Early experiences with Machine Learning and Deep Learning on Summit/Summit-Dev

Junqi Yin

Advanced Data and Workflows Group

Outline

- ML/DL software stack on Summit
- CORAL2 benchmark
  - Data Science benchmark
    - Big Data Analytics Suite
    - Deep Learning Suite
- ML/DL performance model: Summit-Dev to Summit
- Scaling DL
  - Resnet50 on ImageNet
  - Lessons learned from exa-scale DL on Summit
- Discussion: ML vs DL use cases

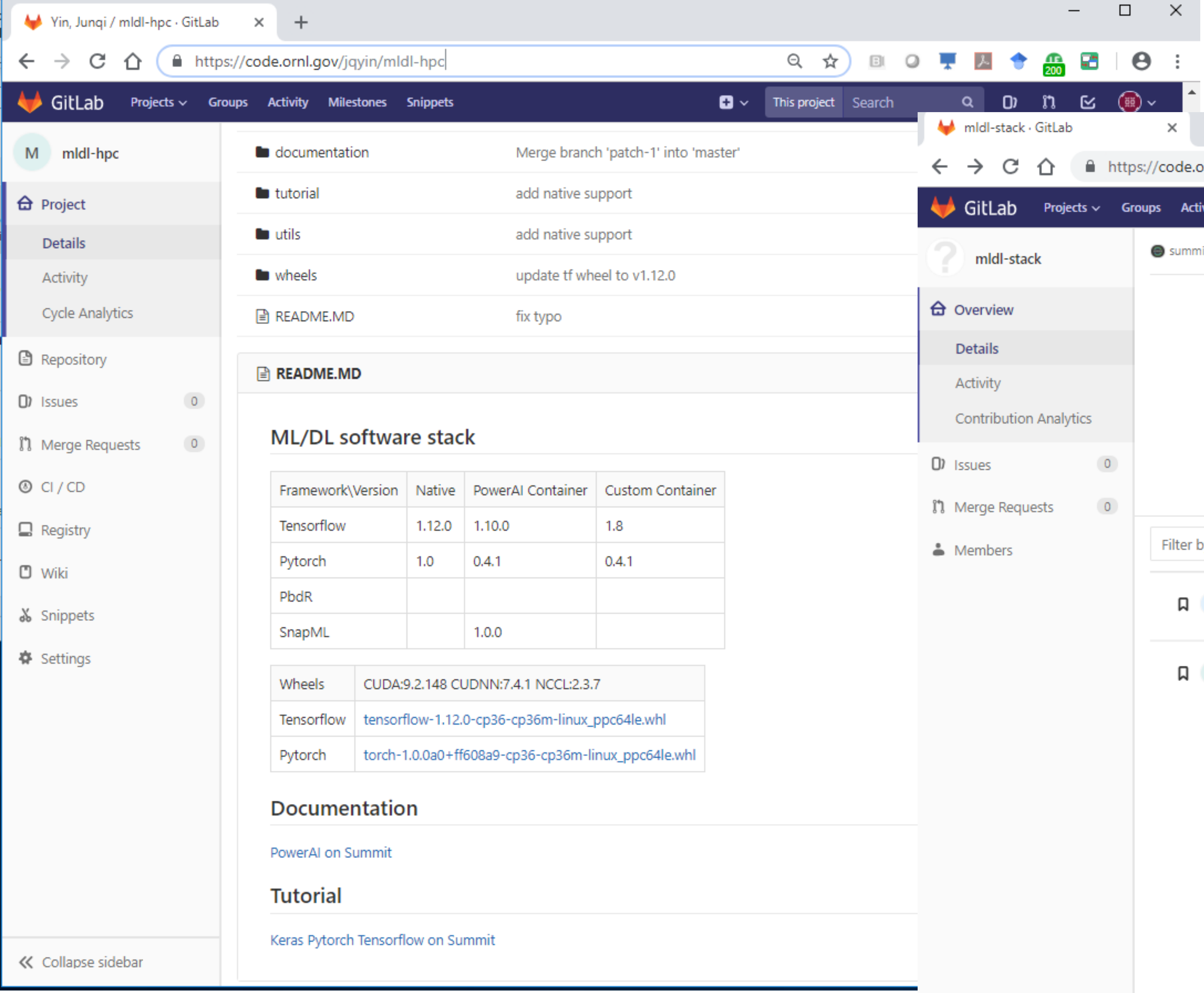
ML/DL software stack on Summit (current plan and subject to change)

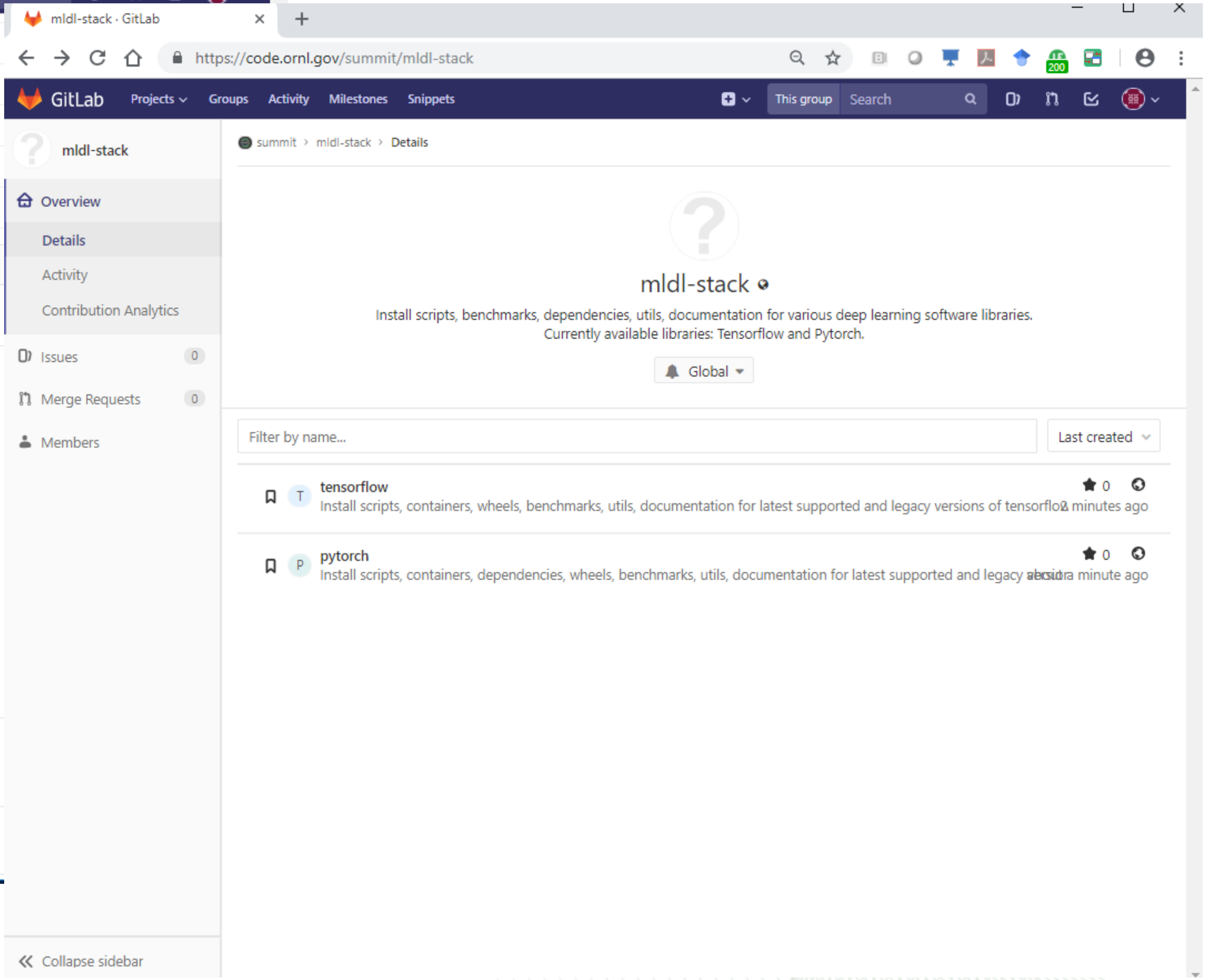
- Native installation
- IBM PowerAI container
- Custom container with Singularity (in planning)

/gpfs/wolf/stf011/world-shared

Framework \Version	Native	PowerAI Container	Custom Container	Python Wheels
Tensorflow	1.12	1.10, 1.8	1.9	tensorflow-1.12.0-cp36-cp36m-linux_ppc64le.whl
Pytorch	1.0rc1	0.4.1	0.4.1	torch-1.0.0a0+ff608a9-cp36-cp36m-linux_ppc64le.whl
R/PbdR	1.1		1.1	
SnapML		1.0.0		

https://code.ornl.gov/summit/mldl-stack



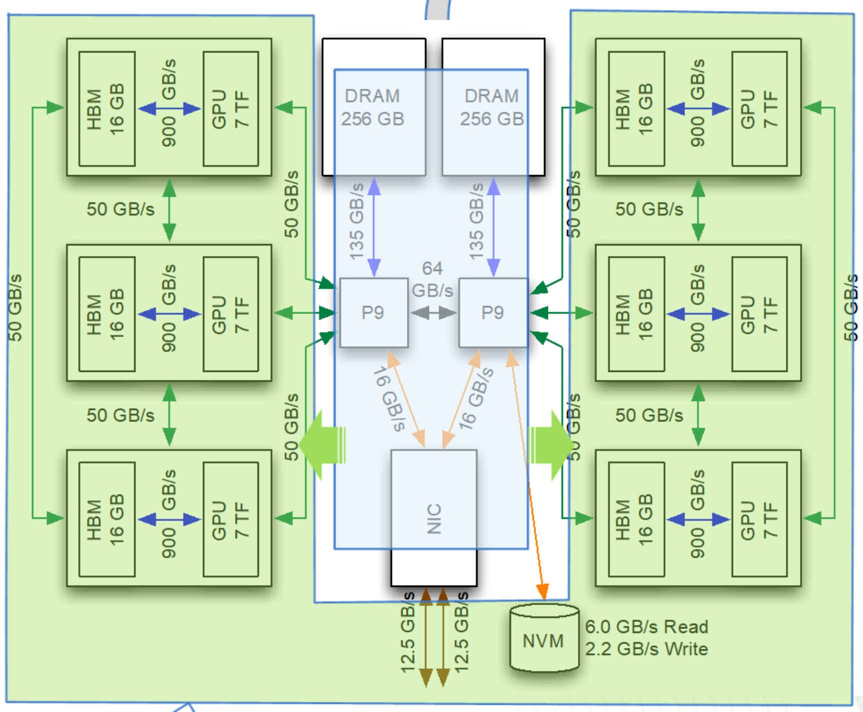


https://code.ornl.gov/jqyin/mldl-hpc

CORAL-2 Data Sciences Benchmarks

Benchmarks	Description
Big Data Analytic Suite	PCA, K-Means, and SVM (based on pbdR)
Deep Learning Suite	CANDLE, CNN, RNN, and ResNet-50 (distributed)

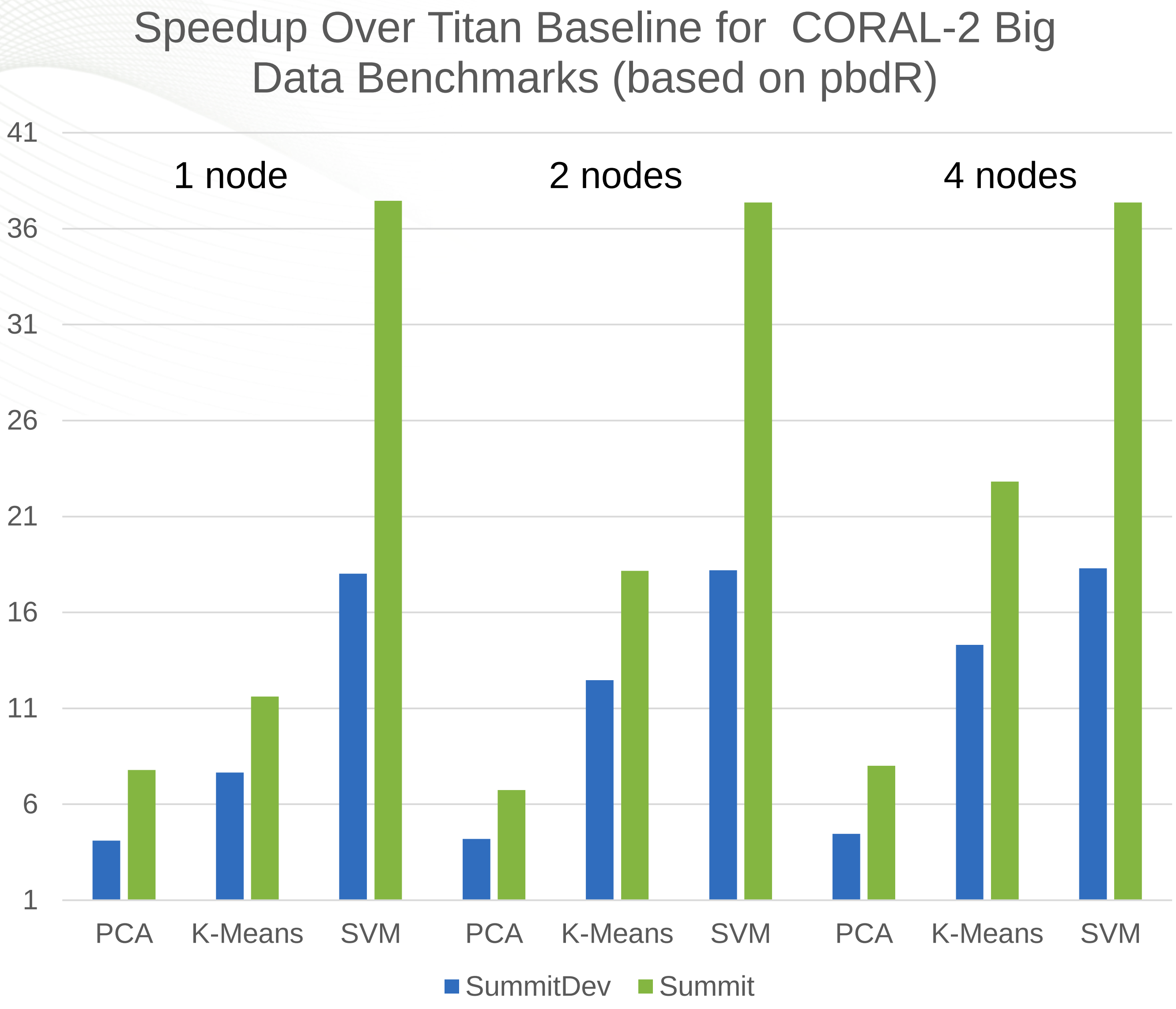
Traditional Node:
PCA, K-Means, etc. excel due to the node's memory, CPU, and on-chip bandwidth

