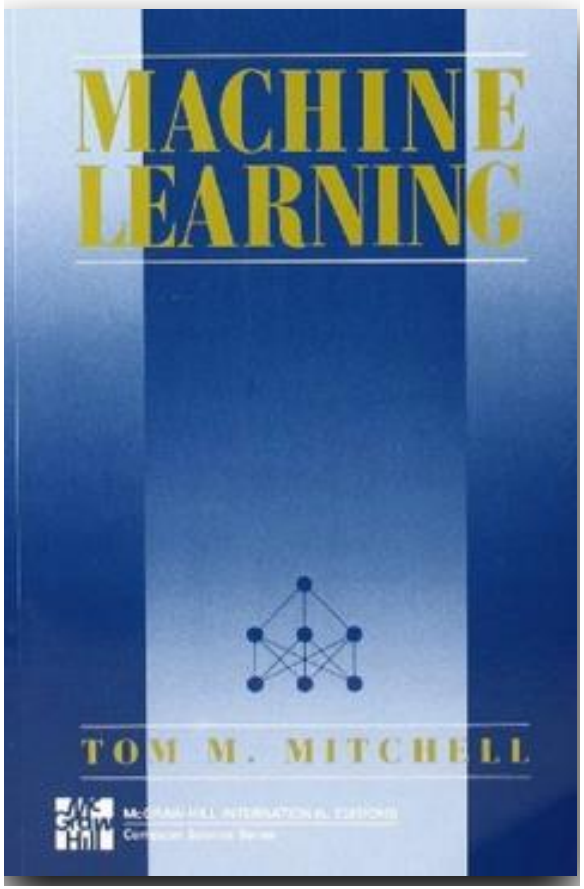
- A practitioner's perspective on the differences between ML and DL
- Where does the computational complexity of DL arise from?

Traditional machine learning paradigm

- Gather as much labelled data as possible
- Throw some at it …
- Pick the best functioning algorithm
- Spend hours hand encoding features:
  - run feature selection/ model selection/ dimensionality reduction/ other techniques
- Repeat…

But, what is machine learning?

- According to Tom Mitchell (CMU) "ML deals with construction and study of systems that learn from data"
- A computer program is said to **learn from experience** (E) with respect to some class of tasks (T) and performance at tasks in T, as measured by P, improves with experience E

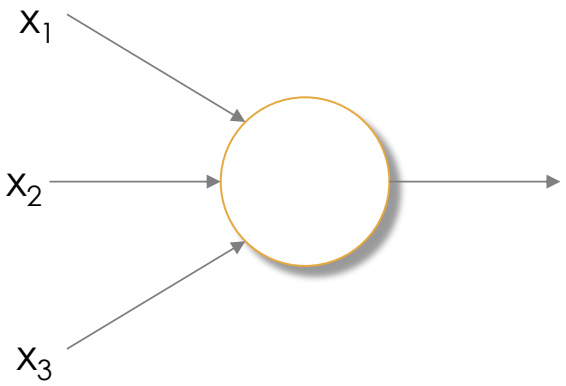


What is machine learning?

- Traditional paradigm:

- Machine learning:

A simple perceptron



$$\text{output} = \begin{cases} 0, \text{if} \sum_j w_j x_j \leq \text{threshold} \\ 1, \text{if} \sum_j w_j x_j > \text{threshold} \end{cases}$$

Can a simple perceptron learn?

- Example: XOR
  - Exclusive OR
  - $[x_1, x_2]^T$
  - y = 1 only when one of x_1 or x_2 is 1
- $X = \{[0,0]^T, [0,1]^T, [1,0]^T, [1,1]^T\}$
- $Y = \{0, 1, 1, 0\}$

X1	X2	Y
0	0	0
0	1	1
1	0	1
1	1	0

Traditional machine learning

- Can
  clas
- Para
- Let's
  linea

$(\vec{x};\vec{\theta}))^2$

model

$+ b$

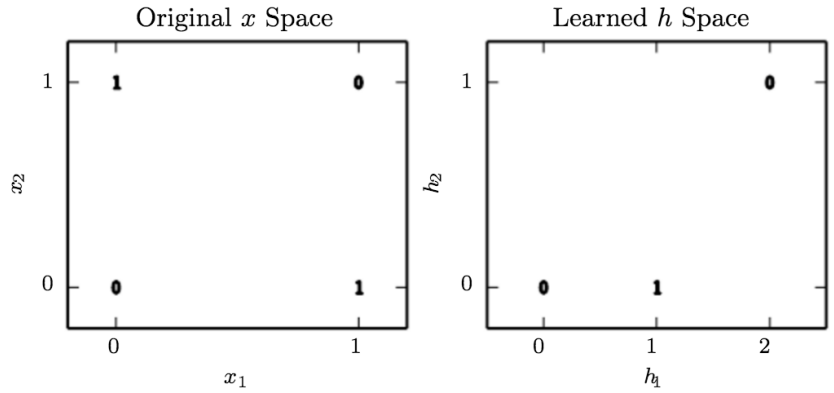


Figure 6.1: Solving the XOR problem by learning a representation. The bold numbers printed on the plot indicate the value that the learned function must output at each point. *(Left)* A linear model applied directly to the original input cannot implement the XOR function. When $x_1 = 0$, the model's output must increase as x_2 increases. When $x_1 = 1$, the model's output must decrease as x_2 increases. A linear model must apply a fixed coefficient w_2 to x_2. The linear model therefore cannot use the value of x_1 to change the coefficient on x_2 and cannot solve this problem. *(Right)* In the transformed space represented by the features extracted by a neural network, a linear model can now solve the problem. In our example solution, the two points that must have output 1 have been collapsed into a single point in feature space. In other words, the nonlinear features have mapped both $\boldsymbol{x} = [1, 0]^\top$ and $\boldsymbol{x} = [0, 1]^\top$ to a single point in feature space, $\boldsymbol{h} = [1, 0]^\top$. The linear model can now describe the function as increasing in h_1 and decreasing in h_2. In this example, the motivation for learning the feature space is only to make the model capacity greater so that it can fit the training set. In more realistic applications, learned representations can also help the model to generalize.

Linear features may not be best for capturing relationships in datasets

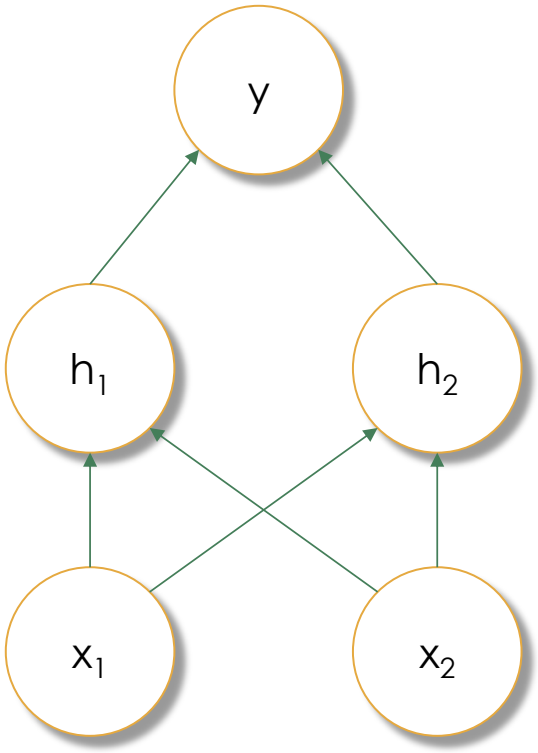
- We output ½ everywhere!
  - Linear features are not good for XOR
- We need other ways to solve this:
  - Two applications of a function in a chain …

$$\vec{h} = f^{(1)}(\vec{x}; \vec{W}, b)$$

$$y = f^{(2)}(\vec{h}; \vec{w}, b)$$

- What does h do?

How does this perceptron look like?



- Linear models will still be problematic
- Neural networks use an affine transformation controlled by learned parameters followed by a non-linear function called activation function

$$\vec{h} = g(\vec{W}^T \vec{x} + \vec{c})$$

- g is referred to as a rectified linear unit (ReLU) defined by

$$g(z) = \max\{0, z\}$$

Learning "XOR" with neural nets

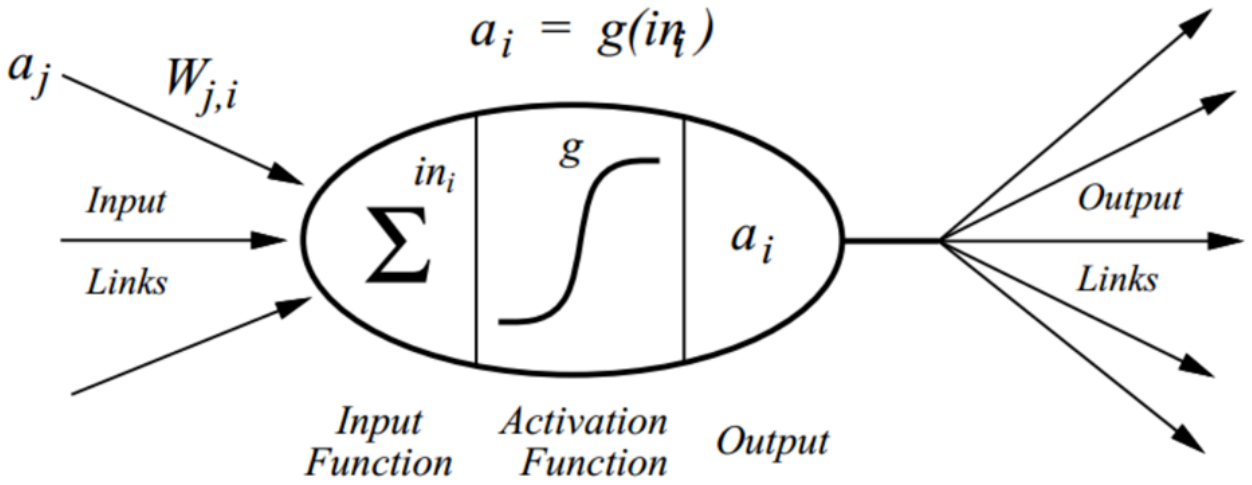
$$f(\vec{x}; \vec{W}, \vec{c}, \vec{w}, b) = \vec{w}^T \max\{0, \vec{W}^T \vec{x} + \vec{c}\} + b$$

$$W = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} \quad c = \begin{bmatrix} 0 \\ -1 \end{bmatrix} \quad w = \begin{bmatrix} 1 \\ -2 \end{bmatrix}$$

$$x = \begin{bmatrix} 0 & 0 \\ 0 & 1 \\ 1 & 0 \\ 1 & 1 \end{bmatrix} \qquad xW = \begin{bmatrix} 0 & 0 \\ 1 & 1 \\ 1 & 1 \\ 2 & 2 \end{bmatrix} \qquad xW + c = \begin{bmatrix} 0 & -1 \\ 1 & 0 \\ 1 & 0 \\ 2 & 1 \end{bmatrix}$$

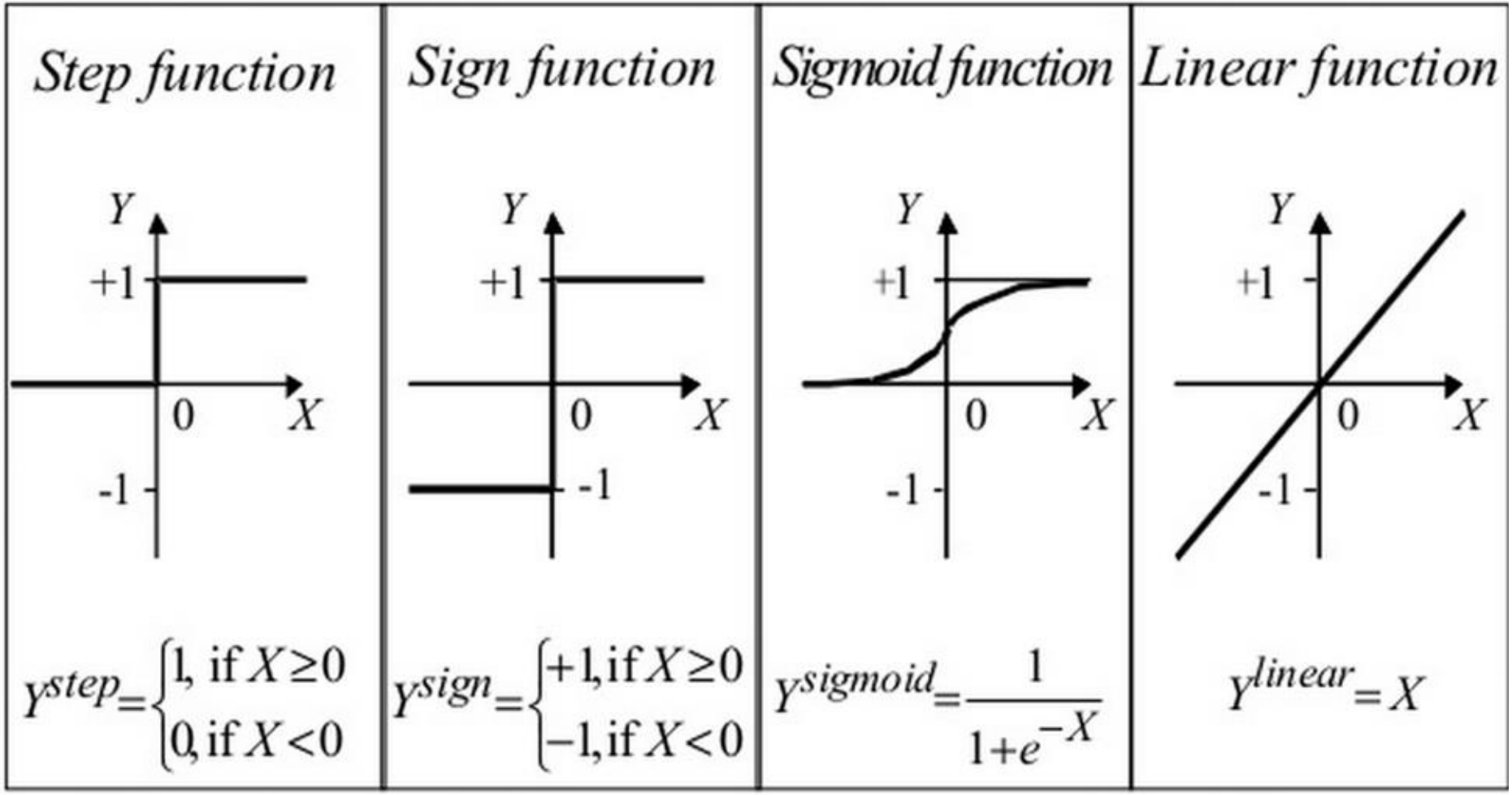
$$\vec{w}^T \max\{0, \vec{W}^T \vec{x} + \vec{c}\} = \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 1 & 0 \\ 2 & 1 \end{bmatrix}$$

Neural nets: all we need to know

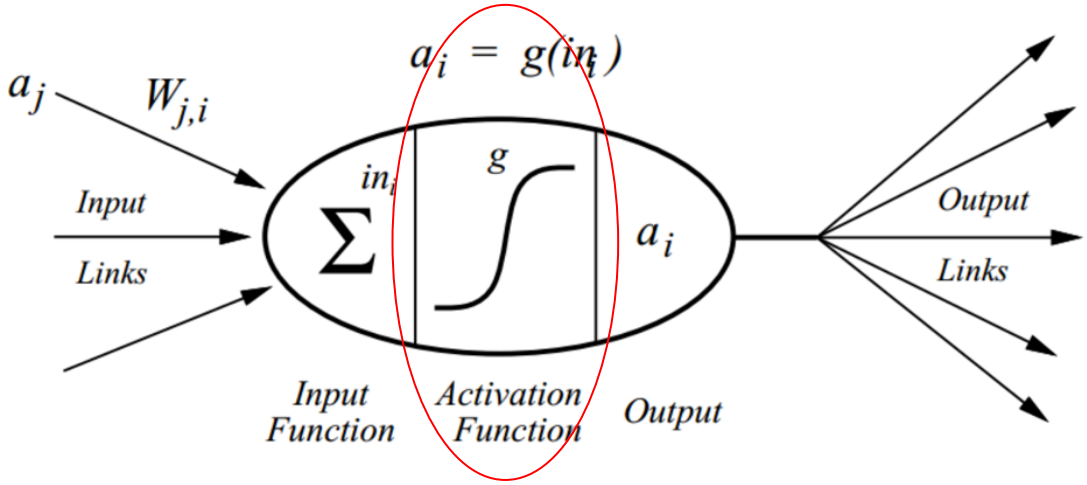


- Typically modeled using a "neuron like model"
  - Activation functions can be many types, but only one can be applied to the neuron at hand!

Types of activation functions…

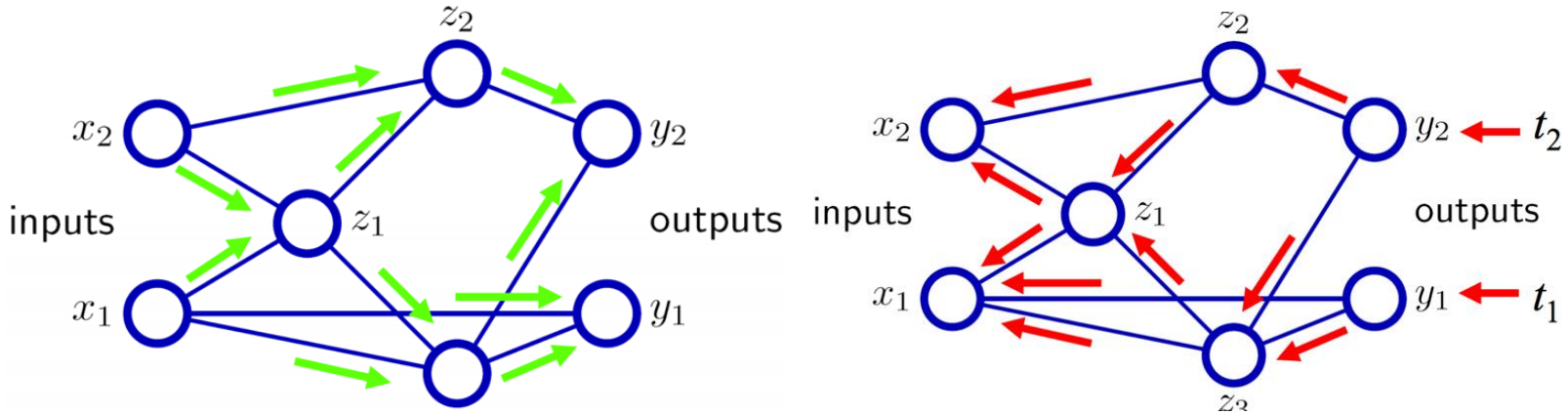


How to train a neural network?



- Activation function is non-linear!!!
- Gradient descent can be used to train neural network

Forwardprop and Backprop!



- Compute gradient of example-wise loss wrt parameters

$$y = f(W); W = g(x); \frac{\partial y}{\partial x} = \frac{\partial y}{\partial W}\frac{\partial W}{\partial x}$$

- Computing loss function is O(n) computation

ForwardProp: compute output based on the neural network

Algorithm 6.1 A procedure that performs the computations mapping n_i inputs $u^{(1)}$ to $u^{(n_i)}$ to an output $u^{(n)}$. This defines a computational graph where each node computes numerical value $u^{(i)}$ by applying a function $f^{(i)}$ to the set of arguments $\mathbb{A}^{(i)}$ that comprises the values of previous nodes $u^{(j)}$, $j < i$, with $j \in Pa(u^{(i)})$. The input to the computational graph is the vector $\boldsymbol{x}$, and is set into the first n_i nodes $u^{(1)}$ to $u^{(n_i)}$. The output of the computational graph is read off the last (output) node $u^{(n)}$.

for $i = 1, \ldots, n_i$ **do**
  $u^{(i)} \leftarrow x_i$
end for
for $i = n_i + 1, \ldots, n$ **do**
  $\mathbb{A}^{(i)} \leftarrow \{u^{(j)} \mid j \in Pa(u^{(i)})\}$
  $u^{(i)} \leftarrow f^{(i)}(\mathbb{A}^{(i)})$
end for
return $u^{(n)}$

Backprop Algorithm: propagate errors wrt model and data

Algorithm 6.2 Simplified version of the back-propagation algorithm for computing the derivatives of $u^{(n)}$ with respect to the variables in the graph. This example is intended to further understanding by showing a simplified case where all variables are scalars, and we wish to compute the derivatives with respect to $u^{(1)}, \ldots, u^{(n_i)}$. This simplified version computes the derivatives of all nodes in the graph. The computational cost of this algorithm is proportional to the number of edges in the graph, assuming that the partial derivative associated with each edge requires a constant time. This is of the same order as the number of computations for the forward propagation. Each $\frac{\partial u^{(i)}}{\partial u^{(j)}}$ is a function of the parents $u^{(j)}$ of $u^{(i)}$, thus linking the nodes of the forward graph to those added for the back-propagation graph.

Run forward propagation (Algorithm 6.1 for this example) to obtain the activations of the network

Initialize `grad_table`, a data structure that will store the derivatives that have been computed. The entry `grad_table`$[u^{(i)}]$ will store the computed value of $\frac{\partial u^{(n)}}{\partial u^{(i)}}$.

`grad_table`$[\partial u^{(n)}] \leftarrow 1$

for $j = n - 1$ down to 1 **do**

  The next line computes $\frac{\partial u^{(n)}}{\partial u^{(j)}} = \sum_{i:j \in Pa(u^{(i)})} \frac{\partial u^{(n)}}{\partial u^{(i)}} \frac{\partial u^{(i)}}{\partial u^{(j)}}$ using stored values:

  `grad_table`$[u^{(j)}] \leftarrow \sum_{i:j \in Pa(u^{(i)})}$ `grad_table`$[u^{(i)}] \frac{\partial u^{(i)}}{\partial u^{(j)}}$

end for

return $\{$`grad_table`$[u^{(i)}] \mid i = 1, \ldots, n_i\}$

Backprop can be generalized…

- Backprop can be generalized beyond the scalar case

$$\vec{x} \in R^m; \vec{y} \in R^n$$

- then, computing the error is nothing but:

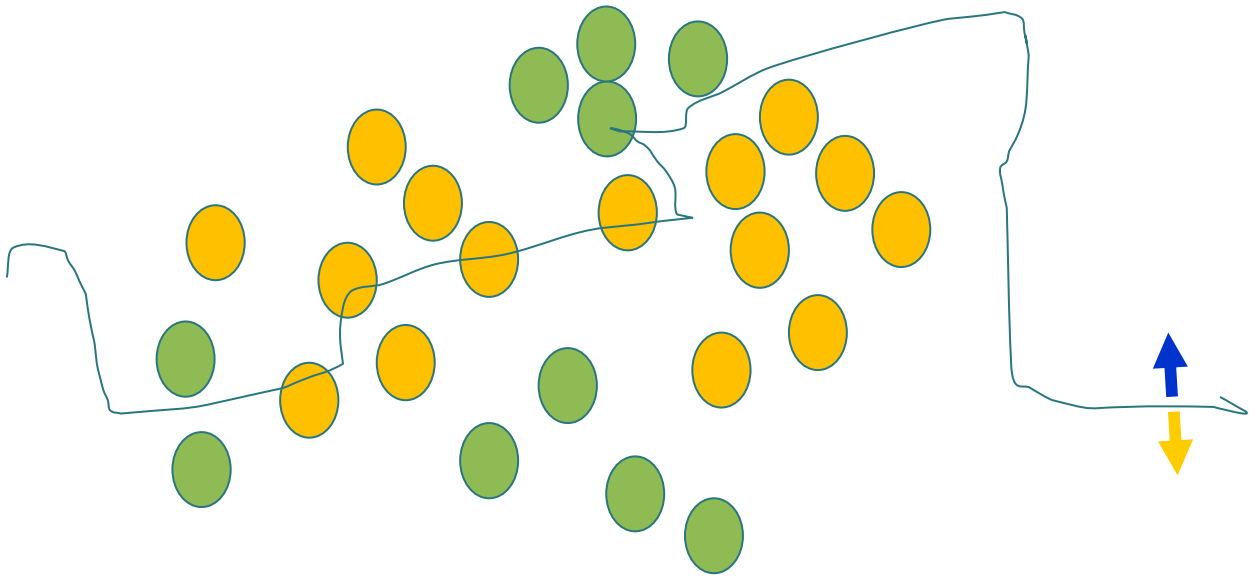
$$\vec{y} = g(\vec{x}); z = f(\vec{y})$$

$$\frac{\partial z}{\partial x_i} = \sum_j \frac{\partial z}{\partial y_j} \frac{\partial y_j}{\partial x_i}$$

$$\nabla_{\vec{x}} z = (\frac{\partial \vec{y}}{\partial \vec{x}})^T \nabla_{\vec{y}} z$$

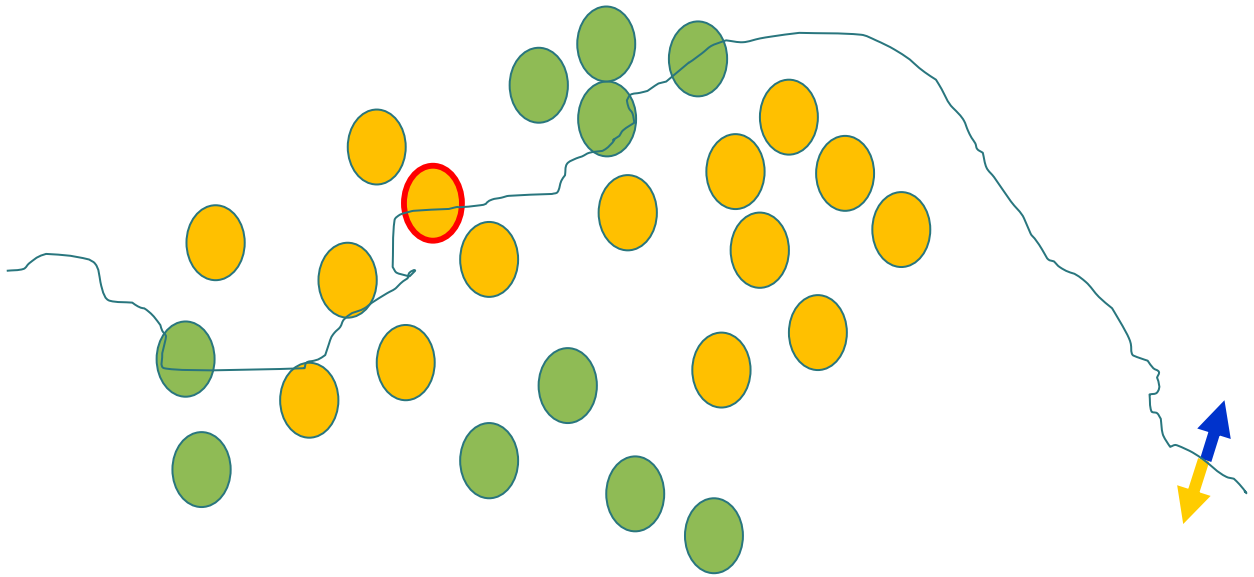
The decision boundary perspective…

Initial random weights



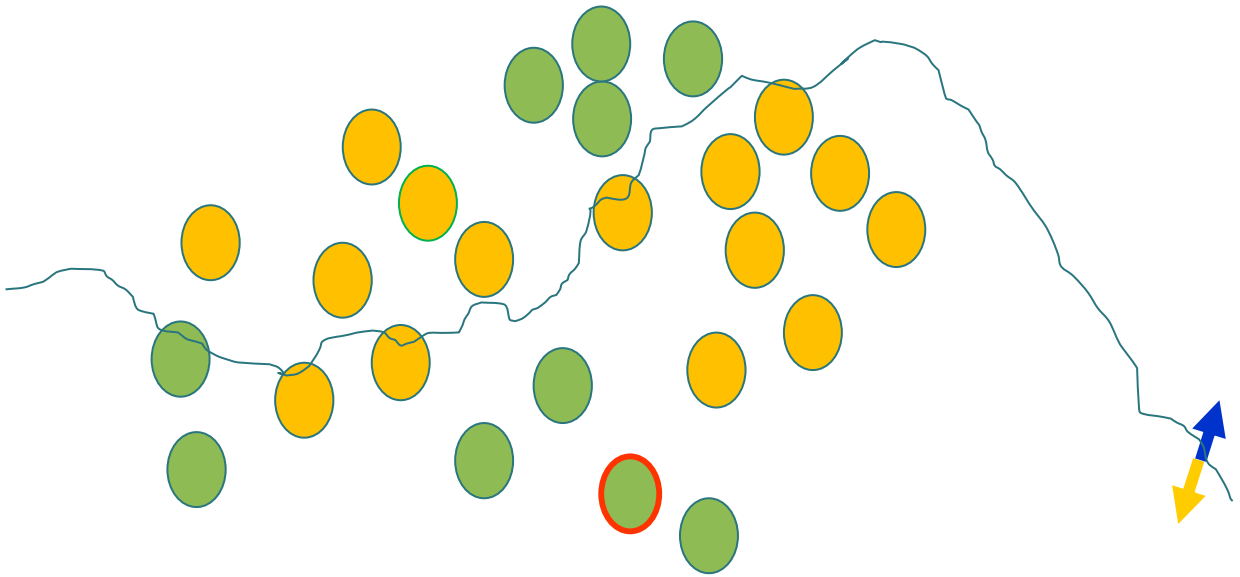
The decision boundary perspective…

Present a training instance / adjust the weights



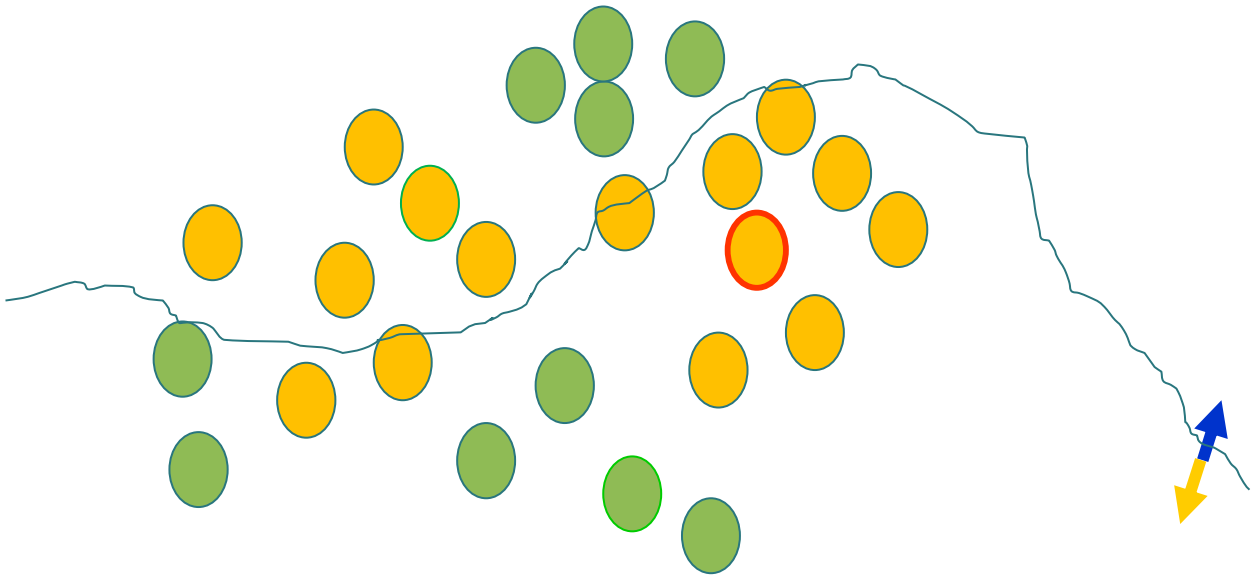
The decision boundary perspective…

Present a training instance / adjust the weights



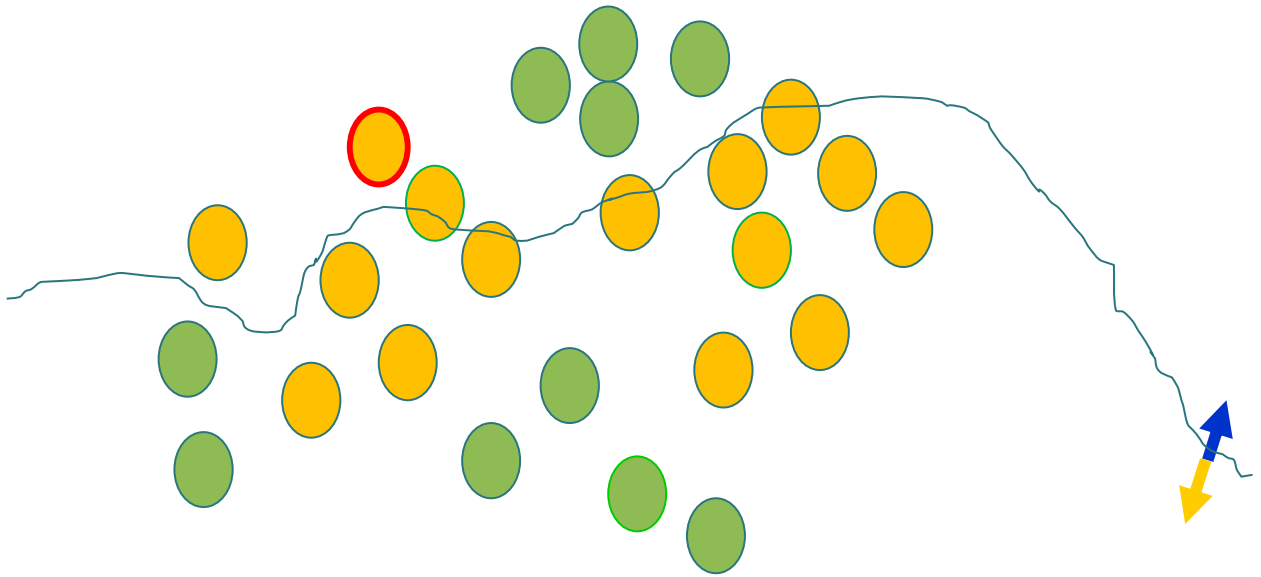
The decision boundary perspective…

Present a training instance / adjust the weights



The decision boundary perspective…

Present a training instance / adjust the weights



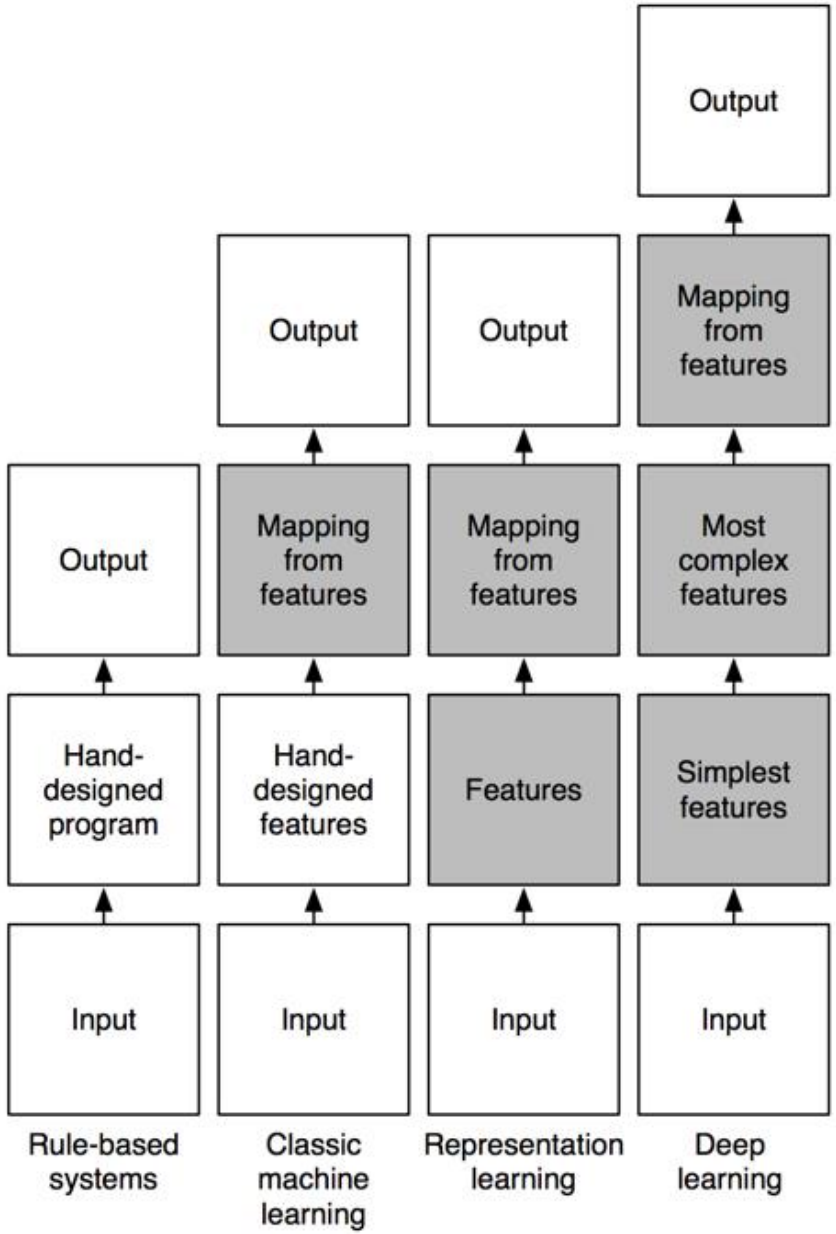
The decision boundary perspective…

Eventually ….

