MVAPICH
MPI, PGAS and Hybrid MPI+PGAS Library

Follow us on
https://twitter.com/mvapich

HiDL
High-Performance Deep Learning

High Performance Distributed Deep Learning

Tutorial at ISC '21

by

Dhabaleswar K. (DK) Panda
The Ohio State University
E-mail: panda@cse.ohio-state.edu
http://www.cse.ohio-state.edu/~panda

Hari Subramoni
The Ohio State University
E-mail: subramon@cse.ohio-state.edu
http://www.cse.ohio-state.edu/~subramon

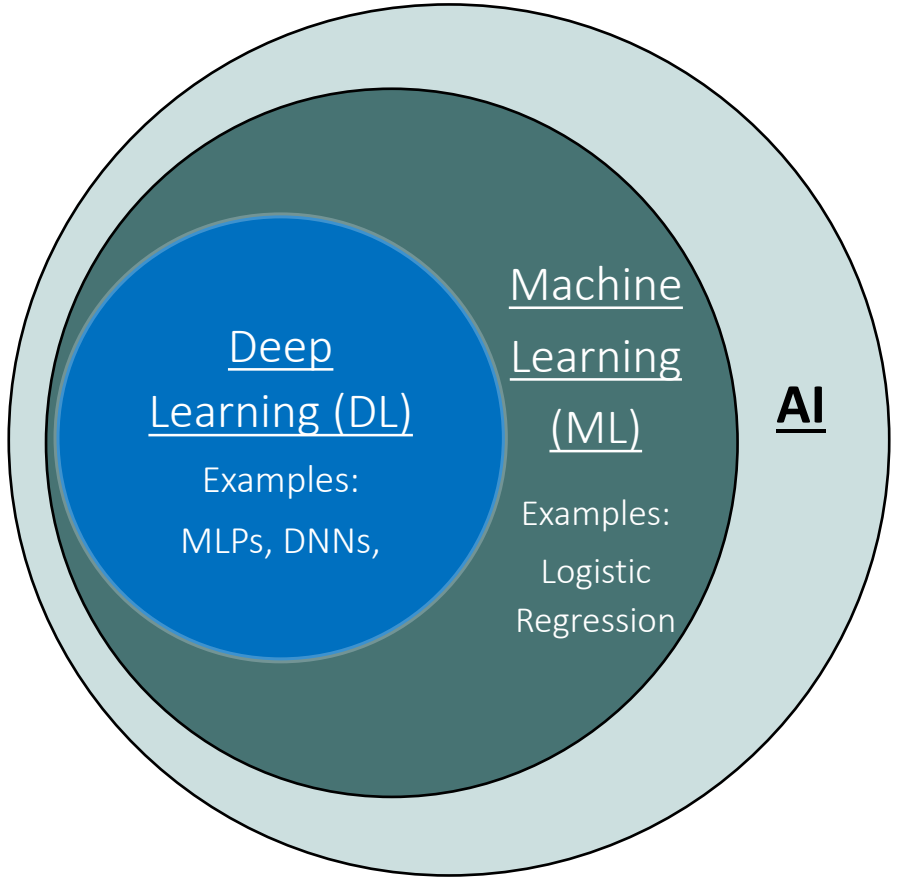
Arpan Jain
The Ohio State University
E-mail: jain.575@osu.edu
http://u.osu.edu/jain.575

Outline

- **Introduction**
  - **The Past, Present, and Future of Deep Learning**
  - What are Deep Neural Networks?
  - Diverse Applications of Deep Learning
  - Deep Learning Frameworks
- Overview of Execution Environments
- Parallel and Distributed DNN Training
- Latest Trends in High-Performance Computing Architectures
- Challenges in Exploiting HPC Technologies for Deep Learning
- Solutions and Case Studies
- Open Issues and Challenges
- Hands-on Exercises
- Conclusion