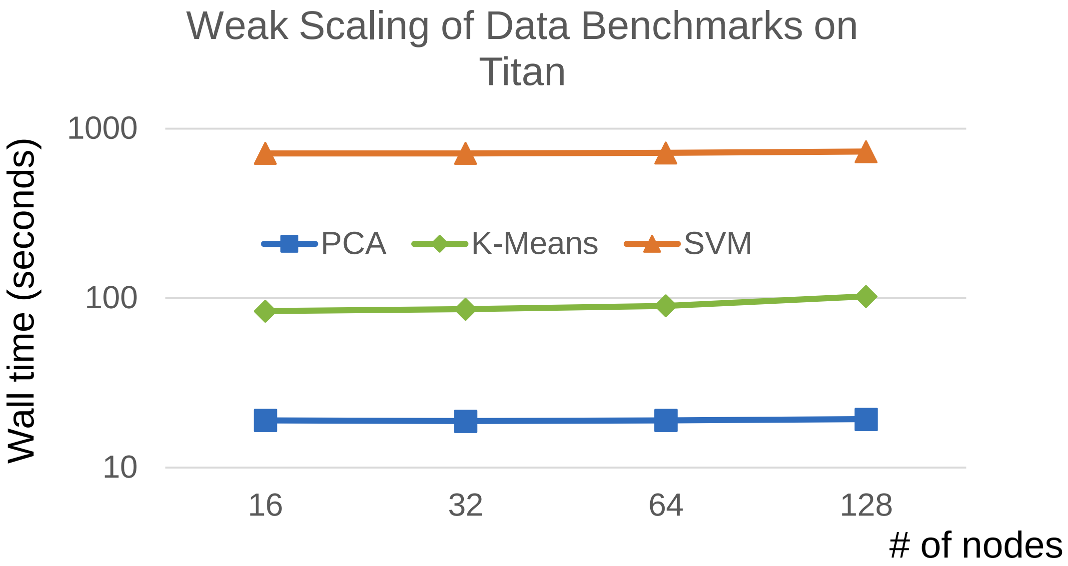


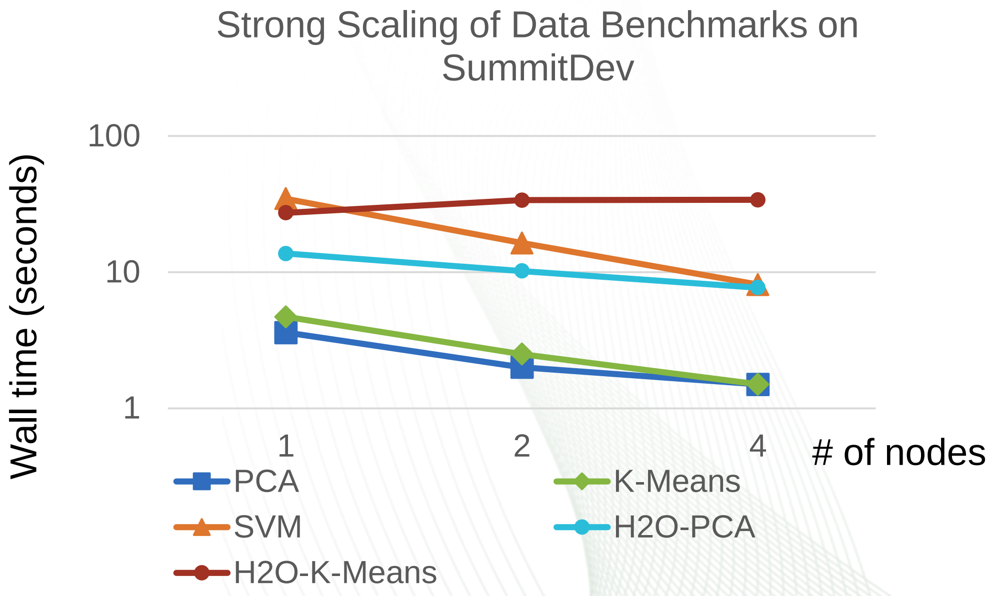
Deep Learning Codes (CNN; ResNet50; ..) excel here with NVM and GPUs enabling tensor operations.

https://asc.llnl.gov/coral-2-benchmarks/

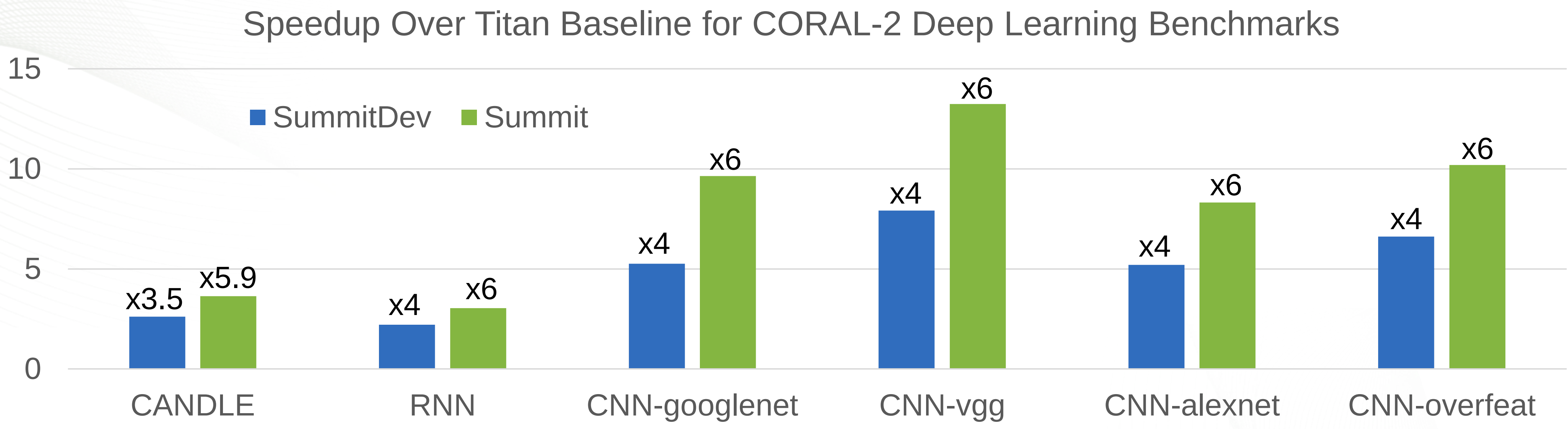
Big Data Analytic Suite

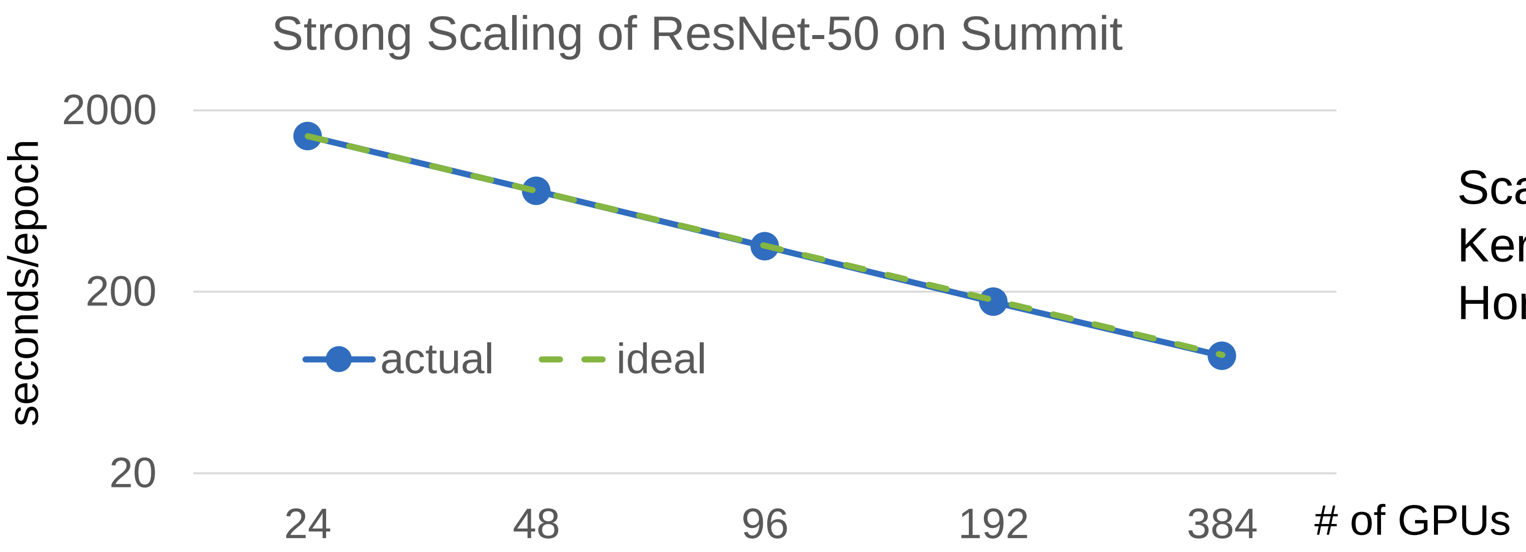






Deep Learning Suite





Scaling of Resnet-50 based on Keras (Tensorflow backend) and Horovod on ImageNet data

Performance model for BDAS

Architecture					Power9												
	Workload(4aff10a)				PCA						Kmeans						
		Input size			8GB			64GB			8GB			64GB			
			SMT (thread|rank)		1	2(2|42)	4(4|42)	1	2(1|84)	4(2|84)	1	2(2|42)	4(4|42)	1	2(2|42)	4(4|42)	1
				walltime(s)	1.9	2.1	2.4	13	12.2	13.5	3.1	3	3	26.2	26.5	26.6	16.7
Power8	PCA	8GB	1	6.8	3.58x	3.24x	2.83x										
			2(1|40)	3.6	1.89x	1.71x	1.50x										
			4(2|40)	3.7	1.95x	1.76x	1.54x										
			8(4|40)	4	2.11x	1.90x	1.67x										
		64GB	1	53.1				4.08x	4.35x	3.93x							
			2(1|40)	28.2				2.17x	2.31x	2.09x							
			4(1|80)	24.8				1.91x	2.03x	1.84x							
			8(2|80)	25.9				1.99x	2.12x	1.92x							
	Kmeans	8GB	1	4.7							1.52x	1.57x	1.57x				
			2(2|20)	4.8							1.55x	1.60x	1.60x				
			4(4|20)	4.8							1.55x	1.60x	1.60x				
			8(8|20)	4.9							1.58x	1.63x	1.63x				
		64GB	1	72.8										2.78x	2.75x	2.74x	
			2(1|40)	49.2										1.88x	1.86x	1.85x	
			4(1|80)	35.3										1.35x	1.33x	1.33x	
			8(2|80)	35.8										1.37x	1.35x	1.35x	
	SVM	8GB	1	34.6													2.07x
			2(1|40)	33.1													1.98x
			4(2|40)	34.7													2.08x
			8(2|80)	37.5													2.25x
		64GB	1	286.6													
			2(1|40)	285.7													
			4(1|80)	283.1													
			8(1|160)	291.5													

$$\log\left(Perf_{Power}\right) = Architecture + Size + Workload + Threads$$

Performance model for DL workloads

Architecture					Volta											
	Workload				CNN				RNN				Comm			
		Implementation			WINOGRAD_NONFUSED		IMPLICIT_PRECOMP_GEMM		LSTM		GRU		NCCL		MPI	
			Precision		fp32	fp16	fp32	fp16	fp32	fp16	fp32	fp16	fp32	fp16	fp32	fp16
				walltime(s)		2.99		1.98		5.1		227.98		0.33		0.46
Pascal	CNN	WINOGRAD_NONFUSED	fp32													
			fp16	5.1		1.71x										
		MPLICIT_PRECOMP_GEMM	fp32													
			fp16	3.1				1.57x								
	RNN	LSTM	fp32													
			fp16	7.4						1.45x						
		GRU	fp32													
			fp16	359.5								1.58x				
	Comm	NCCL	fp32													
			fp16	0.3										1.27x		
		MPI	fp32													
			fp16	0.9												1.91x

Problem size :

WINOGRAD_NONFUSED: input: 112x112xx64x16 filter: 3x3x128

IMPLICIT_PRECOMP_GEMM: input: 112x112xx64x8 filter: 3x3x128

lstm:1024-64-25 gru: 1024-64-1500 (RNN)

100000 4nodes (Comm)

Takeaway - ML

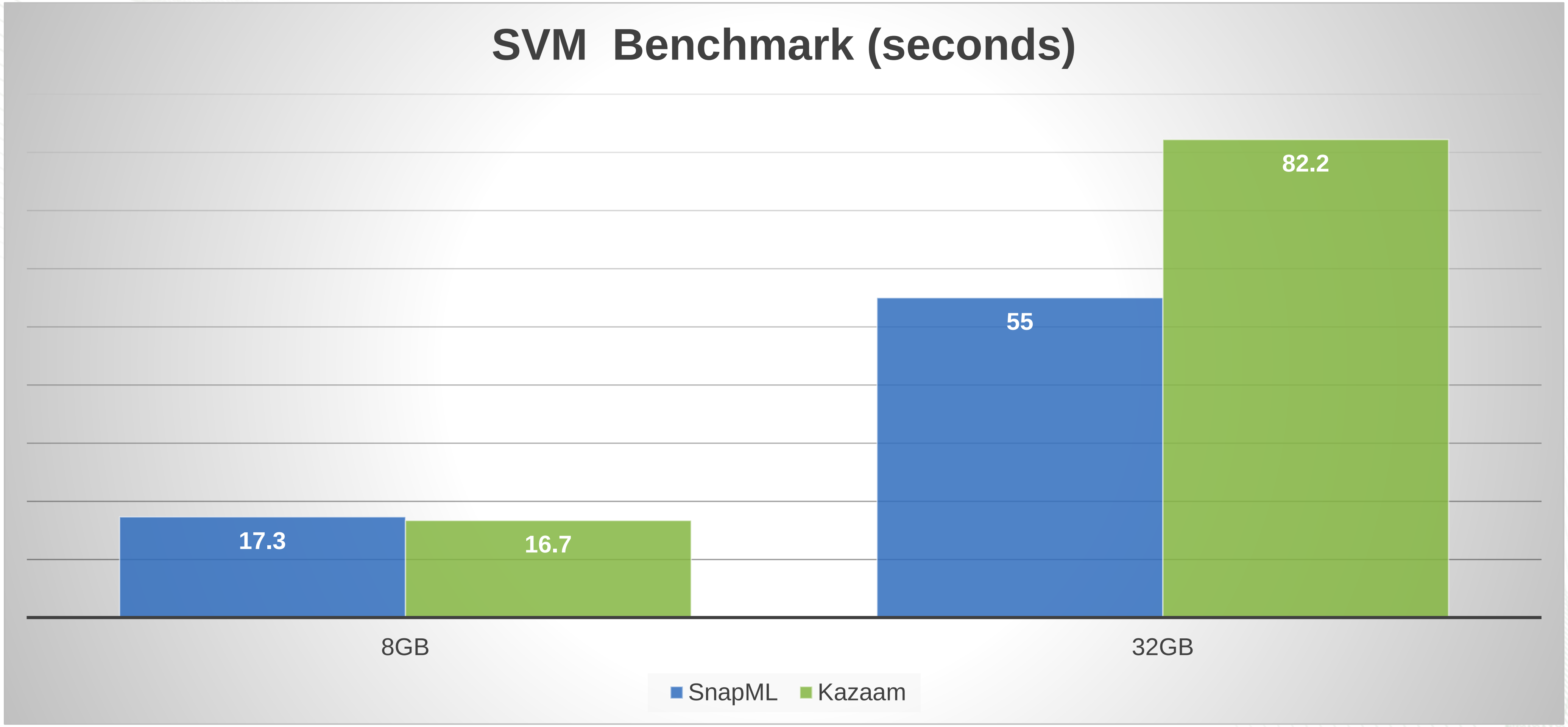
- Per node, expect ~2x over SummitDev, up to ~35x over Titan.
- OpenBLAS provides close performance as IBM ESSL, although ESSL seems to handle SMT better.
- Use SMT=1/2 on Summit SMT=2/4 on SummitDev for pbdR and oversubscribe threads.
- Use RAPIDS, H2O4GPU, SnapML (close source), etc to take advantage of GPUs