Deep Learning

- Hierarchical, Generalizable:
  - and can be unsupervised
- Efficient, Sharable, Distributed

- Downsides:
  - Blackbox
  - Training time?
  - How much data?



OAK RIDGE National Laboratory

CANDLE Workflows and Challenge Problems

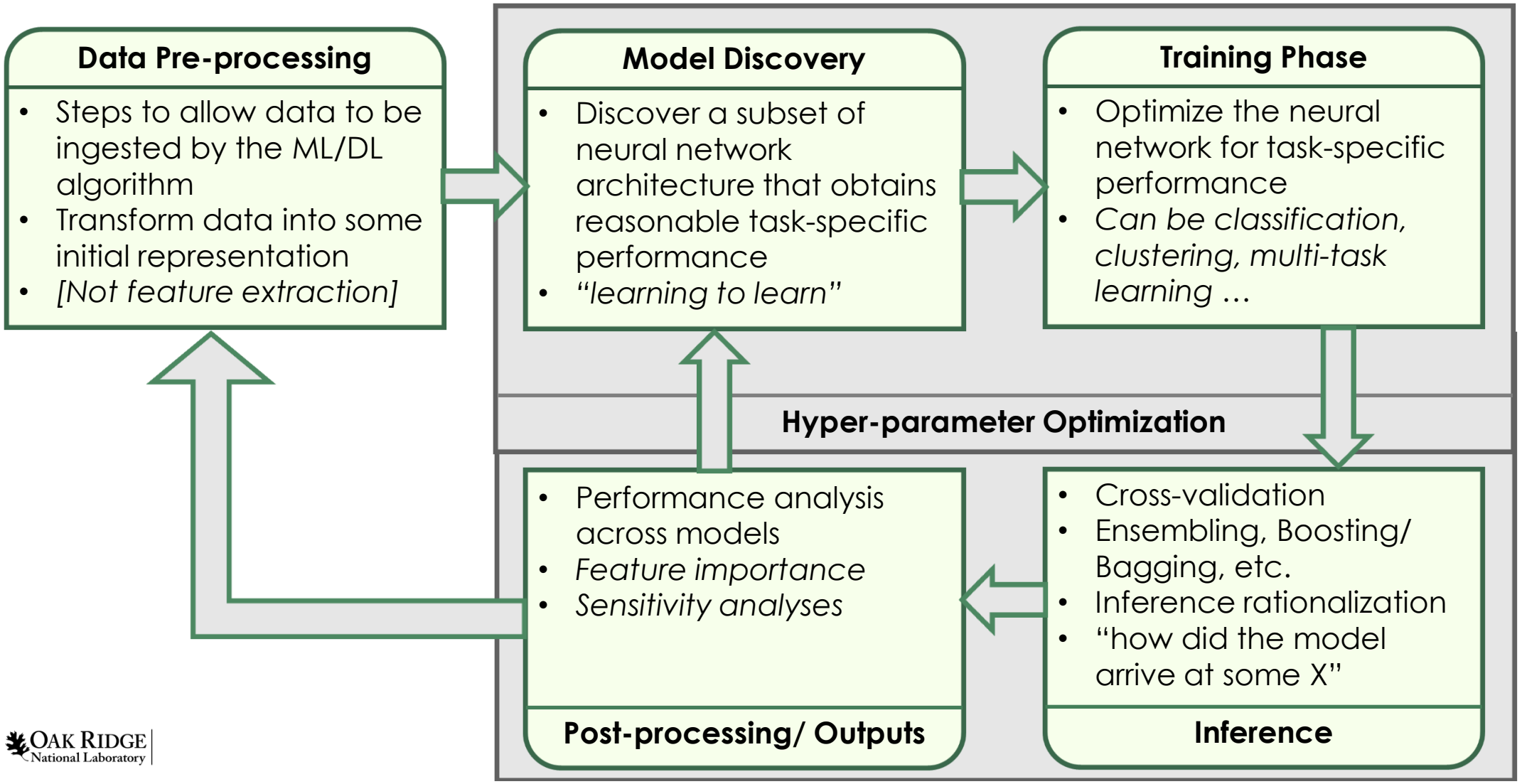
- Some working deep learning terms/ terminology
- Challenge problems
- CANDLE Workflows
- A practical example of clustering biomolecular simulations into conformational states

CANDLE vs. other ECP Applications

- CANDLE supports large-scale deep learning workflows:
  - Goal is to enable massive throughput for:
    - Data pre-processing,
    - Model search, training, and inferencing
    - Post-processing
  - Provide the necessary compute infrastructure that enables massive throughput on emerging Exascale architectures:
    - Support data parallel and model parallel paradigms for efficient use of resources
    - Enable hyperparameter searches over large/ complex models
  - Build on open source platforms that are optimized for CORAL architectures and beyond
  - Enable practical feedback between simulations/ experiments

A practical walk-through of developing a deep learning model

A general deep learning workflow



Example Pilot 1 problem: Modeling Cancer Drug Response

Models vary in {Samples, Drug or Drug Combinations, One or More Studies, Explicit or Implicit Dose, Growth (%, Categorical Response, Time for Doubling)}

Models vary in features used to characterize Samples and Drugs {GE, SNP, Descriptors, Fingerprints}

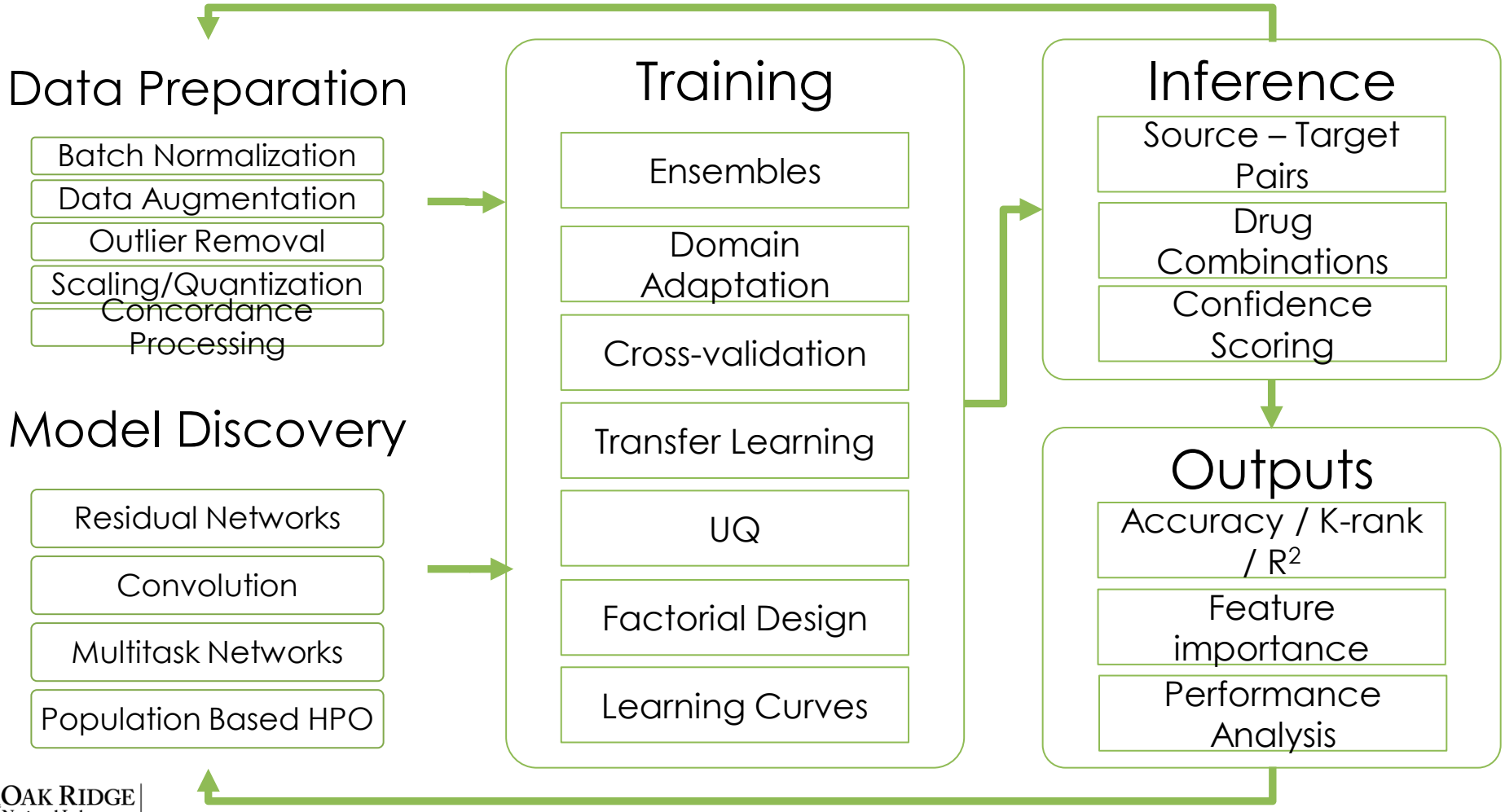
We search machine learning methods (DNN, RF, GB, BMTMK, LightGBM, etc.) and compare them

Goal is is maximizing Model Accuracy, Generalizability and Transferability

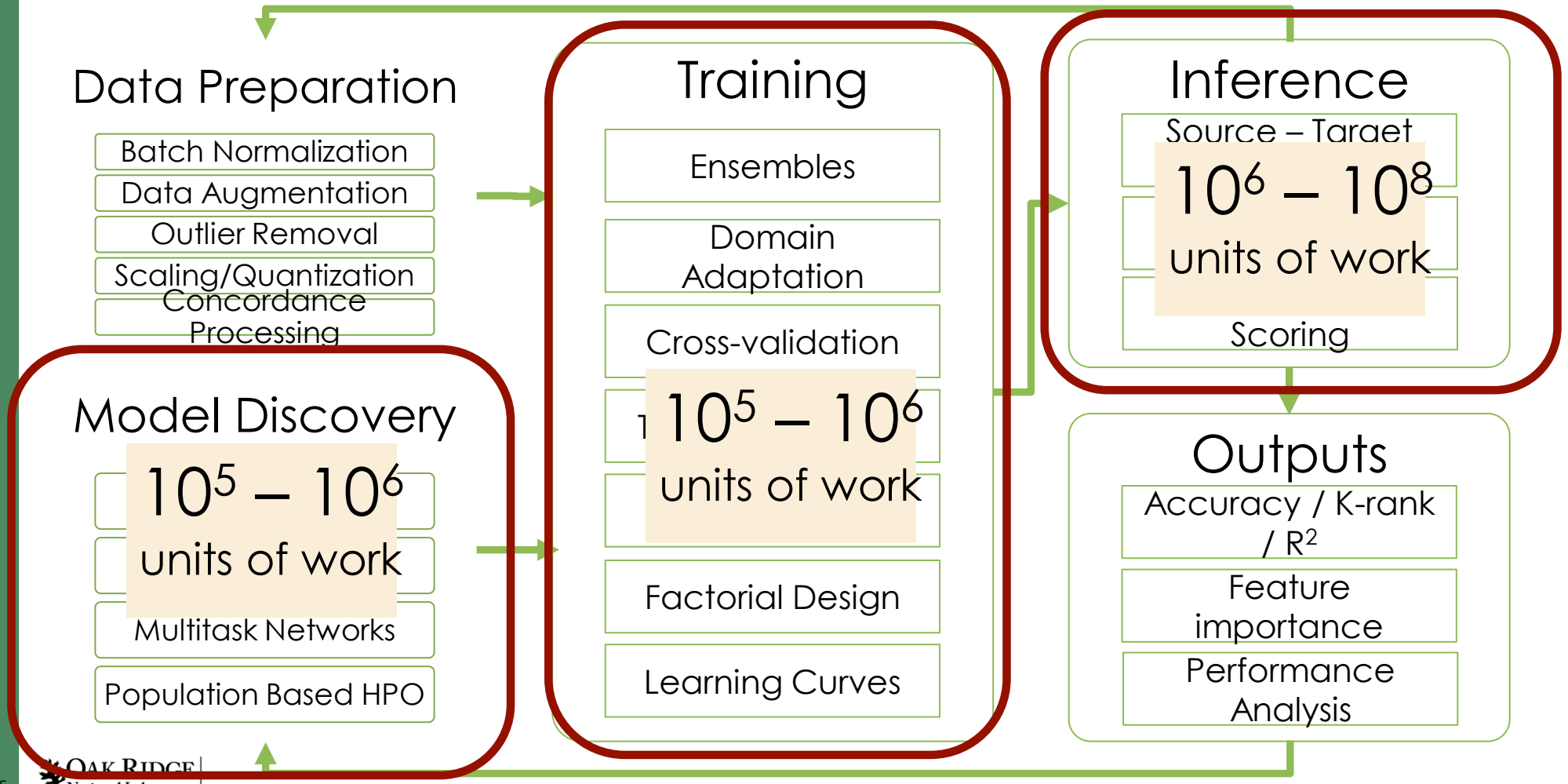
- Single Study (N Sample x Single Drug) One Model per Drug (M_{Dn}) ⟹ {Robot, RF, many models}
- Single Study (N Samples x Drug Pairs) One Model per Drug Pair (M_{Pn}) ⟹ {RF, BMTMK}
- Single Study (N Samples x Drug Pairs) One Model per Study (M_P) ⟹ {COMBO, RF}
- Single Study (N Samples x Single Drug and Drug Pairs) One Model per Study (M_+) ⟹ {UNO}
- M Studies (M*(N_m Samples x Single Drug)) One Model per All Studies (M*) ⟹ {Uno, Robot, Bres, RF}
- M Studies (M*(N_m Samples x Single Drug and Drug Pairs)) One Model per All Studies (M_+*) ⟹ {UNO}

Increase in computational complexity

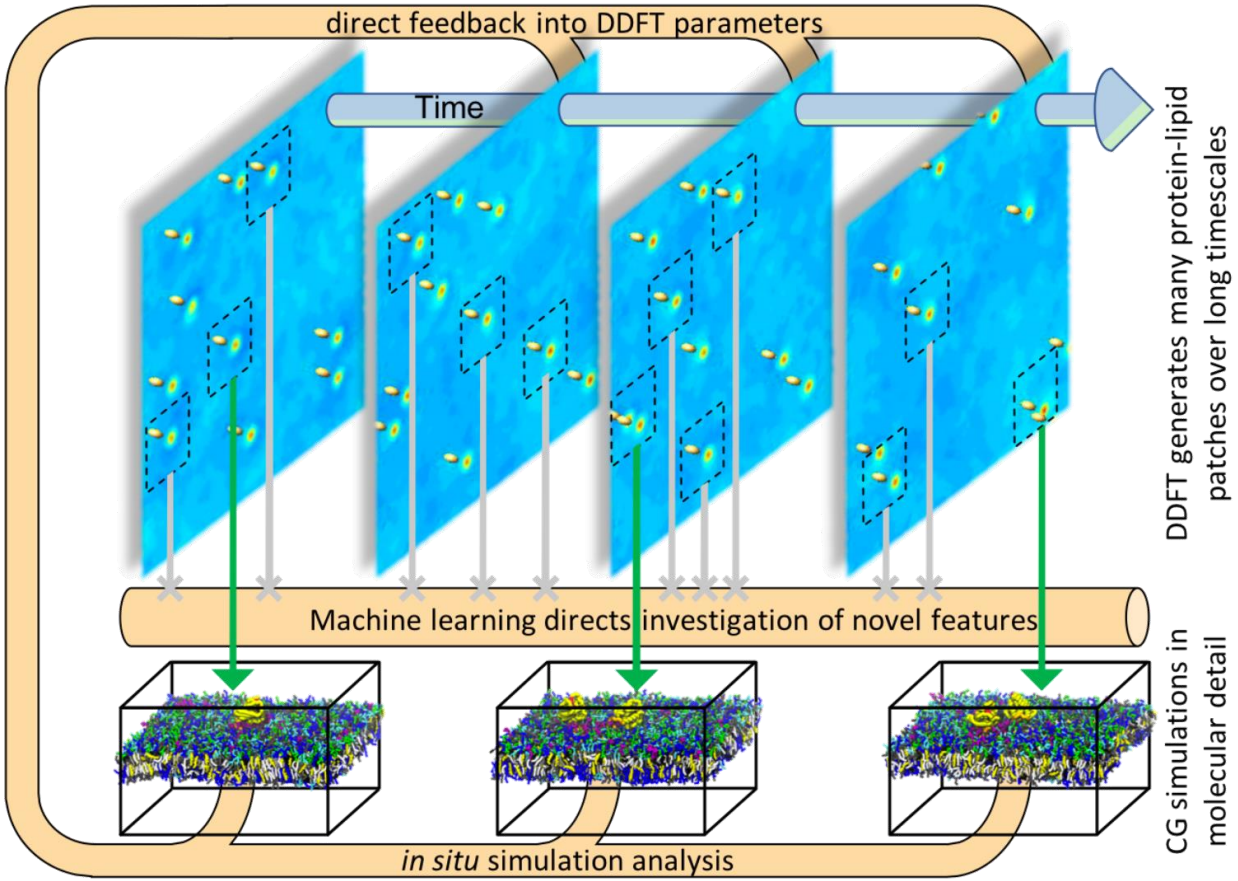
P1 Challenge Problem Workflow(s) Specification



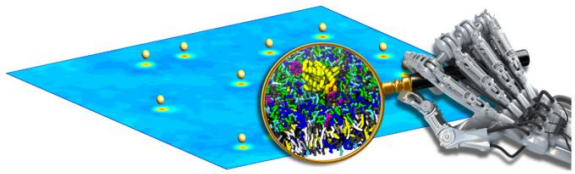
P1 Challenge Problem Workflow(s) Specification



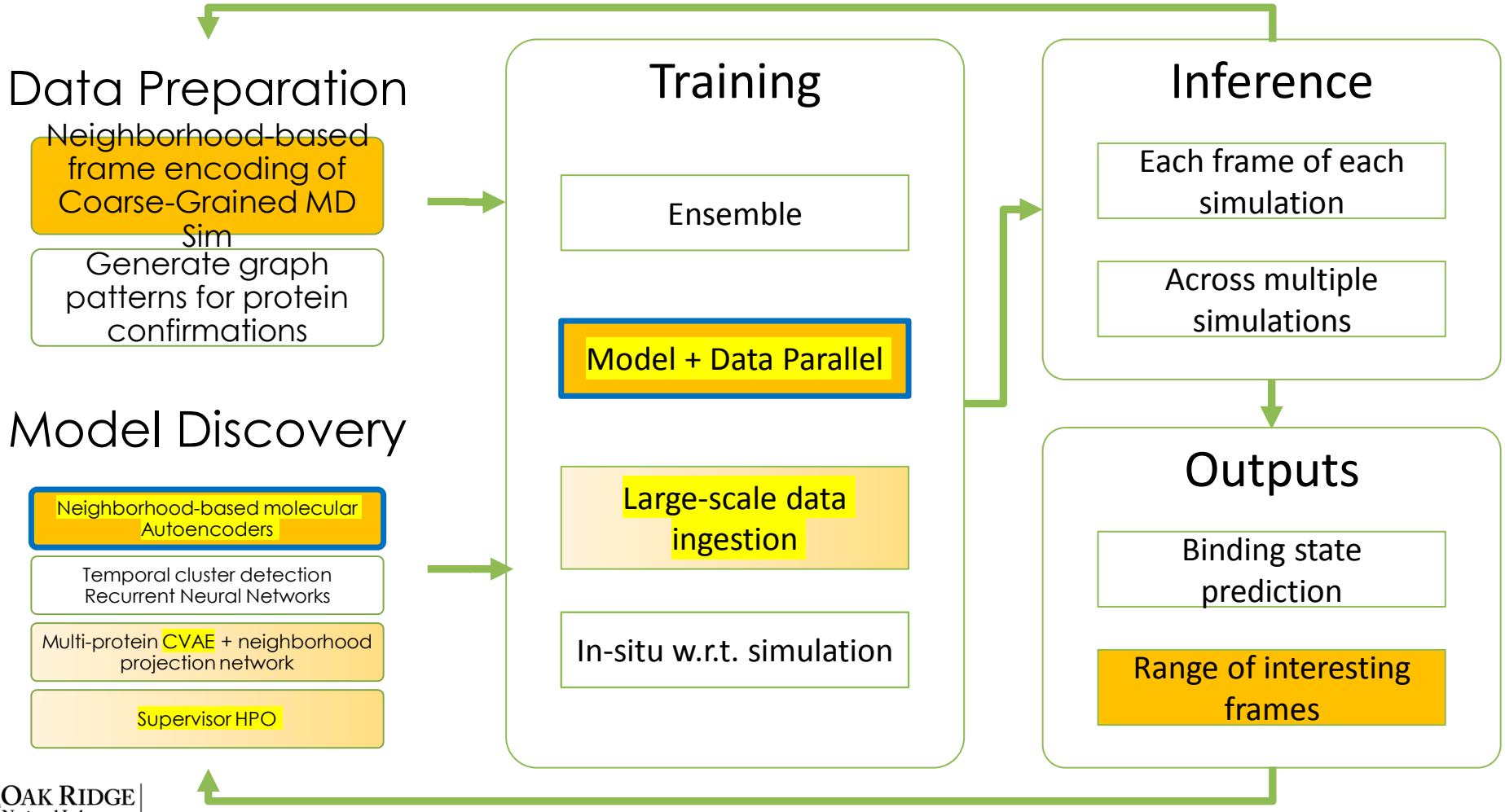
Pilot 2 – Detecting interesting RAS / RAF binding



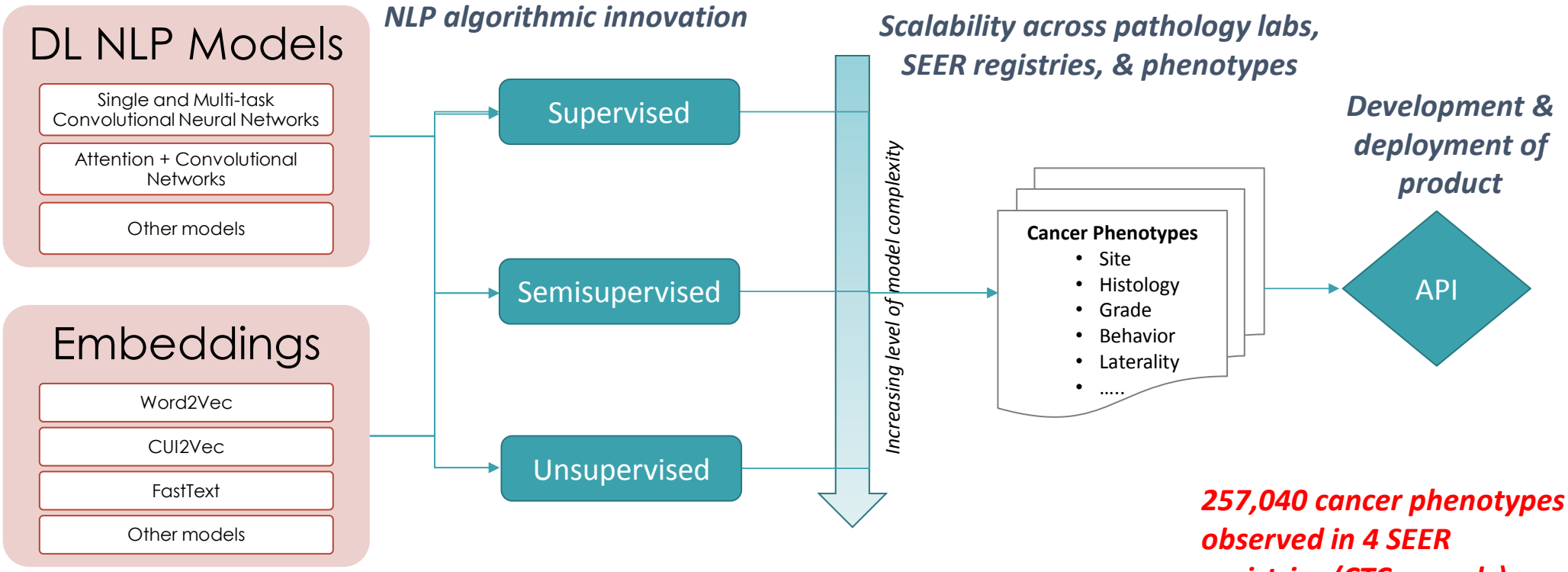
- Identify when the binding event is occurring within the simulation
- Classify the state of the binding as normal or anomalous
- Allow researchers to watch the evolution of the binding to find new methods for intervention



P2 Challenge Problem Workflow(s) Specification



Pilot 3: Treatment Strategy Problem

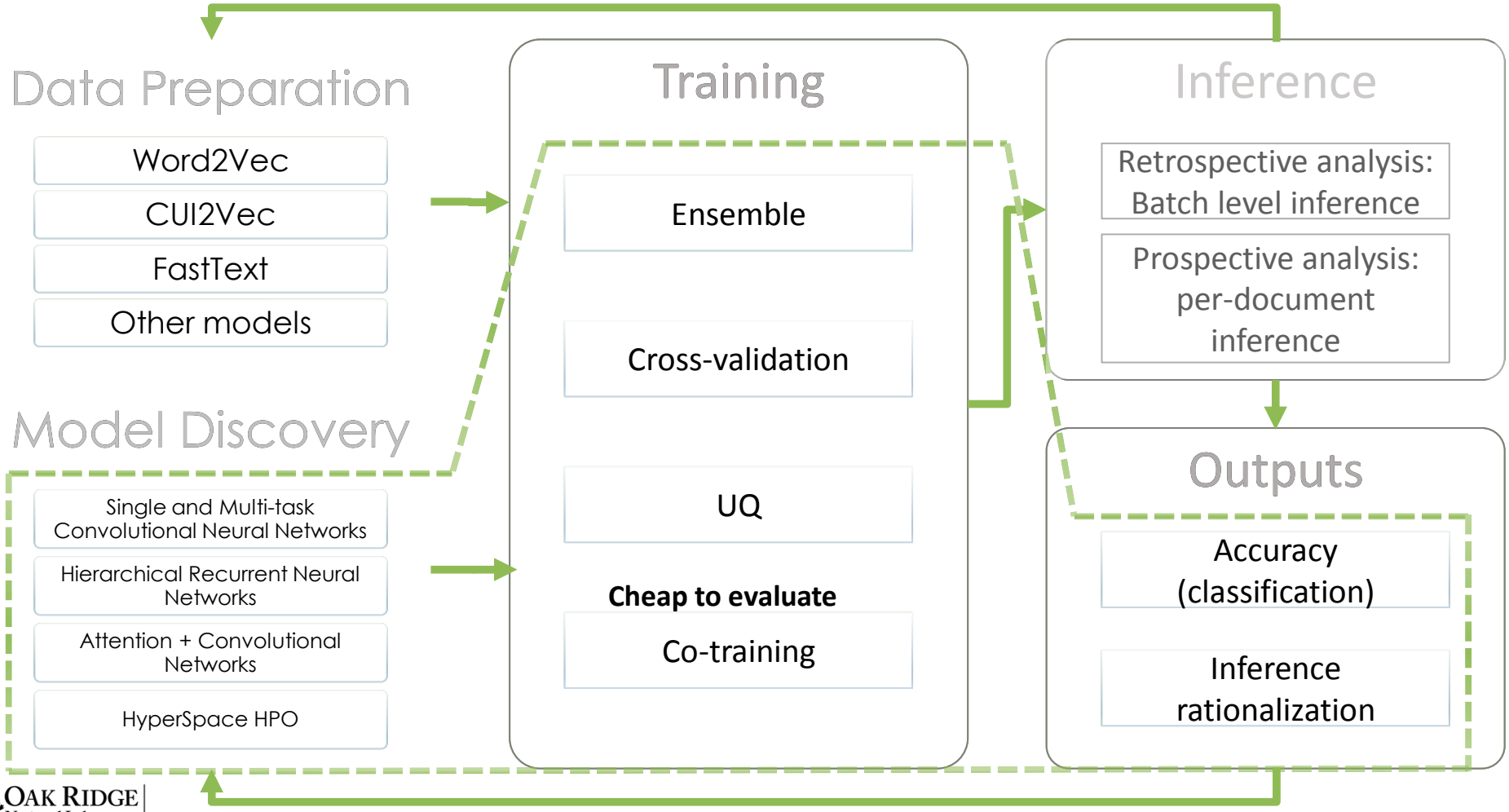


257,040 cancer phenotypes observed in 4 SEER registries (CTC records) based only on 5 attributes

70 cancer sites (306 subsites); 515 histologies; 9 grades; 7 lateralities; 4 behaviors →

Extension to other NLP tasks to extract more data elements (e.g., biomarkers) will increase the number and complexity of cancer phenotypes observed – **combinatorial explosion in computational cancer phenotyping → Exascale computing**

P3 Challenge Problem Workflow(s) Specification



CANDLE Challenge Problem Workflow Structure

Each challenge problem has three compute intensive phases and two data processing phases