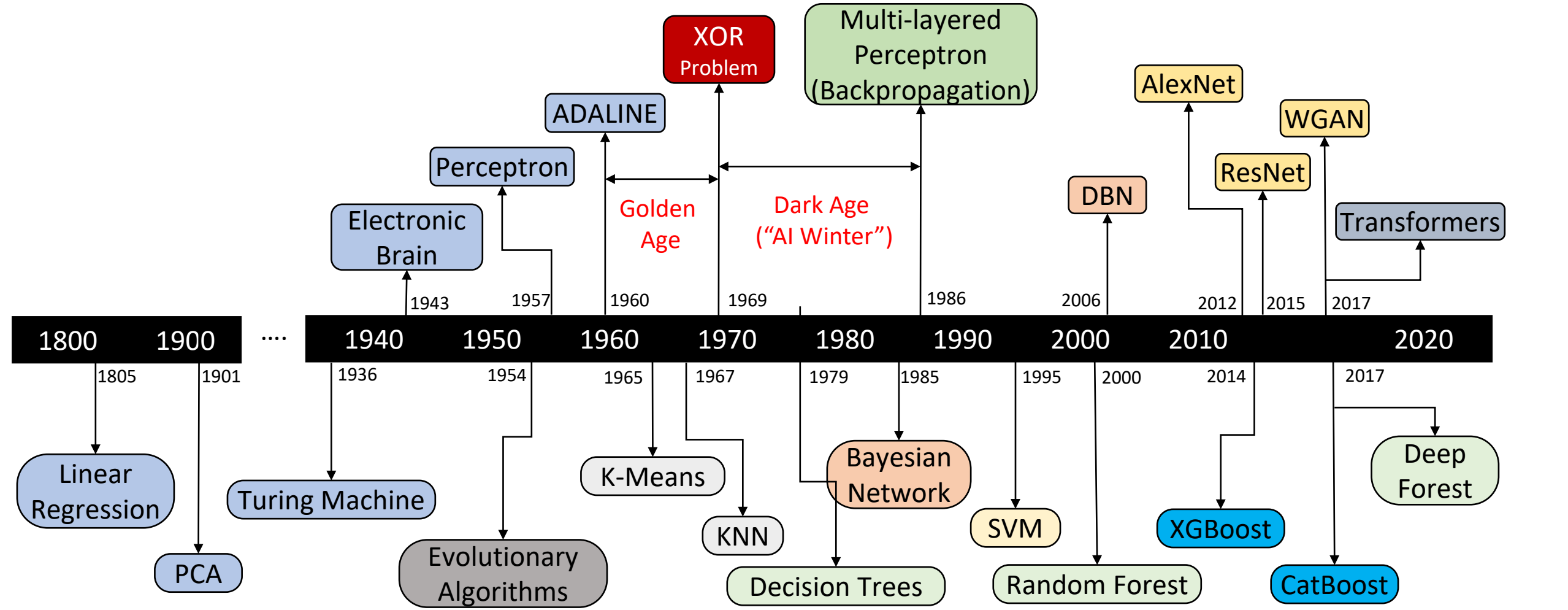
The Artificial Intelligence Revolution

- **Artificial Intelligence (AI) is the science of training machines to perform human tasks.**
  - "Human intelligence exhibited by machines"
- Common use cases
  - Object recognition
  - Speech recognition / sound detection
  - Natural Language Processing
  - Health Systems
  - Marketing / Advertisements
- Learning by example / pattern
  - Machine Learning
  - Deep Learning



Adopted from:
http://www.deeplearningbook.org/contents/intro.html

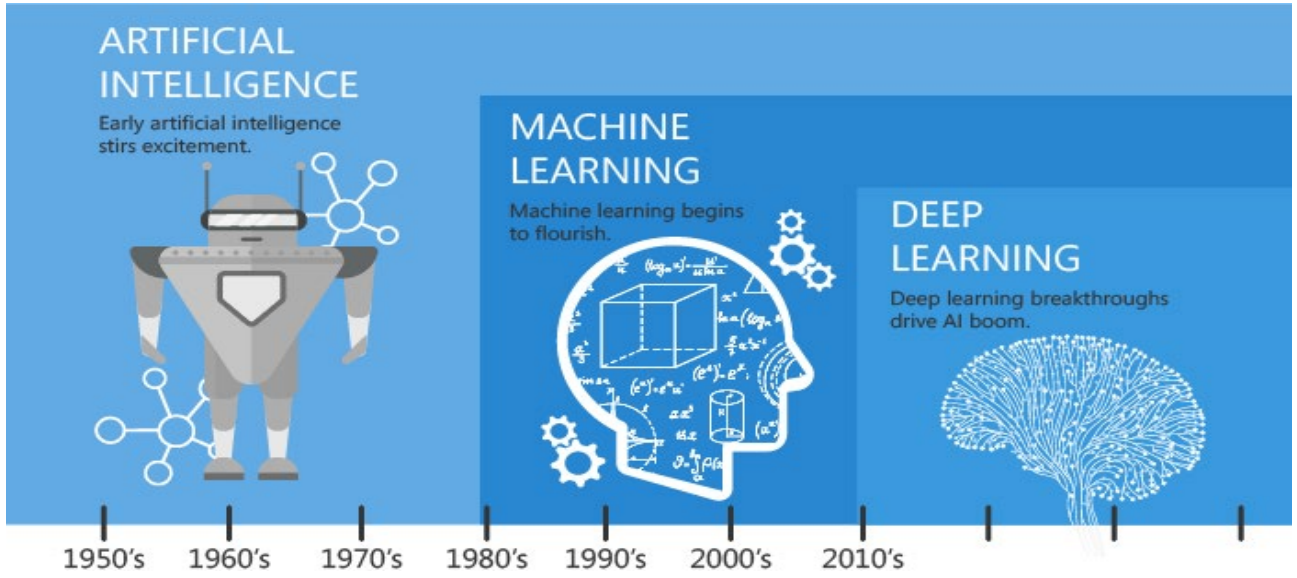
History: Milestones in the Development of Neural Networks

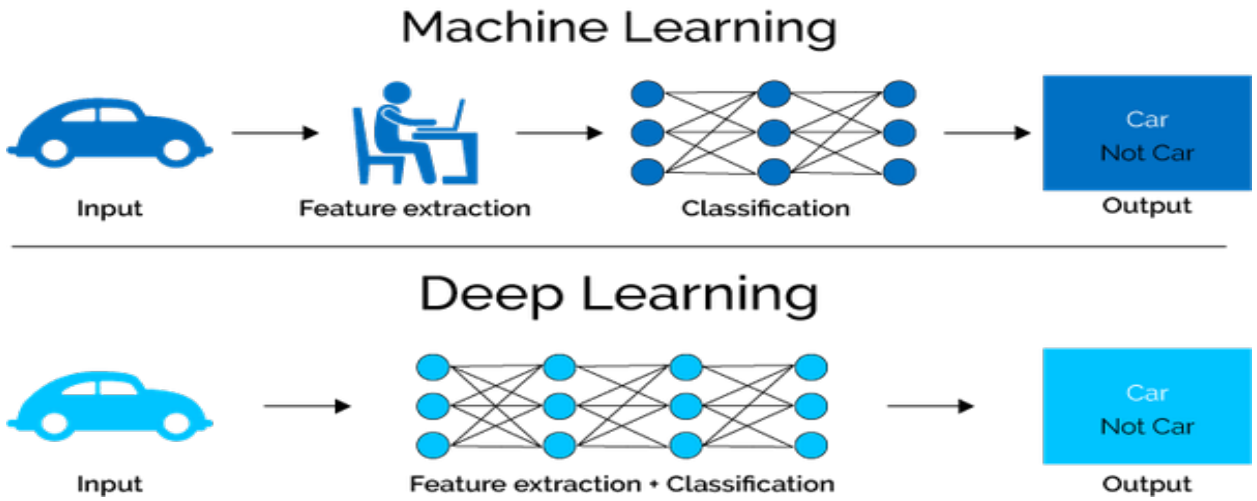


A. Legendre – J. Gauss | K. Pearson | A. Turing | S. McCulloch – W. Pitts | F. Rosenblatt | B. Widrow – M. Hoff | M. Minsky – S. Papert | J. Pearl | D. Rumelhart – G. Hinton – R. Wiliams | V. Vapnik– C. Cortes | A. Krizhevsky | A. Ng | Y. LeCun | Y. Bengio

What is Deep Learning?