For more details, please refer to arXiv:1811.02287

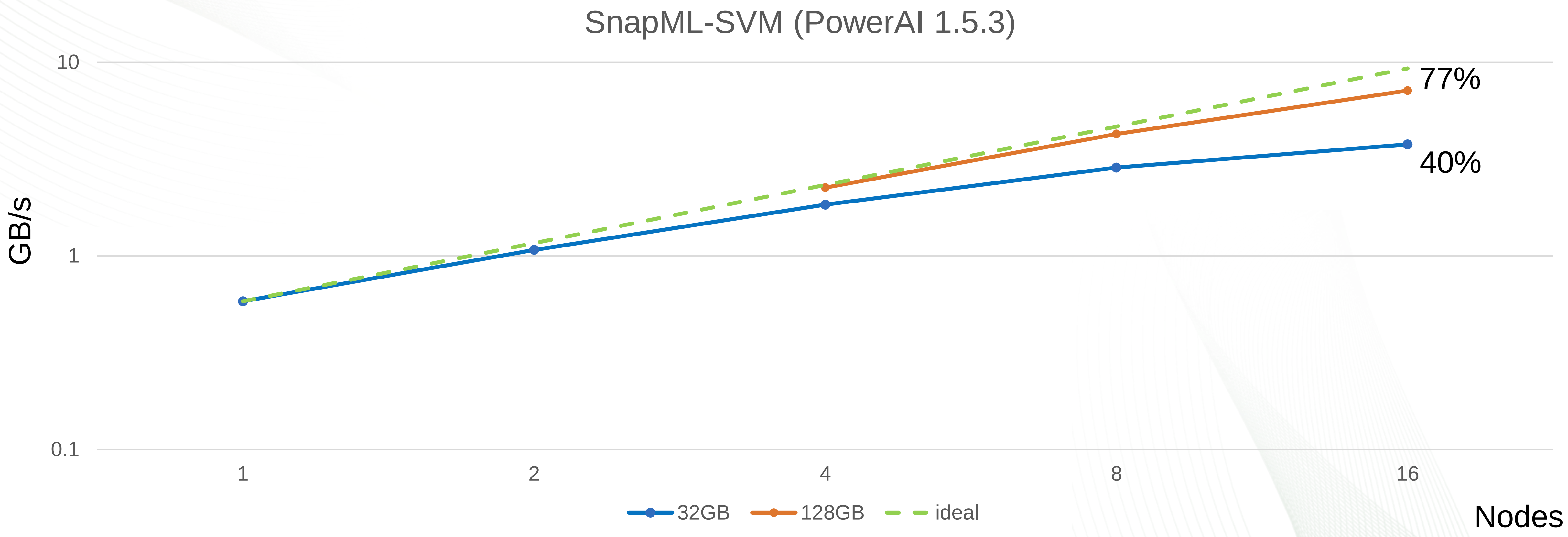
Takeaway - DL

- Per node, expect ~2.5x over Summit-Dev, up to ~ 80x over Titan.
- Average ~60x for CNN workloads, ~20x for RNN workloads, over Titan
- ~1.5x in communication over Summit-Dev
- Near ideal scaling for Keras (Tensorflow backend) + Horovod up to 64 nodes for Resnet50 on ImageNet

IBM’s SnapML



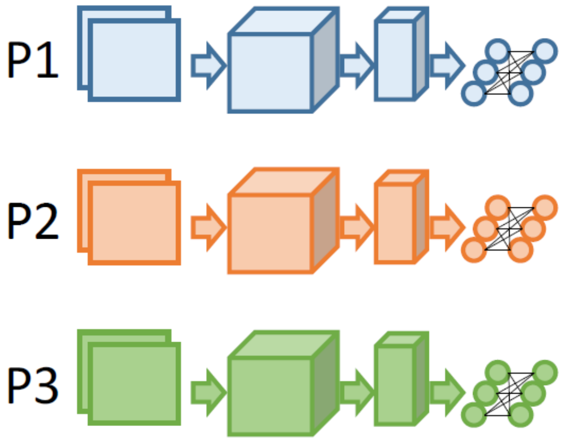
IBM's SnapML

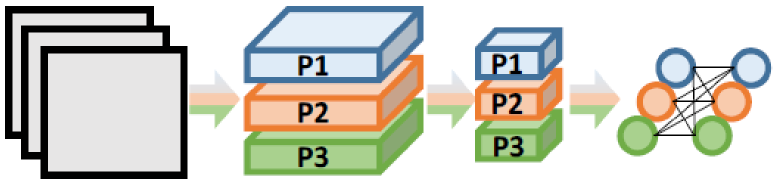


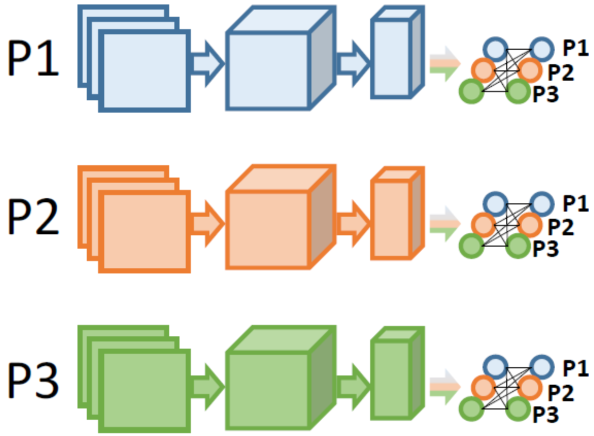
SnapML-SVM (PowerAI 1.5.3)
10
1
0.1
GB/s
1
2
4
8
16
77%
40%
32GB
128GB
ideal
Nodes

Distributed deep learning

- Data parallel
  - **Synchronized**
  - Stale
  - Asynchronized
- Model parallel
- Hybrid

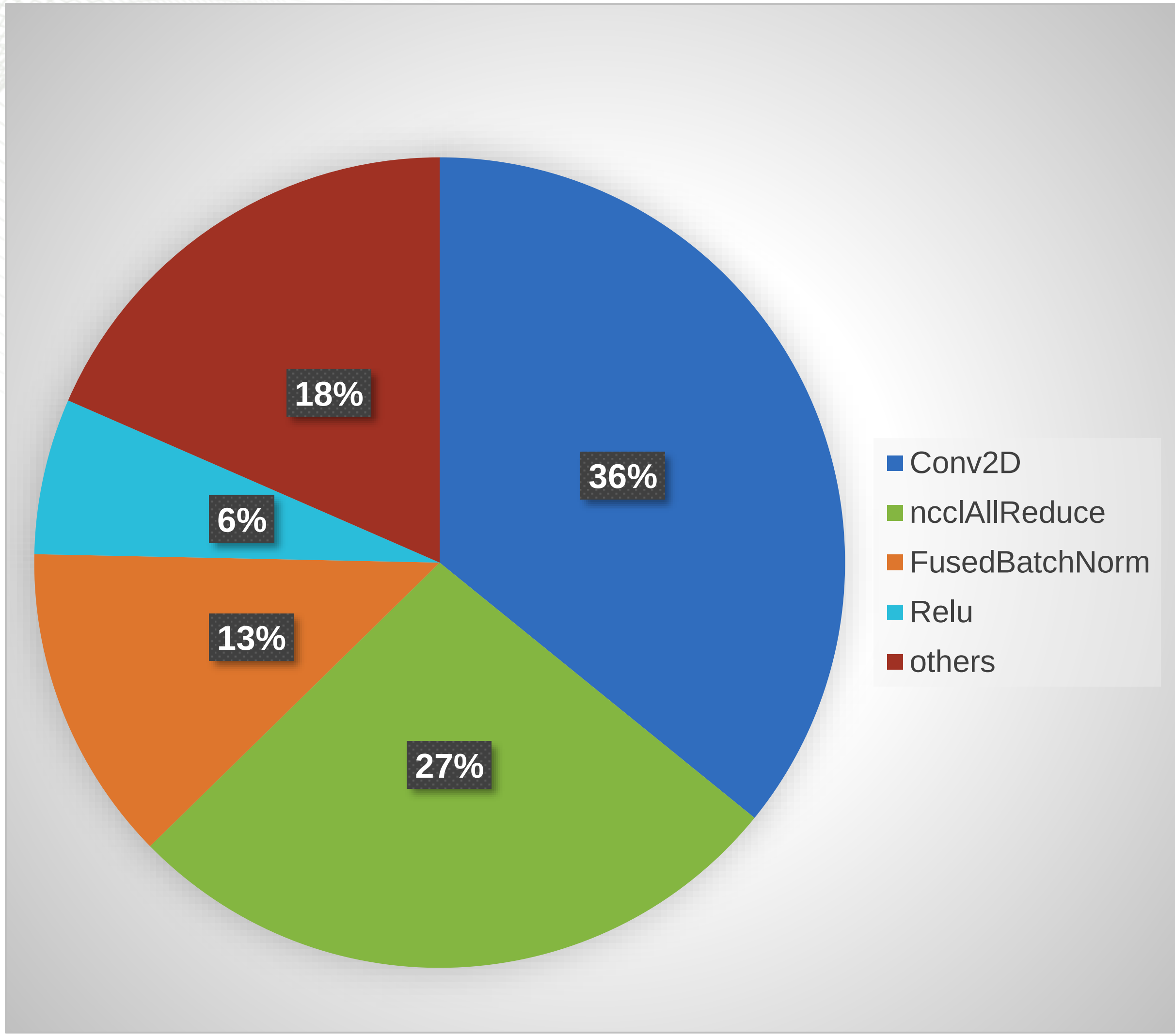


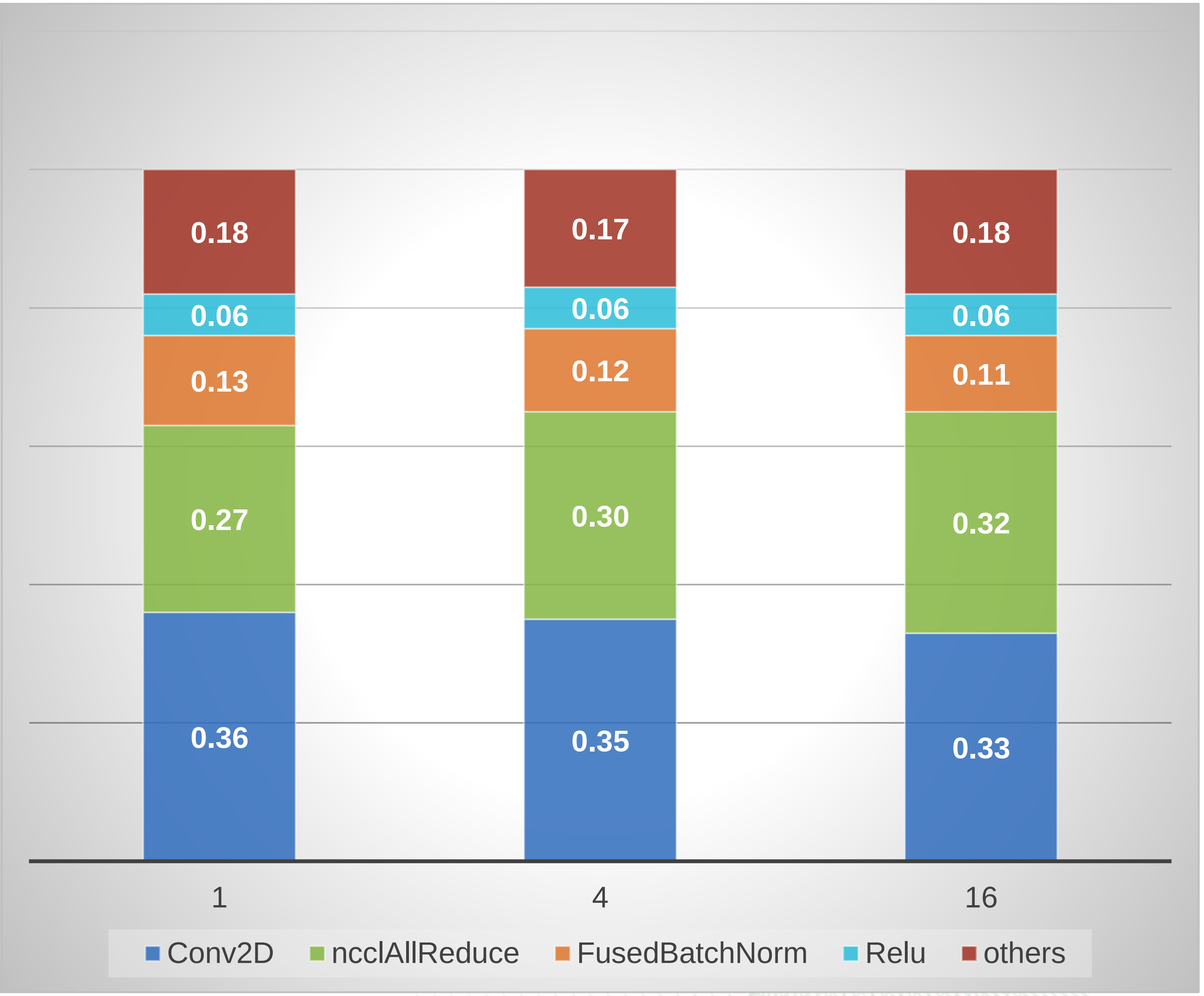




Review: [arXiv:1802.09941]