1. Prepare input datasets, including normalization, outlier removal, joins and merges, etc.
2. Broad search of model space (AutoML+HPO) to find good performing model structures and hyper-parameter settings ⟹ running $O(10^4)$ – $O(10^6)$ model instances (HPO model selection ⟹ small number of model templates < 10)
3. Training a large-scale ensemble of the best model types on relevant training data for a "factorial study" ⟹ $O(10^6)$ models on $O(10^4)$ datasets (5x cross validation and optional bootstrap UQ) (CV model selection ⟹ small number of models < 10 per Source-Target pair)
4. Inferencing with selected models with UQ on new samples ($O(10^7)$ in (and out) of each Source-Target Pair distributions (UQ implies sampling model space $O(10^2)$ times)
5. Post-process inference output for actionable decisions and to capture performance, confidence and scoring (for validation)

Example FOM Calculation

Titan Nodes ~ 18,688 A21 Nodelets ~ 94,000

- P1 unit 60 hrs to train on Titan, 24/60 * 18,688 = 7,475 u/day
- P1 unit 4 hrs to train on A21 "nodelet", 24/4 * 94,000 = 564,000 u/day
- P1 Ratio = 564,000 / 7,475 = 75.45x

- P2 unit 6 hrs to train on Titan, 24/6 * 18,688 = 74,752 u/day
- P2 unit 0.5 hrs to train on A21 "nodelet", 24/0.5 * 94,000 = *4,512,000 u*/day
- P2 Ratio = 4,512,000/ 74,752 = 60.36x

- P3 unit 12 hrs to train on Titan, 24/12 * 18,688 = 37,376 u/day
- P3 unit .25 hrs to train on A21 "nodelet", 24/0.25 * 94,000 = 9,024,000 u/day
- P3 Ratio = 9,024,000 / 37,376 = 241.43x

- $H_{titan} = \frac{1\,676\,313\,600}{97\,177} \approx 17\,250.1$ $H_{a21} = \frac{27\,072\,000}{19} \approx 1.42484 \times 10^6$

- H_{a21} / H_{titan} = FOM = $\frac{68\,509\,785}{829\,426} \approx 82.599$ vs Mean[75.45, 60.36, 241.43] = 125.747

Walk-through of a practical example of identifying conformational states from MD simulations

OAK RIDGE National Laboratory

Capabilities and Ongoing Research

- HyperSpace: Distributed Bayesian Hyperparameter optimization
- Supporting data parallel and model parallel runs with CANDLE

Current platforms for hyperparameter optimization rely on sequential optimization techniques

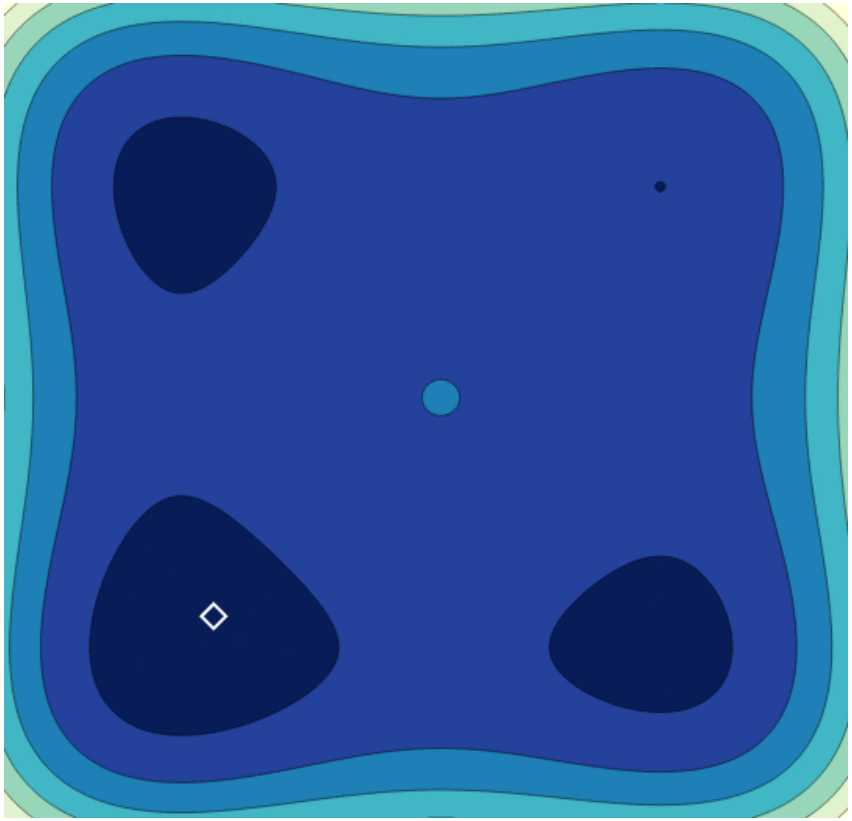
- Bayesian optimization, Bandit
  usually sequential search proc
- Exponential scaling:
  - The number of samples required
    optimization procedure is scales
    search dimensions, as in 2^D, whe
  - Forgotten in the recent exciteme

HyperSpace: Distributed Parallel Bayesian Optimization

- Hyperspace, instead seeks to focus on the search space:
  - Parallelism to exploit the statistical structure of the search space
  - Reveal partial dependencies across parameter spaces
- Build many surrogate functions in parallel
- {Prayer}!

N. Srinivas, A. Krause, S. M. Kakade, and M. W. Seeger. Gaussian pr
abs/0912.3995, 2009.
S. Grunewalder, J.-Y. Audibert, M. Opper, and J. Shawe-Taylor. Regre
Thirteenth International Conference on Artificial Intelligence and Statis
2010.

HyperSpace: Parallel exploration of large search spaces



1. Define the bounds of each hyperparameter search space.

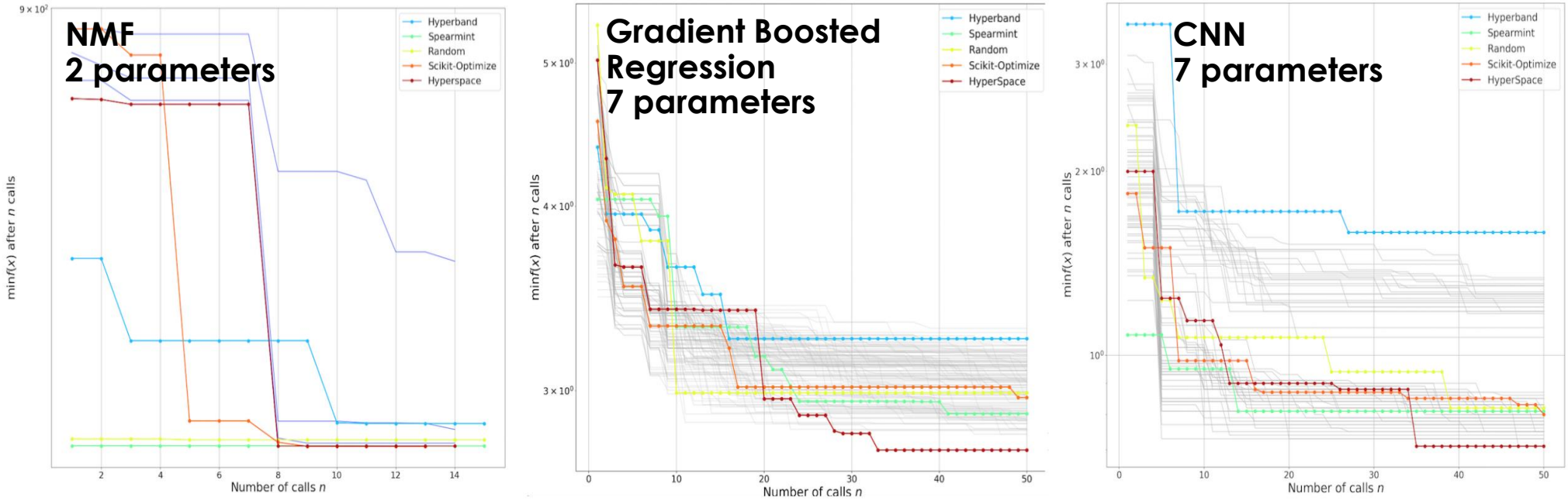
2. Divide each search space bound into two nearly equal sub-bounds with overlap ϕ, where $\{\phi \in \mathbb{R} \mid 0 \leq \phi \leq 1\}$.

3. Create all possible combinations of hyperparameter sub-bounds to form 2^D search spaces (hyperspaces) where D is the number of model hyperparameters.

4. Run Bayesian optimization over each hyperspace in parallel

M. Todd Young, J. D. Hinkle, R. Kannan, A. Ramanathan, HyperSpace: Massively Parallel Bayesian Optimization, Workshop on High Performance Machine Learning, 2018, Lyon, France

HyperSpace can optimize many different ML/DL approaches

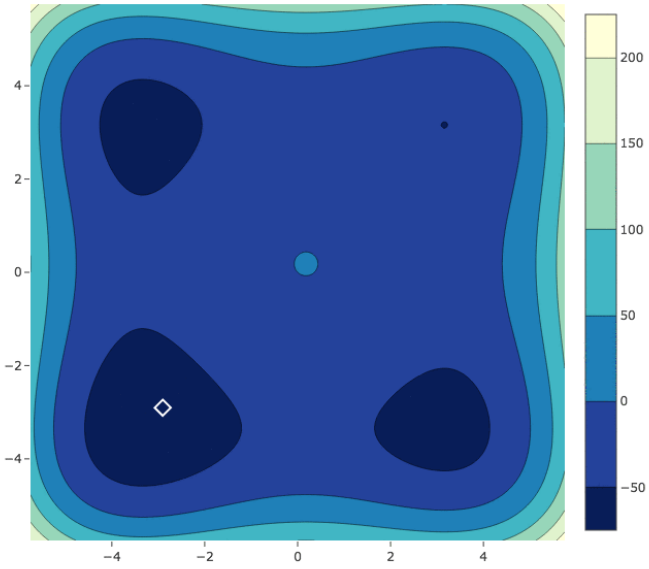


HyperSpace can effectively scale across supercomputing resources to reveal how models perform under many unique hyperparameter configurations.

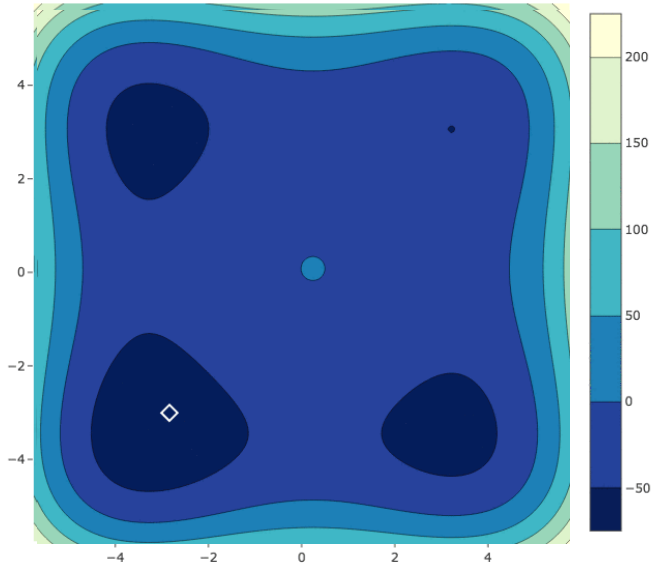
- Discovers regions in the hyperparameter search space where models perform well *and* where they perform poorly
- Finds families of solutions where various settings of hyperparameters perform equally well
- Opens the possibility of meta learning for hyperparameter optimization (future direction)

M. Todd Young, J. D. Hinkle, R. Kannan, A. Ramanathan, HyperSpace: Massively Parallel Bayesian Optimization, Workshop on High Performance Machine Learning, 2018, Lyon, France

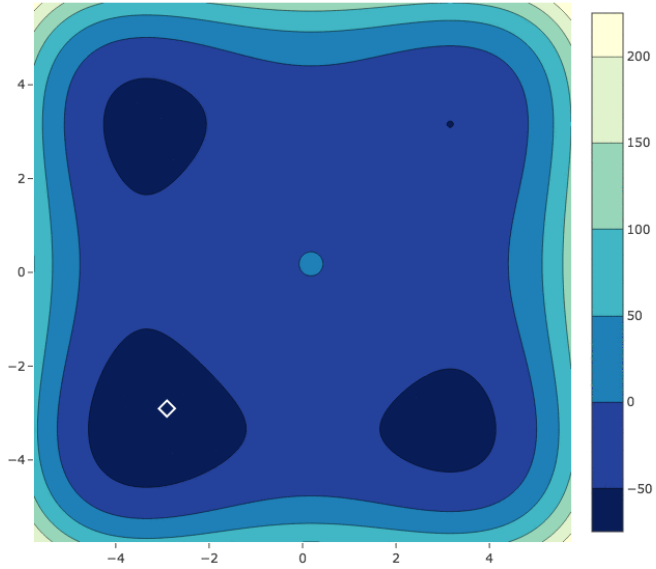
Parallel exploration of large search spaces works better than random/ sequential based optimization



HyperSpace

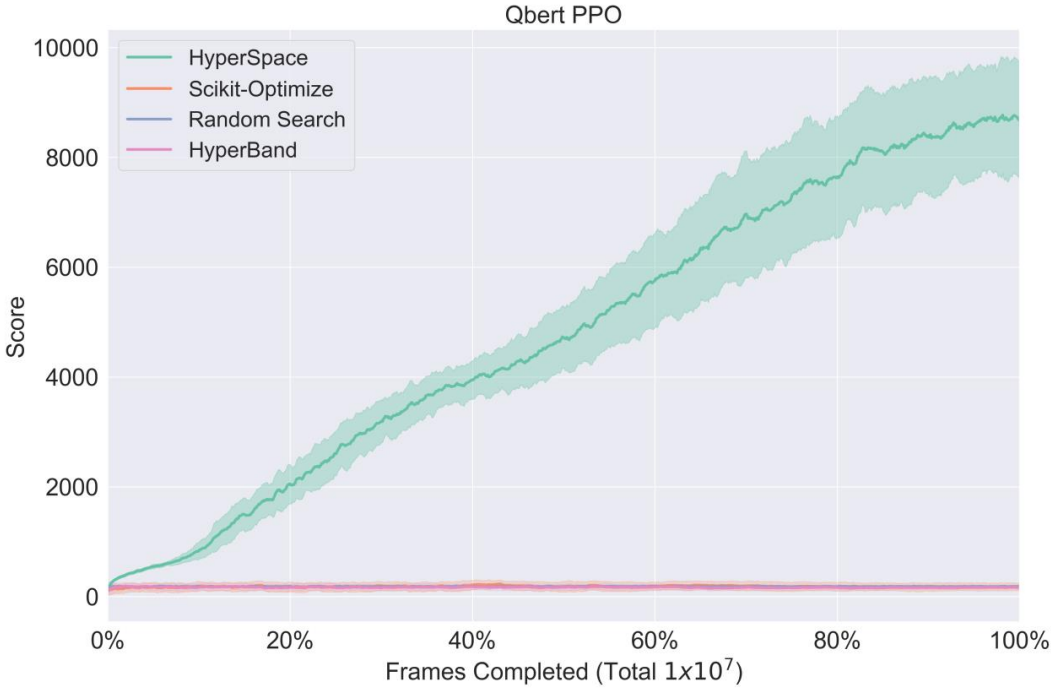


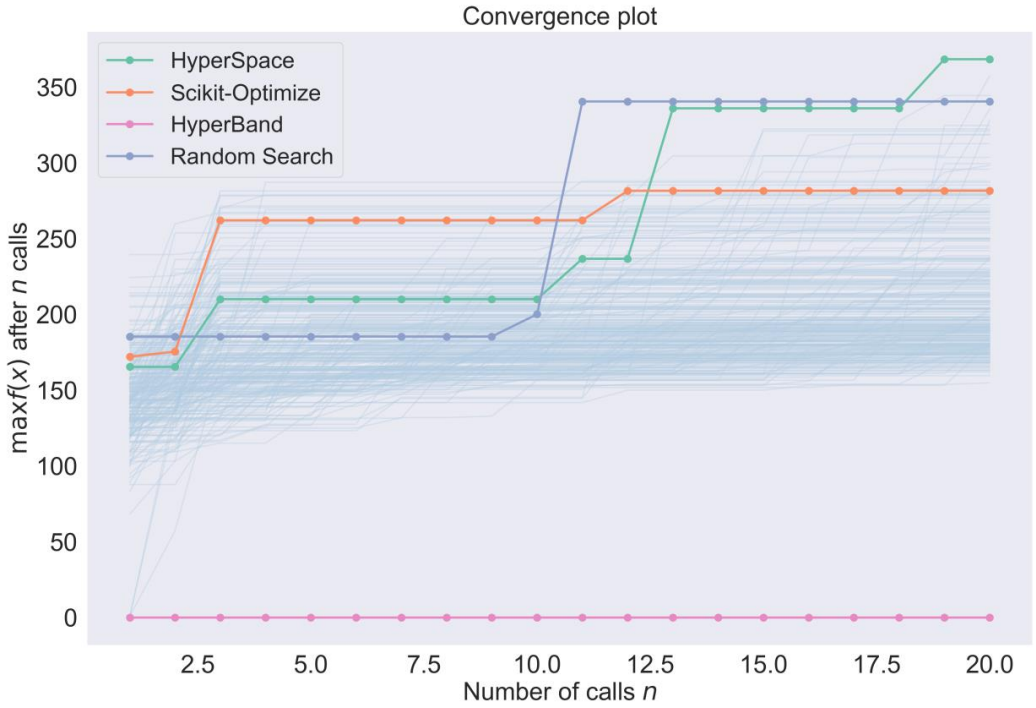
Random search



SMBO

HyperSpace can be used to optimize challenging reinforcement learning approaches





- RL ideas are really hard to optimize!
- HyperSpace performs better models with rewards that are significantly better for a variety of games

Summary

- CANDLE offers a scalable computational infrastructure for deep learning workflows:
  - Minimal "code" modifications for CANDLE compliance
  - Novel, scalable approaches for hyper-parameter optimization and distributed learning
  - Inbuilt tools for UQ for design of robust models
- CANDLE applications are diverse:
  - Genomics,
  - MD simulations, and
  - NLP /text analytics
- Code and examples will demonstrate ideas further… (so, don't go anywhere…)

OAK RIDGE National Laboratory

THANK YOU…

Questions/ Comments:

ramanathana@ornl.gov

ORNL is managed by UT-Battelle, LLC for the US Department of Energy

U.S. DEPARTMENT OF ENERGY

CANDLE Tutorial: Library Overview

ECP Annual Meeting, Houston, TX
January 2019

Jamaludin Mohd Yusof
Advanced Architecures and Applications
Los Alamos National Laboratory
jamal@lanl.gov

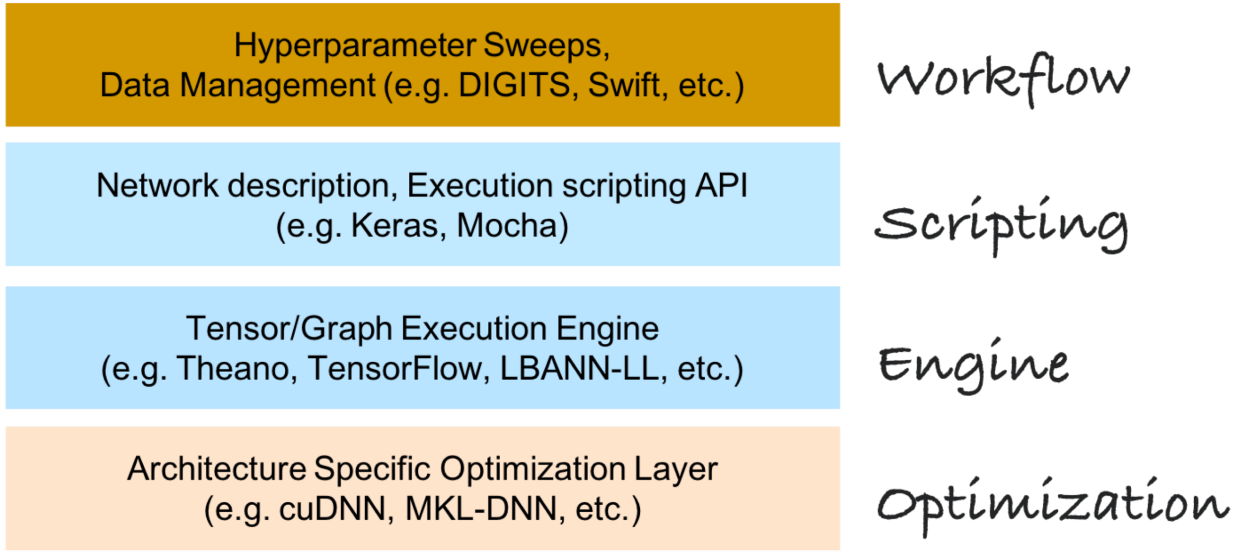
Talk Outline

- Introduction
- Benefits
- Library Overview
- Example Benchmark Workflow
- Simple Parameter Sweeps

Introduction

- Purpose
  - To streamline the writing of CANDLE-compliant codes
  - Allow rapid prototyping and exploration of hyperparameters
  - Integrate with the Supervisor framework
- Historical perspective
  - Consolidation of frequently used functionality from the Benchmark codes
  - Evolving to incorporate new functionality as needed
    - Improved usability over time

The CANDLE Environment



Benefits provided by CANDLE

- Consistent
  - Standardized network specification with a “`default_model_file`”
  - Standardized command line intercept protocol
  - Standardized default values across frameworks
  - Ideal for testing the same problems with consistency on new DOE hardware
- Convenient
  - Pass arguments via command line
    - Standard keywords parsed automatically, user can add new ones
  - Modify the default file
    - Provide a new default model specification ‘`--config_file new_default_model.txt`’

Benefits provided by CANDLE

- Provides various utility packages that promote reuse and streamline code development
  - Actively developed, new functionality based on need
- Provides the pathway for inferencing, data-parallelism, automated sweeps of hyperparameters
- Availability of a robust framework for documentation and testing
- Pre-existing for containers such as Singularity (Ex. machines such as Theta, Titan, Cori, summitdev)
- Documentation: https://ecp-candle.github.io/Candle/html/index.html

Library Overview

- Integrated into the scripting level of CANDLE stack
  - Keras-based (for now)
  - Allows multiple backends (Tensorflow, CNTK, etc)
- Provides a single namespace for inclusion of useful functions into Benchmark codes
- Allows developers to decide which functions are exposed to users
  - `candle_keras` directory with `__init__.py` file sets included functions
- Allows reuse of non-Keras specific functions to create libraries for other languages

Library organization

- Utilities are organized by functionality
  - Transparent to the user, mostly framework agnostic
    - default_utils: create, modify parameter dictionary
    - file_utils: fetch and unpack da†a files
    - data_utils: load and manipulate data, enable UQ (via uq_utils)
    - generic_utils: callback function, standardize screen output
    - keras_utils: translation from CANDLE keywords, enhance Keras functionality
    - solr_keras: database functionality
    - viz_utils: visualize networks and output (prototype)

Example Benchmark Workflow

- `initialize_parameters`:
  - Read default values for the model from `default_model.txt` file
  - Automatically parse many standard ML hyperparameters
    - `learning_rate`, `batch_size`, etc
    - `data_url`, `train_data`, `test_data`...
  - Allows user to add other keywords via `additional_definitions`
  - Sets undefined hyperparameters to the corresponding Keras defaults to ensure consistency across backend frameworks
    - Early work revealed some performance differences were due to mismatched default settings

Example Benchmark Workflow

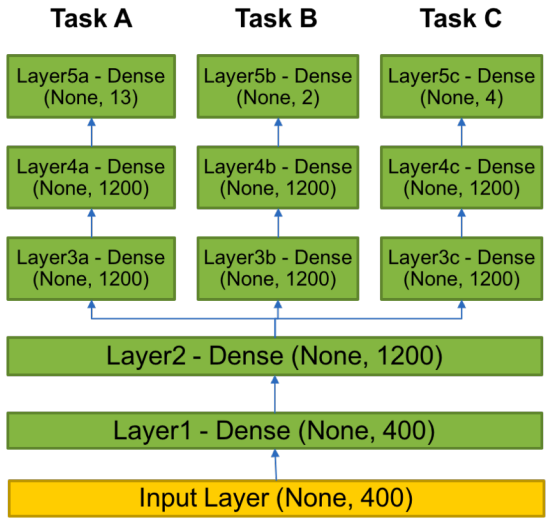
- `Data management`
  - `fetch_data`
    - Check if local data copy exists; if not, fetch data from `data_url`
    - allows separate `train_data` and `test_data` files, check MD5 hash if provided, untar if needed
  - `load_data`
    - Load labeled (xy) or unlabeled (x) data, perform various manipulations
      - Shuffle, scale, split into `{train, validation, test}`
      - Takes UQ operations to provide repeatable cross-validation data splits

Example Benchmark Workflow

- Model build
  - `keras_utils` translates CANDLE keywords into Keras methods
    - e.g. optimizer, initializer specification
  - Extends Keras functionality where useful
    - `PermanentDropout`
  - `default_utils` adds default `EXP000` and `RUN000` directory structure
  - `solr_keras` adds monitoring and logging
    - Timeout functionality to respect job limits

Example benchmark

- **P3B1: Multi-task Deep Neural Net (DNN) for data extraction from clinical reports**
- **Overview**: Given a corpus of patient-level clinical reports, build a deep learning network that can simultaneously identify:(i) b tumor sites, (ii) t tumor laterality, and (iii) g clinical grade of tumors.
- **Relationship to core problem**: Instead of training individual deep learning networks for individual machine learning tasks, Build a multi-task DNN that can exploit task-relatedness to simultaneously learn multiple concepts.
- **Expected outcome**: Multi-task DNN that trains on same corpus and can automatically classify across three related tasks.



Original code

```
shared_nnet_spec= [ 1200 ]
individual_nnet_spec0= [ 1200, 1200 ]
individual_nnet_spec1= [ 1200, 1200 ]
individual_nnet_spec2= [ 1200, 1200 ]
individual_nnet_spec = [ individual_nnet_spec0, individual_nnet_spec1,
individual_nnet_spec2 ]

learning_rate = 0.01
batch_size = 10
n_epochs = 10
dropout = 0.0

## Read files
from data_utils import get_file
origin = 'http://ftp.mcs.anl.gov/pub/candle/public/benchmarks/P3B1/P3B1_data.tgz'
data_loc = get_file('P3B1_data.tgz', origin, untar=True, md5_hash=None,
cache_subdir='P3B1')
```

CANDLE code

```
gParameters = initialize_parameters()
….

# input layer
layer = Input( shape = ( input_dim, ), name= 'input' )
shared_layers.append( layer )

# shared layers
for k in range( len( shared_nnet_spec ) ):
      layer = Dense( shared_nnet_spec[ k ], activation=gParameters['activation'],
            name= 'shared_layer_' + str( k ) )( shared_layers[ -1 ] )

      if gParameters['drop'] > 0:
            layer = Dropout( gParameters['drop'] )( shared_layers[ -1 ] )
      shared_layers.append( layer )

# individual layers
……
```

CANDLE model file

```
data_url = 'ftp://ftp.mcs.anl.gov/pub/candle/public/benchmarks/P3B1/’
train_data = 'P3B1_data.tar.gz’
model_name = 'p3b1’
learning_rate = 0.01
batch_size = 10
epochs = 10
drop = 0.0
activation = 'relu’
out_activation = 'softmax’
loss = 'categorical_crossentropy’
optimizer = 'sgd’
metrics = 'accuracy’
n_fold = 1
shared_nnet_spec = '1200’
ind_nnet_spec = '1200, 1200:1200, 1200:1200, 1200’
feature_names = 'Primary site:Tumor laterality:Histological grade’
timeout =1800
scaling = 'none’
output_dir = '.’
initialization='glorot_uniform’
```

Simple parameter exploration

- Provide a new default model specification
  - `'--config_file new_default_model.txt'`
- Overwrite individual parameters in the default model
  - `'--learning_rate 0.1 -drop 0.1'`
- Provides an easy way to perform individual experiments to probe the hyperparameter space

```
python myDNN.py -learning_rate 0.01 -run_id "run1"
python myDNN.py -learning_rate 0.02 -run_id "run2"
```

- Provides the pathway for automated sweeps of hyperparameters
  - → Supervisor workflows

Acknowledgments

This research was supported by the Exascale Computing Project (ECP), Project Number: 17-SC-20-SC, a collaborative effort of two DOE organizations - the Office of Science and the National Nuclear Security Administration, responsible for the planning and preparation of a capable exascale ecosystem, including software, applications, hardware, advanced system engineering and early testbed platforms, to support the nation's exascale computing imperative.