- Deep Learning (DL)
  - A subset of Machine Learning that uses Deep Neural Networks (DNNs)
  - **Perhaps, the most revolutionary subset!**
- Based on learning data representation
- Examples Convolutional Neural Networks, Recurrent Neural Networks, Hybrid Networks
- Data Scientist or Developer Perspective
  1. Identify DL as solution to a problem
  2. Determine Data Set
  3. Select Deep Learning Algorithm to Use
  4. Use a large data set to train an algorithm

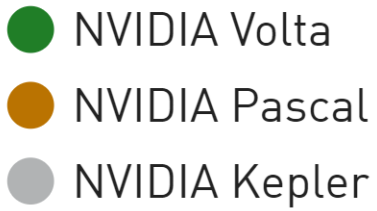




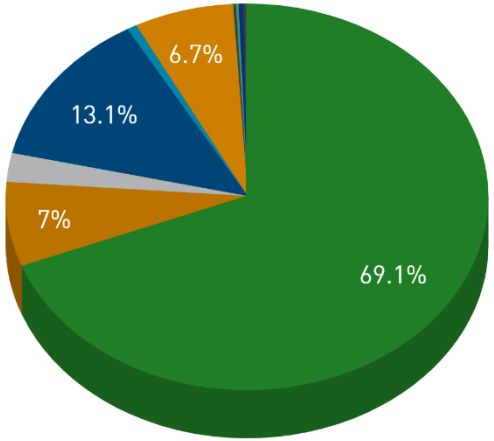
Courtesy: https://hackernoon.com/difference-between-artificial-intelligence-machine-learning-and-deep-learning-1pcv3zeg, https://blog.dataiku.com/ai-vs.-machine-learning-vs.-deep-learning

Deep Learning and High-Performance Architectures

- NVIDIA GPUs are the main driving force for faster training of DL models
  - The ImageNet Challenge - (ILSVRC) -- 90% of the teams used GPUs (2014)*
  - Deep Neural Networks (DNNs) like ResNet(s) and Inception



- However, High Performance Architectures for DL and HPC are evolving
  - 110/500 Top HPC systems use NVIDIA Volta GPUs (Nov '20)
  - DGX-1 (Pascal) and DGX-2 (Volta)
    - Dedicated DL supercomputers
  - Cascade-Lake Xeon CPUs have 28 cores/socket (TACC Frontera– #9 on Top500)
  - AMD EPYC (Rome) CPUs have 64 cores/socket
  - AMD GPUs will be powering Frontier – DOE's Exascale System at ORNL
  - Domain Specific Accelerators for DNNs are also emerging