TensorFlow Resnet50 profiling on Summit

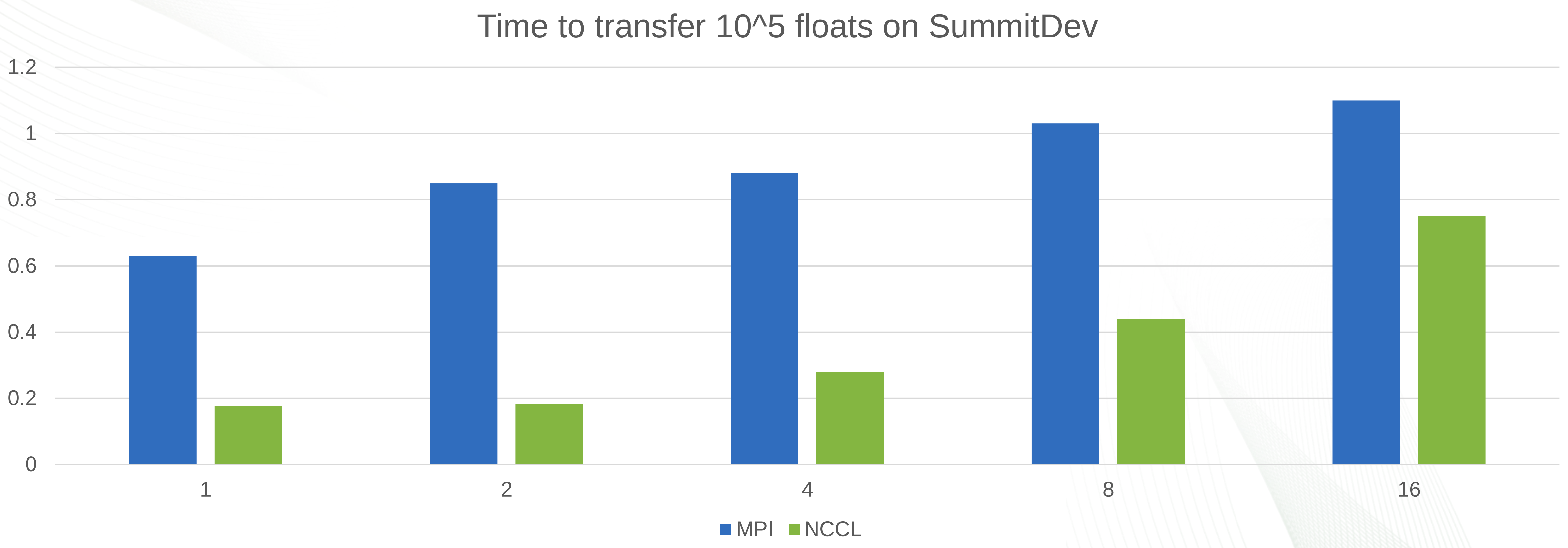




“mini-MPI” for distributed deep learning

- NCCL (Nvidia): collective multi-GPU communication
- Horovod (Uber): Tensorflow and Pytorch support
  - NCCLReduceScatter - MPIAllreduce – NCCLAllgather for data divisible by local_rank()
  - NCCLReduce - MPIAllreduce – NCCLBcast for the remainder
  - Tensor Fusion: fuse small allreduce tensor operations into larger ones for performance gain
  - Compression (cast vars to fp16) before allreduce
- GLOO (Facebook): Pytorch support
- DDL (IBM): Tensorflow, Pytorch, Caffe support. Close source.

NCCL vs MPI allreduce



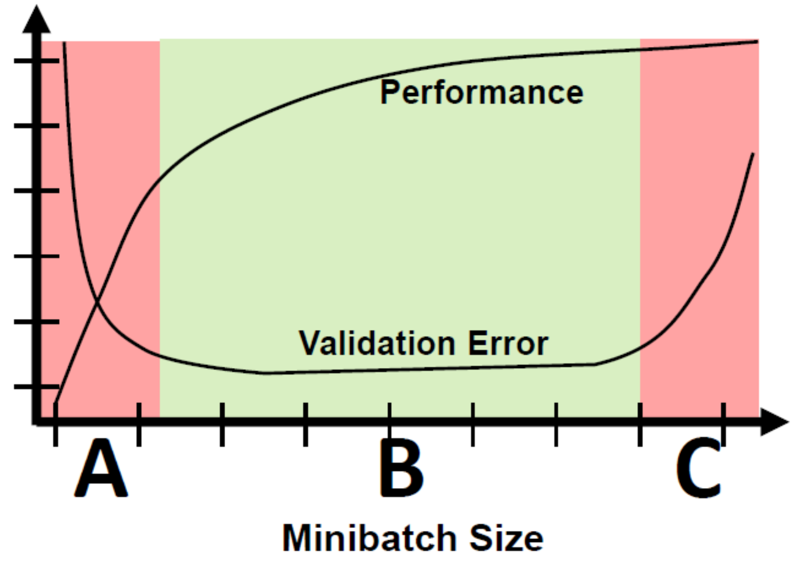
Differences in scaling up: DL VS simulation

- DL is a global optimization, changing scale -> changing solution space.
  - DL usually requires changing network architecture, update scheme, etc

- Scale in OPS $\neq$Scale in time-to-solution (accuracy)
  - Tradeoff between more epochs and faster convergence

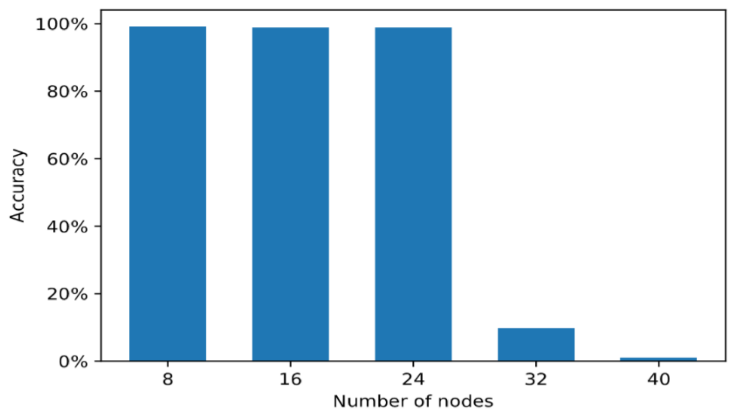
- High per-node OPS makes DL comm- and/or IO- bound at relatively small node count.
  - DL requires special designed comm (mainly all-reduce) and IO pipeline

Synchronized data parallel: scaling vs convergence

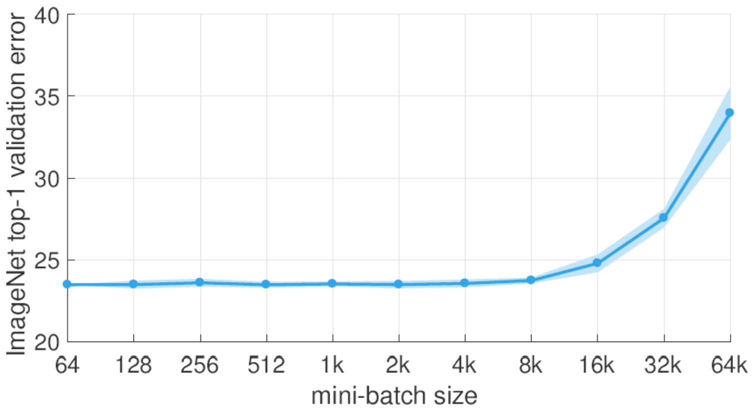


(a) Minibatch Effect on Accuracy and Performance (Illustration)

- Possible causes: **"generalization gap"** (Keskar et al. 2017)
  - loss of the explorative properties
  - tend to converge to sharp minimizers
  - model overfits the training data



Convergence of MNIST with increasing mini-batch size



(b) Empirical Accuracy (ResNet-50, figure adapted from [Goyal et al. 2017], lower is better)

Large mini-batch size training

- mini-batch size 8K (arXiv:1706.02677)
  - Warmup with default learning rate for optimizer
  - Start with learning rate multiplying # of workers
  - Decay learning rate periodically

- mini-batch size 32K
  - Layer-wise adaptive rate scaling (LARS) (arXiv:1711.04325)

State-of-the-art Imagenet training

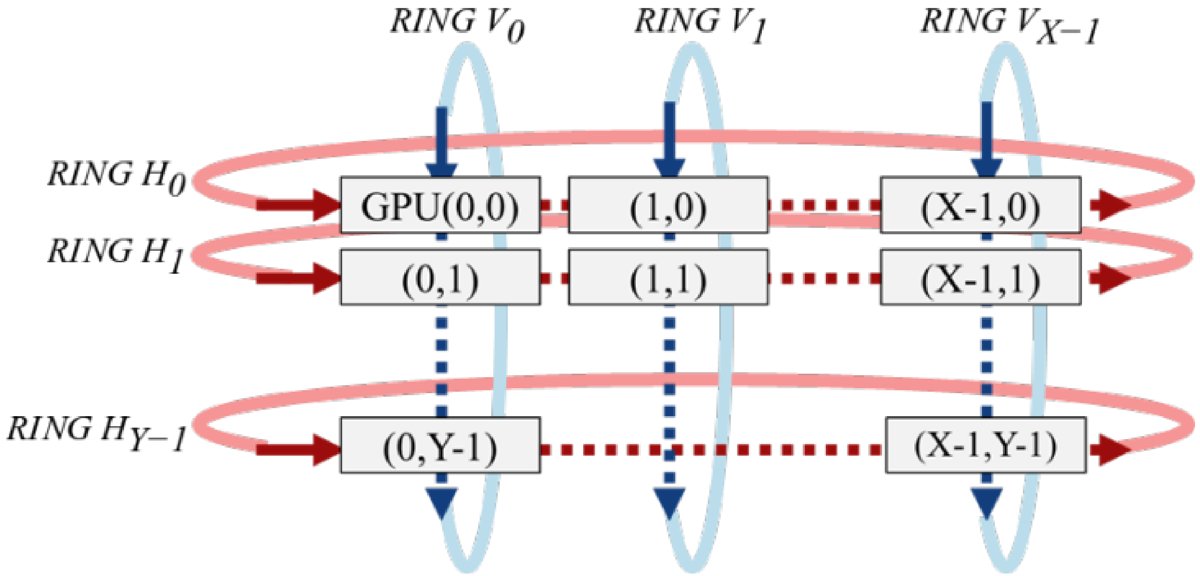
Chronology of Distributed Deep Learning Records

Table 1 : Training time and top-1 1-crop validation accuracy with ImageNet/ResNet-50

		Batch Size	Processor	DL Library	Time	Accuracy
2016	He et al.	256	Tesla P100 x8	Caffe	29 hours	75.3%
2017	Goyal et al.	8K	Tesla P100 x256	Caffe2	1 hour	76.3%
2017	Smith et al.	8K→16K	full TPU Pod	TensorFlow	30 mins	76.1%
2017	Akiba et al.	32K	Tesla P100 x1024	Chainer	15 mins	74.9%
2018	Jia et al.	64K	Tesla P40 x2048	TensorFlow	6.6 mins	75.8%
2018	Mikami et al.	**34K→68K**	**Tesla V100 x2176**	**NNL**	**224 secs**	**75.03%**

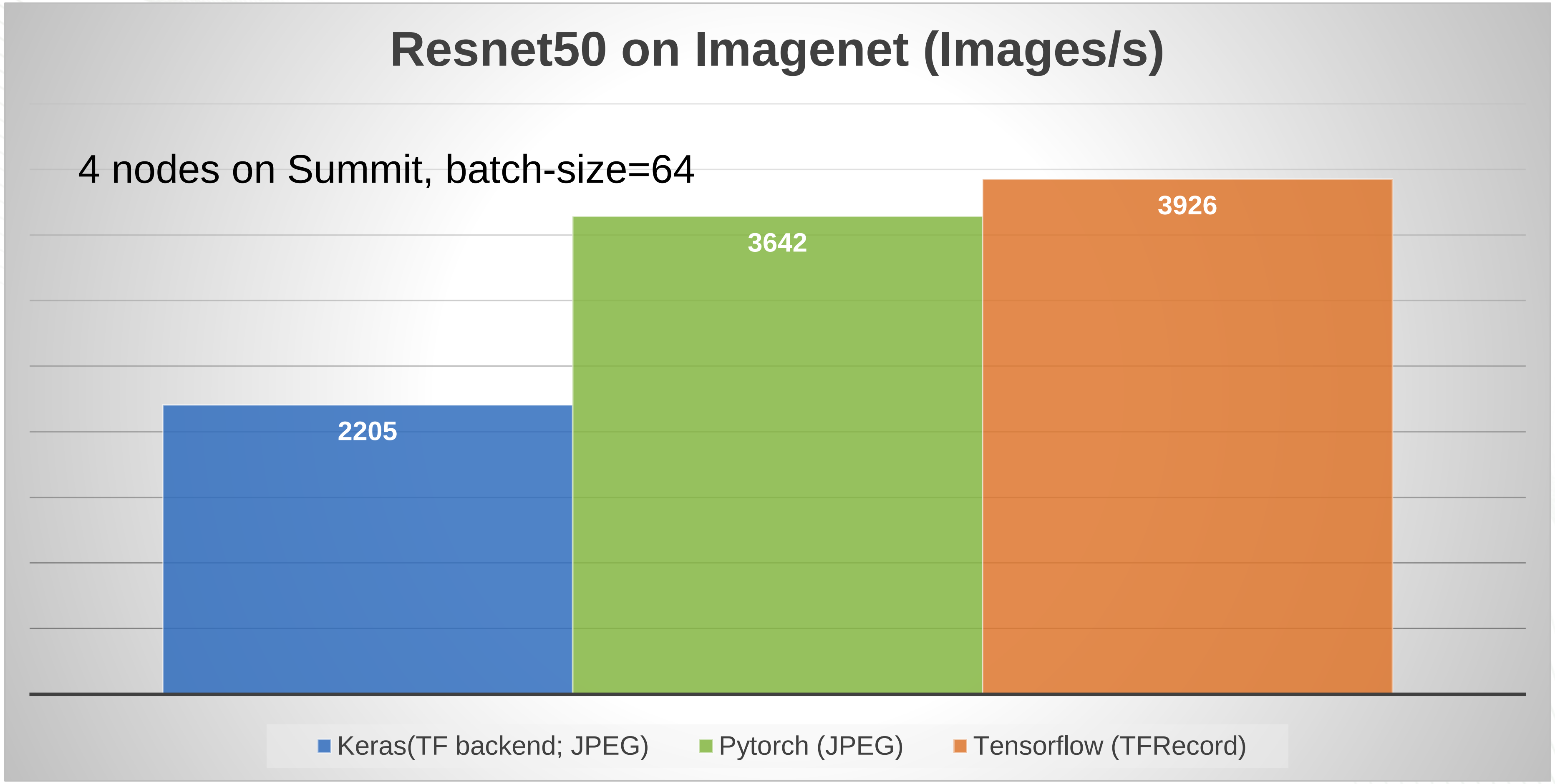
State-of-the-art Imagenet training (arXiv:1811.05233)

- Batch size control + LARS -> 68K mini-batch size
- 2D-Torus All-reduce communication

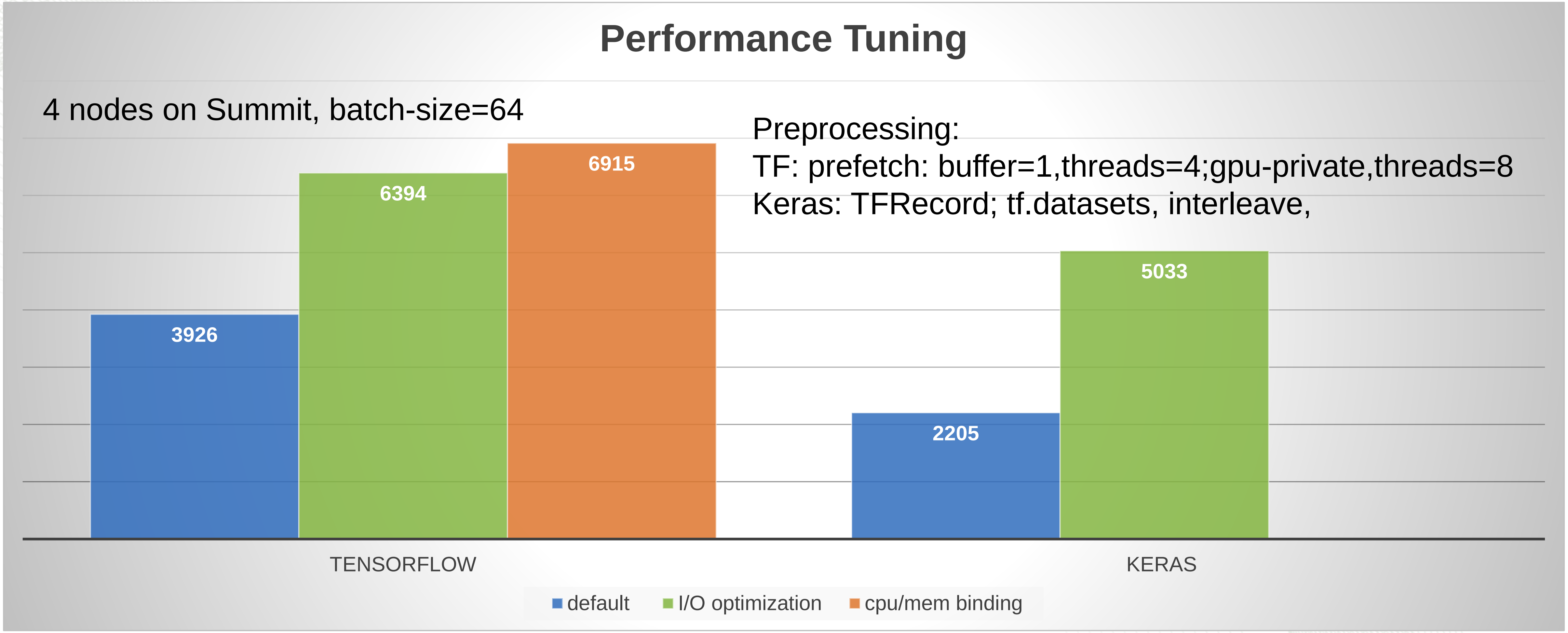


- 224s training -> 75.03% top1 accuracy and 66% scaling efficiency on 2176 V100.