Argonne NATIONAL LABORATORY

MIGRATING YOUR ML TO CANDLE

HARRY YOO
Software Engineer
Data Science and Learning Division

hsyoo@anl.gov
Mon, Jan 14th 2019,
ECP Annual Meeting

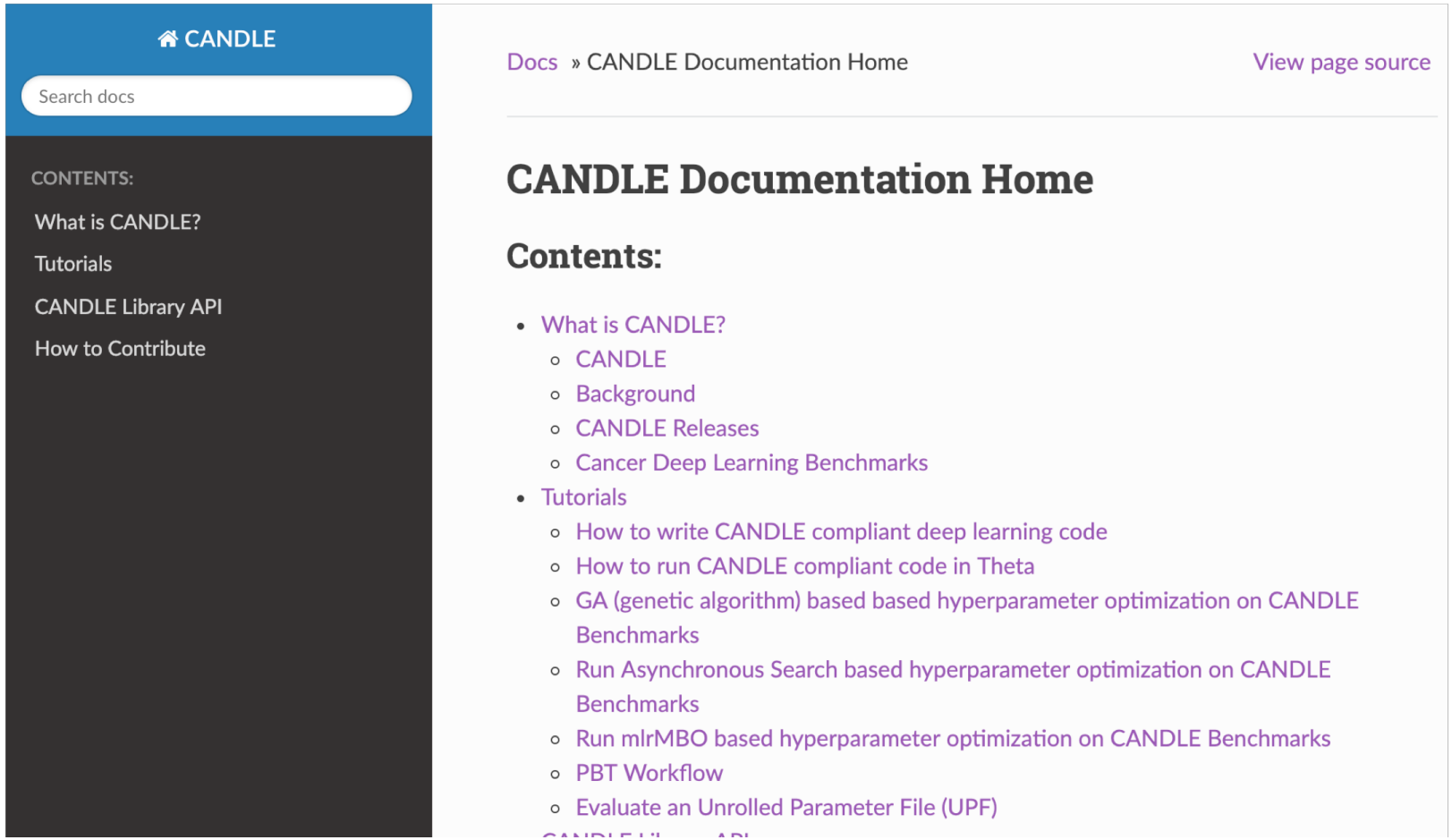
Argonne NATIONAL LABORATORY

AGENDA

- Documentation Site
- Code Structure
- Parameters
- Custom data loading

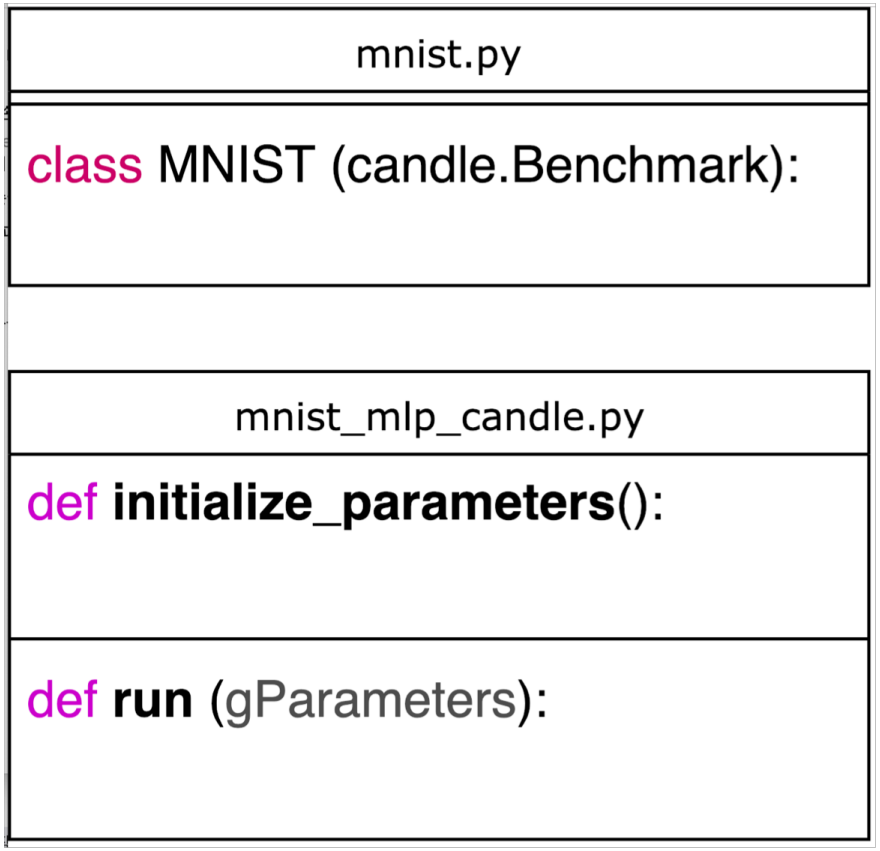
DOCUMENTATION

https://ecp-candle.github.io/Candle/



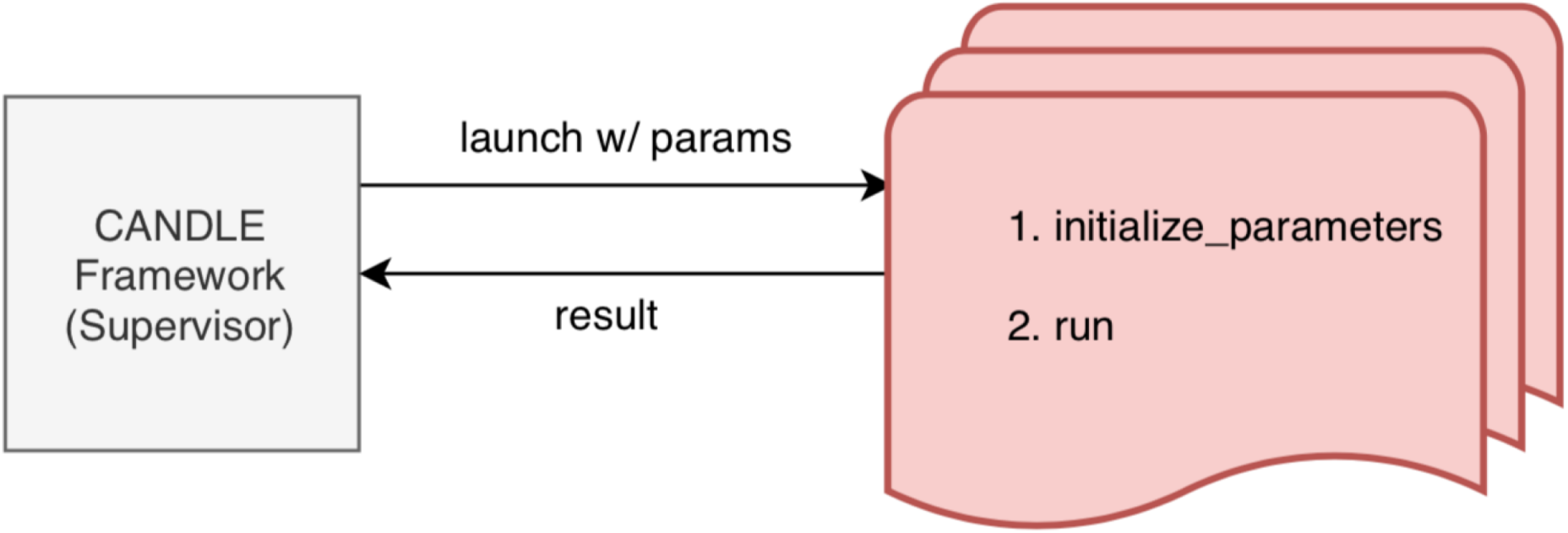
CODE STRUCTURE

1. `import candle_keras as candle`
2. Construct a class extending `candle.Benchmark`
3. Implement
   - `initialize_parameters`
   - `run`
4. (later) implement data loading for your data



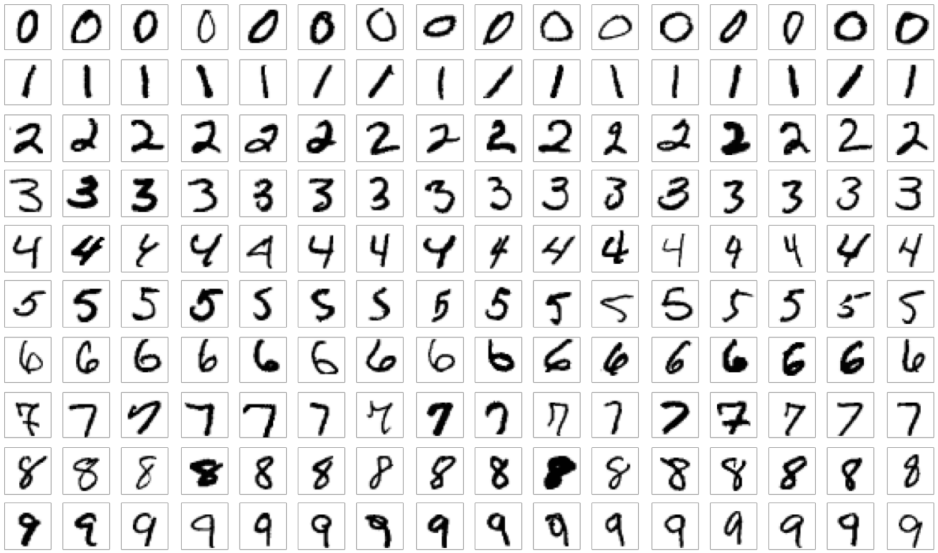
`https://github.com/ECP-CANDLE/Candle/tree/master/examples/mnist`

WHY


CANDLE
Framework
(Supervisor)
launch w/ params
result
1. initialize_parameters
2. run

ABOUT MNIST

- Handwritten digits (0 – 9)
- 28 x 28 pixels
- 60,000 training, 10,000 test samples
- http://yann.lecun.com/exdb/mnist/
- `from keras.datasets import mnist`



MNIST MLP

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_1 (Dense)              (None, 512)               401920
_________________________________________________________________
dropout_1 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_2 (Dense)              (None, 512)               262656
_________________________________________________________________
dropout_2 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_3 (Dense)              (None, 10)                5130
=================================================================
Total params: 669,706
Trainable params: 669,706
Non-trainable params: 0
_________________________________________________________________
```

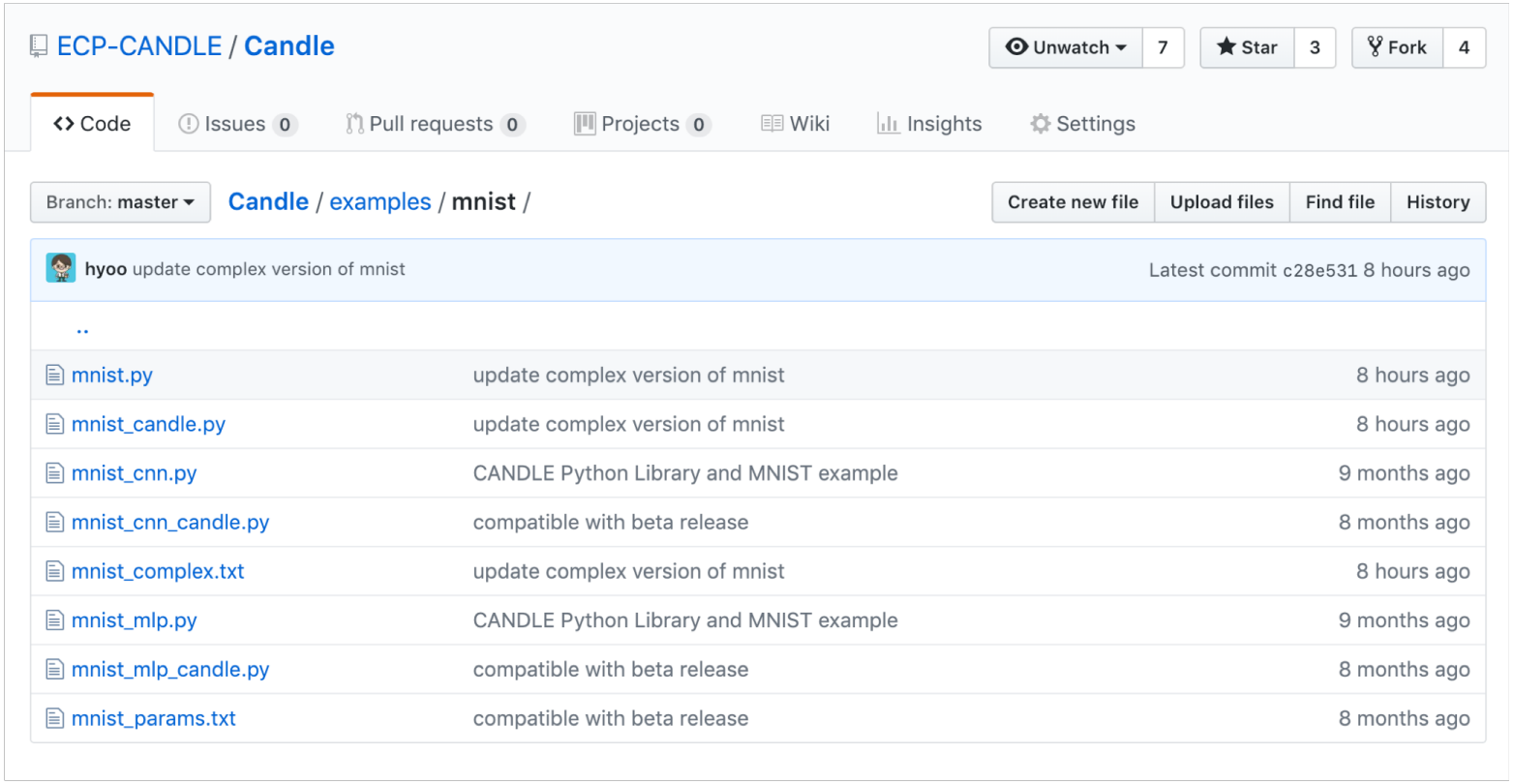
https://github.com/ECP-CANDLE/Candle/blob/master/examples/mnist/mnist_mlp.py

MNIST CNN

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 26, 26, 32)        320
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 24, 24, 64)        18496
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 12, 12, 64)        0
_________________________________________________________________
dropout_1 (Dropout)          (None, 12, 12, 64)        0
_________________________________________________________________
flatten_1 (Flatten)          (None, 9216)              0
_________________________________________________________________
dense_1 (Dense)              (None, 128)               1179776
_________________________________________________________________
dropout_2 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_2 (Dense)              (None, 10)                1290
=================================================================
Total params: 1,199,882
Trainable params: 1,199,882
Non-trainable params: 0
```

MNIST CODE REVIEW

https://github.com/ECP-CANDLE/Candle/tree/master/examples/mnist



HANDS ON

```
(keras) ❯ python mnist_mlp_candle.py
Using TensorFlow backend.
Configuration file:  /Users/hsyoo/dev/CANDLE/Candle/examples/mnist/mnist_params.txt
{'activation': 'relu', 'batch_size': 128, 'epochs': 20, 'optimizer': 'rmsprop'}
Params:
{'activation': 'relu',
 'batch_size': 128,
 'datatype': <class 'numpy.float32'>,
 'epochs': 20,
 'experiment_id': 'EXP000',
 'gpus': [],
 'logfile': None,
 'optimizer': 'rmsprop',
 'output_dir': '/Users/hsyoo/dev/CANDLE/Candle/examples/mnist/Output/EXP000/RUN000',
 'rng_seed': 7102,
 'run_id': 'RUN000',
 'shuffle': False,
 'timeout': -1,
 'train_bool': True,
 'verbose': None}
```

PARAMETERS

- Default Parameters
  - Use `-help` to check
- Add / mandate parameters
- Use model parameter file to set default sets
  - `mnist_params.txt`
- Overwrite parameters as needed

OVERWRITING PARAMETERS

1. Default value defined in CANDLE Library
2. Values defined in default parameter file. e.g. `mnist_params.txt`
3. Parameters file overwritten by `--conf`
4. Individual parameter values provided by argument. e.g. `-e 1`

HANDS ON

- Run without arguments
  - `python mnist_mlp_candle.py`
- Run using model file
  - `python mnist_mlp_candle.py -conf model_complex.txt`
- Run overwriting with argument
  - `python minst_mlp_candle.py -e 1`
- Let make it more complicated
  - `python mnist_candle.py`

ADDITIONAL PARAMS

- Declare additional parameters in JSON format

```
additional_definitions = [
{'name':'latent_dim',
    'action':'store',
    'type': int,
    'help':'latent dimensions'},
{'name':'model',
    'default':'ae',
    'choices':['ae', 'vae', 'cvae'],
    'help':'model to use: ae, vae, cvae'},
{'name':'use_landmark_genes',
    'type': candle.str2bool,
    'default': False,
    'help':'use the 978 landmark genes from LINCS (L1000) as expression features'},
```

https://github.com/ECP-CANDLE/Benchmarks/blob/master/Pilot1/P1B1/p1b1.py

REQUIRED PARAMS

```
74  required = [
75      'activation',
76      'batch_size',
77      'dense',
78      'drop',
79      'epochs',
80      'initialization',
81      'learning_rate',
82      'loss',
83      'noise_factor',
84      'optimizer',
85      'rng_seed',
86      'model',
87      'scaling',
88      'validation_split',
```

https://github.com/ECP-CANDLE/Benchmarks/blob/master/Pilot1/P1B1/p1b1.py

LOADING DATA

```
def load_data(params, seed):
    drop_cols = ['case_id']
    onehot_cols = ['cancer_type']
    y_cols = ['cancer_type']

    if params['use_landmark_genes']:
        lincs_file = 'lincs1000.tsv'
        lincs_path = candle.fetch_file(params['url_p1b1'] + lincs_file, 'Pilot1')
        df_l1000 = pd.read_csv(lincs_path, sep='\t')
        x_cols = df_l1000['gdc'].tolist()
        drop_cols = None
    else:
        x_cols = None

    train_path = candle.fetch_file(params['url_p1b1'] + params['file_train'], 'Pilot1')
    test_path = candle.fetch_file(params['url_p1b1'] + params['file_test'], 'Pilot1')

    return candle.load_csv_data(train_path, test_path,
                                x_cols=x_cols,
                                y_cols=y_cols,
                                drop_cols=drop_cols,
                                onehot_cols=onehot_cols,
                                n_cols=params['feature_subsample'],
                                shuffle=params['shuffle'],
                                scaling=params['scaling'],
                                dtype=params['datatype'],
                                validation_split=params['validation_split'],
                                return_dataframe=False,
                                return_header=True,
                                seed=seed)
```

Check CANDLE Library API for more detail

Argonne NATIONAL LABORATORY

THANKS

Argonne NATIONAL LABORATORY

UNCERTAINTY QUANTIFICATION

HARRY YOO
Software Engineer
Data Science and Learning Division

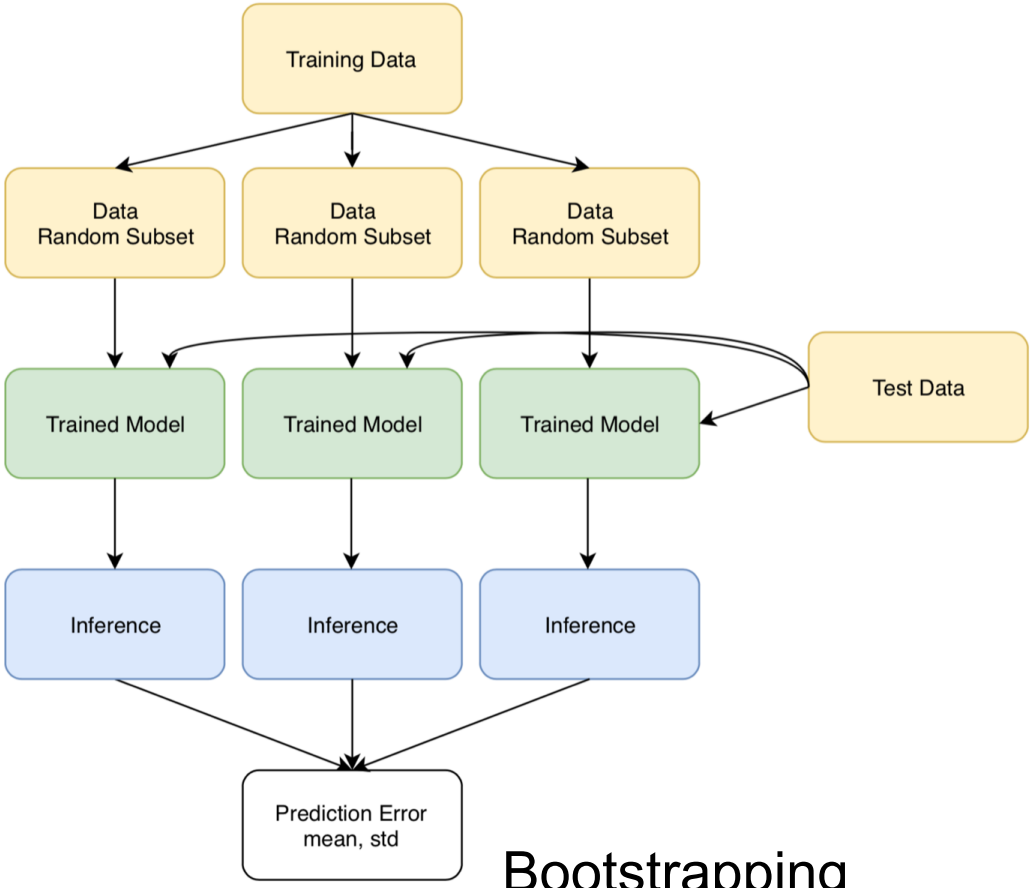
hsyoo@anl.gov
Mon, Jan 14th 2019,
ECP Annual Meeting

Argonne NATIONAL LABORATORY

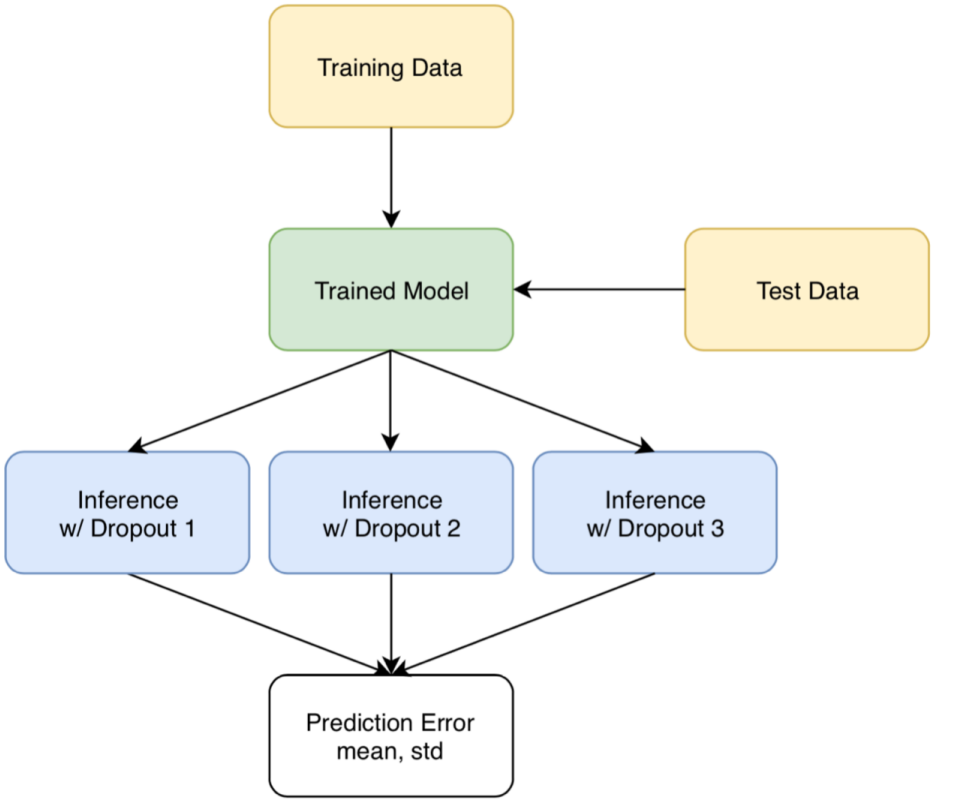
OUTLINE

- Uncertainty Quantification Methods: Bootstrap, Dropout
- Comparing DL UQ Variants
- CANDLE UQ Utilities & Class
- CANDLE UPF workflow
- Demo

CONCEPT DIAGRAM



Bootstrapping

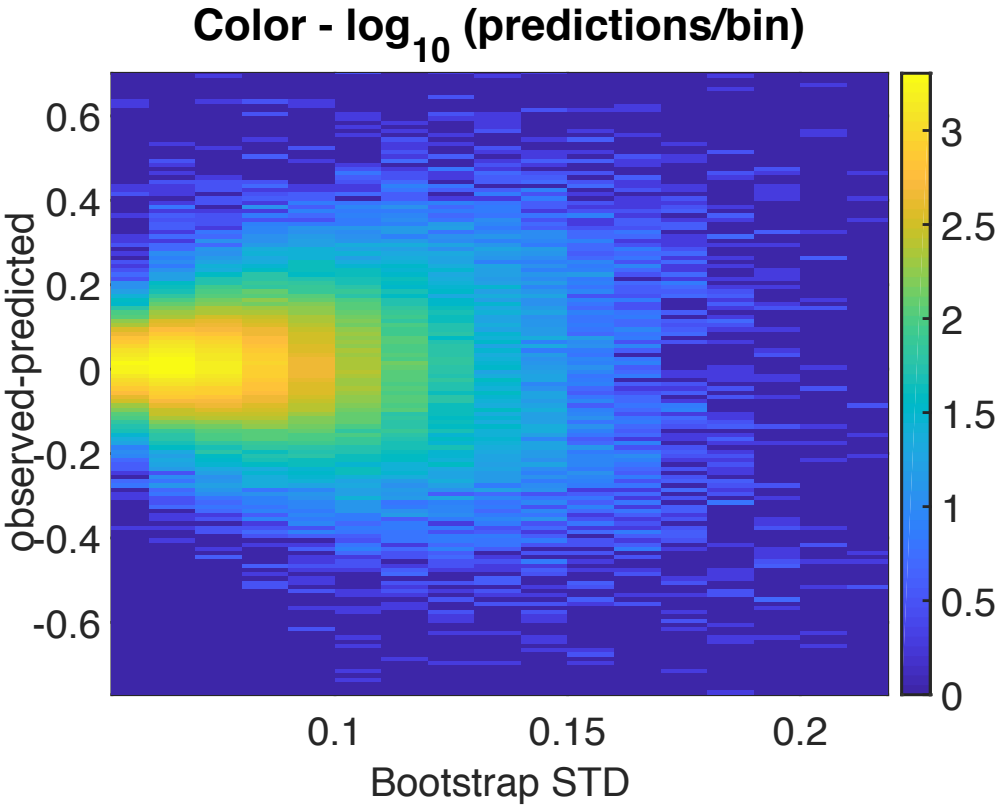


Dropout

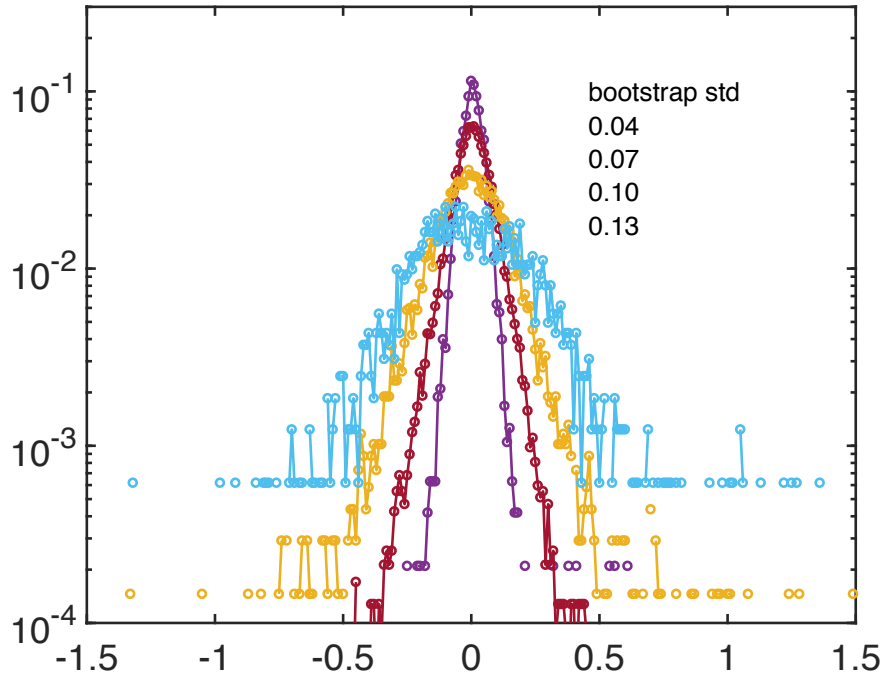
Bootstrap UQ for DL

- We sample, with repetition, N = 100 training samples using the ALMANAC data set.
- For each sample we train a DL network to estimate a function that maps from

  (drug 1, drug 2, CL_GE, concentration) -> measured ALMANAC growth values
- This produces N DL functions that make growth predictions on the test set.
- For each test point we have N predictions from which we estimate the mean and standard deviation; the prediction error is (mean – measured growth)
- We analyze the correlation between the prediction error (accuracy) and the standard deviation (confidence) of the predictive distribution.

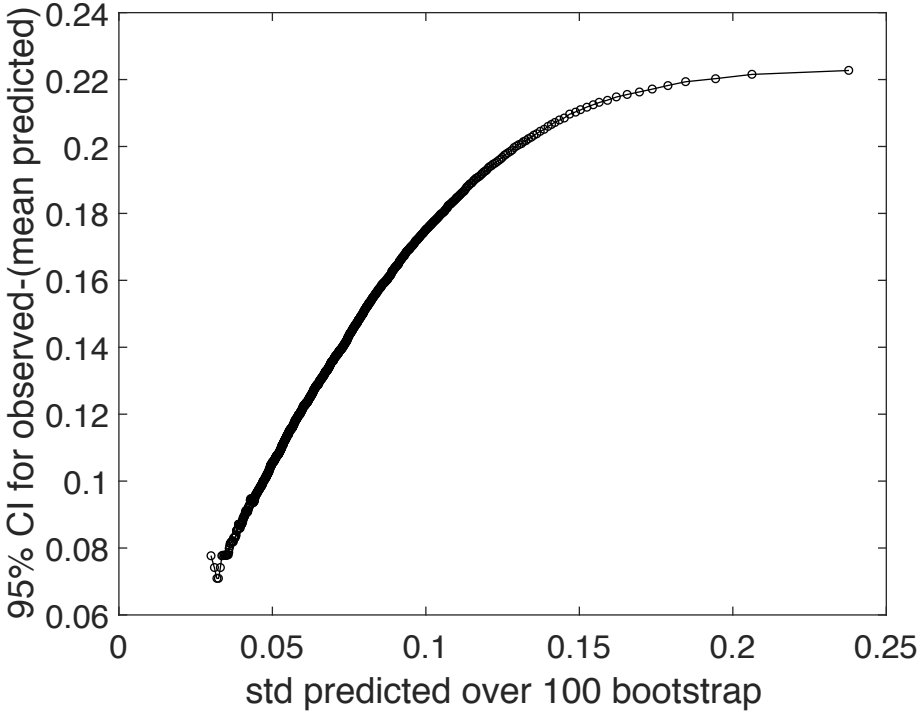


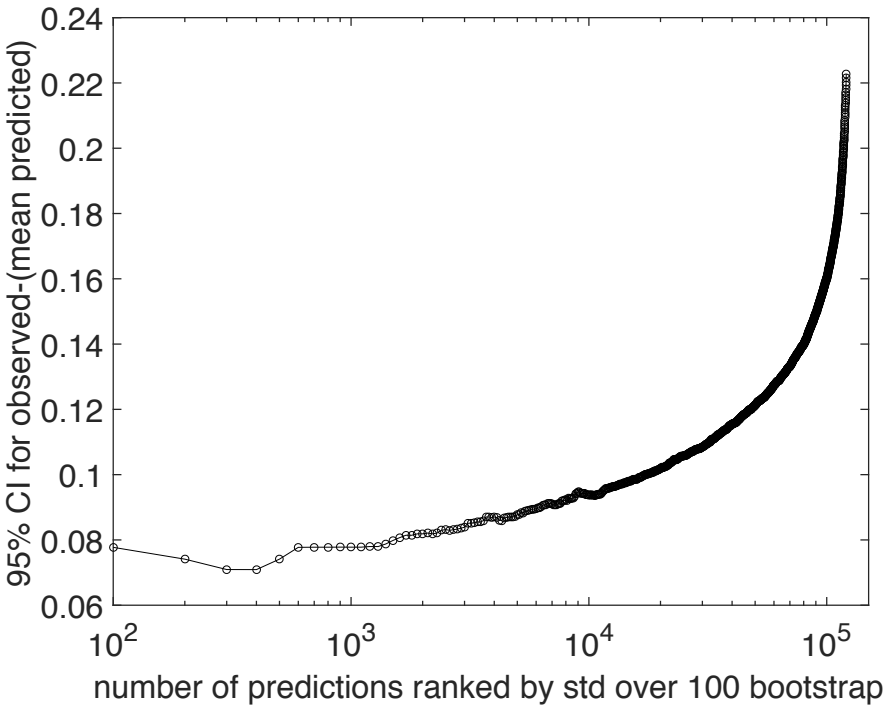
A) Bootstrap histogram in (std, error) space.



B) Probability distribution of prediction errors given bootstrap std; this is the empirical PDF for the columns in A).

Highly confident predictions (small bootstrap std) have high accuracy (small error).





Highly confident predictions (small bootstrap std) have high accuracy (with high confidence the predictions are in a small interval around the true value).