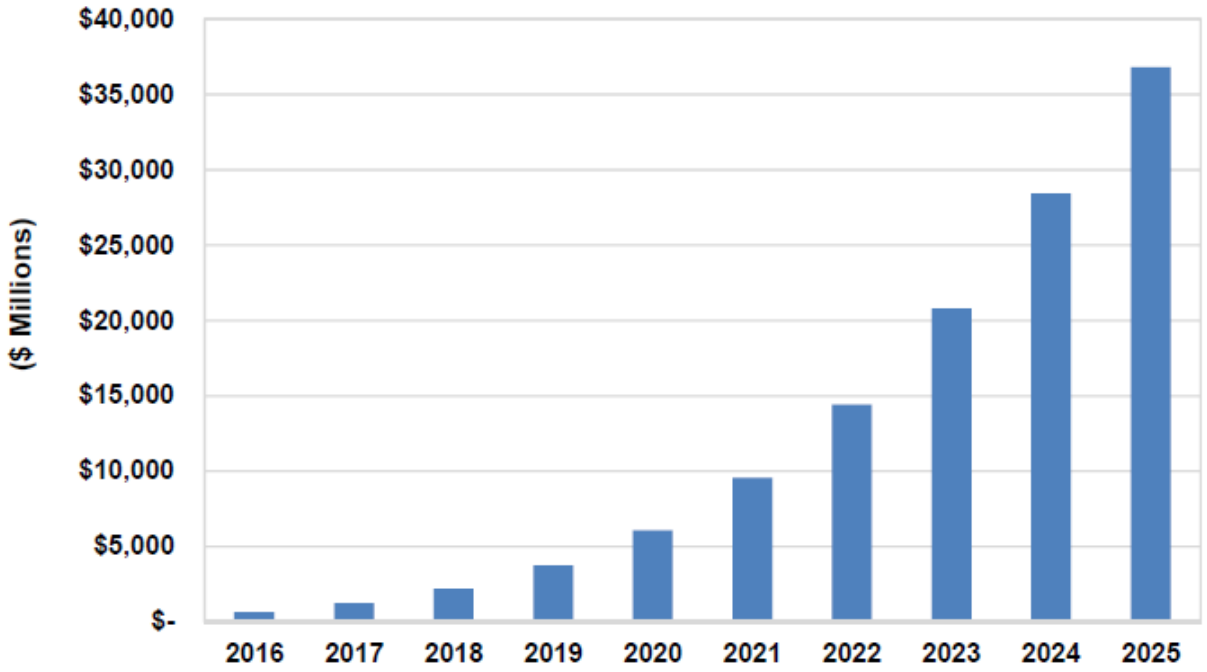


Accelerator/CP Performance Share www.top500.org

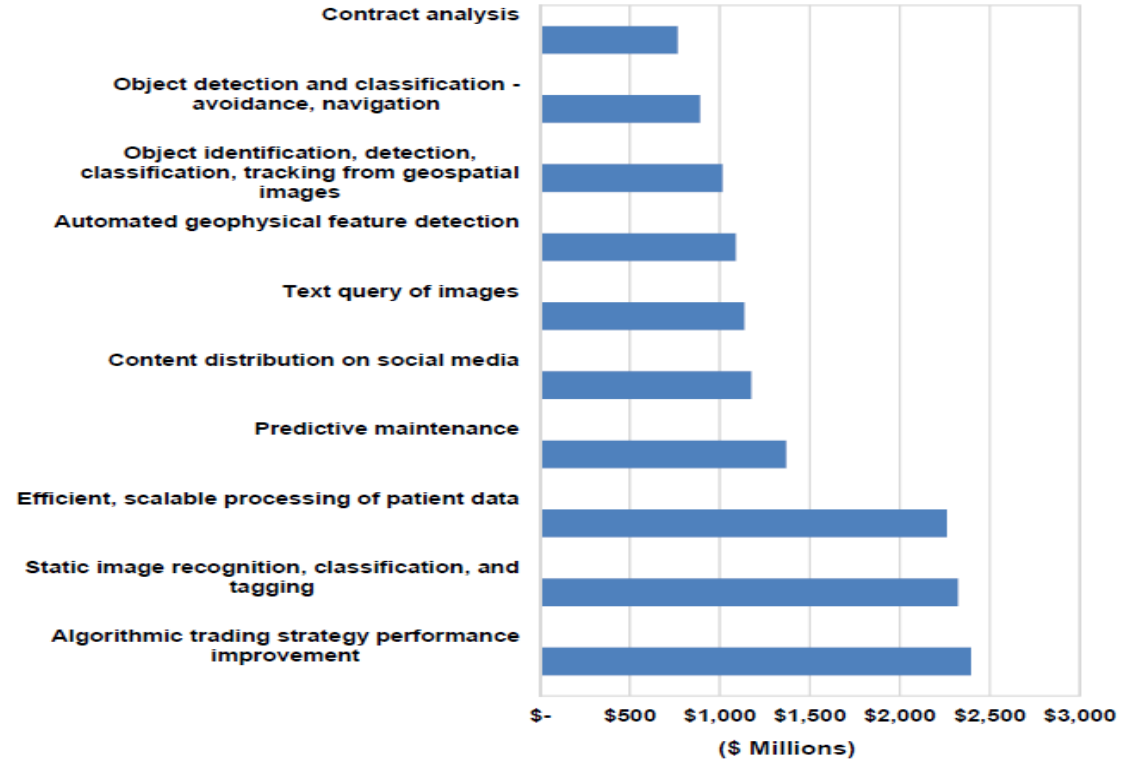
*https://blogs.nvidia.com/blog/2014/09/07/imagenet/

Deep Learning Use Cases and Growth Trends

1.1 *Artificial Intelligence Revenue, World Markets: 2016-2025*



1.2 *Artificial Intelligence Revenue, Top 10 Use Cases, World Markets: 2025*



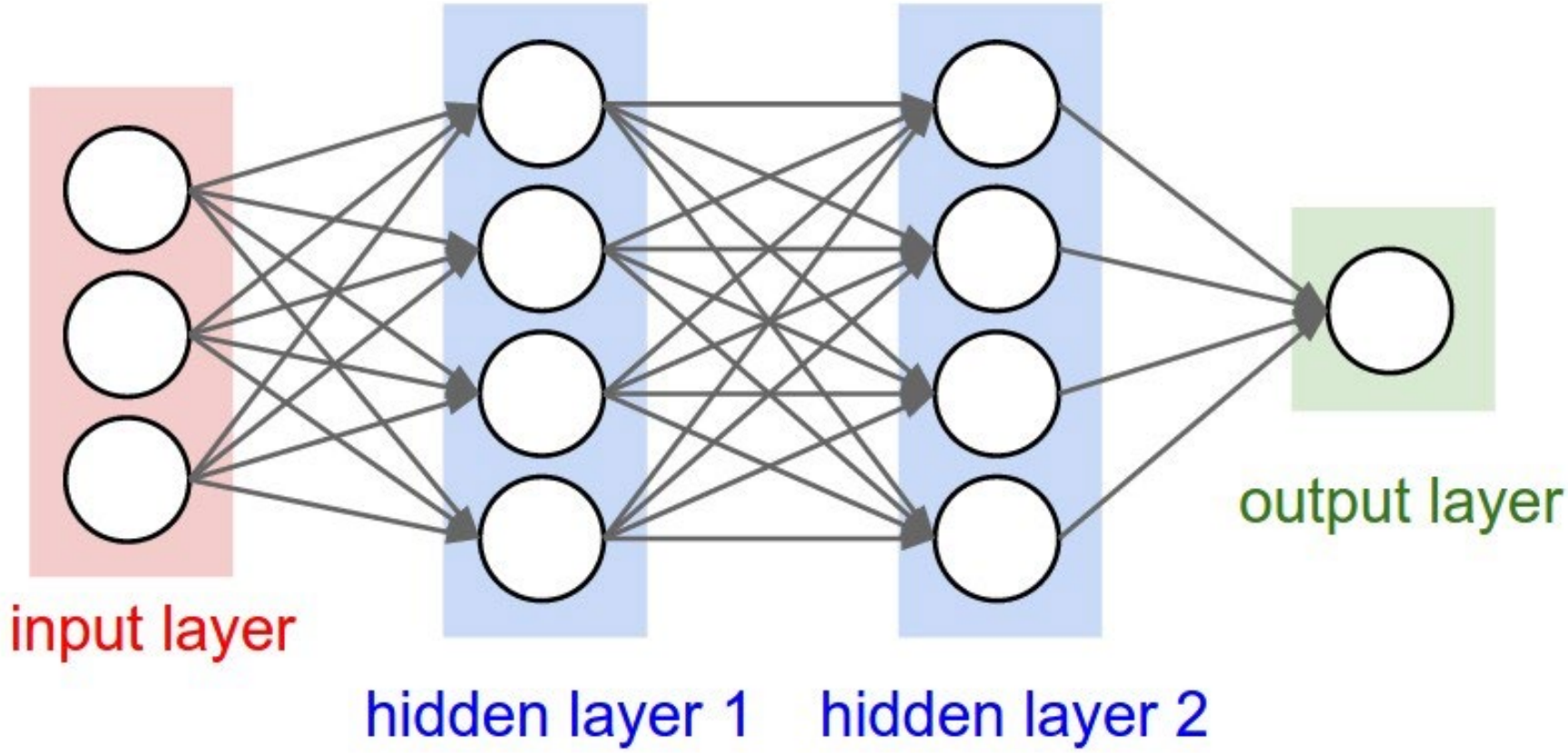
Courtesy: https://www.top500.org/news/market-for-artificial-intelligence-projected-to-hit-36-billion-by-2025/