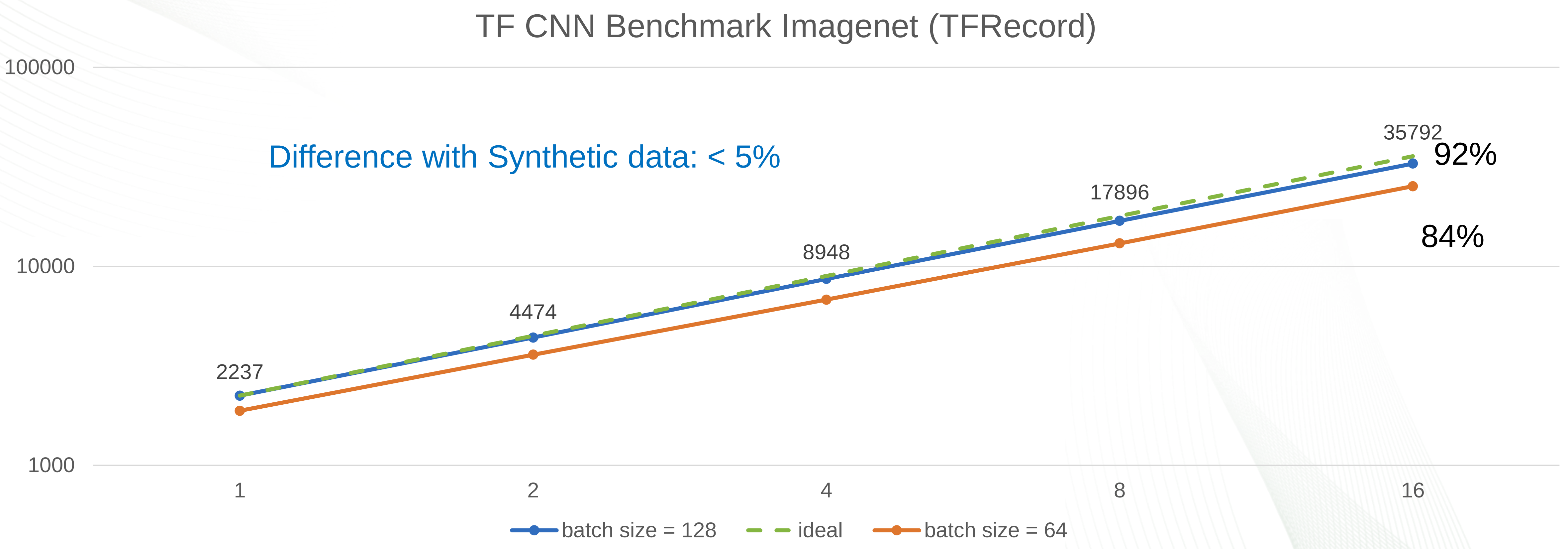
Without tuning



Tuning of Tensorflow and Keras

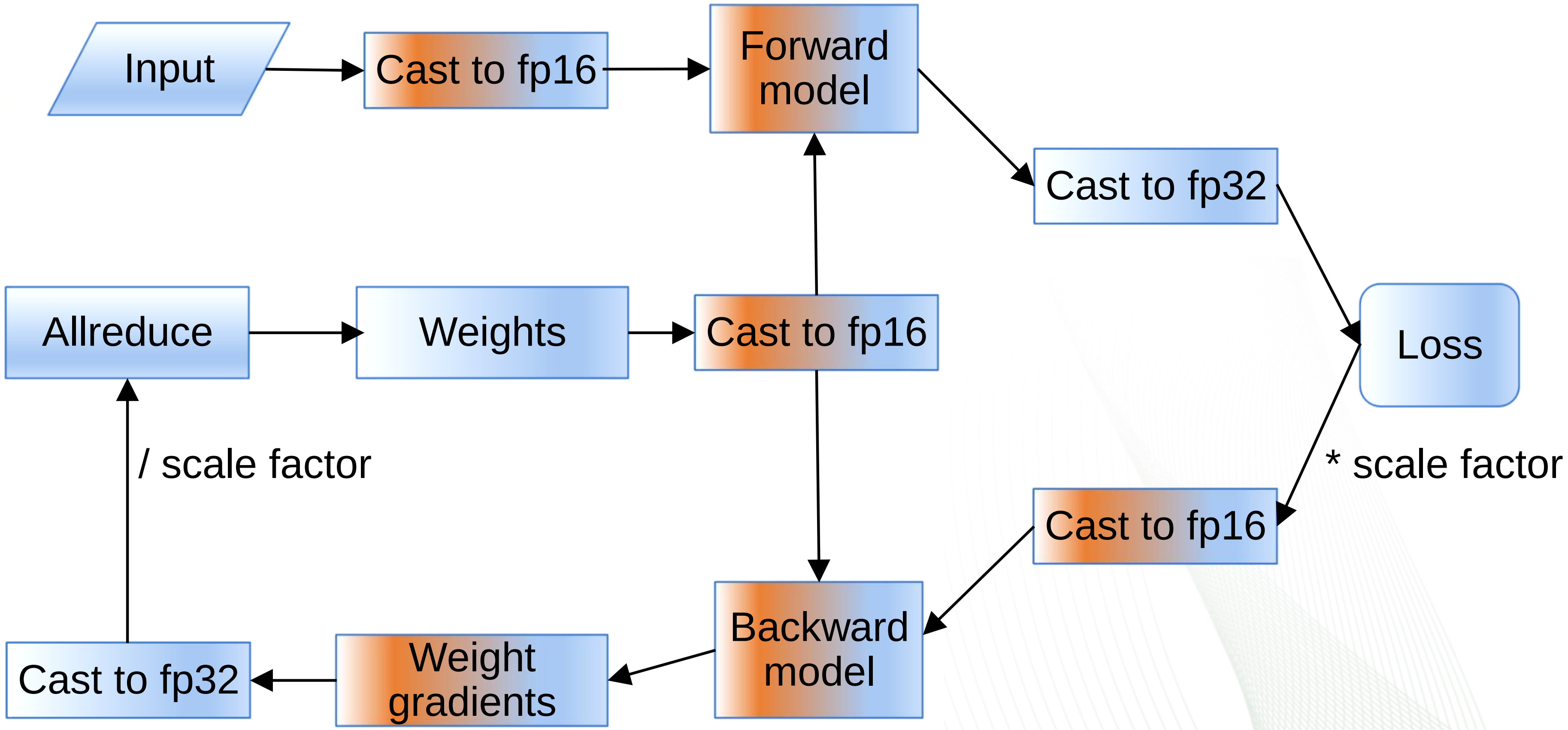


TF benchmark on Summit



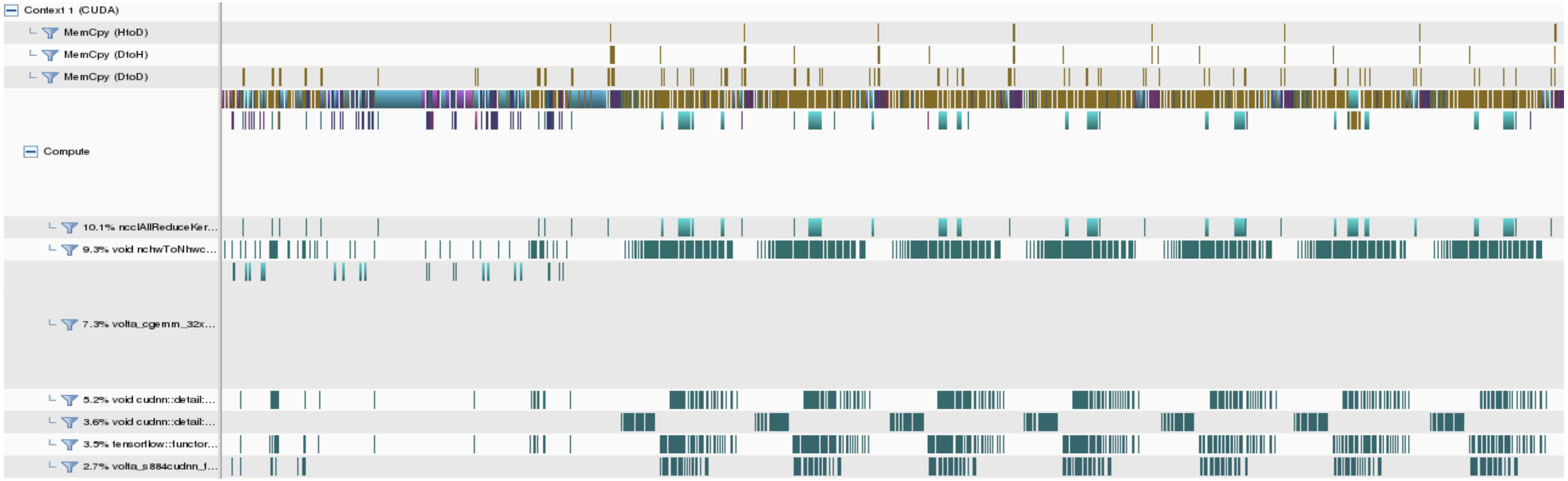
Mixed precision & Tensorcore

- Consideration
  - Imprecise weights
  - Gradients underflow
  - Reduction overflow
- Verification
  - s884cudnn

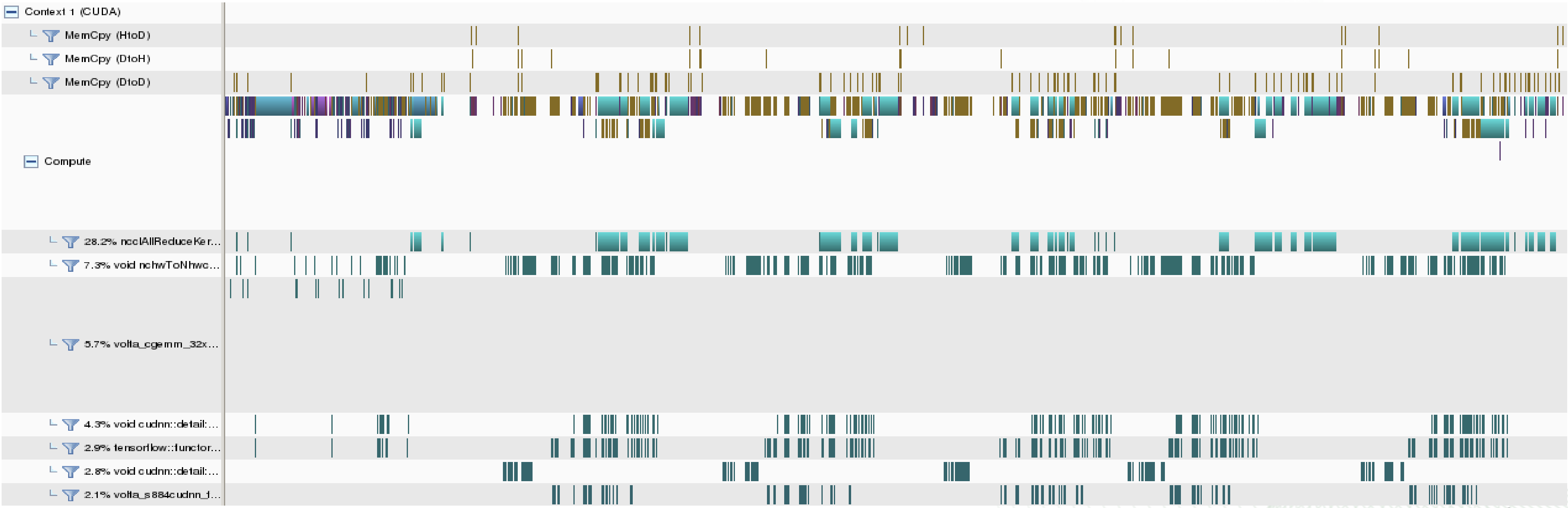


NVProf

Synthetic Data



TFRecord



Lessons learned from Exa-scale DL on Summit (arXiv:1810.01993)

- Data ingestion (mostly coincide with TF performance guide)
  - Input pipeline, queueing input for compute
  - Concurrent processing with map
- Communication
  - Broadcast tree
    - Hierarchical aggregation of the control message (the order of tensors to be reduced)
  - Hybrid NCCL-MPI allreduce
    - NCCL intra node allreduce
    - 4 ranks (2 on each socket, b/c 4 IB devices) per node each MPI_Allreduce on a quarter of the data
    - NCCL intra node broadcast