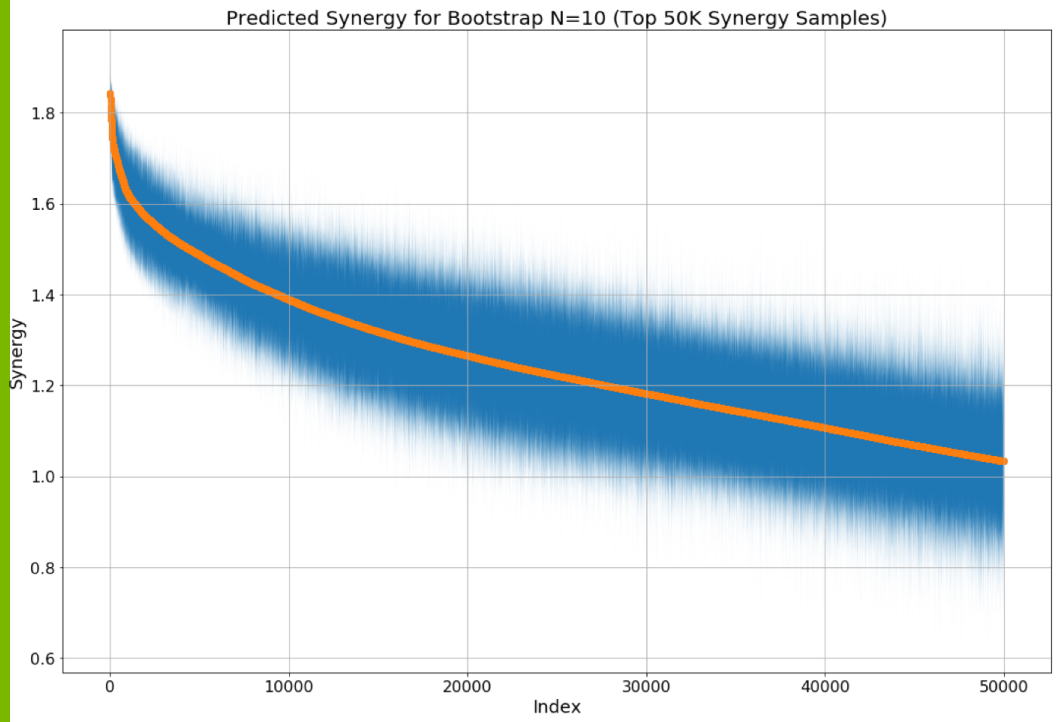
Dropout UQ for DL

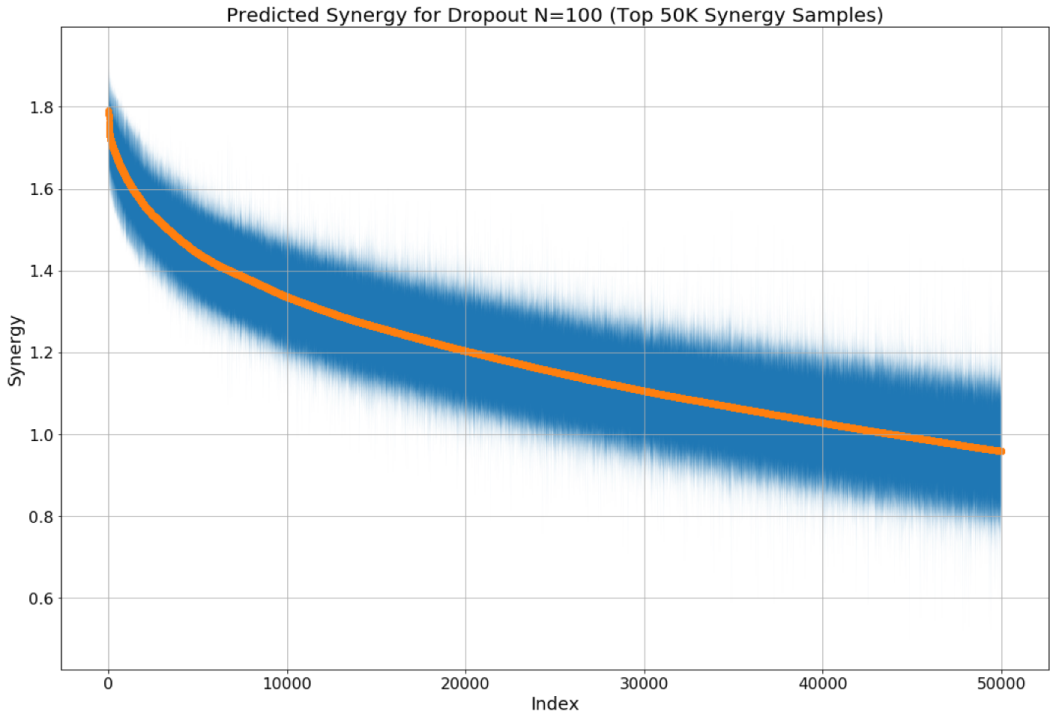
- Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning (Yarin Gal, Zoubin Ghahamani, 2016)
- https://arxiv.org/abs/1506.02142

Comparing DL UQ Variants

- Synergy for combo drugs predicted with deep learning (DL) models.
- Two UQ methods:
  - Bootstrap: bootstrap training data resulting in N models and then doing N inference runs.
  - Dropout: train once with dropout enabled and do inferencing with dropout in place N times.
- GDSC cell lines and Almanac compounds (no ground truth)
- Not all results are available (top 50K synergy samples predicted, top 50K uncertainty samples predicted)

Different methods: dropout vs bootstrap



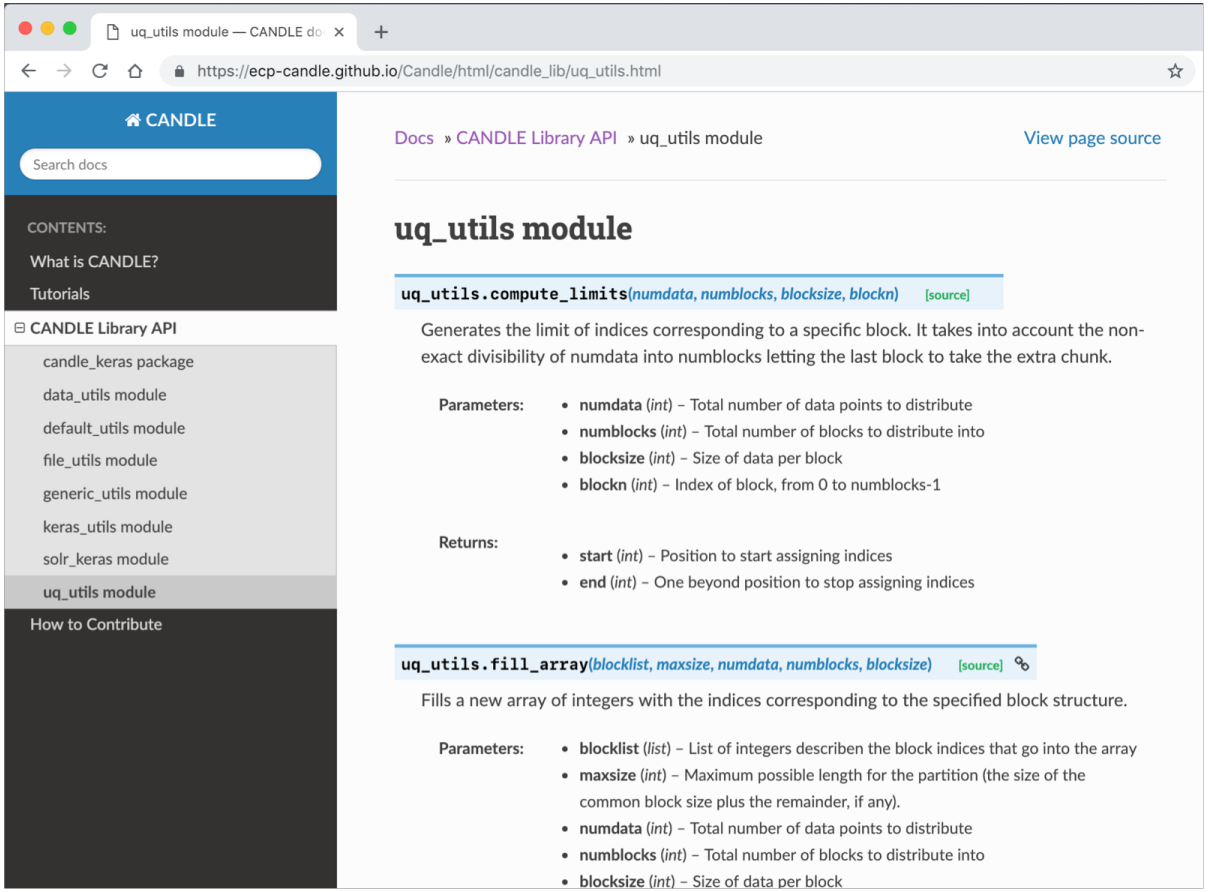


Scaling Inferencing for UQ Measurements of Drug Synergy Predictions

- 680 Drugs in Combination
- 670 Samples
- 30 Replicates
- In each node, 680 * 680 * 30 = 13,872,000

JobID	Node Size	Successful Node	Failed Node	Infers/Node:	Infers/Job	Duration	Infer/Sec/Node	Dropout Rate
X004	670	670	0	13,872,000	9,294,240,000	6:56:08	555.587	0.1
X006	670	670	0	13,872,000	9,294,240,000	6:58:10	553.875	0.2
X007	670	670	0	13,872,000	9,294,240,000	6:56:18	555.364	0.3
X012	670	670	0	13,872,000	9,294,240,000	6:57:44	554.447	0.4
X018	670	670	0	13,872,000	9,294,240,000	6:57:33	553.686	0.5
Totals	3350	3350	0	69,360,000	46,471,200,000	34:45:53	554.5918	

CANDLE UQ Utilities



https://ecp-candle.github.io/Candle/html/candle_lib/uq_utils.html

PermanentDropout

class **keras_utils.PermanentDropout**(*rate*, ***kwargs*) [source]

Bases: `keras.layers.core.Dropout`

call(*x*, *mask=None*) [source]

This is where the layer's logic lives.

Arguments

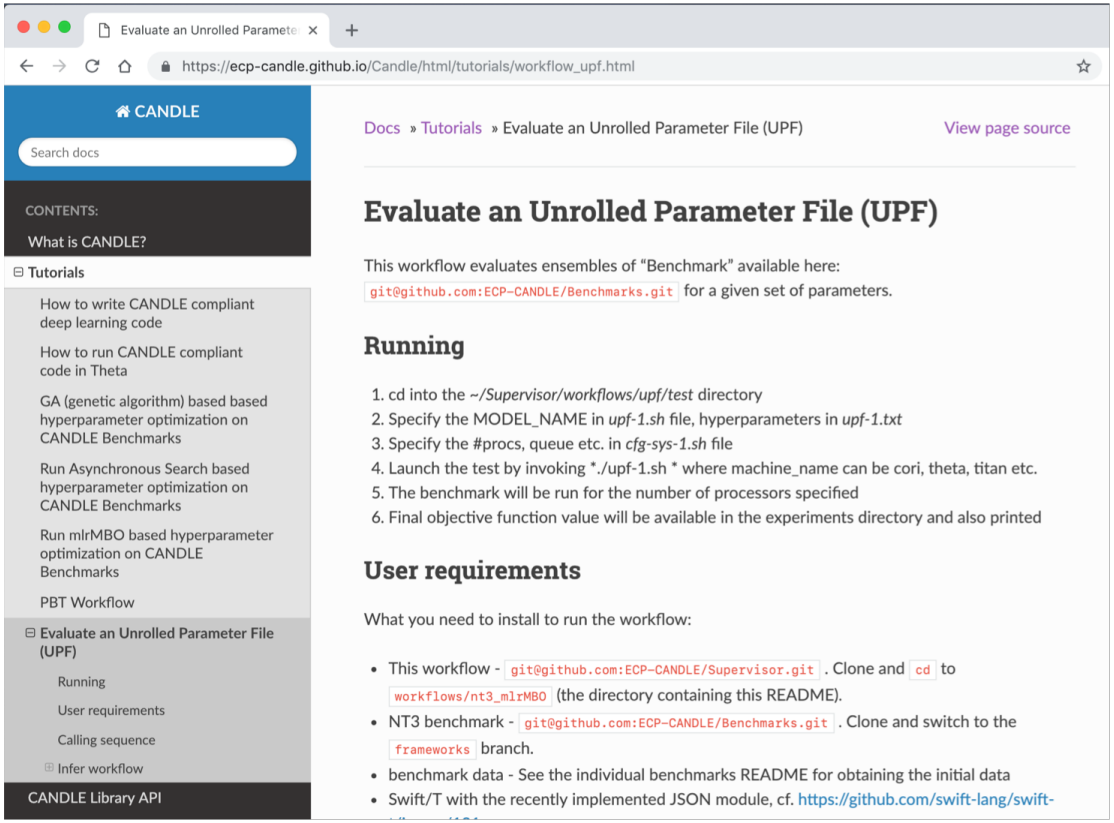
inputs: Input tensor, or list/tuple of input tensors. **kwargs: Additional keyword arguments.

Returns

A tensor or list/tuple of tensors.

https://ecp-candle.github.io/Candle/html/candle_lib/keras_utils.html#keras_utils.PermanentDropout

CANDLE UPF Workflow



https://ecp-candle.github.io/Candle/html/tutorials/workflow_upf.html

Steps for Dropout UQ

- Add Permanent Dropout Layers to DNN
- Add Permanent Dropout Layers to Inference Script
- Modify the JSON Model Representation

- Set Up the UPF Workflow
- Run the Experiment

Demo

- ```
  # training
  ```
- ```
  python mnist_candle_uq.py -e 1 --dropout 0.1 --model_file
  checkpoint.01.model.json
  ```
- ```
  # inference individually
  ```
- ```
  python mnist_infer.py --model_file checkpoint.01.model.json
  ```
- ```
  python mnist_infer.py --model_file checkpoint.02.model.json
  ```
- ```
  # Or use UPF workflow
  ```
- ```
  QUEUE=debug-cache-quad PROJECT=CSC249ADOA01 PROCS=6
  WALLTIME=00:30:00 ./upf-1.sh theta
  ```
  ```

UPF param example

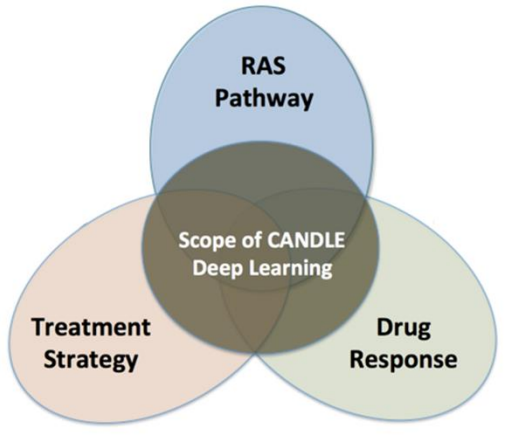
```
{
  "id": "dropout1",
  "model_file": "/gpfs/mira-home/hsyoo/prj/demo/Candle/examples/mnist/ckpt.01.model.json",
  "weights_file": "/gpfs/mira-home/hsyoo/prj/demo/Candle/examples/mnist/ckpt.weights.h5"
}
{
  "id": "dropout2",
  "model_file": "/gpfs/mira-home/hsyoo/prj/demo/Candle/examples/mnist/ckpt.02.model.json",
  "weights_file": "/gpfs/mira-home/hsyoo/prj/demo/Candle/examples/mnist/ckpt.weights.h5"
}
{
  "id": "dropout3",
  "model_file": "/gpfs/mira-home/hsyoo/prj/demo/Candle/examples/mnist/ckpt.03.model.json",
  "weights_file": "/gpfs/mira-home/hsyoo/prj/demo/Candle/examples/mnist/ckpt.weights.h5"
}
{
  "id": "dropout4",
  "model_file": "/gpfs/mira-home/hsyoo/prj/demo/Candle/examples/mnist/ckpt.04.model.json",
  "weights_file": "/gpfs/mira-home/hsyoo/prj/demo/Candle/examples/mnist/ckpt.weights.h5"
}
{
  "id": "dropout5",
  "model_file": "/gpfs/mira-home/hsyoo/prj/demo/Candle/examples/mnist/ckpt.05.model.json",
  "weights_file": "/gpfs/mira-home/hsyoo/prj/demo/Candle/examples/mnist/ckpt.weights.h5"
}
```

THANKS

OPTIMAL DEEP LEARNING ON EXASCALE COMPUTERS

Argonne NATIONAL LABORATORY

HYPERPARAMETER OPTIMIZATION WORKFLOWS WITH CANDLE



JUSTIN M WOZNIAK
Computer Scientist
Data Science & Learning
Argonne National Laboratory

CANDLE Tutorial @ Exascale Computing Project Annual Meeting
January 14, 2019

COLLABORATORS

- CANDLE Infrastructure:
  Tom Brettin, Jon Ozik, Nick Collier, Rajeev Jain (ANL), Harry Yoo (ANL)
  Jamal Mohd-Yusof, Cristina Garcia Cardona (LANL),
  George Zaki (NIH)

- Pilot benchmarks
  Fangfang Xia (ANL), Brian Van Essen (LLNL), Arvind Ramanathan (ORNL)

- PI
  Rick Stevens (ANL)

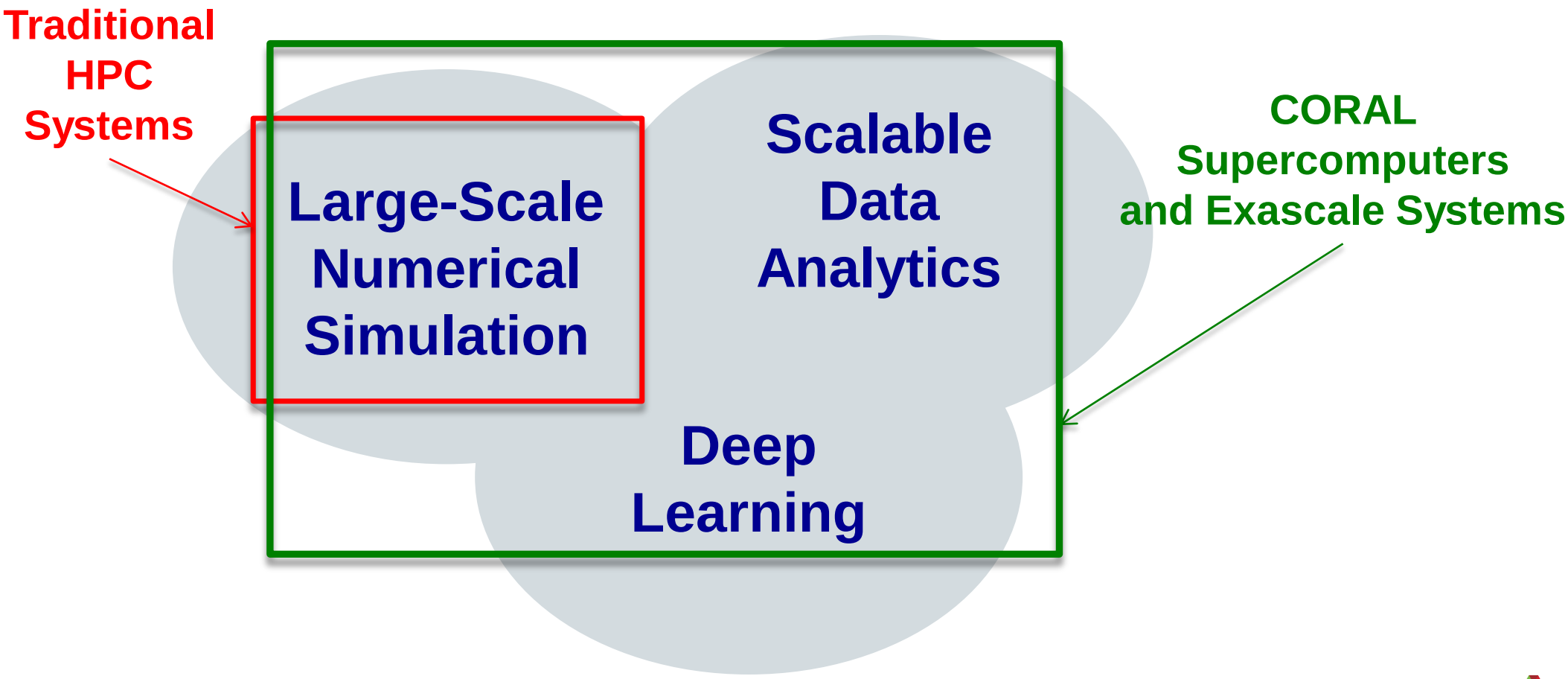
OUTLINE

- Overview of CANDLE project
- Overview of hyperparameter optimization
  - Introduction to hyperparameter optimization
  - Workflow-based solution: EMEWS
- Demo of Swift/T and CANDLE/Supervisor

CANDLE OVERVIEW

Argonne NATIONAL LABORATORY

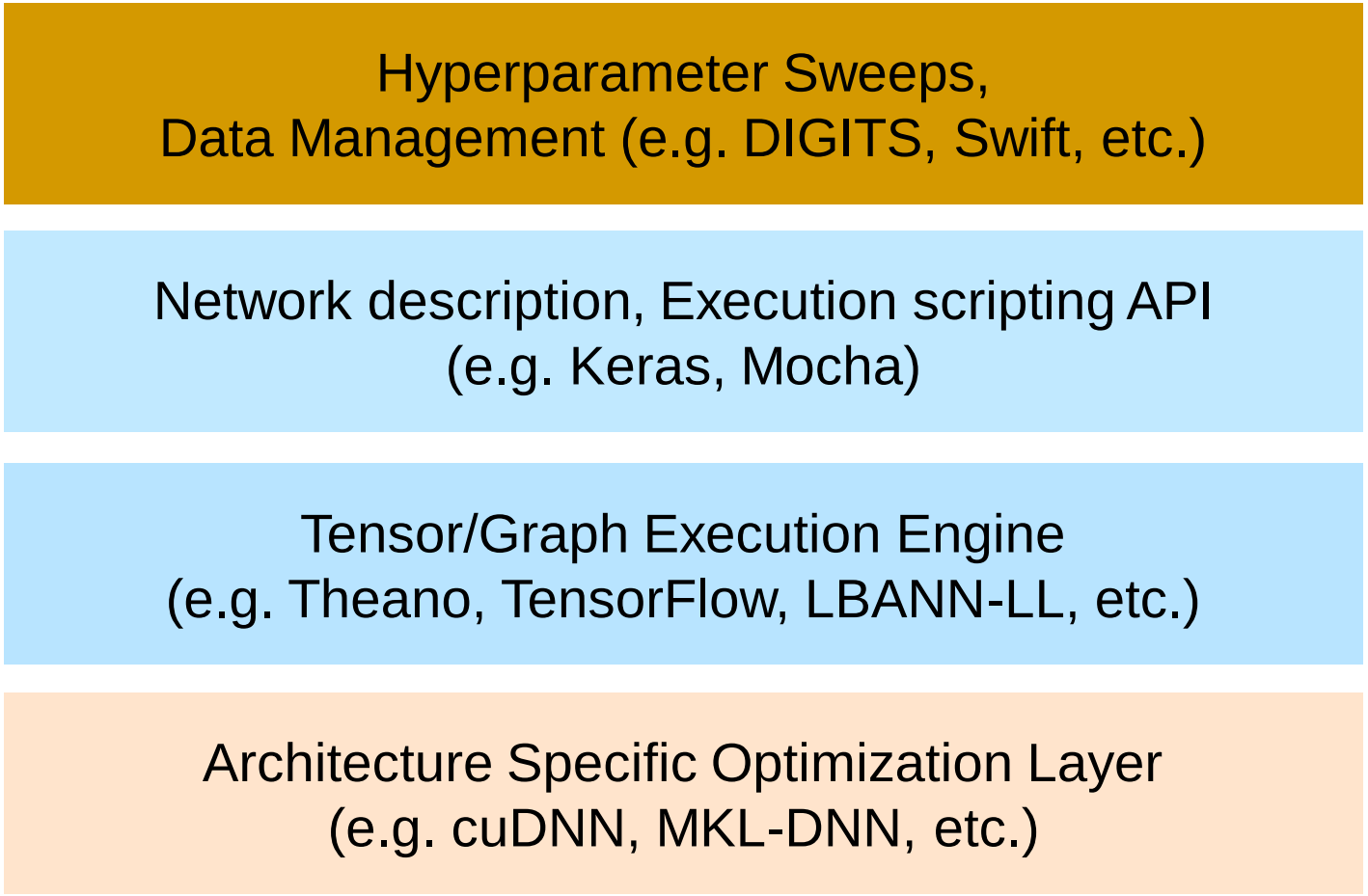
DRIVING INTEGRATION OF SIMULATION, DATA ANALYTICS AND MACHINE LEARNING



CANDLE WORKFLOWS: GOALS

- Develop an exascale deep learning environment for cancer
- Building on open source deep learning frameworks and middleware
- Optimization for CORAL and exascale platforms
- Support all three pilot project needs for deep learning – common abstractions
- Collaborate with DOE computing centers, HPC vendors and ECP co-design and software technology projects
- Mission statement: Enable the most challenging deep learning problems in cancer research to run on the most capable supercomputers in the DOE

CANDLE SOFTWARE STACK



Workflow

Scripting

Engine

Optimization

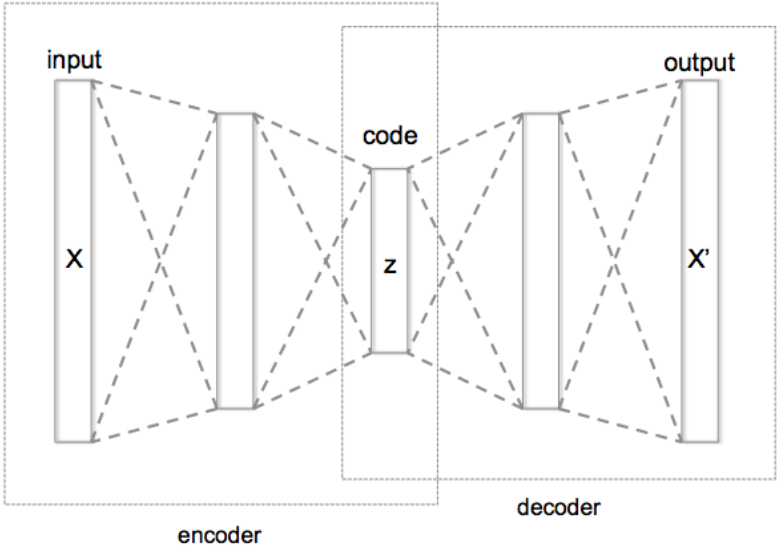
HYPERPARAMETER OPTIMIZATION

WHAT IS HYPERPARAMETER OPTIMIZATION

Hyperparameter optimization = HPO

- Neural networks have a large number of possible configuration parameters, called *hyperparameters*
  - Avoids collision with NN *weights*, which are sometimes called *parameters*
- Applying optimization can automate part of the design of the neural network

- In the cancer Pilot 1 autoencoder shown, the system can determine
  - How many neurons to put in each layer
  - What activation function to use
  - What batch size to use
  - Etc.



MATHEMATICAL EXPRESSION FOR HPO

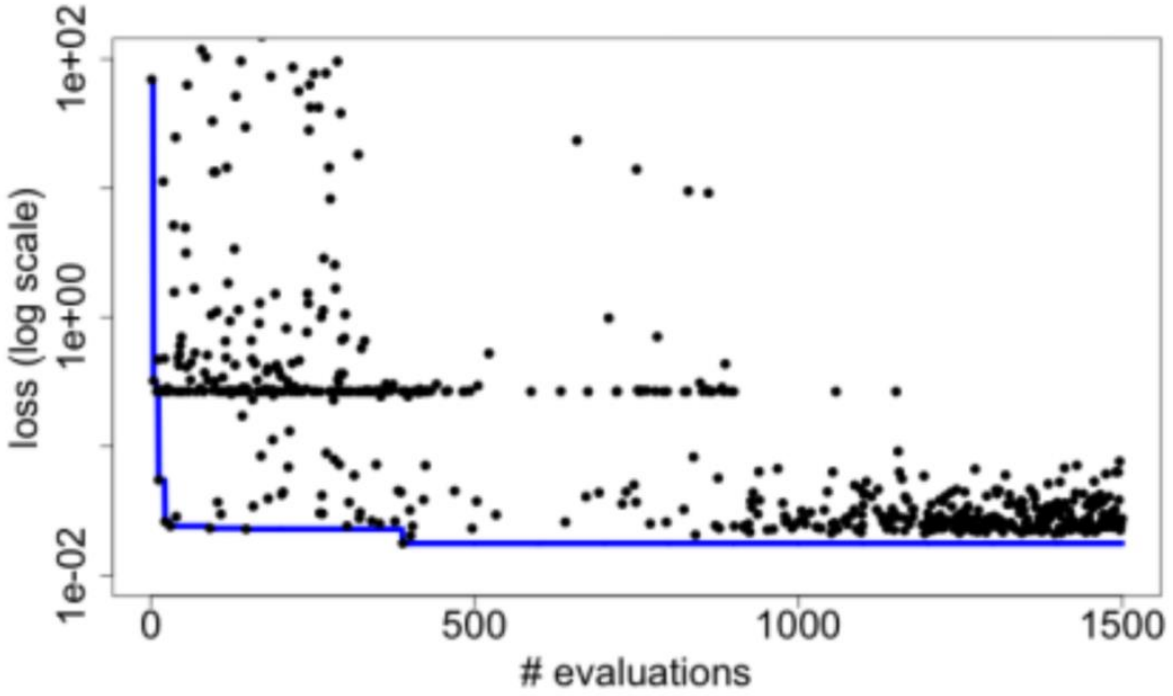
- For a given problem:
  - A loss function ***F*** is determined on a given NN (usually accuracy)
  - The hyperparameter optimization problem is to minimize ***F(p)***,
    - for all hyperparameter sets ***p*** in the valid parameter space ***P***,
    - however, ***P*** is large and ***F*** is expensive.
    - ***P*** is the cross product of all valid network settings,
      - some of which may be categorical, some integer, some continuous.
    - Evaluating ***F*** involves training the network on a training data set and applying it to the validation set
- We can use a generic, previously developed method to optimize ***F*** !
- These methods require and can use large compute resources

BASIC STRATEGIES

- Grid search
- Random search
- Generic optimization
  - Stochastic gradient descent
  - Evolutionary algorithms
  - Model-based optimization (mlrMBO in R)
- NN hyperparameter-specific optimization
  - Hyperopt, NEAT, Optunity, …

CANDLE HYPERPARAMETER LEARNING

- Search trajectory of mlrMBO (R model-based optimization) algorithm
- Each iteration does 300 evaluations (batch size)
- Minimum and average performance on validation data set decreases as the ME algorithm learns

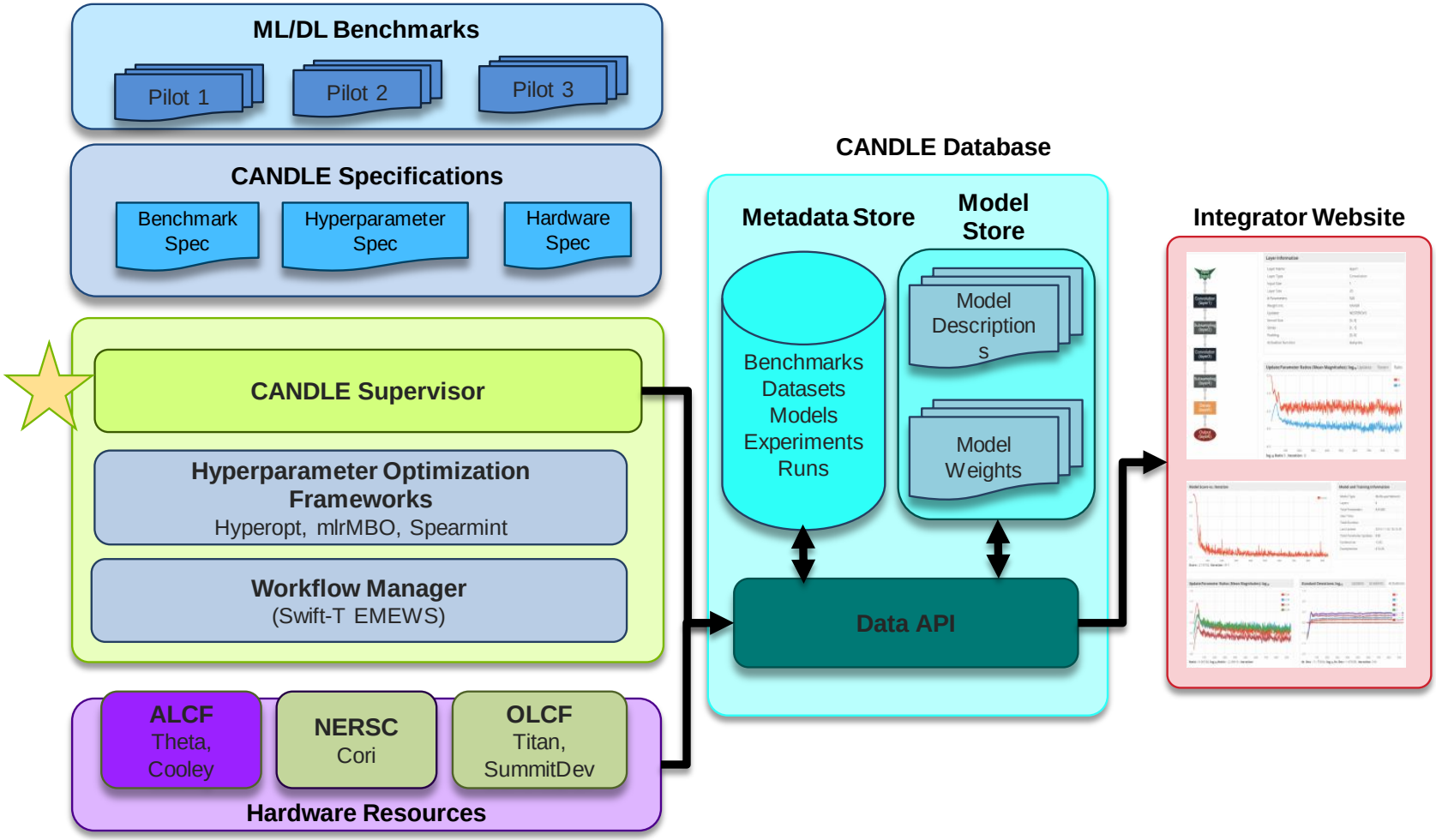


Predicting Tumor Cell Line Response to Drug Pairs with Deep Learning, F. Xia, M. Shukla, T. Brettin, C. Garcia-Cardona, J. Cohn, J. Allen, S. Maslov, Y. Evrard, S. Holbeck, J. Doroshow, E. Stahlberg, and R. Stevens (Computational Approaches for Cancer Workshop @ SC 2017)

CANDLE: WORKFLOWS

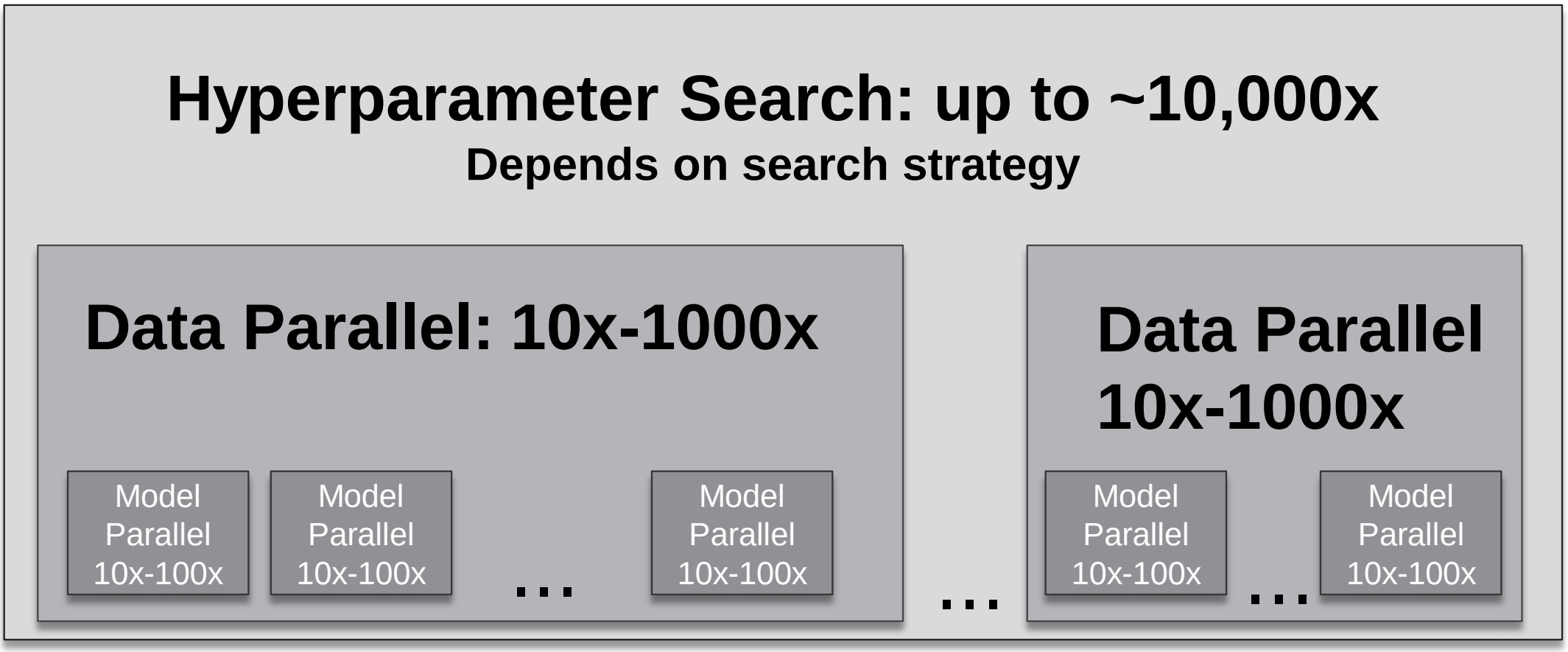
Argonne NATIONAL LABORATORY

CANDLE SYSTEM OVERVIEW

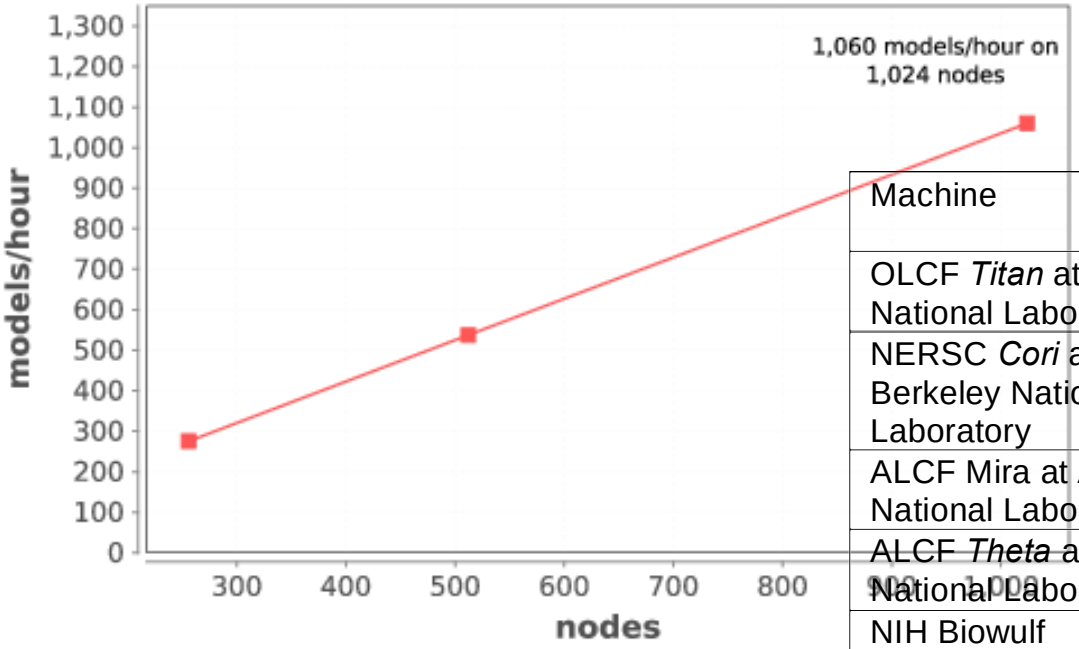


PARALLELISM STRATEGIES

10,000 x 10-1000 x 10-100 = 1M – 1000M processing elements



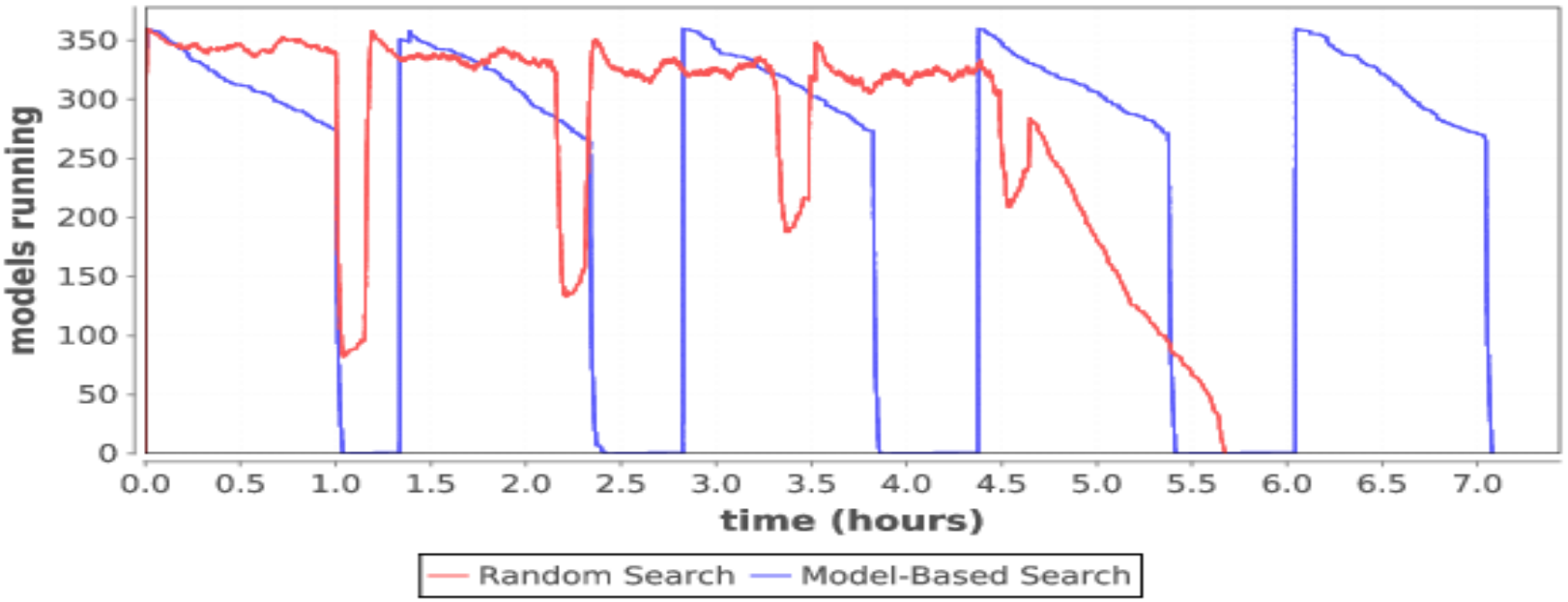
CANDLE PERFORMANCE



- Delivers 1+ petaflop!

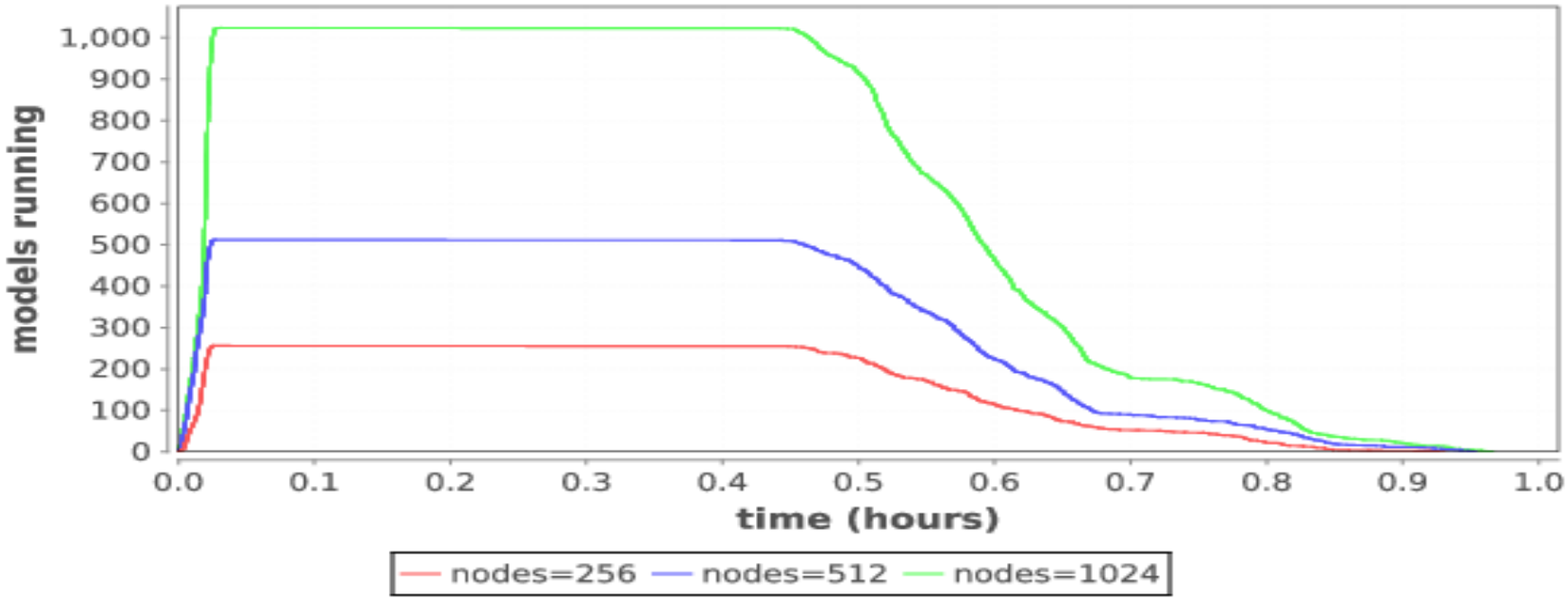
Machine	Architecture (Nov 2017 Top 500 ranking)	Scheduler
OLCF *Titan* at Oak Ridge National Laboratory	Cray XK7, AMD Opteron / K20X (5)	PBS
NERSC *Cori* at Lawrence Berkeley National Laboratory	Cray XC40, Intel Xeon Phi / E5 (8)	SLURM
ALCF Mira at Argonne National Laboratory	BlueGene/Q, PowerPC A2 (11)	Cobalt
ALCF *Theta* at Argonne National Laboratory	Cray XC40, Intel Xeon Phi (18)	Cobalt
NIH Biowulf	HP Apollo XL1x0r, Intel Xeon E5 (66)	SLURM
Blues, Bebop at Argonne National Laboratory	Intel Xeon Phi / E5 (148)	PBS/SLURM
Beagle2 at University of Chicago	Cray XE6, AMD Operton (NA)	PBS
Midway, Midway2 at University of Chicago	Intel various / AMD Opteron (NA)	SLURM

LOAD OVER TIME FOR SEARCH



- Typical load plot for NT3 workflow on Cori

RAMP UP / RAMP DOWN



- Zoom in on single iteration on Titan

SYSTEMS CHALLENGES

WORKFLOW SUPPORT FOR ML FRAMEWORKS

- Concurrency:
  - Scalable task distributor
  - Intranode concurrency, accelerators left up to the framework
  - Multinode ML tasks are future work (already basically supported)
- Data management:
  - Input staging methods have been developed
  - Intermediate caches via DataSpaces
- Software integration:
  - Usually launch frameworks in separate process
  - Launching within process is a configuration challenge
  - Search methods launched within process

WORKFLOW GOALS

Hierarchical, naturally parallel, script-like programming

- Make it easy to run large batteries of external program or library executions
- Provide rich programming language at the top level – fully generic
- Support implicit concurrency and conventional programming constructs
- Enable complex tasks based in other scripting languages (e.g., Python) or parallel MPI tasks

THE SWIFT PROGRAMMING MODEL

All progress driven by concurrent dataflow