Outline

- **Introduction**
  - The Past, Present, and Future of Deep Learning
  - **What are Deep Neural Networks?**
  - Diverse Applications of Deep Learning
  - Deep Learning Frameworks
- Overview of Execution Environments
- Parallel and Distributed DNN Training
- Latest Trends in High-Performance Computing Architectures
- Challenges in Exploiting HPC Technologies for Deep Learning
- Solutions and Case Studies
- Open Issues and Challenges
- Hands-on Exercises
- Conclusion

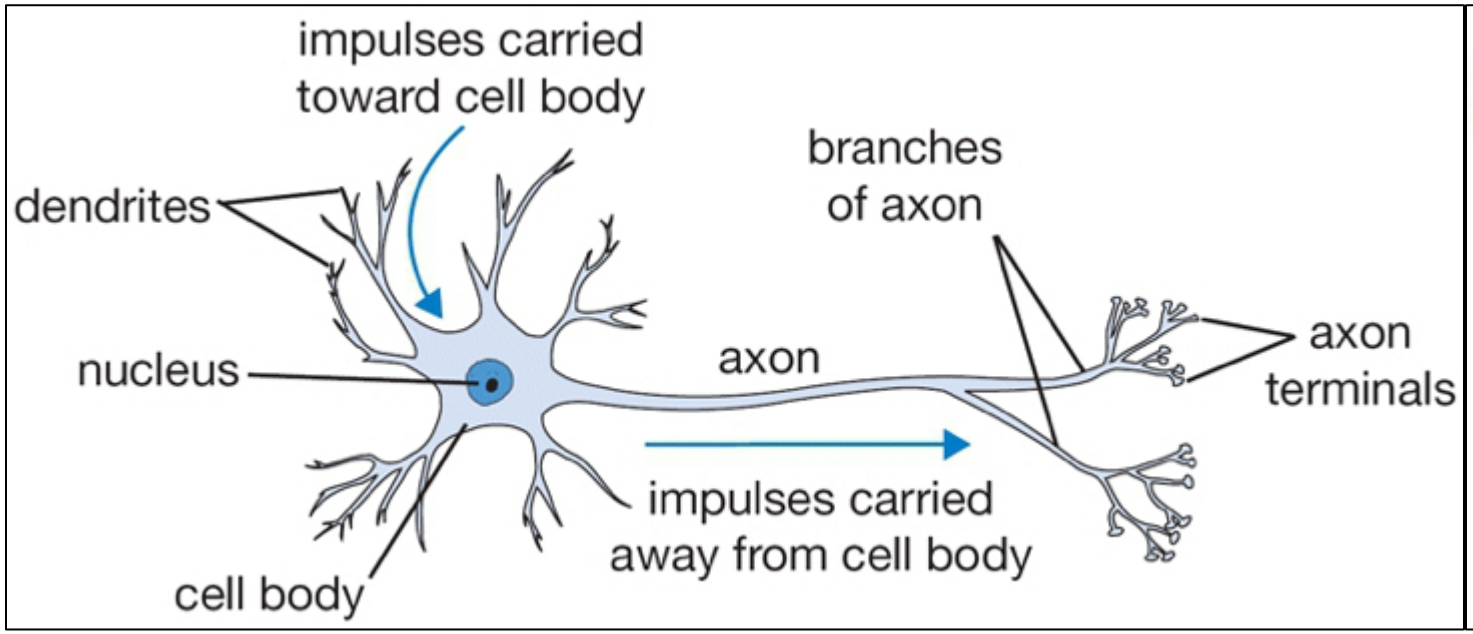
So what is a Deep Neural Network?

- Example of a 3-layer Deep Neural Network (DNN) – (input layer is not counted)