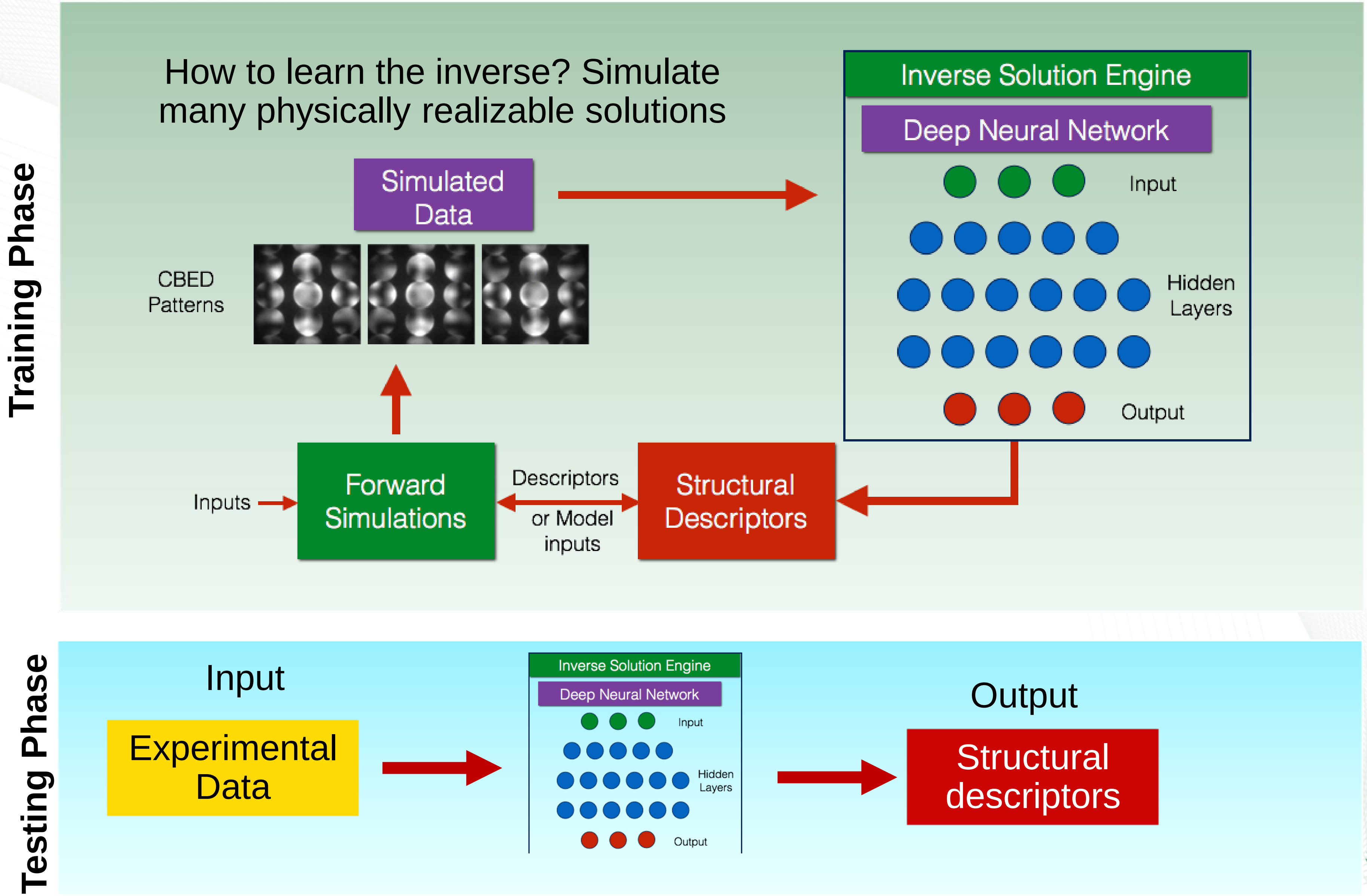
Lessons learned from Exa-scale DL on Summit (arXiv:1810.01993)

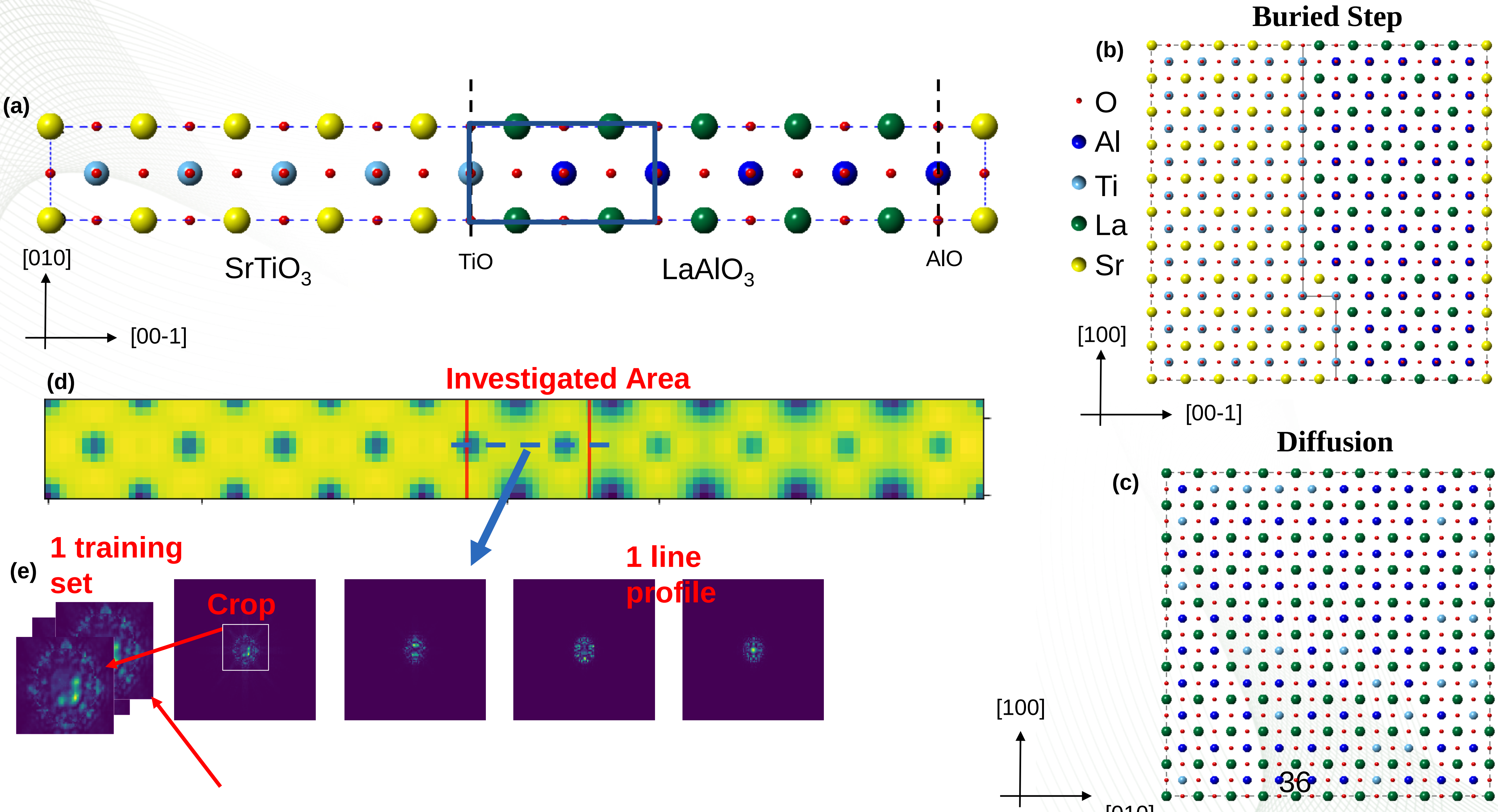
- Algorithmic considerations
  - Weighted loss, i.e. each pixel contributes differently to the loss function, specific to application (background vs area of interest)
  - LARC, a variant on LARS, for large batch sizes.
  - Multi-channel (16), more compute, more accurate
  - Gradient lag, overlap communication and computation
  - Network, larger layer, less number of layers, to improve compute intensity.

DL vs conventional ML

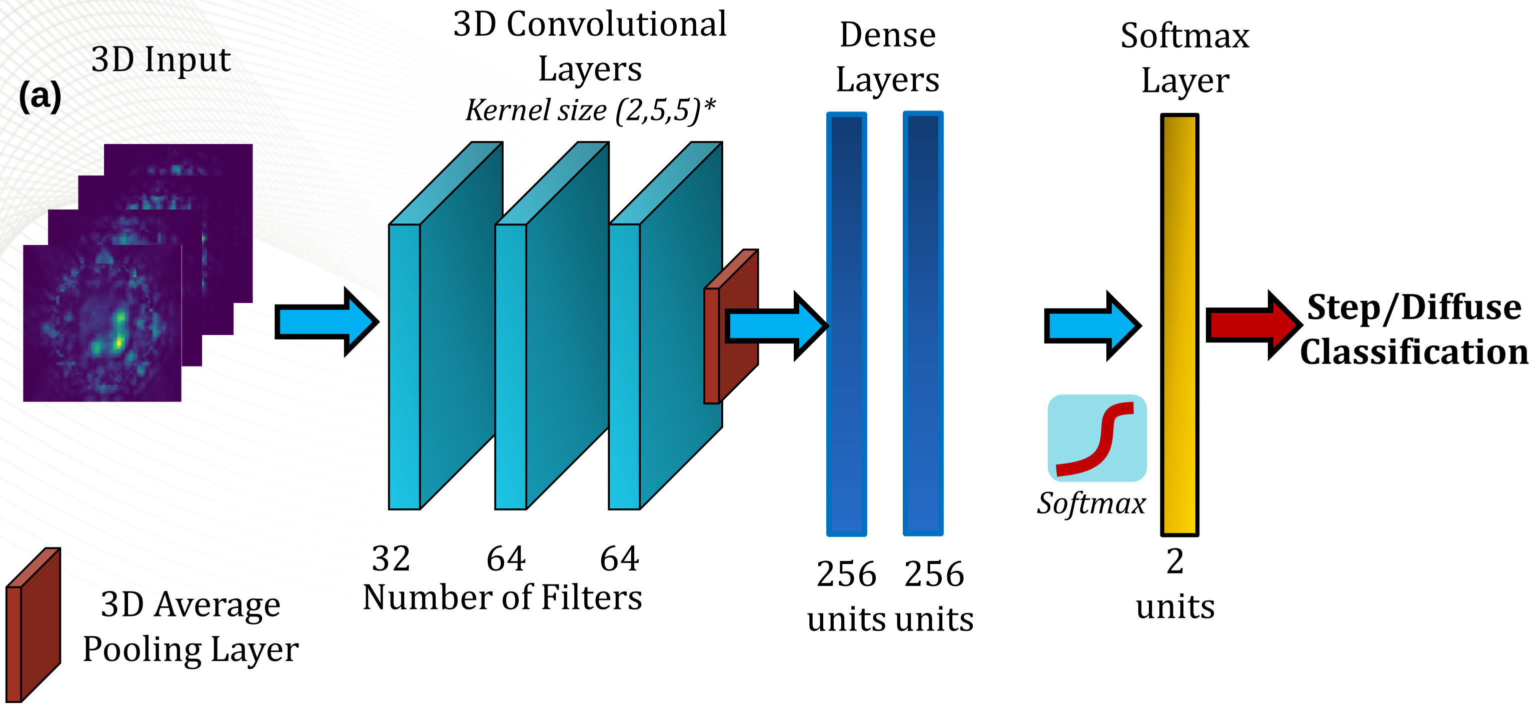
- It depends.
  - In general, DL works better for unstructured features, e.g. images, text; gradient boosting works better for data with structured ones, e.g. tabulated data; feature selection + gaussian process (equivalent infinite width neural network) works better for limited data and explainability.
- Explored in several use cases.
  - Simulation energy prediction
  - Material design (High entropy alloy)
  - Climate surrogate modelling
  - Microscopic images classification

Backup slides: Use Case 1 (LDRD PI: Rama Vasudevan)

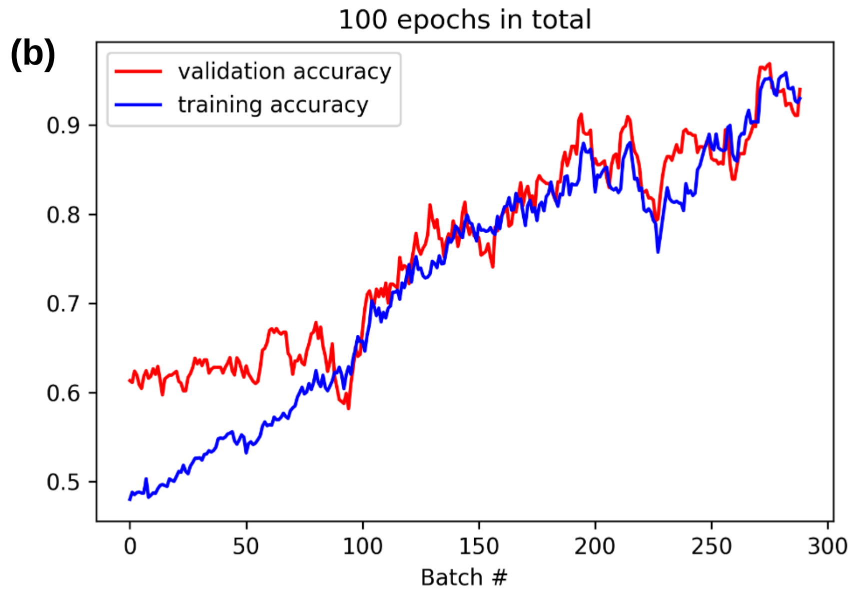



Buried Step
(a)
(b)
O
Al
Ti
La
Sr
[010]
[00-1]
SrTiO3
TiO
LaAlO3
AlO
[100]
[00-1]
(d)
Investigated Area
Diffusion
(c)
(e)
1 training set
Crop
1 line profile
[100]
[010]

(a)



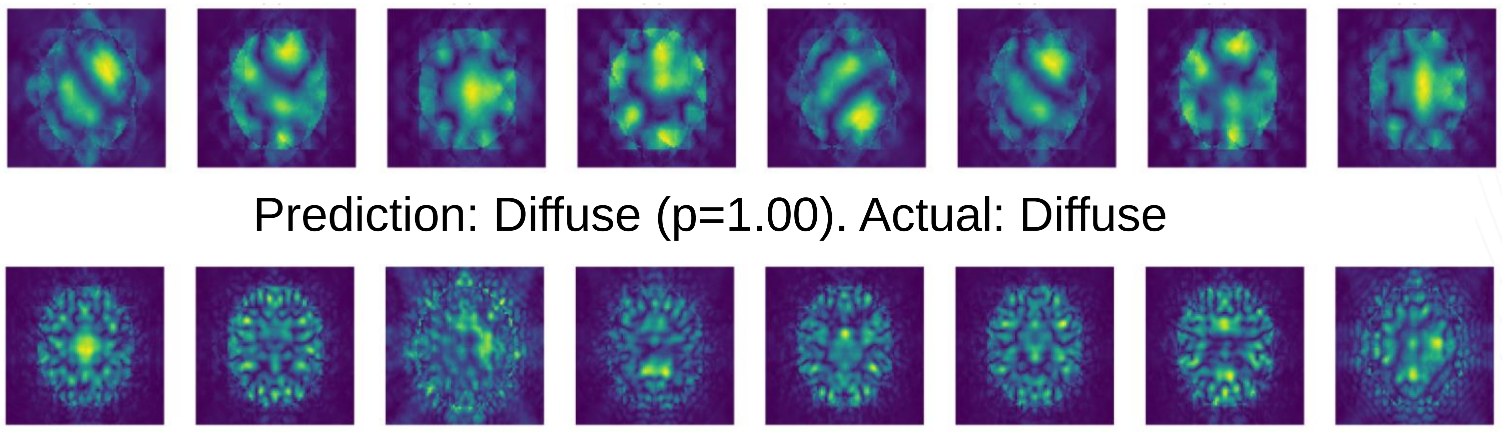
(b)