```
(int r) myproc (int i, int j)
{
    int x = F(i);
    int y = G(j);
    r = x + y;
}
```

- `F()` and `G()` implemented in native code or external programs
- `F()` and `G()` run in concurrently in different processes
- `r` is computed when they are both done
- This parallelism is *automatic*
- Works recursively throughout the program's call graph

SWIFT SYNTAX

- Data types

```
int i = 4;
string s = "hello world";
file image<"snapshot.jpg">;
```

- Shell access

```
app (file o) myapp(file f, int i)
{ mysim  "-s" i @f @o; }
```

- Structured data

```
typedef image file;
image A[];
type protein_run {
  file pdb_in; file sim_out;
}
bag<blob>[] B;
```

- Conventional expressions

```
if (x == 3) {
    y = x+2;
    s = strcat("y: ", y);
}
```

- Parallel loops

```
foreach f,i in A {
    B[i] = convert(A[i]);
}
```

- Data flow

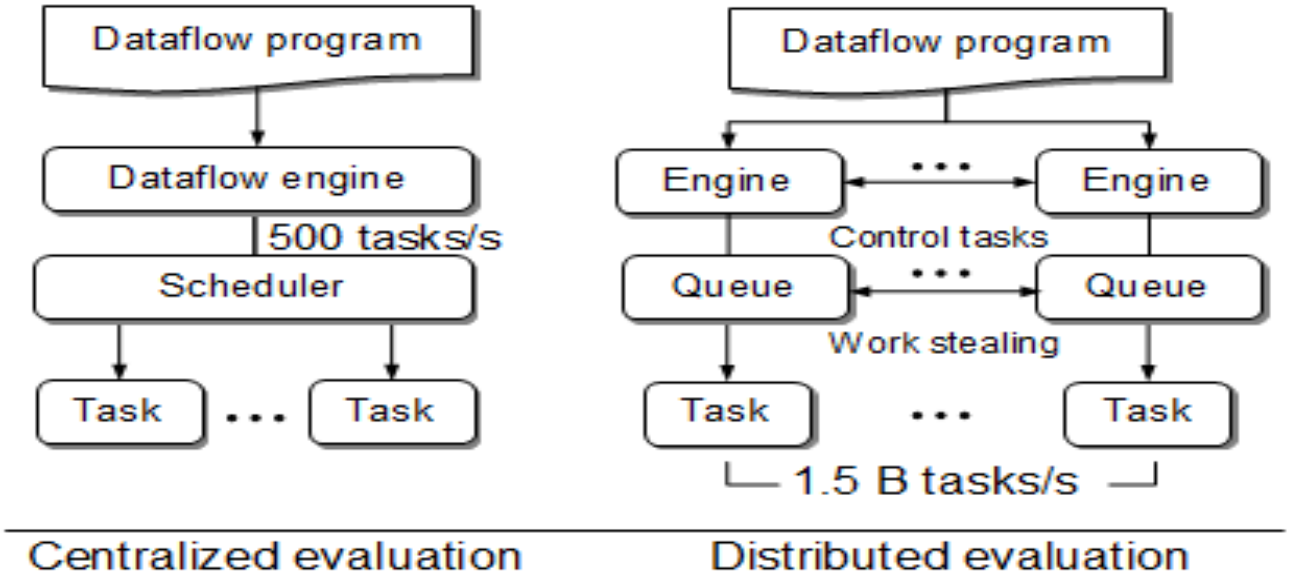
```
merge(analyze(B[0], B[1]),
      analyze(B[2], B[3]));
```

- **Swift: A language for distributed parallel scripting.** J. Parallel Computing 2011
- **Compiler techniques for massively scalable implicit task parallelism.** Proc. SC 2014

CENTRALIZED EVALUATION IS A BOTTLENECK AT EXTREME SCALES

Had this (Swift/K):

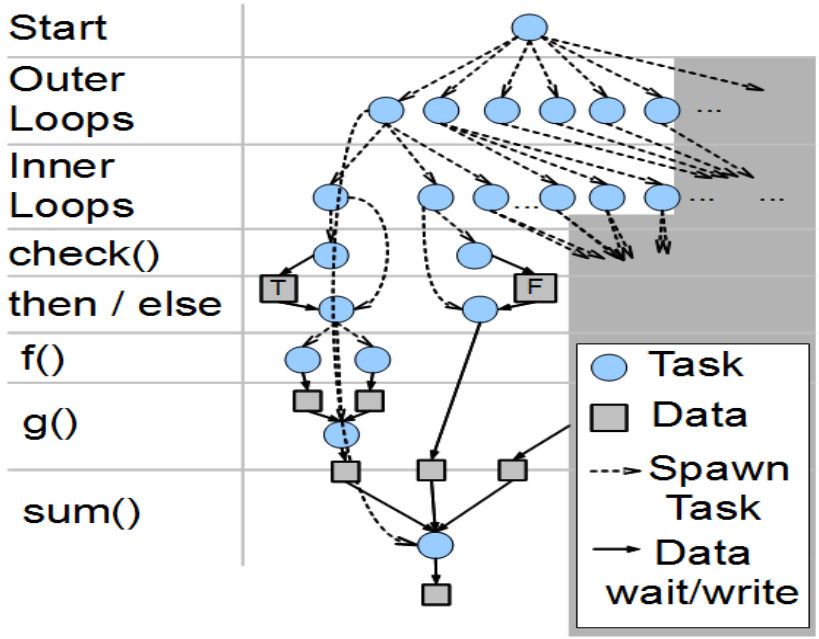
Now have this (Swift/T):



- **Turbine: A distributed-memory dataflow engine for high performance many-task applications.** Fundamenta Informaticae 28(3), 2013

SWIFT/T: FULLY PARALLEL EVALUATION OF COMPLEX SCRIPTS

```
int X = 100, Y = 100;
int A[][];
int B[];
foreach x in [0:X-1] {
  foreach y in [0:Y-1] {
    if (check(x, y)) {
      A[x][y] = g(f(x), f(y));
    } else {
      A[x][y] = 0;
    }
  }
  B[x] = sum(A[x]);
}
```



- **Swift/T: Scalable data flow programming for distributed-memory task-parallel applications .** Proc. CCGrid, 2013.

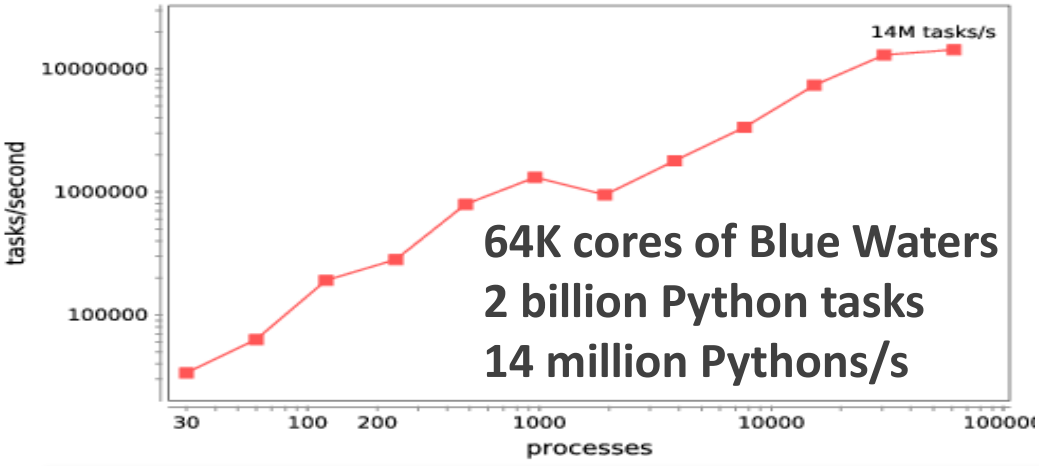
SWIFT/T: ENABLING HIGH-PERFORMANCE SCRIPTED WORKFLOWS

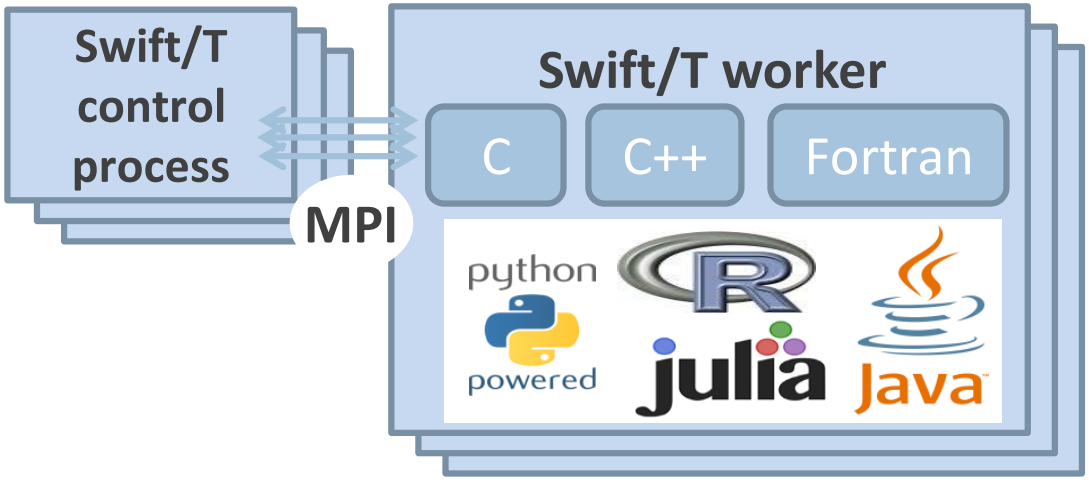
Supports tasks written in many languages

- Write site-independent scripts, translates to MPI
- Automatic task parallelization and data movement
- Invoke native code, script fragments
- Rapidly subdivide large partitions for MPI jobs in multiple ways

SWIFT/T

2018 R&D 100 WINNER