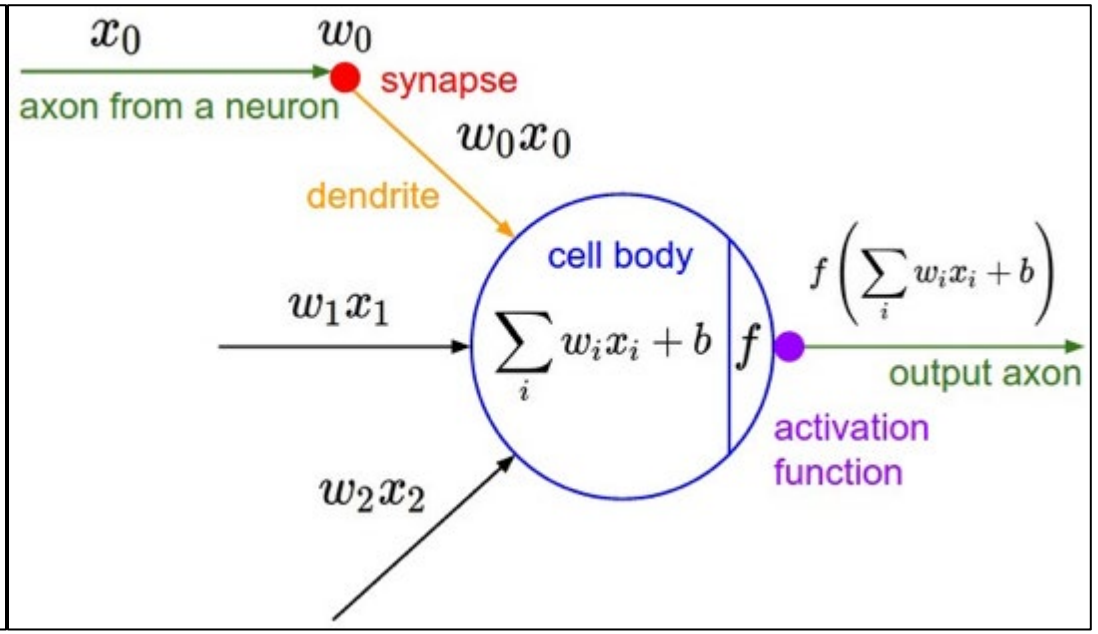


Courtesy: http://cs231n.github.io/neural-networks-1/

Graphical/Mathematical Intuitions for DNNs



Drawing of a Biological Neuron

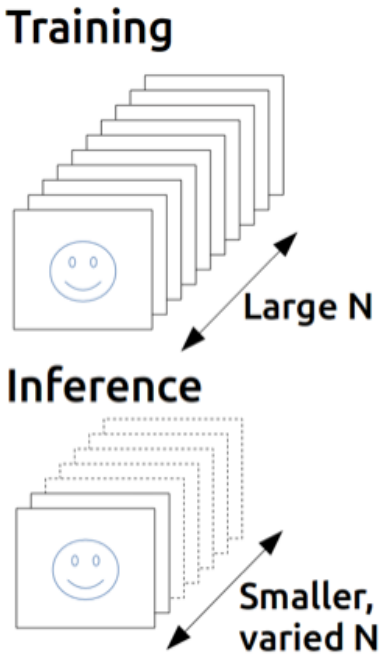


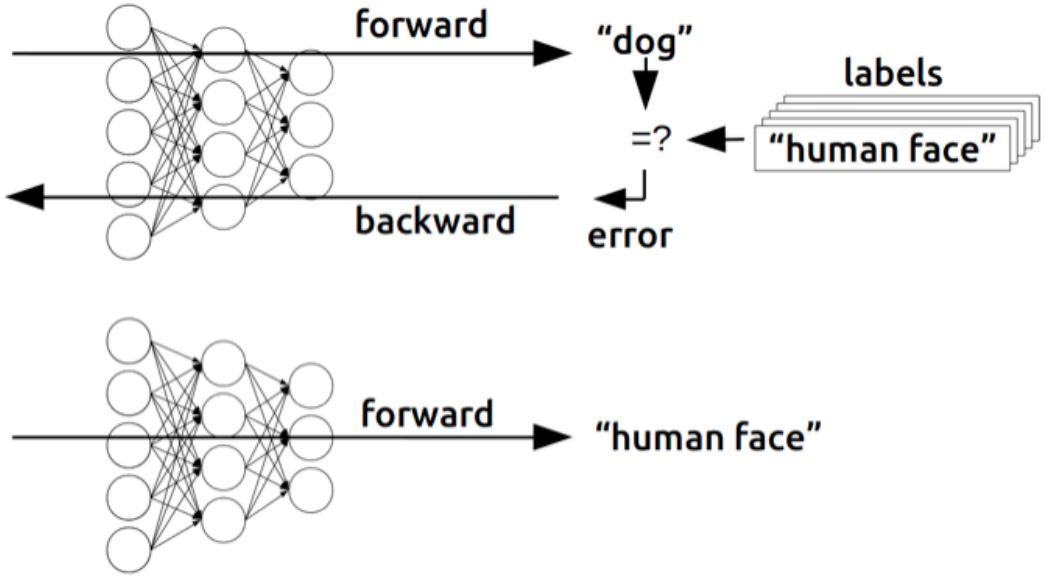
The Mathematical Model

Courtesy: http://cs231n.github.io/neural-networks-1/

Key Phases of Deep Learning

- Training is compute intensive
  - Many passes over data
  - Can take days to weeks
  - Model adjustment is done
- Inference
  - Single pass over the data
  - Should take seconds
  - No model adjustment
- Challenge: How to make **"Training"** faster?
  - Need Parallel and Distributed Training…