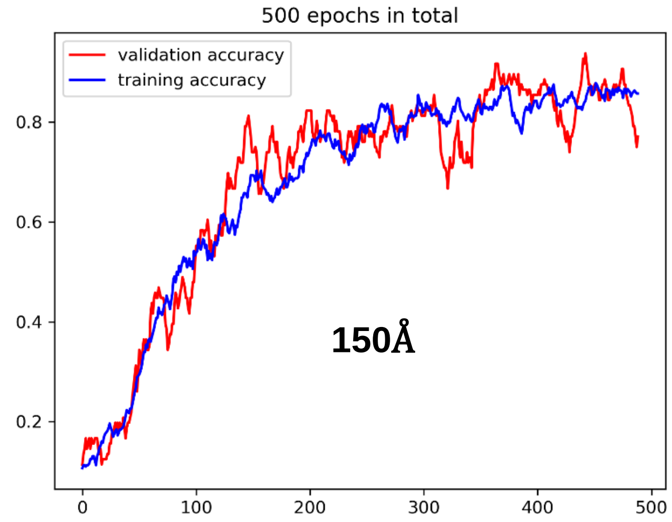
(c) **Along interface**

Prediction: Step (p=0.96). Actual: Step

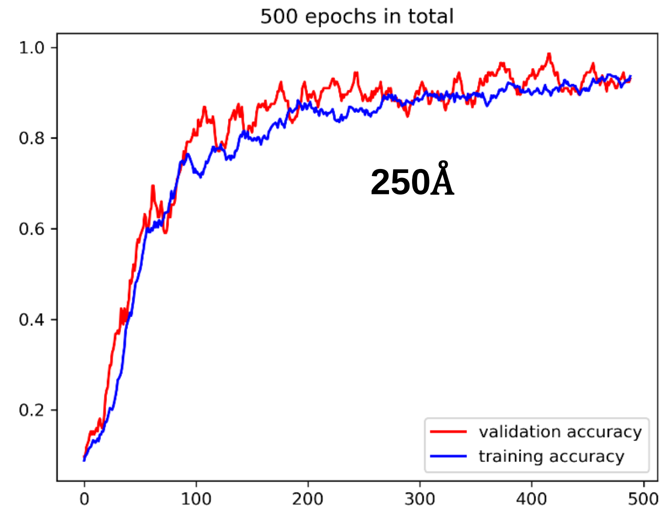
Prediction: Diffuse (p=1.00). Actual: Diffuse



(a)

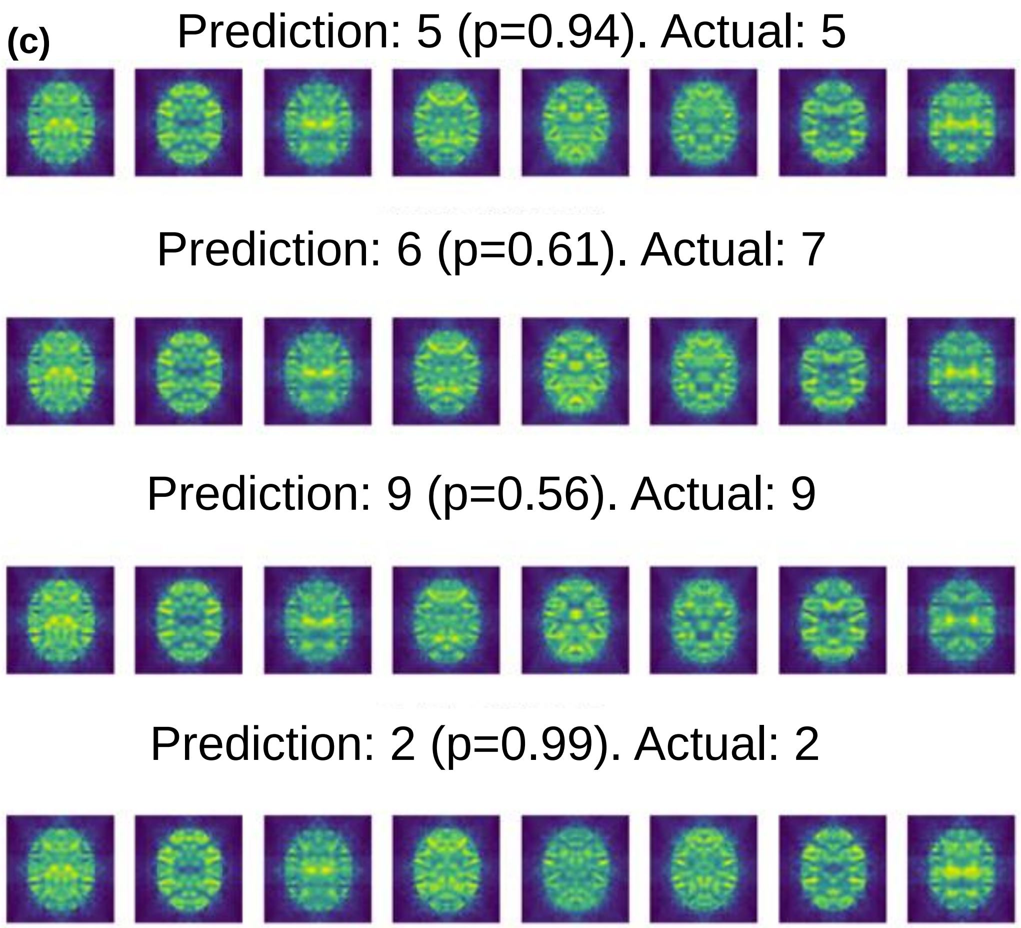


(b)

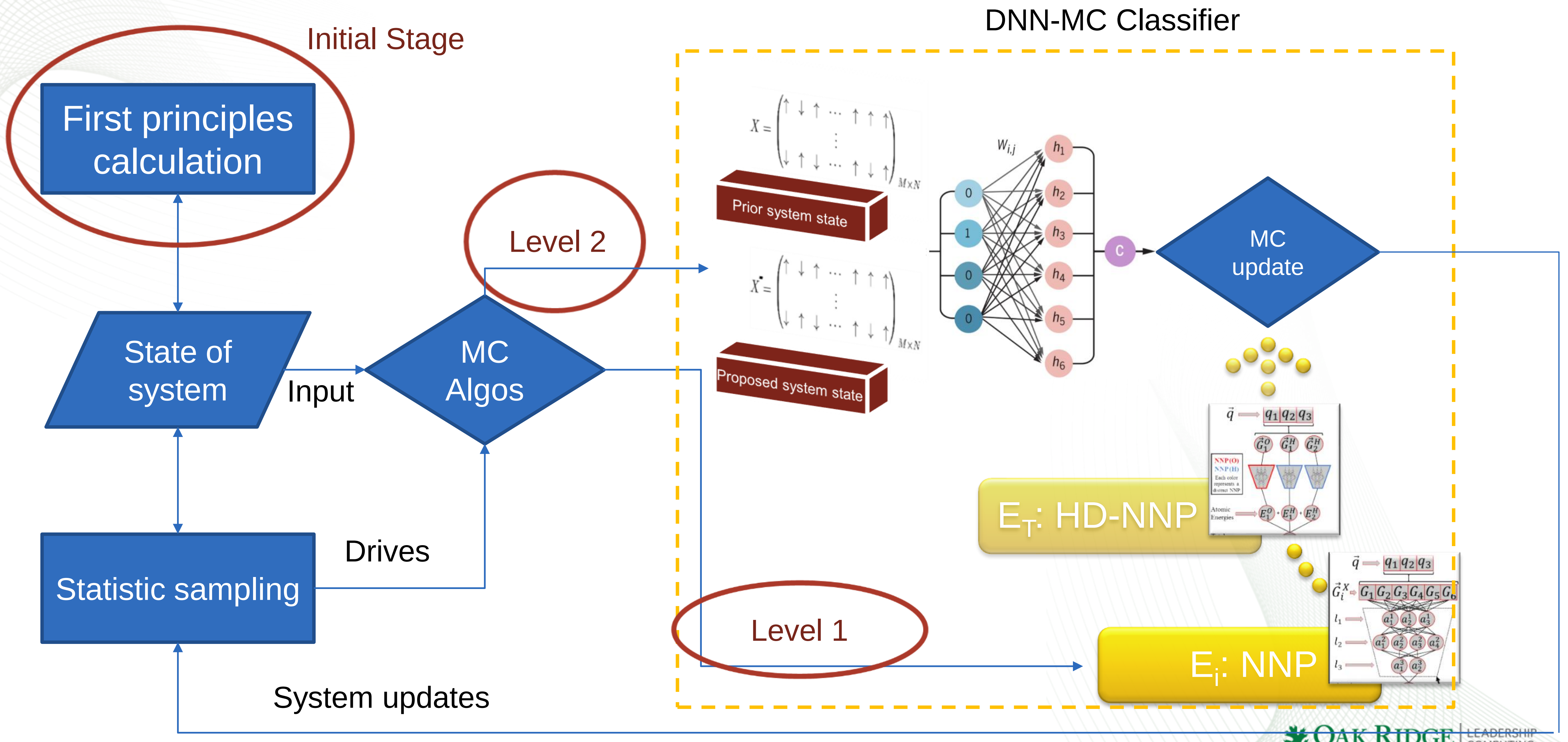


Thickness: 250Å

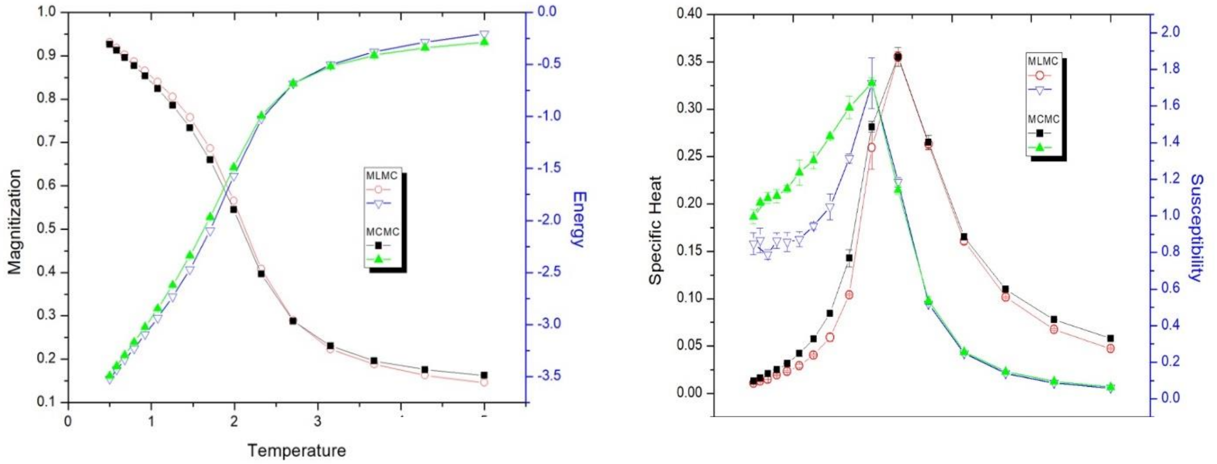
(c)



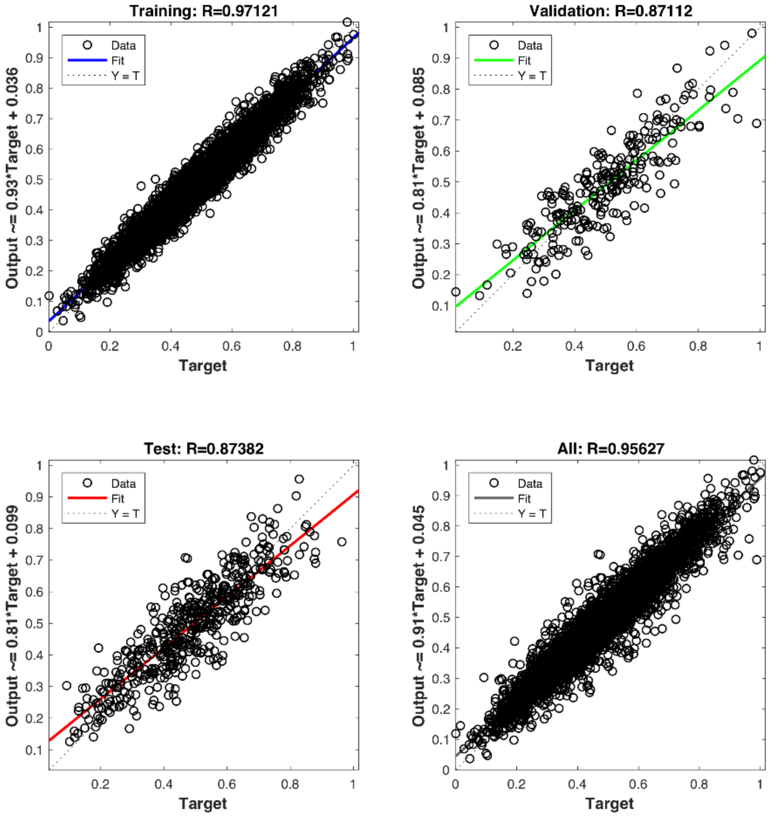
Use Case 2 (LDRD PI: Markus Eisenbach)