```
$ conda install -c lightsource2-tag swift-t
```

Compiler techniques for massively scalable implicit task parallelism. Proc. SC 2014.

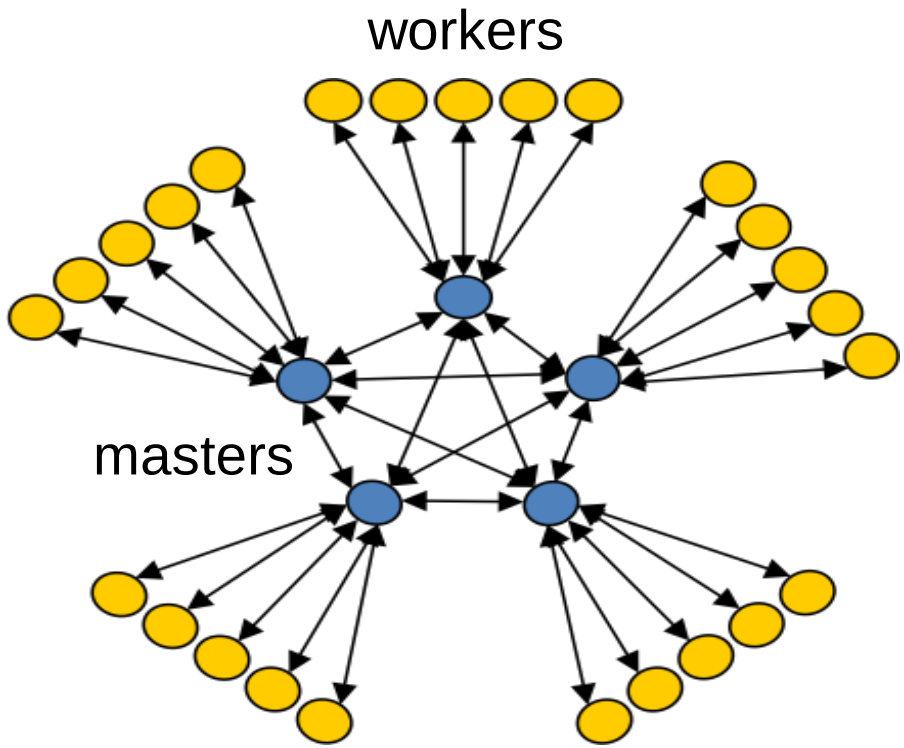
Argonne NATIONAL LABORATORY

ASYNCHRONOUS DYNAMIC LOAD BALANCER

ADLB for short

- An MPI library for master-worker workloads in C
- Uses a variable-size, scalable network of servers
- Servers implement work-stealing
- The work unit is a byte array
- Optional work priorities, targets, types

- For Swift/T, we added:
  - Server-stored data
  - Data-dependent execution
  - Parallel tasks

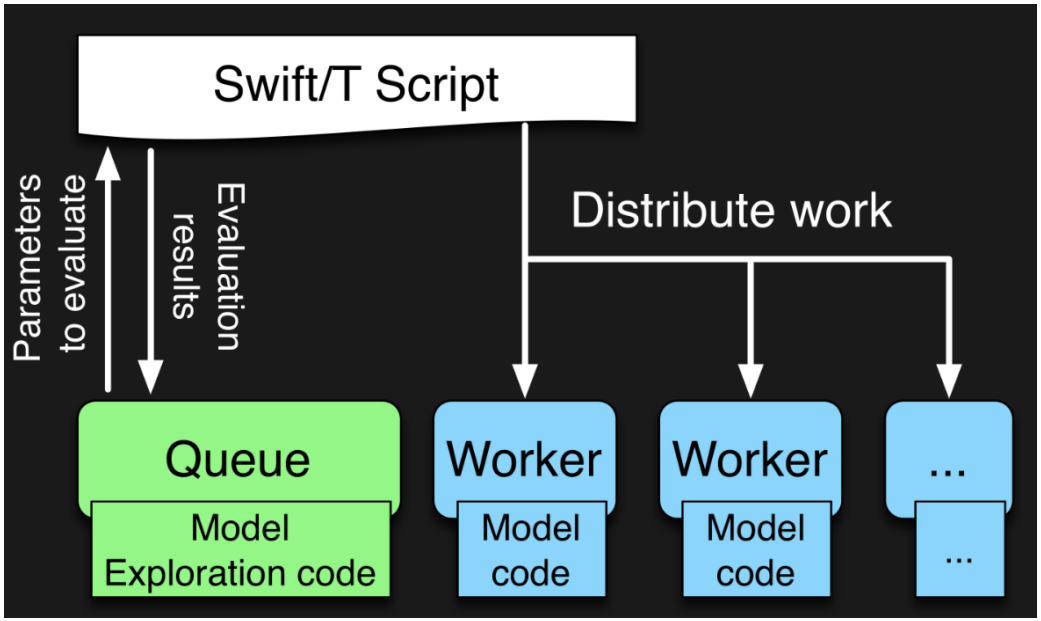


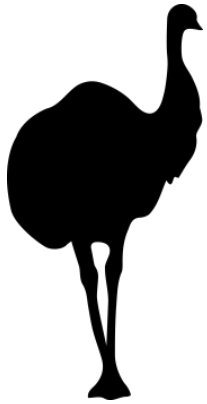
- Lusk et al. **More scalability, less pain: A simple programming model and its implementation for extreme computing.** SciDAC Review 17, 2010

EXTREME-SCALE MODEL EXPLORATION WITH SWIFT (EMEWS)

Argonne NATIONAL LABORATORY

EMEWS WORKFLOW STRUCTURE



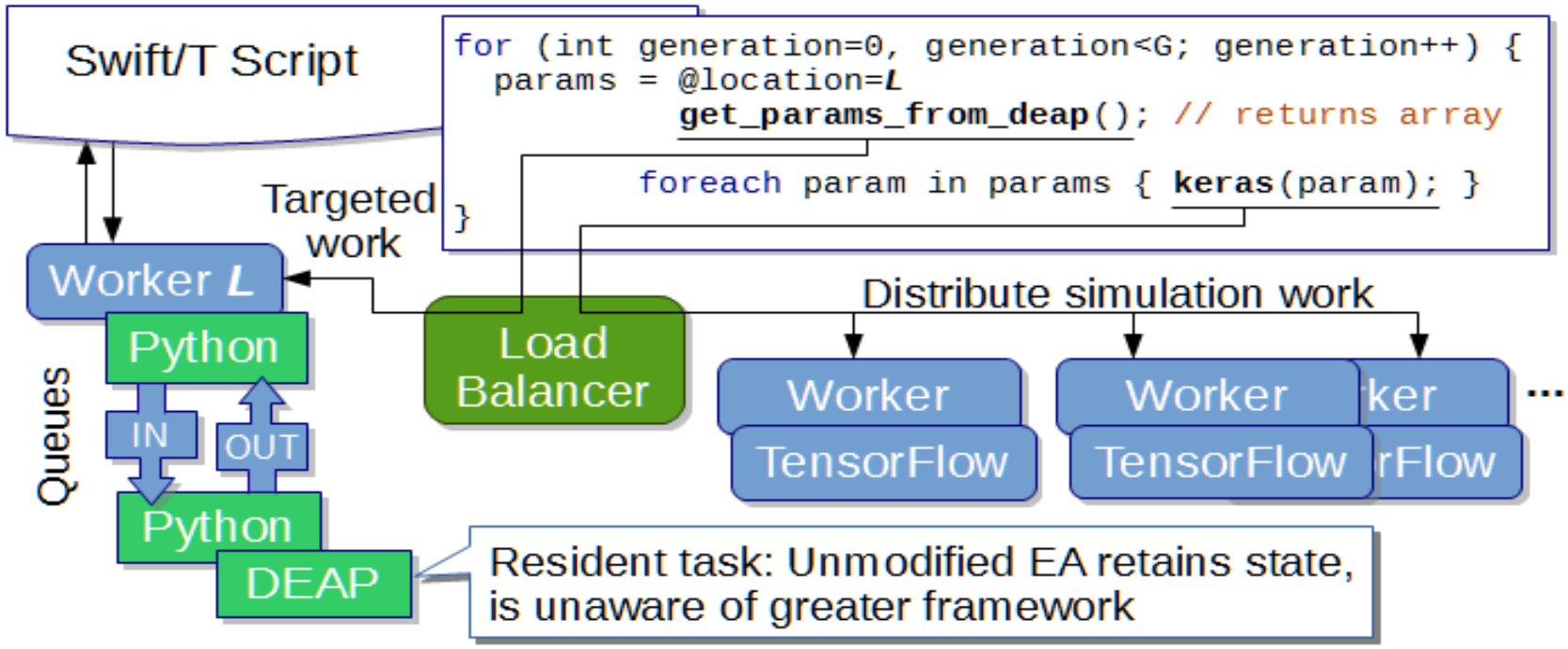


emews.org

- The core novel contributions of EMEWS are shown in green, these allow the Swift script to access a running **Model Exploration (ME)** algorithm, and create an **inversion of control (IoC)** workflow
- Both green and blue boxes accept **existing multi-language code**

EMEWS: EXTREME-SCALE MODEL EXPLORATION WORKFLOWS IN SWIFT/T

- To query the state of the EA, we designate one worker on location L for exclusive use by DEAP. Other optimizers can easily be used (e.g., mlrMBO in CANDLE)



PREVIOUS WORK ON HPC WORKFLOWS

Other uses of workflows to control model exploration (ME) typically take one of two approaches

1. They provide rich support for arithmetic operations so that ME algorithms can be constructed (ported)
   - requires that algorithm be **coded from scratch**
   - **impossible to reuse code** in other languages
2. The ME algorithm is provided as a built-in feature of the system
   - does not allow the end users much **control over the algorithm** used
   - may require **access to workflow system source code** in order to incorporate external ME algorithms or to modify built-in algorithms

In both cases, the many libraries being actively developed and implemented as free and open source software in programming languages such as R and Python **cannot be directly/easily utilized**.

SUMMARY OF KEY SYSTEM POINTS

- What about Swift/T enables CANDLE?
  - A workflow system that is actually a hierarchical programming language
  - Runs entirely on the compute nodes
  - Uses standard APIs for HPC (MPI), allows for minimal OS environment
  - Very scalable
  - Supports MPI tasks, embedded Python, R interpreters
- What about EMEWS enables CANDLE?
  - Allows user to focus on two sequential codes
    - The optimizer
    - Their objective function code
  - Everything else is managed by the system

THANKS

- Thanks to the organizers

- Code and guides:
  - CANDLE GitHub Organization: https://github.com/ECP-CANDLE
  - Swift/T Home: http://swift-lang.org/Swift-T
  - EMEWS Tutorial: http://emews.org

- This research was supported by the Exascale Computing Project (17-SC-20-SC), a joint project of the U.S. Department of Energy's Office of Science and National Nuclear Security Administration, responsible for delivering a capable exascale ecosystem, including software, applications, and hardware technology, to support the nation's exascale computing imperative.

TUTORIAL: SUPERVISOR

Argonne NATIONAL LABORATORY

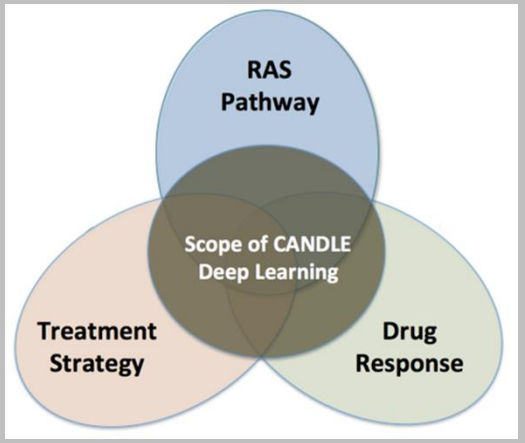
HANDS-ON TUTORIALS

- May be found here:
  - `git@github.com:brettin/candle_tutorials.git`
  - Subdirectory `Topics/2_hyperparameter_optimization`
- See the top-level README to get started with the installation

OPTIMAL DEEP LEARNING ON EXASCALE COMPUTERS

Argonne NATIONAL LABORATORY

DATA PARALLEL TRAINING WITH CANDLE



JUSTIN M WOZNIAK
Computer Scientist
Data Science & Learning
Argonne National Laboratory

CANDLE Tutorial @ Exascale Computing Project Annual Meeting
January 14, 2019

COLLABORATORS

- CANDLE Infrastructure:
  Tom Brettin, Jon Ozik, Nick Collier, Rajeev Jain (ANL), Harry Yoo (ANL)
  Jamal Mohd-Yusof, Cristina Garcia Cardona (LANL)
  George Zaki (NIH)

- Pilot benchmarks
  Fangfang Xia (ANL), Brian Van Essen (LLNL), Arvind Ramanathan (ORNL)

- PI
  Rick Stevens (ANL)

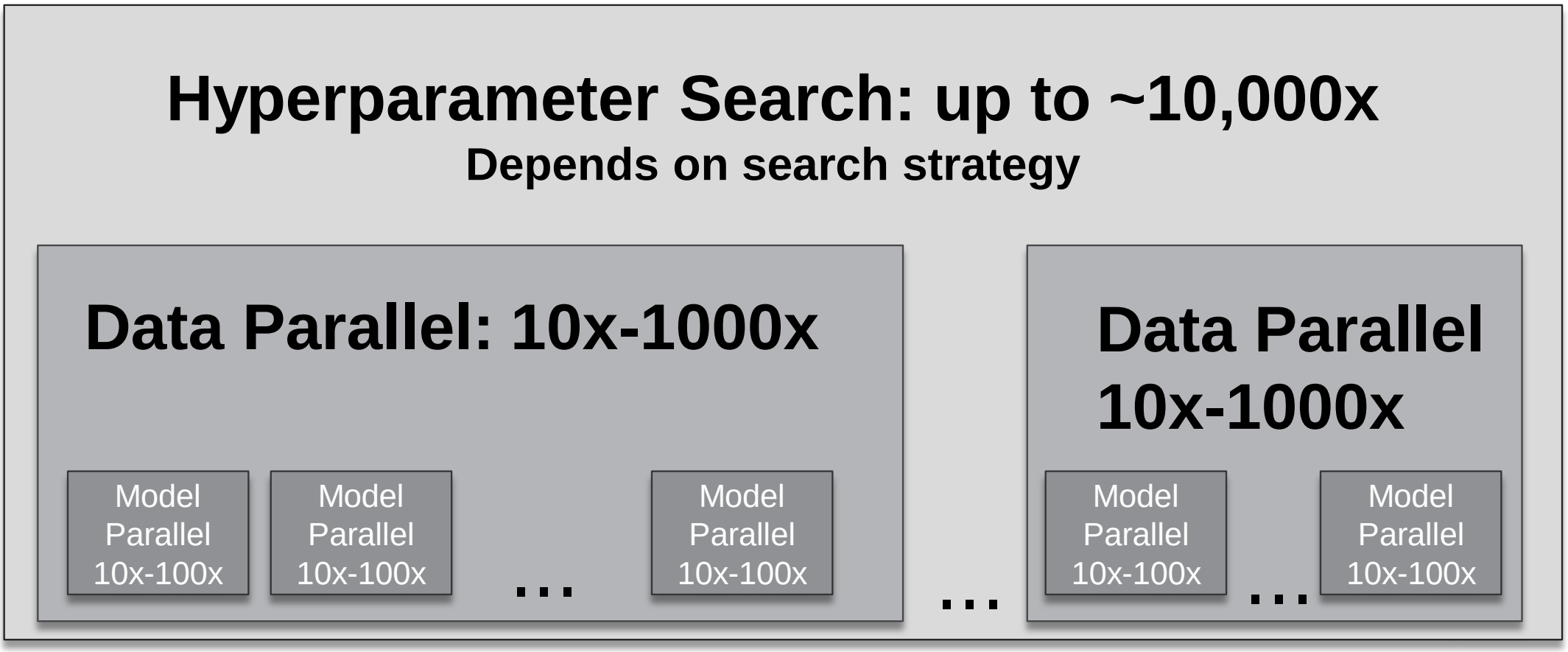
OUTLINE

- Introduction to parallel tasks in workflows
- Data parallel training in CANDLE
  - Horovod
  - Performance analysis
- Big data management
  - Interaction with ECP CODAR

PARALLEL TASK SUPPORT IN WORKFLOWS

PARALLELISM STRATEGIES

10,000 x 10-1000 x 10-100 = 1M – 1000M processing elements

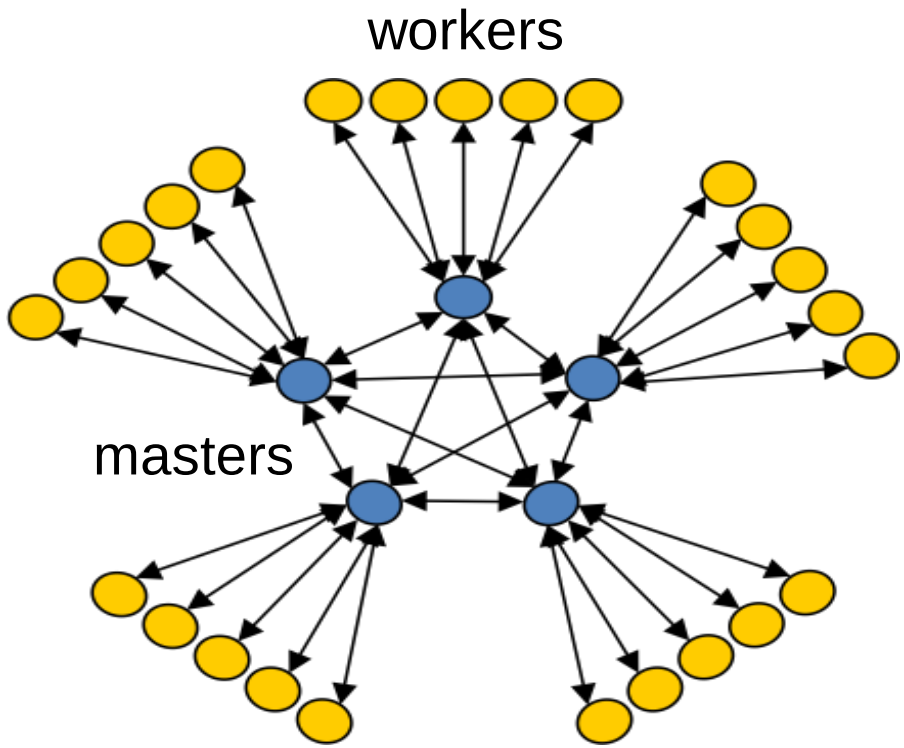


ASYNCHRONOUS DYNAMIC LOAD BALANCER

ADLB for short

- An MPI library for master-worker workloads in C
- Uses a variable-size, scalable network of servers
- Servers implement work-stealing
- The work unit is a byte array
- Optional work priorities, targets, types

- For Swift/T, we added:
  - Server-stored data
  - Data-dependent execution
  - Parallel tasks



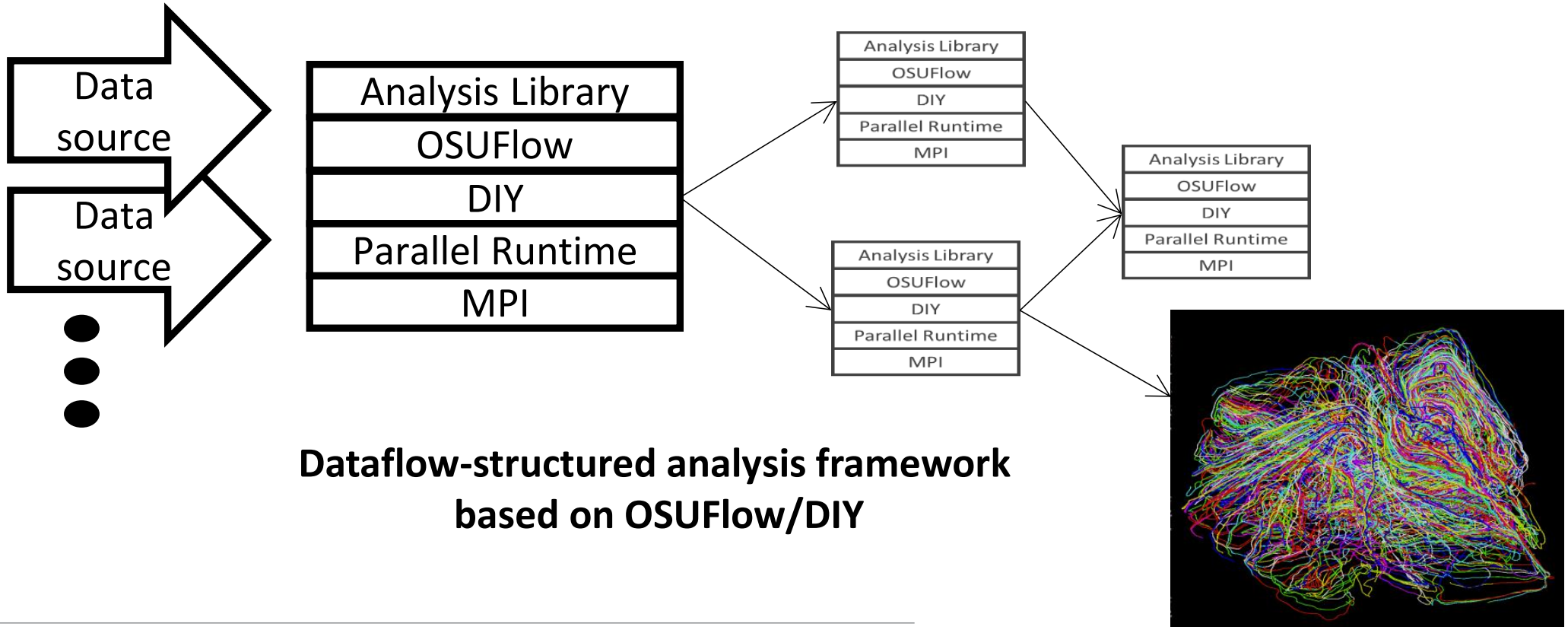
- Lusk et al. **More scalability, less pain: A simple programming model and its implementation for extreme computing.** SciDAC Review 17, 2010

PARALLEL TASKS IN CANDLE WORKFLOWS

Complex concurrency structures

- Training the same network across nodes
- Library approach:
  - Use Swift/T @par syntax
  - Uses MPI 3 to dynamically create communicator from group
  - User task library accepts communicator via function input
  - Approach developed for other scientific computing cases, LAMMPS, NAMD, DIY, etc.
- MPI_Launch approach
  - Use Swift/T launch() function
  - Creates MPI 3 group
  - Launches mpiexec on those resources, creating a new MPI_COMM_WORLD and separate processes (fault tolerance)
  - Works on clusters
  - Has initial vendor support

DATAFLOW+DATA-PARALLEL ANALYSIS/VISUALIZATION

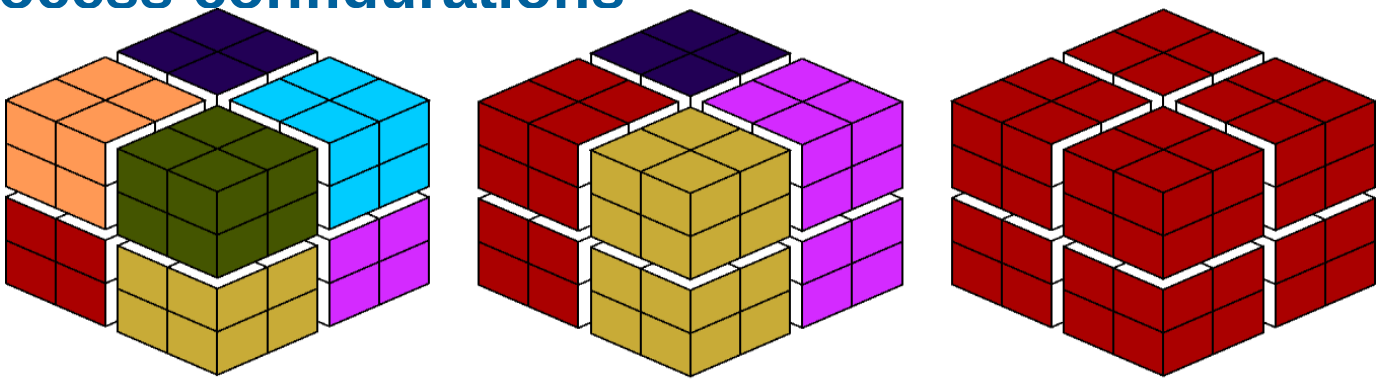


Dataflow-structured analysis framework based on OSUFlow/DIY

- **Dataflow coordination of data-parallel tasks via MPI 3.0**
  Proc. EuroMPI, 2013

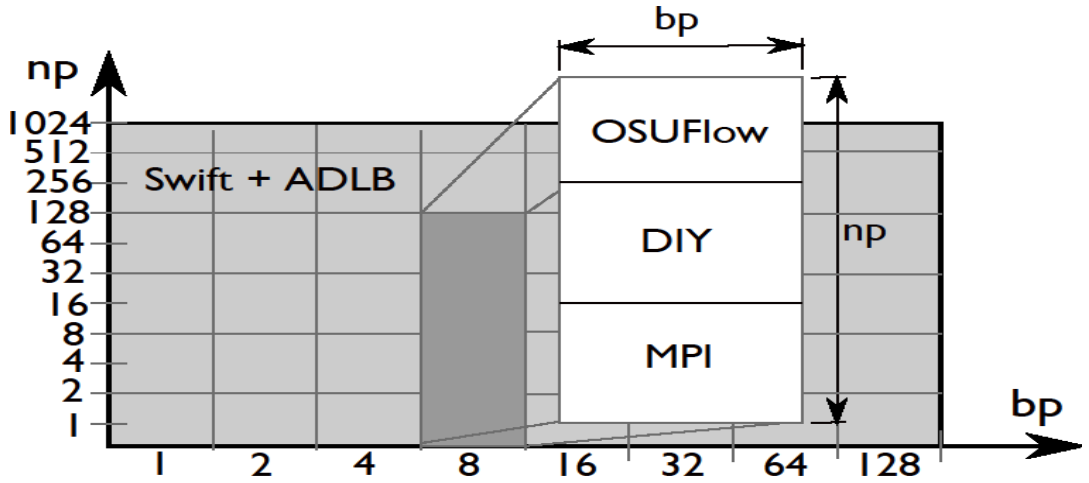
PARAMETER OPTIMIZATION FOR DATA-PARALLEL ANALYSIS

Process configurations



8 processes
1 block per process

4 processes
2 blocks per process

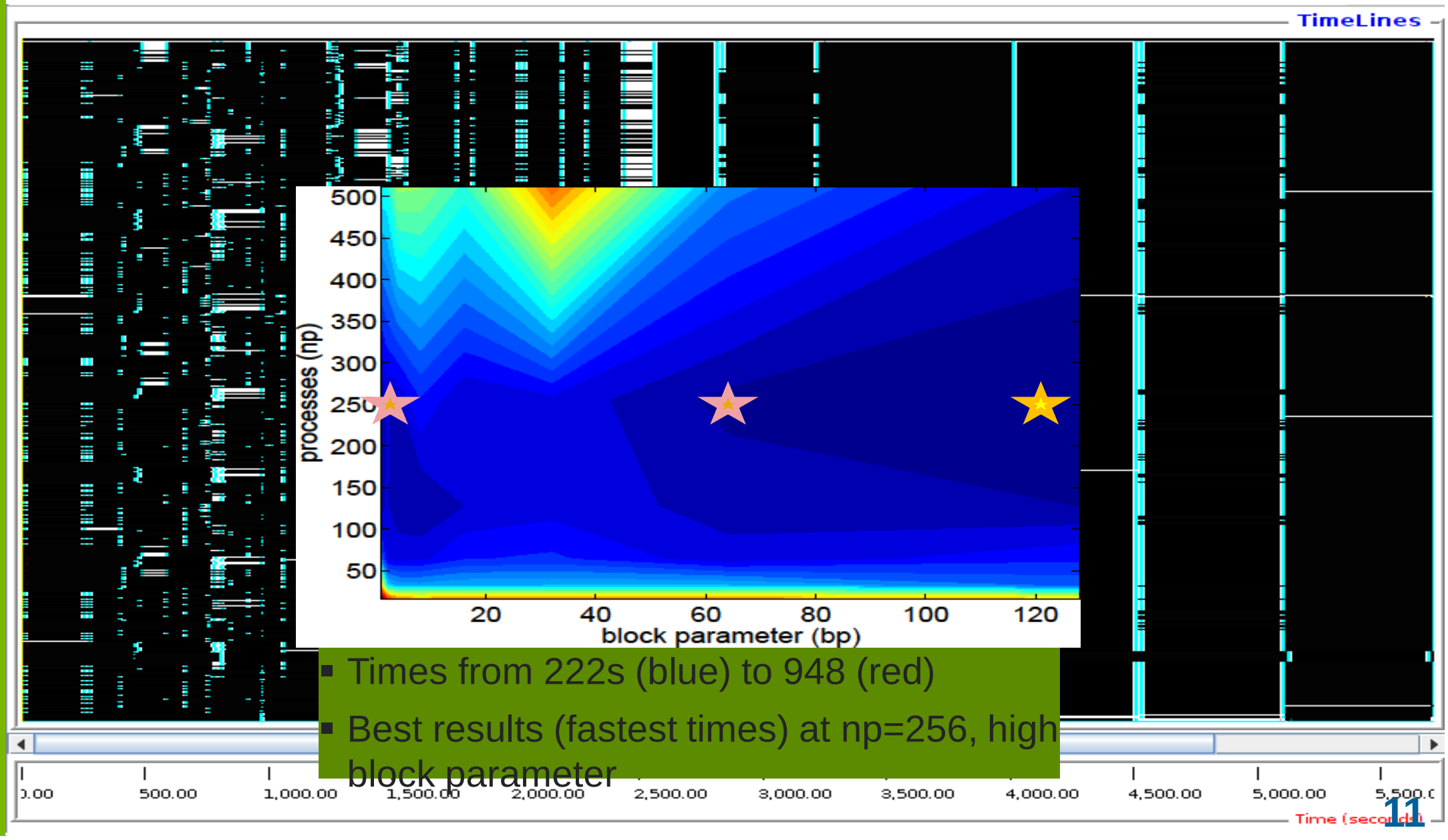


OSUFLOW APPLICATION

Complete code!

```
// Define call to OSUFlow feature MpiDraw
@par (float t) mpidraw(int bf) "mpidraw";

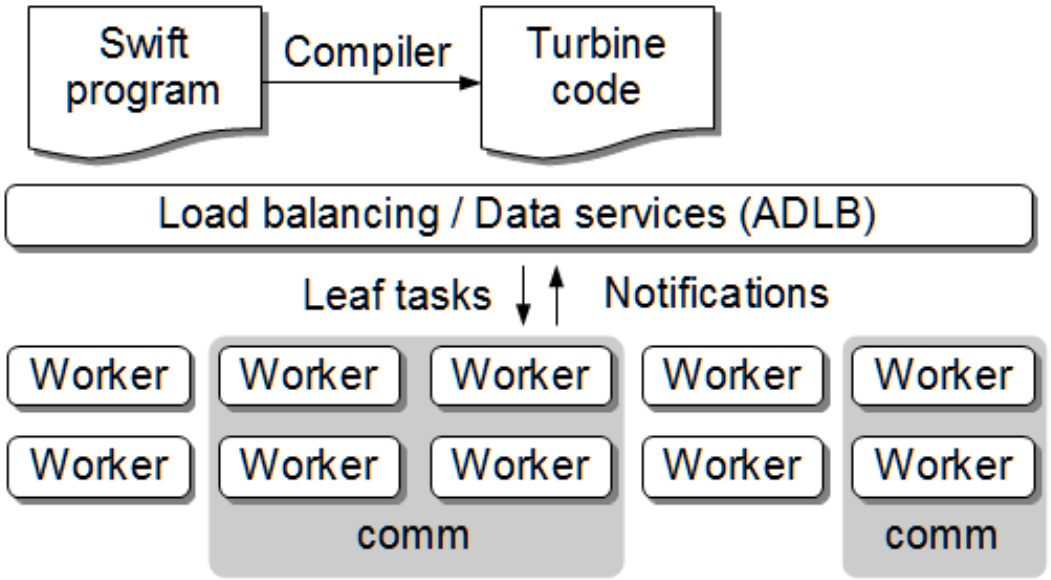
foreach b in [0:7] {
  // Block factor: 1-128
  bf = round(2**b);
  foreach n in [4:9] {
    // Number of processes/task: 16-512
    np = round(2**n);
    t = @par=np mpidraw(bf);
    printf("RESULT: bf=%i np=%i -> time=%0.3f",
                    bf,   np,      t);
  }
}
```



- Times from 222s (blue) to 948 (red)
- Best results (fastest times) at np=256, high block parameter

MULTIPLE PROCESS ALLOCATION

- Swift/T runs an MPI-based load balancer
- When necessary, multiple idle workers may be pooled together
- Swift/T runs MPI_Comm_create_group() on the workers to create a subcommunicator
- Swift/T then invokes the user code on that communicator



LAMMPS PARALLEL TASKS

LAMMPS provides a convenient C++ API

```
foreach i in [0:20] {
  t = 300+i;
  sed_command = sprintf("s/_TEMPERATURE_/%i/g", t);
  lammps_file_name = sprintf("input-%i.inp", t);
  lammps_args = "-i " + lammps_file_name;
  file lammps_input<lammps_file_name> =
    sed(filter, sed_command) =>
    @par=8 lammps(lammps_args);
}
```

This example can be found on GitHub:
https://github.com/b240/Workflows/tree/master/demo/LAMMPS-1
See the README.md file for more information.

MPI_COMM_CREATE_GROUP

- This approach fundamentally uses `MPI_Comm_create_group()`
- Child code runs in same address space as parent code
- Works well on existing systems (even Blue Gene)
- Fault tolerance: if child crashes, all of Swift/T crashes
- Code coupling: may be difficult to integrate different codes into the same workflow if there are compiler or link-time conflicts
- In situ communication: may be difficult to couple libraries with external communication techniques

CONTEXT: PARALLEL LAUNCH IN MPI

- Want to control scientific ensembles from MPI:
  - Parameter sweeps, searches, optimizations
  - Tests under varying parameters and process counts
  - Workflows and code coupling cases where jobs can exit or fail

- State of the art:
  - `MPI_Comm_spawn()` has limited availability
    - in part due to complexity of implementing?
  - `MPI_Comm_spawn()` does not support job exit detection and failures
  - Users write complex shell scripts against vendor-specific job launchers

- Proposing an alternate function: `MPI_Comm_launch()`
- Implemented on clusters (via MPICH/mpiexec hack) and by a vendor (internal)

MPI_COMM_LAUNCH(): DETAILS

- ```
  int MPIX_Comm_launch(const char *cmd, char **argv,
                       MPI_Info info, MPI_Comm comm,
                       int* exit_code);
  ```

- Runs “in-place” on given communicator- no interaction with scheduler, etc.
- Parent is blocked
- No communication between parent and child
- Parent gets exit code- easy recovery from child failures
- Presented at MPI Forum December 2018- assigned to WG for discussion
  ```

MPI_COMM_LAUNCH(): BENEFITS

- Allows user to use familiar communicator management to setup subjob
  - We have done things with `MPI_Comm_split()` and `MPI_Comm_create_group()`
- Easy to work with unmodified child codes
- Allows for the development of MPI-based workload management systems
  - Write simple test harnesses or parameter sweeps in C or Fortran + MPI
  - Or use an MPI-based system like ADLB, Swift/T, or MPI-Bash (all implemented!)
  - Possibly work with other parallel programming systems?

SWIFT/T EXAMPLE

When file `A` is created, launch `N` sub jobs of varying size

```
file B[]; // Define an array of file variables
A => {
  foreach i in [0:N-1] {
    file B_i<"B-%i.txt"%i>;
    string args_B[] = [ int2string(i),
                        filename(A), filename(B_i) ];
    @par=i launch("./child.x", args_B) => B_i = touch();
    B[i] = B_i;
  }}
```

- Child tasks are load-balanced, `MPI_Comm_create_group()` is done automatically!

MPI-BASH

Author: Scott Pakin (LANL)

- Forked and extended for `MPI_Comm_launch()` by Wozniak

```
#! /usr/bin/env mpibash
enable -f mpibash.so
mpi_init mpi_init
mpi_comm_rank rank

mpi_barrier
if [ $rank -eq 0 ] ; then
           iter=0
           while [ $iter -lt $niters ] ; do
               mpi_send 1 X
               mpi_recv 1 msginfo
               let iter++
           done
. . .
# Run with:
$ mpirun -np 16 ./my-script.sh
```

```
#! /usr/bin/env mpibash
enable -f mpibash.so
mpi_init mpi_init
mpi_comm_rank rank
mpi_comm_split $rank $rank newcomm

echo newcomm: $newcomm

mpi_comm_set $newcomm

mpi_comm_rank rank

mpi_comm_launch hostname
echo exit_code $?

. . .
```

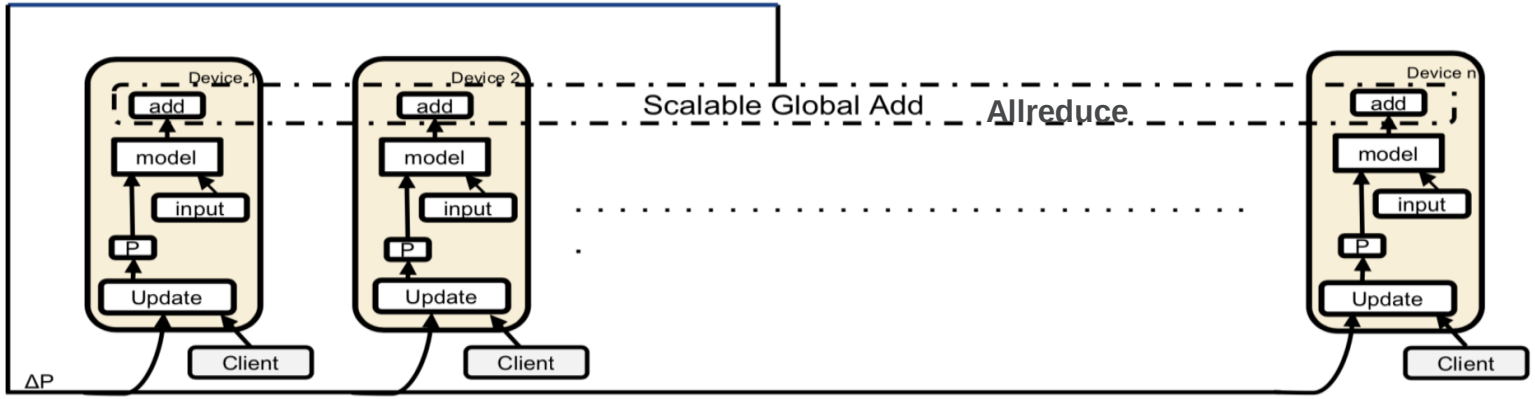
INTEGRATING HOROVOD WITH CANDLE

Argonne NATIONAL LABORATORY

Horovod

- Horovod developed by Uber Engineering in September 2017
  - An open source component of Uber internal ML-as-a-service platform Michelangelo's deep learning toolkit
  - A distributed deep learning framework to speed up TensorFlow

- Replaces native optimizer class with a new class DistributedOptimizer
  - Adds an allreduce operation between gradient computation and model update to average the gradients, then applies those averaged gradients



Source: Peter Mendygral, Scaling Deep Learning, ALCF SDL Workshop 2018

HOROVOD AS A LIBRARY

- Horovod is a Python-wrapped C++ library that is injected into TensorFlow
- Normally invoked as a stand-alone MPI application

```
$ mpiexec –n 4 python train.py
```

- Need to load the library and pass it a communicator to run on from C/C++
- Ultimately callable from any C/C++ program, notably Swift/T

CALLING HOROVOD FROM SWIFT/T

- Example Swift/T script to carry out a scaling sweep:

```
code = read(input("test.py"));
foreach p in [1:10] {
  @par=p horovod(code);
 }
```

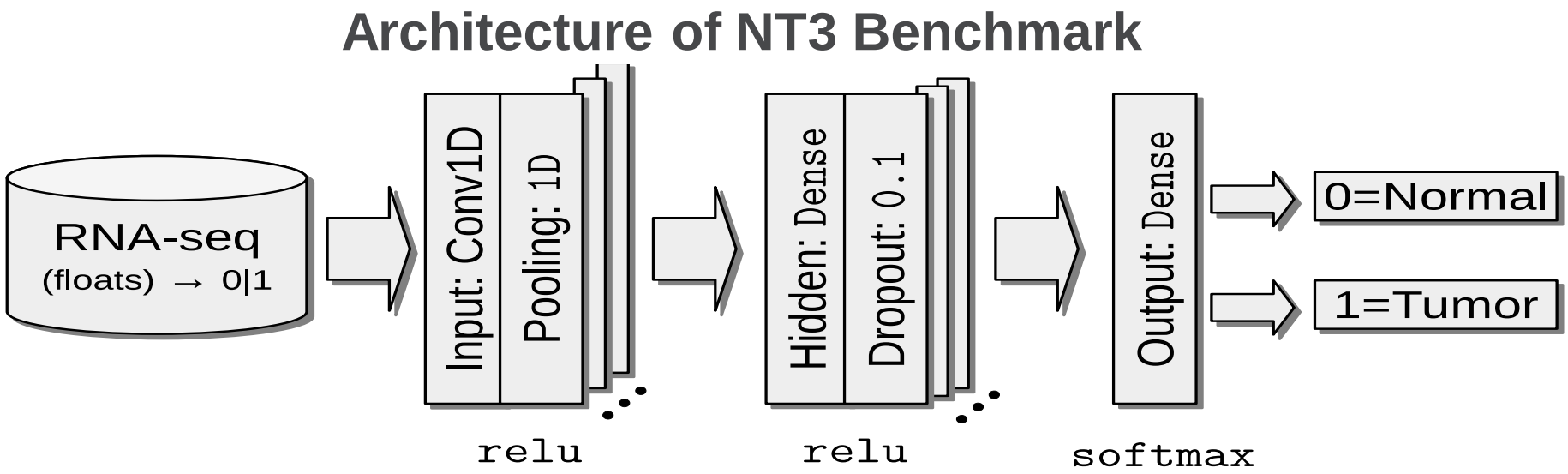
- Invokes a C function:

```
int controller(MPI_Comm comm, char* code);
```

- The C function can be called from any MPI program!

CANDLE Benchmarks

- **Pilot 1 Benchmarks**: P1B1, P1B2, P1B3, **NT3** (TensorFlow)
  - At the cellular level to predict drug response based on molecular features of tumor cells and drug descriptors

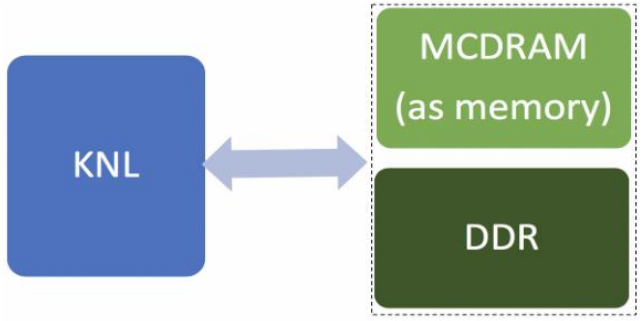


Source: CANDLE Benchmarks, https://github.com/ECP-CANDLE/Benchmarks

PERFORMANCE ANALYSIS

- Our collaborators built a Horovod-based version of the CANDLE app NT3
- Studied how the app behaves on Theta in its two memory modes using
- Overhead in NT3 increases significantly with the number of nodes although Horovod has the ability to scale up.