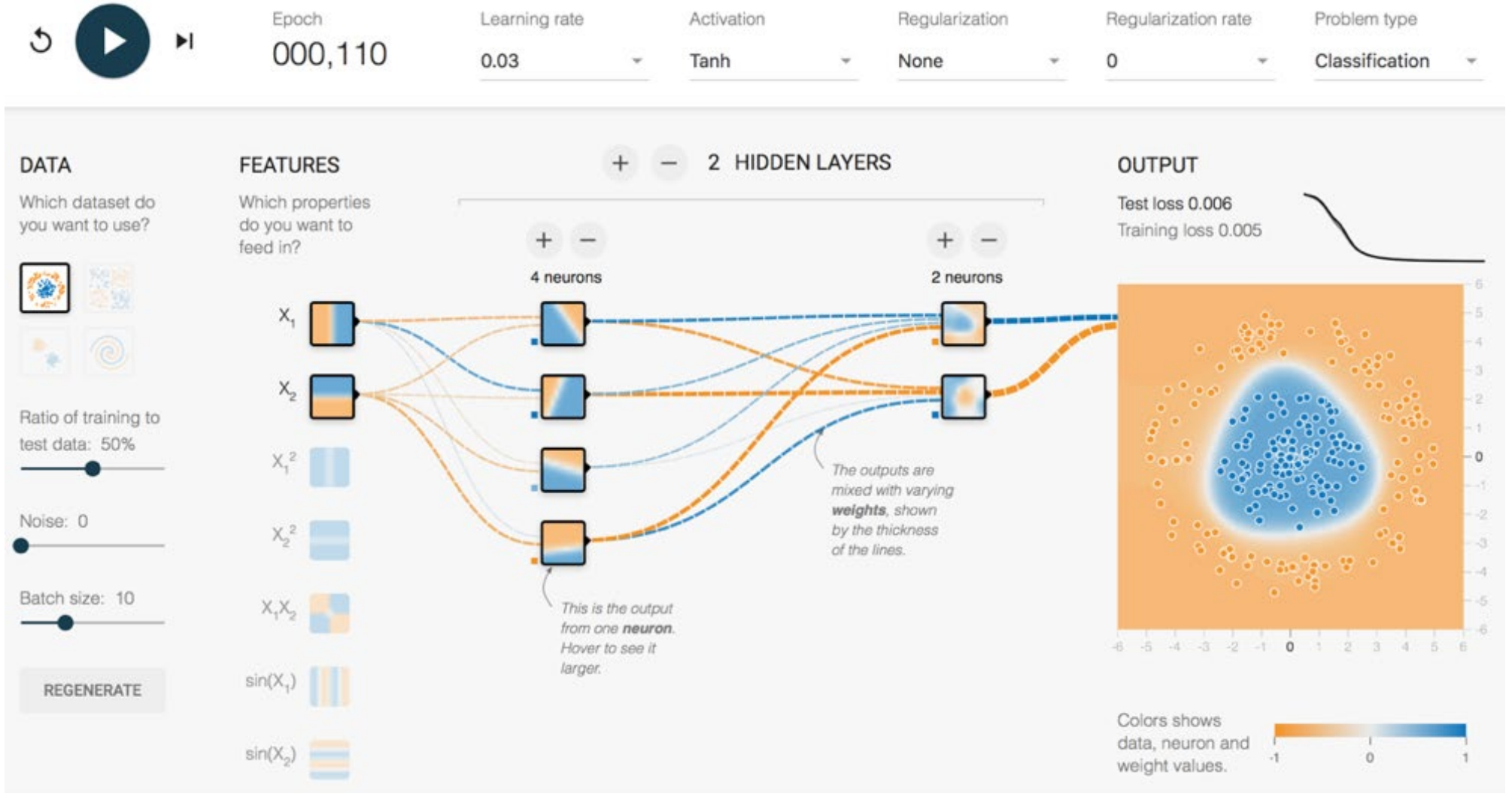




Courtesy: https://devblogs.nvidia.com/

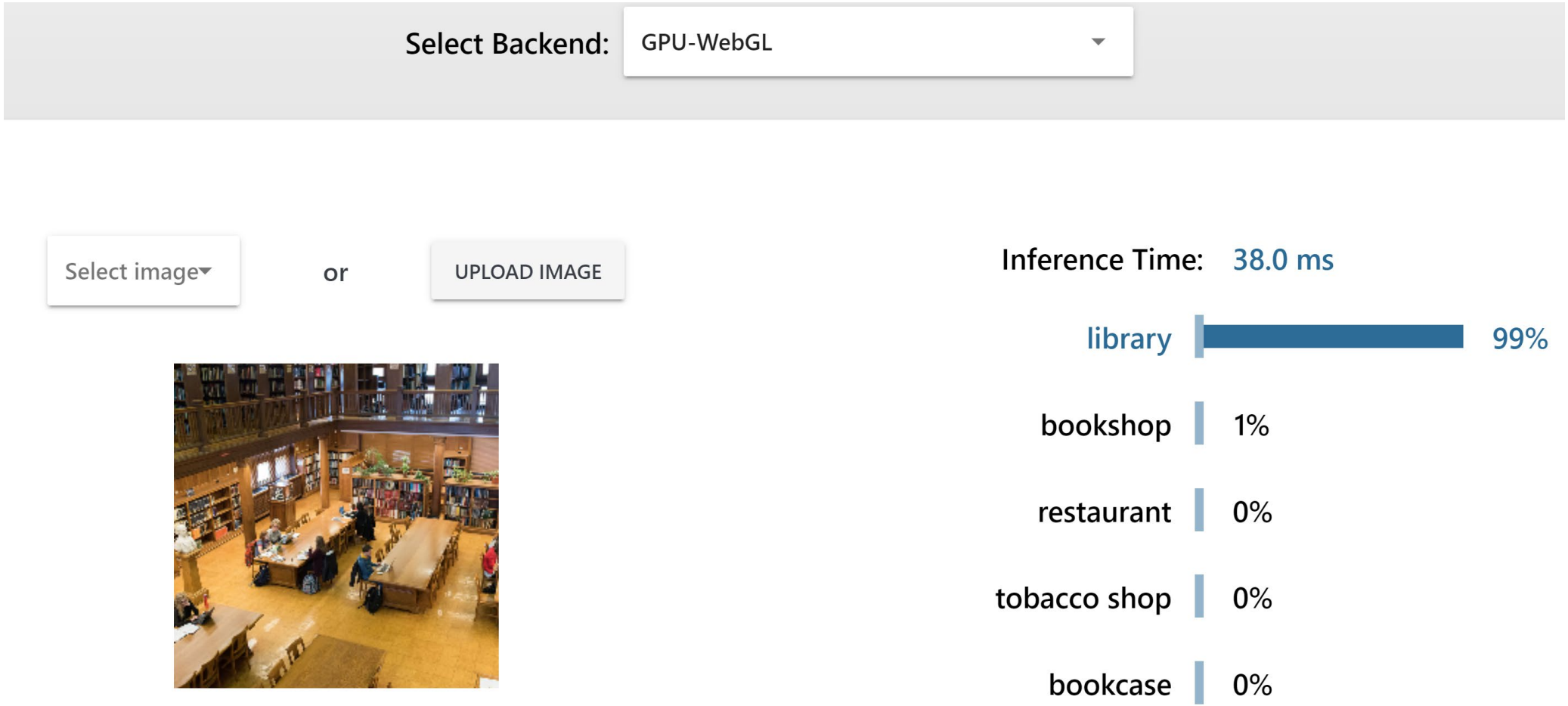
TensorFlow playground (Quick Demo)

- To actually train a network, please visit: http://playground.tensorflow.org



Inference on trained ResNet50 (Quick Demo)

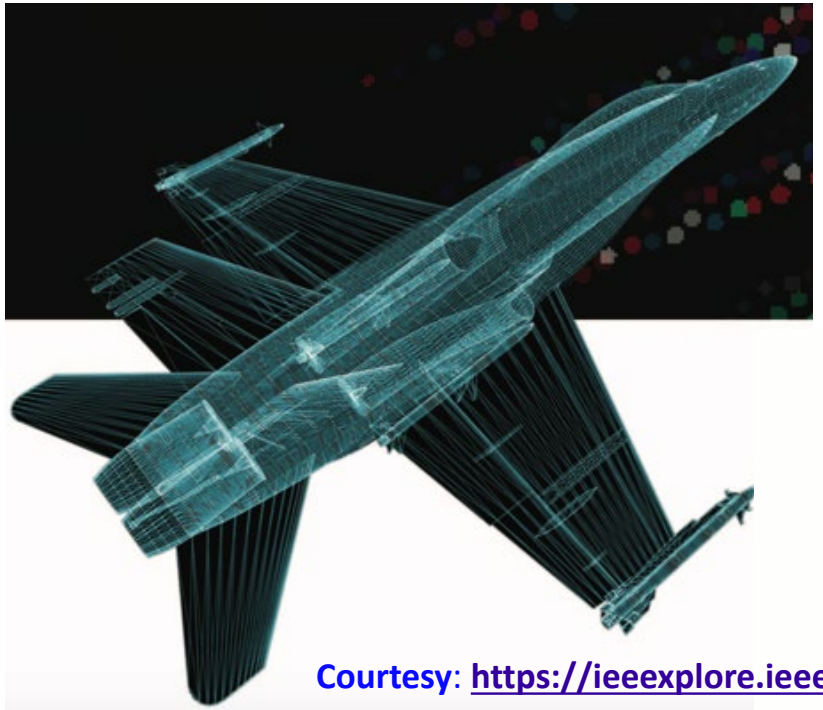
- To try your own image, please visit: https://microsoft.github.io/onnxjs-demo/#/resnet50