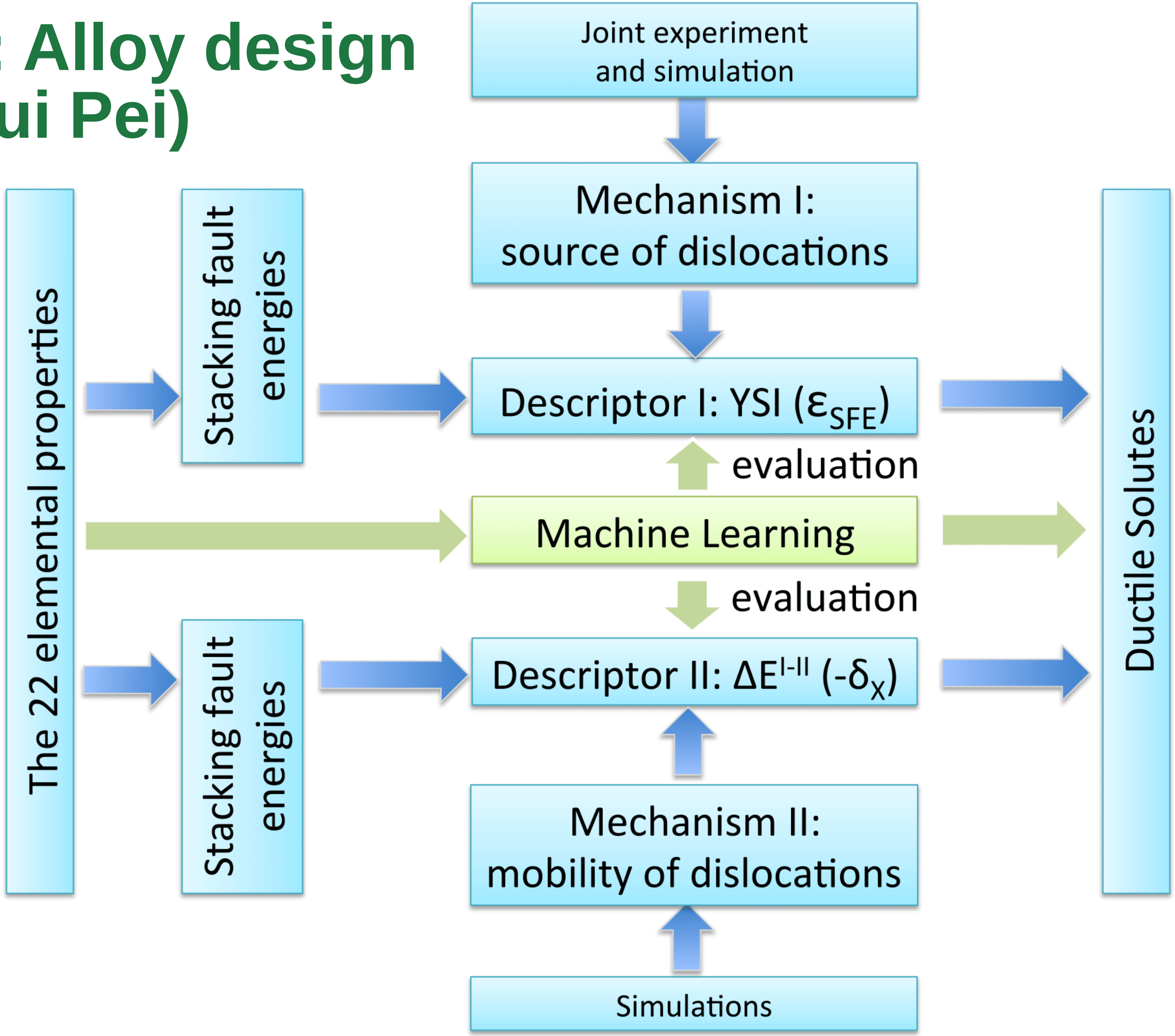
- Proof of concept for an online model (Heisenberg)
- Offline models for complex systems (Water cluster, FeCo alloy)
- Exploration of sampling algorithms (Metropolis, Wang-Landau, Nested Sampling)

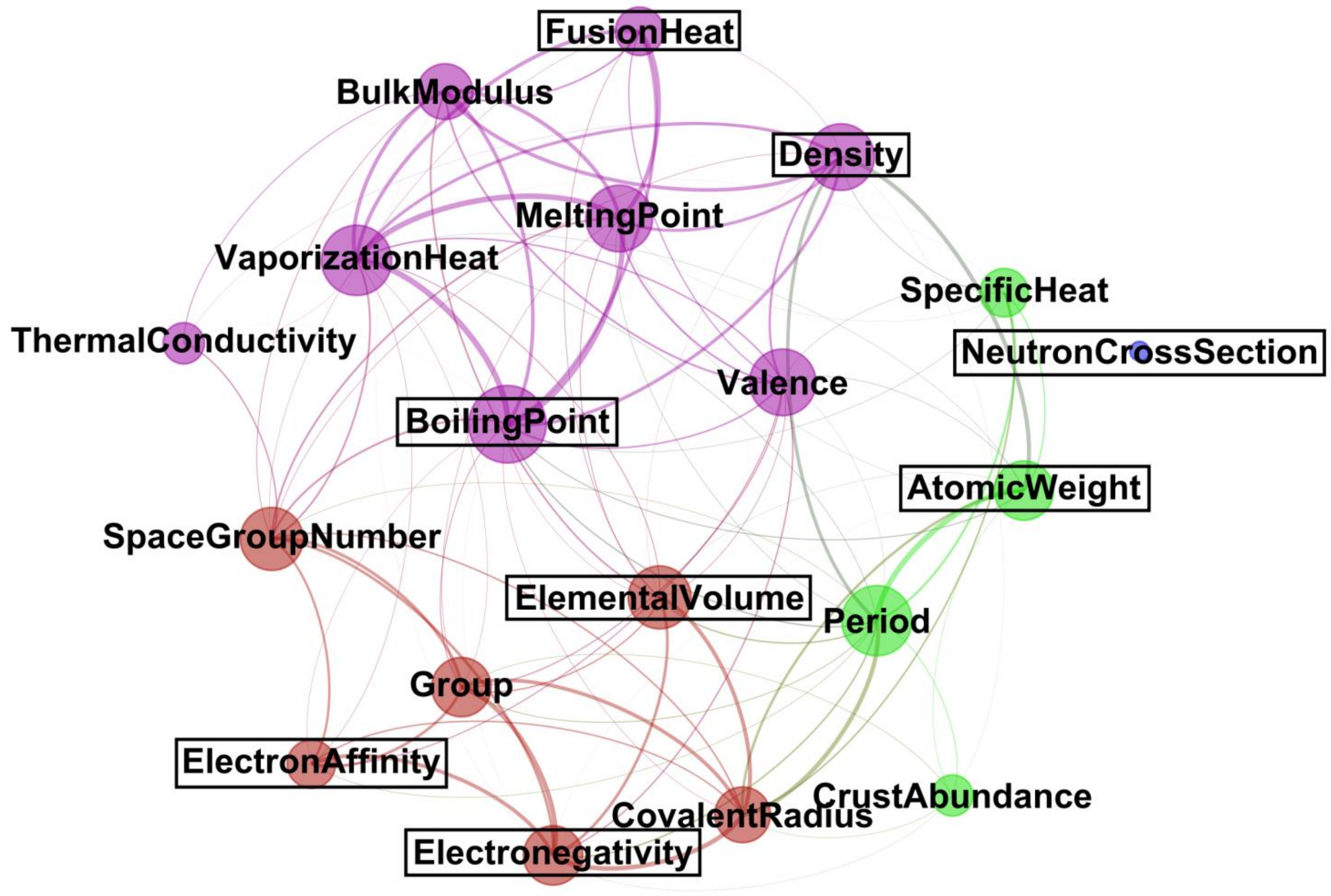


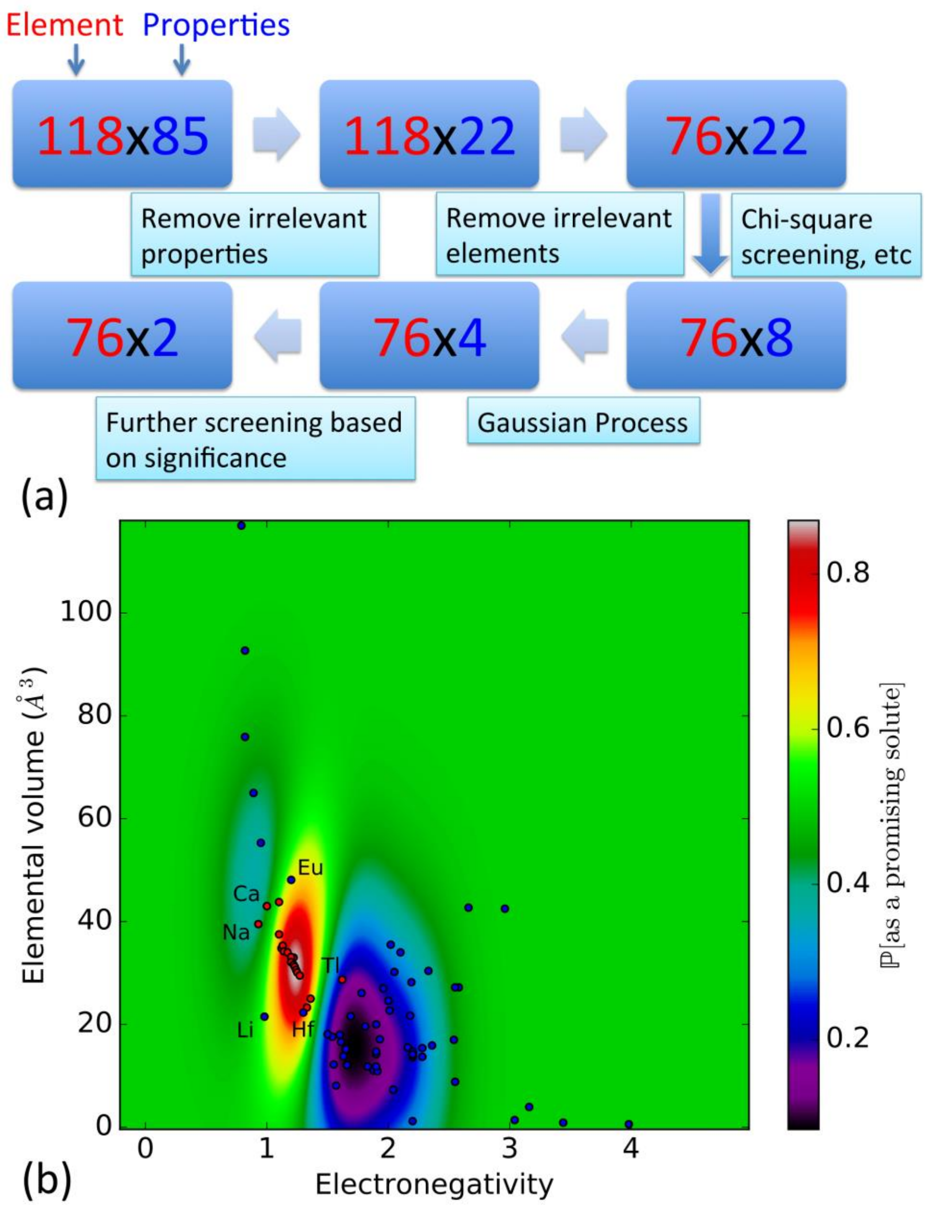
System	Sampling algorithms	Model	Accuracy
Heisenberg	Metropolis Nested Sampling	XGBoost | DNN	87%
Water cluster	Wang-Landau	XGBoost | DNN	91%
FeCo alloy	Metropolis	DNN	87%

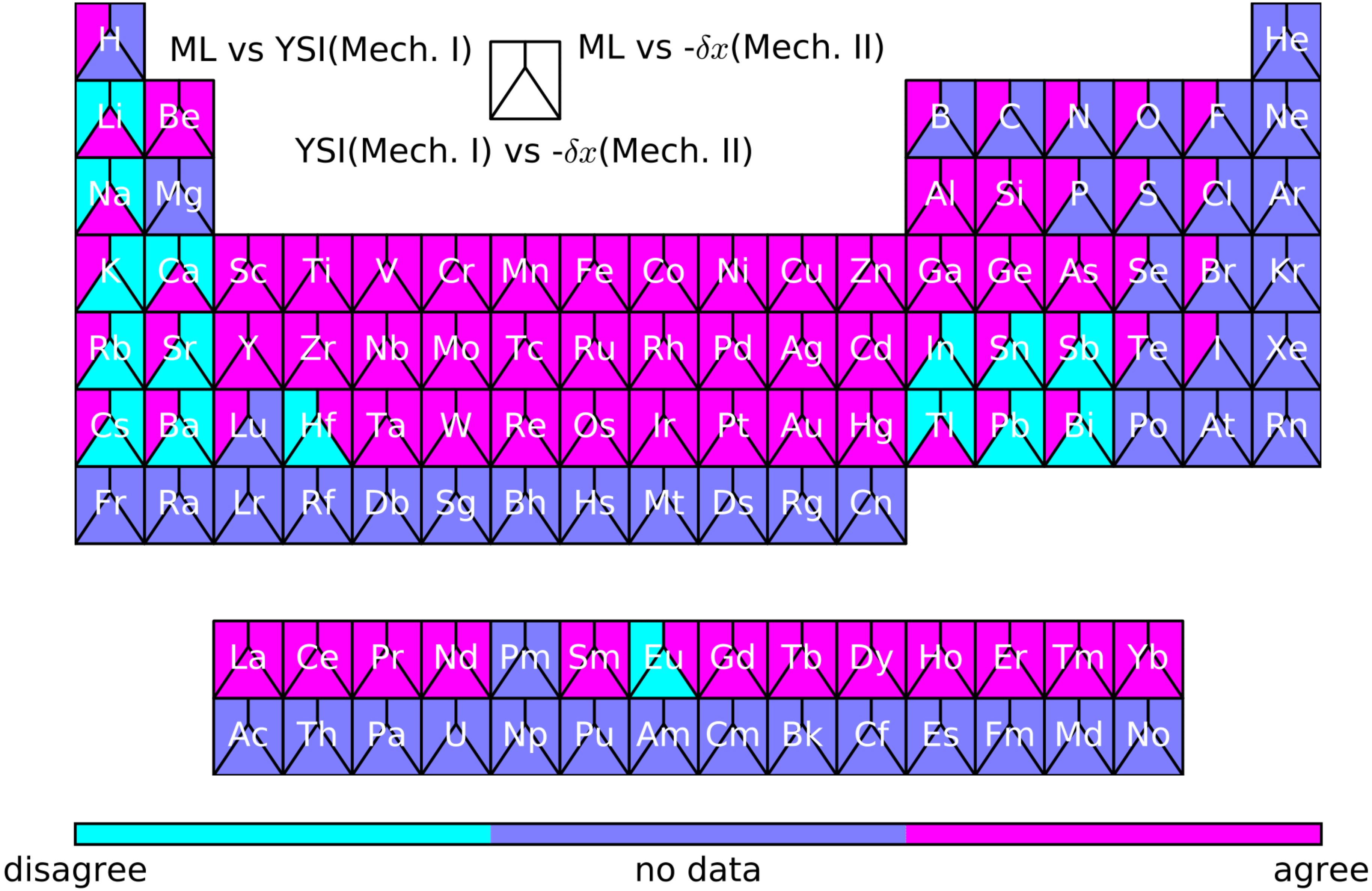


Use Case 3: Alloy design (with Zongrui Pei)




FusionHeat
BulkModulus
Density
MeltingPoint
VaporizationHeat
SpecificHeat
ThermalConductivity
NeutronCrossSection
Valence
BoilingPoint
AtomicWeight
SpaceGroupNumber
ElementalVolume
Period
Group
ElectronAffinity
CrustAbundance
CovalentRadius
Electronegativity


Element
Properties
118x85
118x22
76x22
Remove irrelevant properties
Remove irrelevant elements
Chi-square screening, etc
76x2
76x4
76x8
Further screening based on significance
Gaussian Process
(a)
Elemental volume ($\AA^3$)
Electronegativity
ℙ[as a promising solute]
Eu
Ca
Na
Tl
Li
Hf
(b)


ML vs YSI(Mech. I)
ML vs -δx(Mech. II)
YSI(Mech. I) vs -δx(Mech. II)
H He
Li Be B C N O F Ne
Na Mg Al Si P S Cl Ar
K Ca Sc Ti V Cr Mn Fe Co Ni Cu Zn Ga Ge As Se Br Kr
Rb Sr Y Zr Nb Mo Tc Ru Rh Pd Ag Cd In Sn Sb Te I Xe
Cs Ba Lu Hf Ta W Re Os Ir Pt Au Hg Tl Pb Bi Po At Rn
Fr Ra Lr Rf Db Sg Bh Hs Mt Ds Rg Cn
La Ce Pr Nd Pm Sm Eu Gd Tb Dy Ho Er Tm Yb
Ac Th Pa U Np Pu Am Cm Bk Cf Es Fm Md No
disagree
no data
agree

Questions?