Cache mode

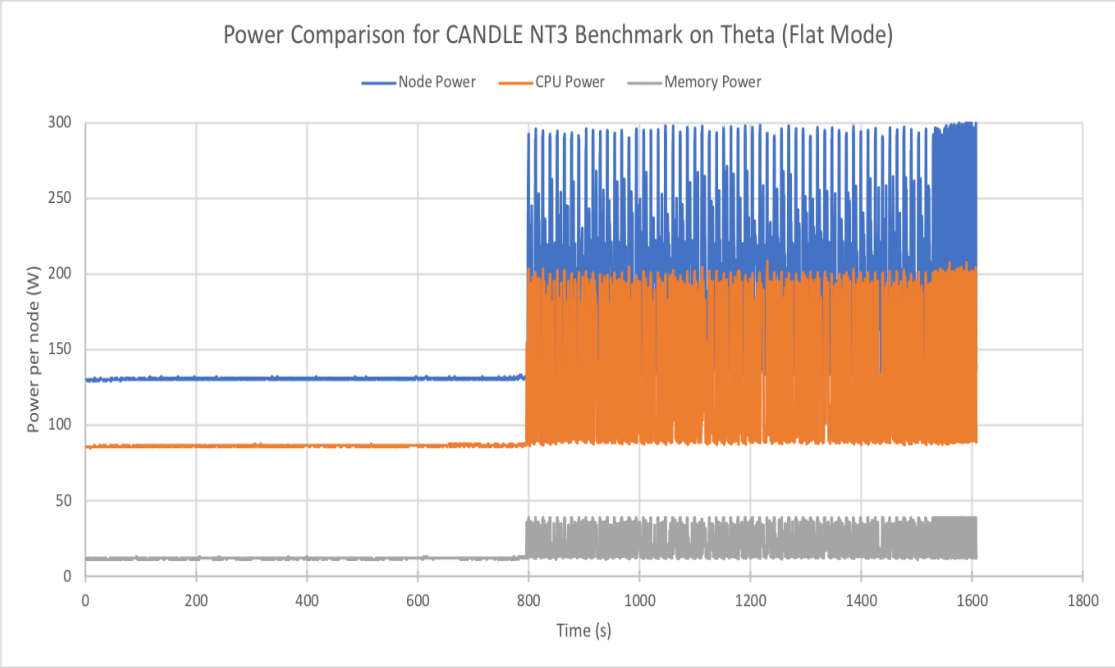


Flat mode

INTERACTION WITH MCS

With Xingfu Wu and Valerie Taylor

- **CANDLE** benchmarks perform differently under different hyperparameters, parallelization
- **Goal:** Analyze the performance and power usage of CANDLE Benchmarks
- Use PoLiMeR to measure power and performance of CANDLE Benchmark
- Could be used to determine
- Shows difference between single-node parallelism and Horovod usage



Performance, Power, and Scalability Analysis of the Horovod Implementation of the CANDLE NT3 Benchmark on the Cray XC40. Xingfu Wu, Valerie Taylor, Justin M. Wozniak, Rick Stevens, Thomas Brettin, and Fangfang Xia. Proc. PyHPC @ SC 2018.

Summary

- We analyzed the performance, power, and scalability of the Horovod NT3 with weak scaling and strong scaling
- Power profiling is useful for showing how the Horovod NT3 benchmark behaves on the underlying system.
- The Horovod overhead increased significantly with the number of nodes although Horovod has the ability to scale up.
- The benchmark under the cache mode resulted in smaller runtime and lower power consumption for the node and CPU as compared with the flat mode.
- Increasing the batch size led to a runtime decrease and slightly impacted the power
- and the model training in NT3 requires the proper number of epochs to achieve the high accuracy.

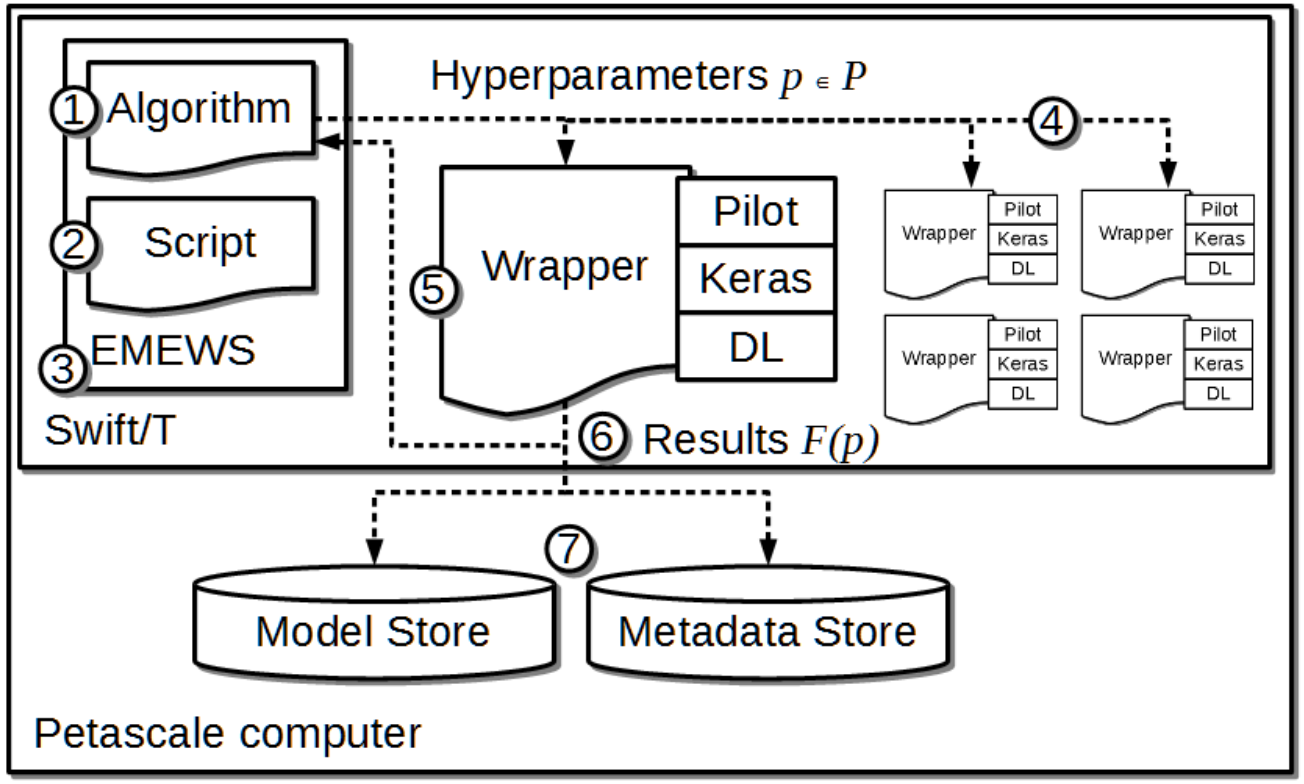
CONTAINERIZATION OF PARALLEL TASK WORKFLOWS

Argonne NATIONAL LABORATORY

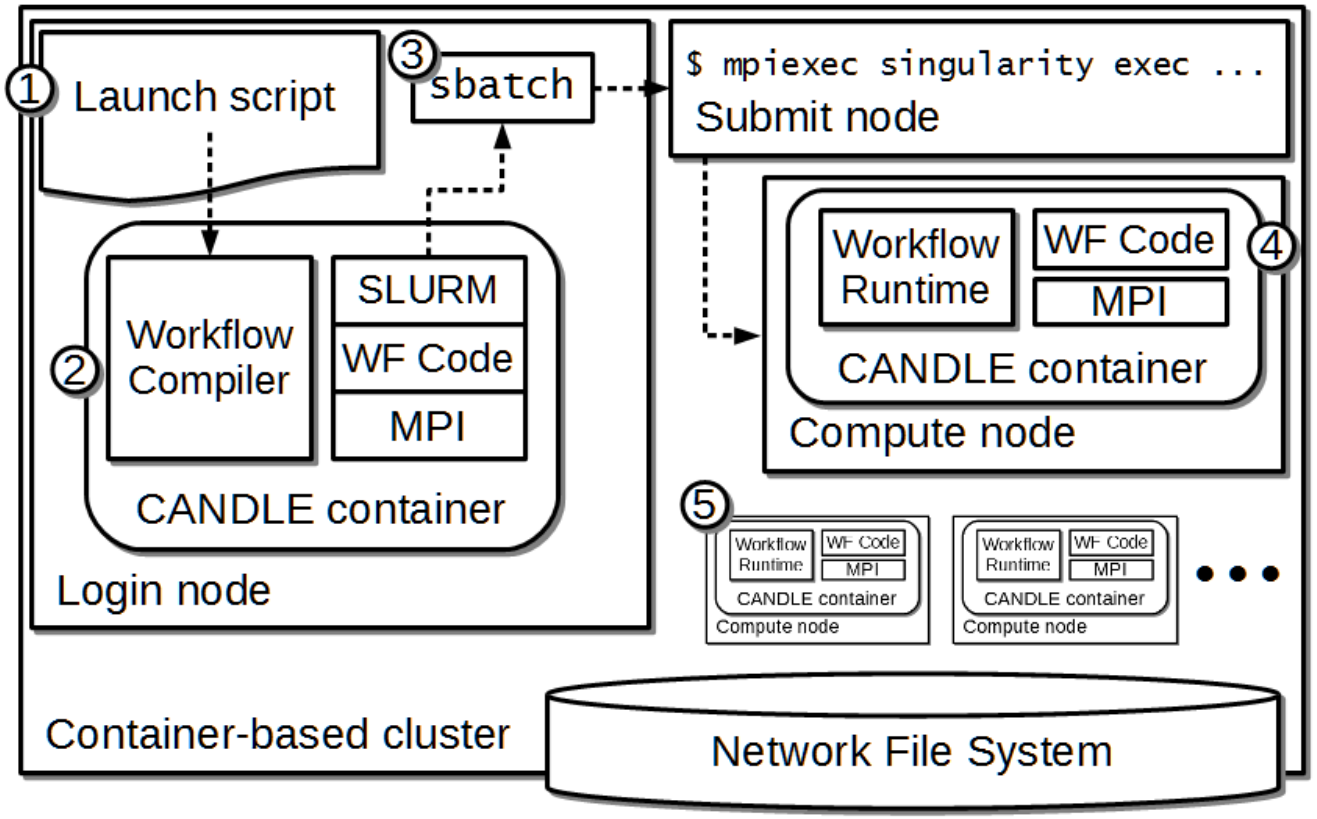
CANDLE CONTAINERS: MOTIVATION

- Portability and reproducibility are major requirements in life science applications
- Containers technologies help life scientist to focus on their domain problem
- This is a prototype implementation running on Biowulf, we are testing it on Summit
- Future work includes adding data parallelism capabilities to the container in addition to HPO

BASIC CANDLE ARCHITECTURE



CANDLE ARCHITECTURE WITH CONTAINERS



CANDLE CONTAINER PERFORMANCE STUDY

- **HPO:** Grid search
- **Model:**
- **Time:** 3 minutes
- **Workers:** 2 MPI tasks
- **Native run:** **891** evaluations
- **Container run:** **869** evaluations

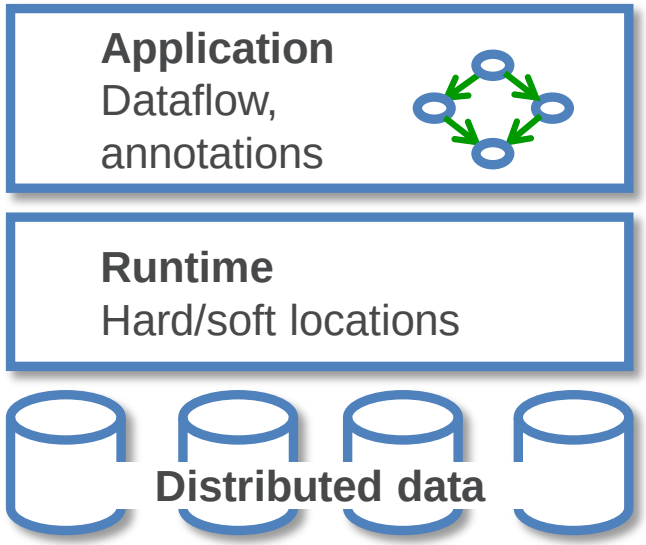
- In our tests on the NIH HPC cluster, the container overhead is negligible.

- **Portable and reusable deep learning infrastructure with containers to accelerate cancer studies.** Zaki, Wozniak, et al. Proc. PyHPC @ SC 2018.

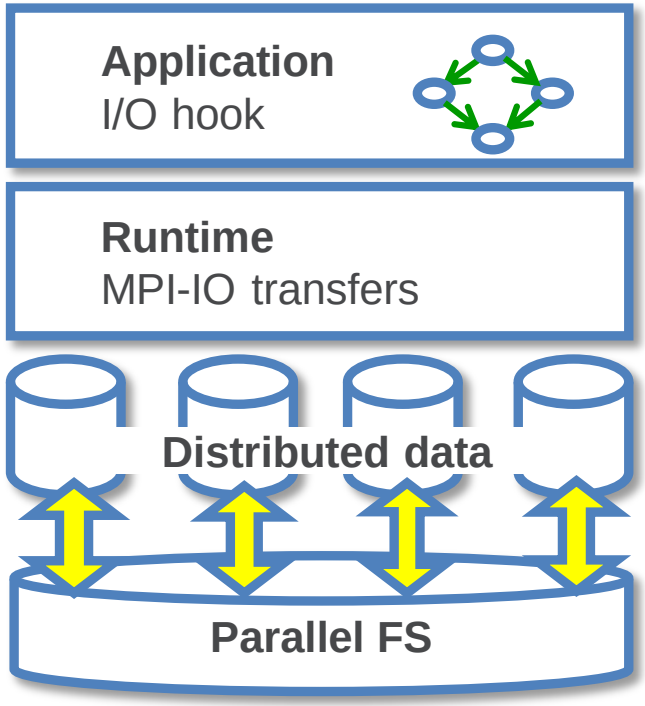
FEATURES FOR BIG DATA MANAGEMENT

FEATURES FOR BIG DATA ANALYSIS

- **Location-aware scheduling**
  User and runtime coordinate data/task locations



- **Collective I/O**
  User and runtime coordinate data/task locations



- F. Duro et al**. Flexible data-aware scheduling for workflows over an in-memory object store**. Proc. CCGrid, 2016.

TASK LOCATIONS

- User-written annotation on function call
- Swift/T provides a **hostmap** library that maps host names to MPI ranks
- User annotation sends function to rank:

```
foreach i in 0:N-1 {
  location L = hostmap_lookup("file"+i);
  @location=L f(i);
}
```

- Useful for data-intensive applications or leaf functions with state
- Soft locations: allow queued tasks to be stolen and execute anywhere

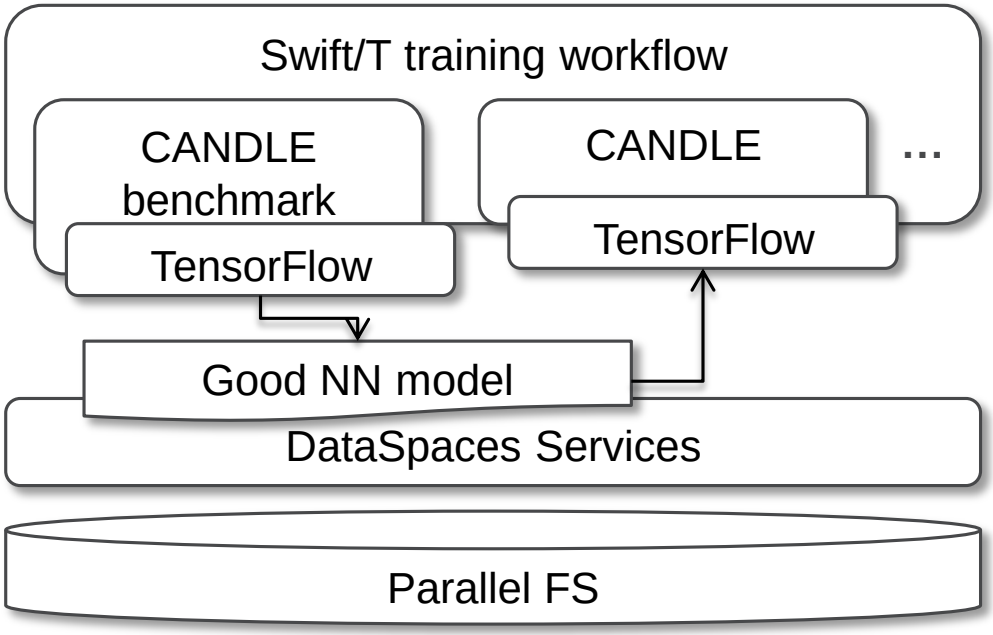
```
foreach i in 0:N-1 {
  location L = hostmap_lookup("file"+i);
  @location=(L, SOFT) f(i);
}
```

DATA RATES FOR MODEL GENERATION

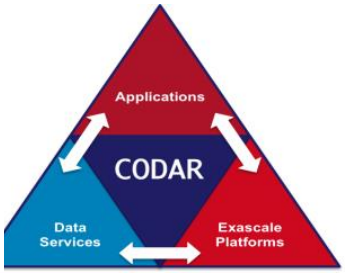
- Model generation rates are manageable now but could become an unnecessary drain of system resources
- ***M***: Memory of a node (assumed full of NN weights)
- ***N***: Number of nodes ***T***: Time to train a model
- Data is generated at a rate of ***M*** x ***N*** / ***T*** bytes/second.
- On Cori: ***M*** = 64 GB; ***N*** = 9,688; ***T*** = 3,600 for a sustained rate of 43 GB/s.
- On Summit: ***M*** = 512 GB; ***N*** = 4,608; ***T*** = 3,600 for a sustained rate of 655 GB/s.
- As computers increate in speed and scale, these rates will increase rapidly
- ***Most of these models will be simply thrown out!***

USING WORKFLOW-INTEGRATED STORAGE

- **CANDLE** workflows produce a great number of medium-sized ML models
- **Goal:** Cache these on compute node storage for *possible* later use
- Need to flush to global FS before end of run, but many models will be discarded
- **Accomplishment:** Integrated Swift/T workflow system used in CANDLE with DataSpaces client
- Accelerate CANDLE workflow performance, enable novel training strategies (parameter sharing)
- Demonstrate the utility of node-local storage for complex workflows



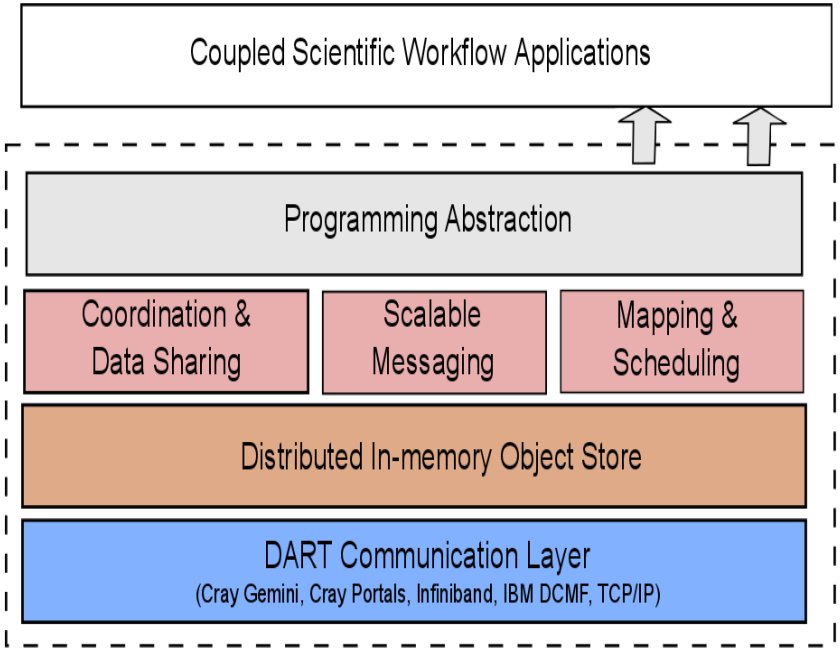
Scaling Deep Learning for Cancer with Advanced Workflow Storage Integration. Justin M. Wozniak, Philip E. Davis, Tong Shu, Jonathan Ozik, Nicholson Collier, Manish Parashar, Ian Foster, Thomas Brettin, and Rick Stevens. Proc. MLHPC @ SC 2018.



Argonne NATIONAL LABORATORY

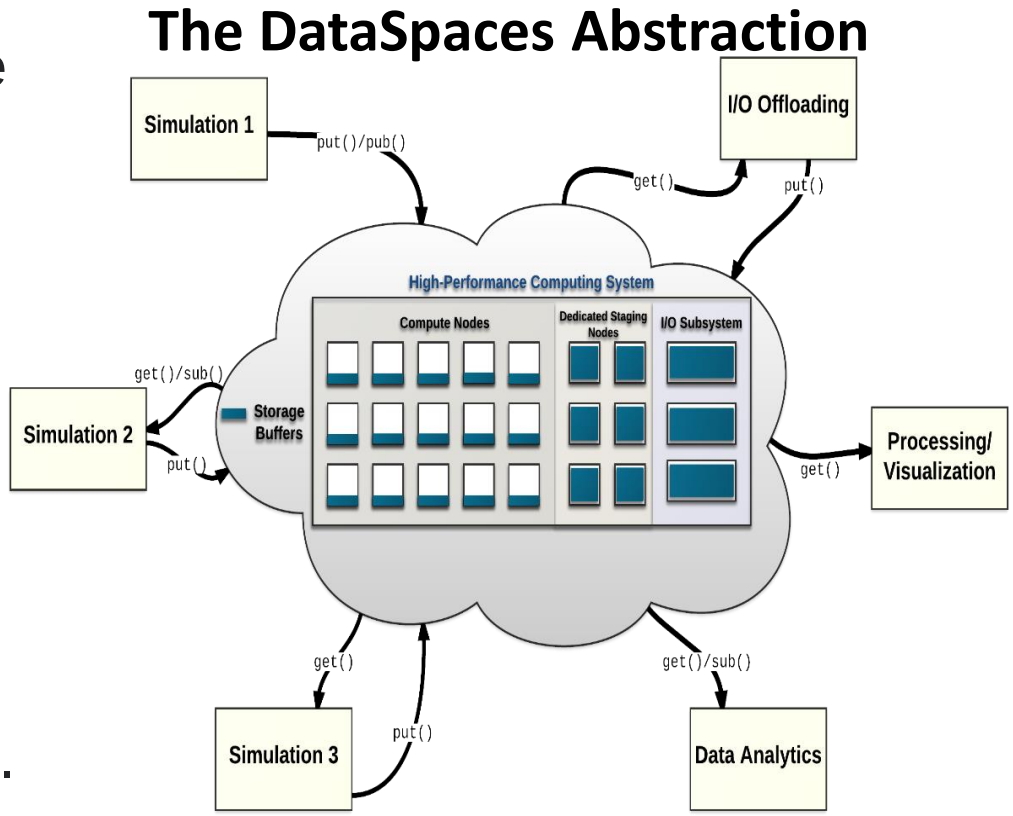
DATASPACES CODE COUPLING LIBRARY

- Client library and storage servers
  - Storage servers run on separate nodes
- Provides shared storage space using DRAM from server nodes
  - Uses RDMA for low-latency transfer
  - Enables communication between tasks, decoupled in **space** and **time**
  - Optimizations for HPC workflows
  - Assumes a shared space abstraction
  - Provides spatially-significant put/get primitives (variable X bounding box X version)
  - Indexing optimized for spatial locality of access with a SFC, allowing a high degree of scalability using a DHT.



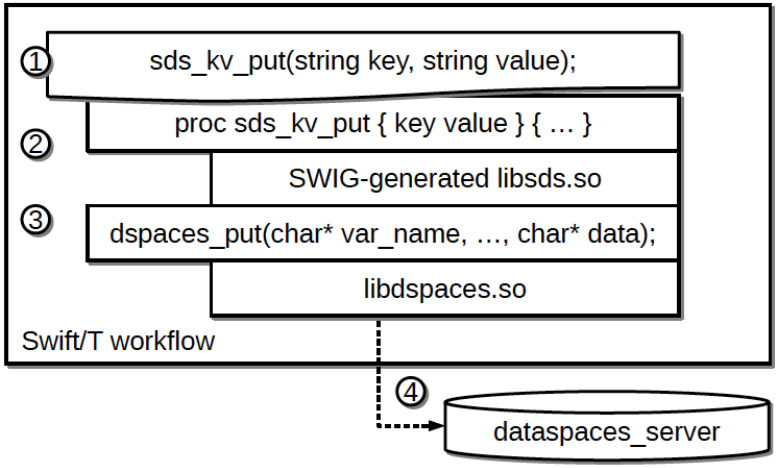
DATASPACES INTERFACE AND TYPICAL USE

- Data is manipulated with a put/get interface
  - `dspaces_put()` – provide variable name, size, version, and bounding box along with input data buffer
  - `dspaces_get()` – arguments symmetric to `dspaces_put()`, put data into provided output buffer
- Also provide simple synchronization primitives to check if a `dspaces_put()` operation has completed, and to stall a reader if a put operation has not completed.



SWIFT/T-DATASPACES (SDS) INTEGRATION

- Extend Swift/T with DataSpaces support
  - Implement a key/value store, where the value can be an array or file
  - Create small C wrapper around DataSpaces put/get primitives
  - Use SWIG to generate a shared object which a Swift module can load and use
- Place key/value pairs into globally-accessible space
  - Decouple data access from data generation
  - Fast RDMA access vs shared FS access
  - SDS also provides APIs to load node-local files into DataSpaces



DataSpaces & Swift/T Callstack

PERFORMANCE RESULTS

- Ran workflows in Swift/T using DataSpaces and also the basic parallel file system (PFS) as a control
- Systems:
  - Blues (ANL/LCRC)
    - 306 nodes with 16-core Intel Xeon E5-2670 @ 2.6GHz, 64 GB RAM
  - Cori (LBNL/NERSC)
    - 9,688 nodes with Intel Xeon Phi, 16 GB MCDRAM, 96 GB RAM

```
foreach i in [0:N-1] {
  file f<"key"+i> = write("value");
  string s = read(f);
}
```

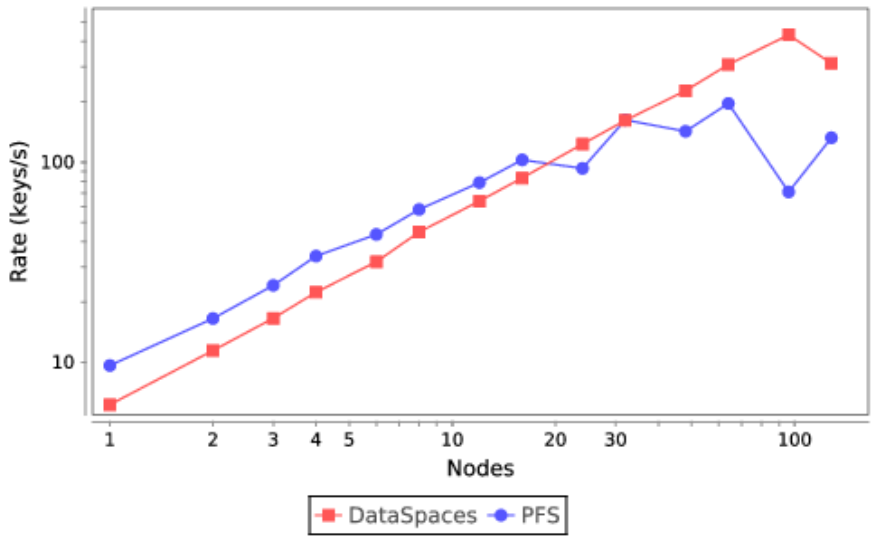
PFS

```
foreach i in [0:N-1] {
  sds_kv_put("key"+i, "value1") =>
  sds_kv_get("key"+i, 100);
}
```

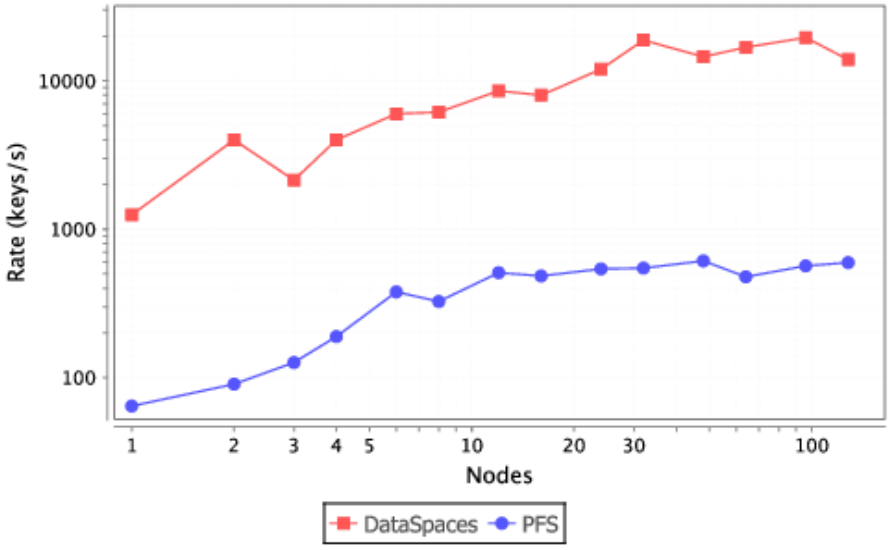
DataSpaces

PERFORMANCE RESULTS: SMALL I/O

- Small write/read operations from Swift/T on Blues and Cori



Blues



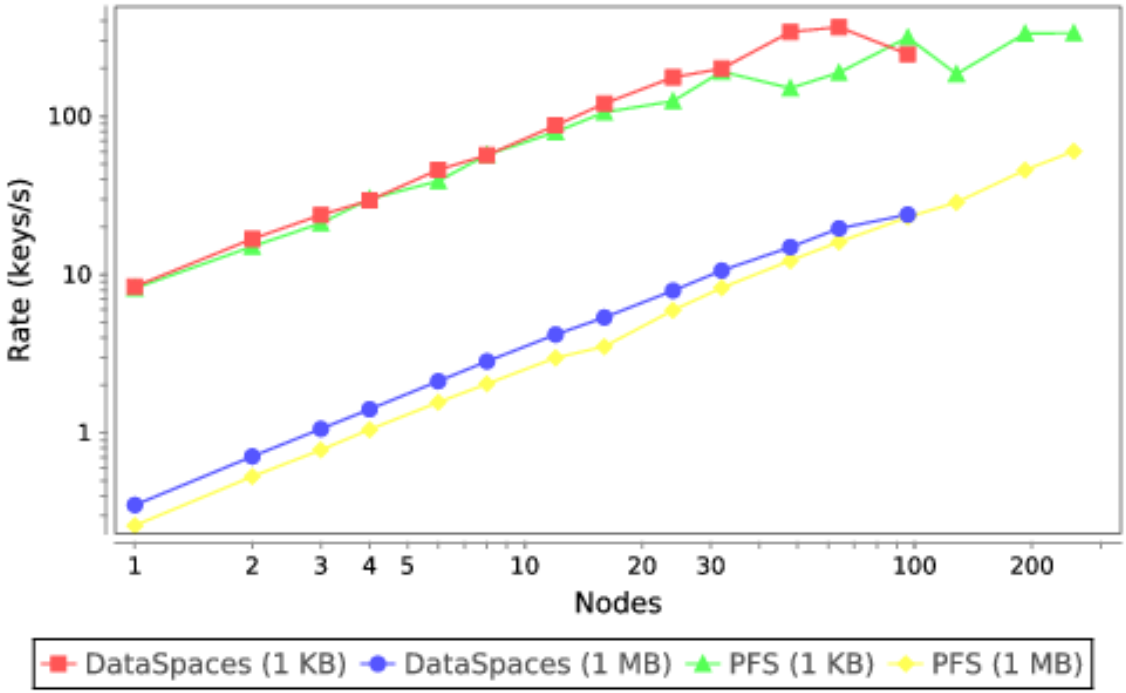
Cori

TWO-STEP I/O

- File is written to `/tmp` , then copied into DataSpaces
- 1 KB and 1 MB files

```
foreach j in [0:M-1] {
  foreach r in [0:N-1] {
    // in /tmp
    name = make_file(j,r);
    file f<name> = make_data(SIZE)
         => sds_kvf_put(name, name);
  }
}
```

SDS



CONCLUSIONS

- Presented an overview of the data parallel workflows in the CANDLE project
- Described use of Horovod in CANDLE and the use of containers
- Described the output data challenge in model generation
- Presented performance results from a tight coupling of Swift/T and DataSpaces

THANKS

- Thanks to the organizers

- Code and guides:
  - CANDLE GitHub Organization: https://github.com/ECP-CANDLE
  - Swift/T Home: http://swift-lang.org/Swift-T
  - EMEWS Tutorial: http://emews.org

- This research was supported by the Exascale Computing Project (17-SC-20-SC), a joint project of the U.S. Department of Energy's Office of Science and National Nuclear Security Administration, responsible for delivering a capable exascale ecosystem, including software, applications, and hardware technology, to support the nation's exascale computing imperative.

TUTORIAL: SUPERVISOR

Argonne NATIONAL LABORATORY

HANDS-ON TUTORIALS

- May be found here:
  - `git@github.com:brettin/candle_tutorials.git`
  - Subdirectory `Topics/2_hyperparameter_optimization`
- See the top-level README to get started with the installation