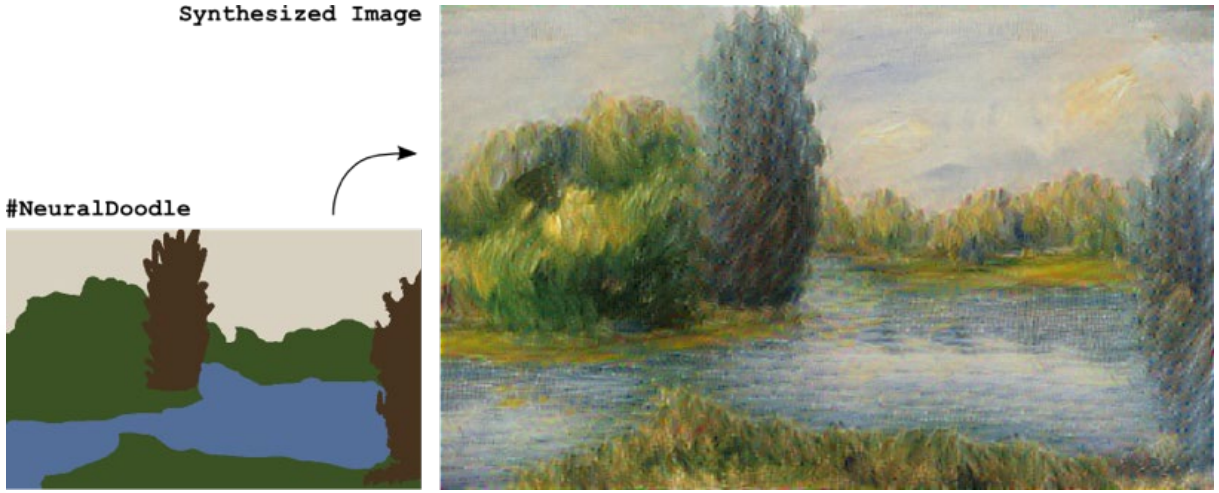
Outline

- **Introduction**
  - The Past, Present, and Future of Deep Learning
  - What are Deep Neural Networks?
  - **Diverse Applications of Deep Learning**
  - Deep Learning Frameworks
- Overview of Execution Environments
- Parallel and Distributed DNN Training
- Latest Trends in High-Performance Computing Architectures
- Challenges in Exploiting HPC Technologies for Deep Learning
- Solutions and Case Studies
- Open Issues and Challenges
- Hands-on Exercises
- Conclusion

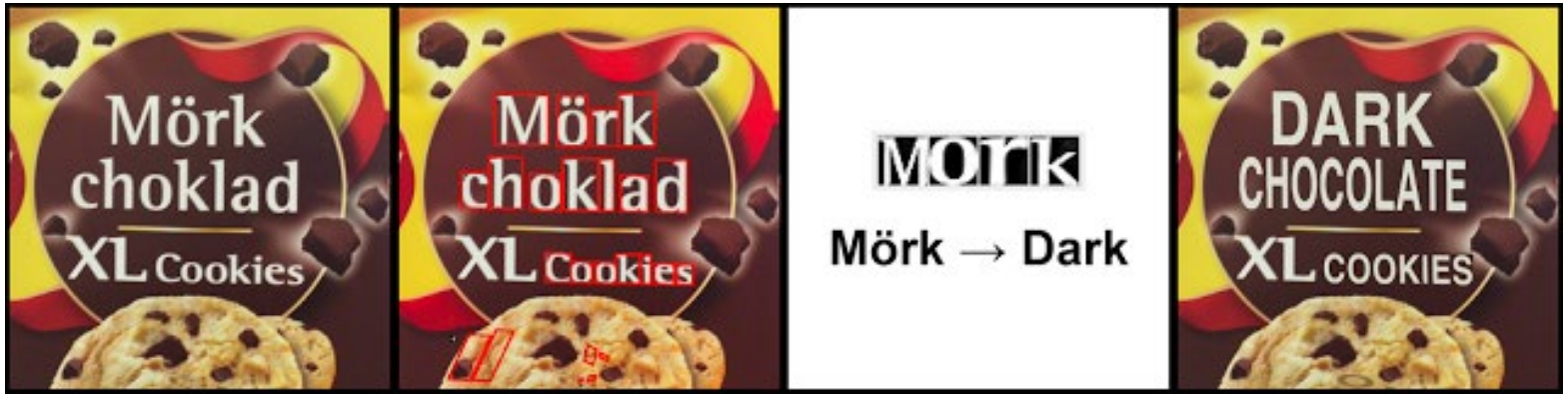
The Impact of Deep Learning on Application Areas



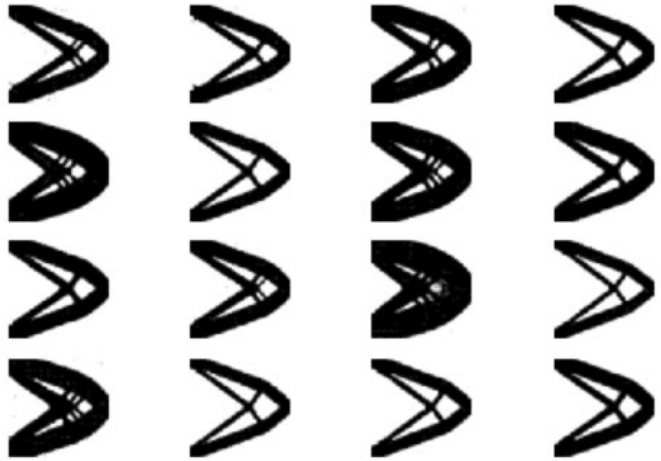
Courtesy: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8065136



Courtesy: https://github.com/alexjc/neural-doodle

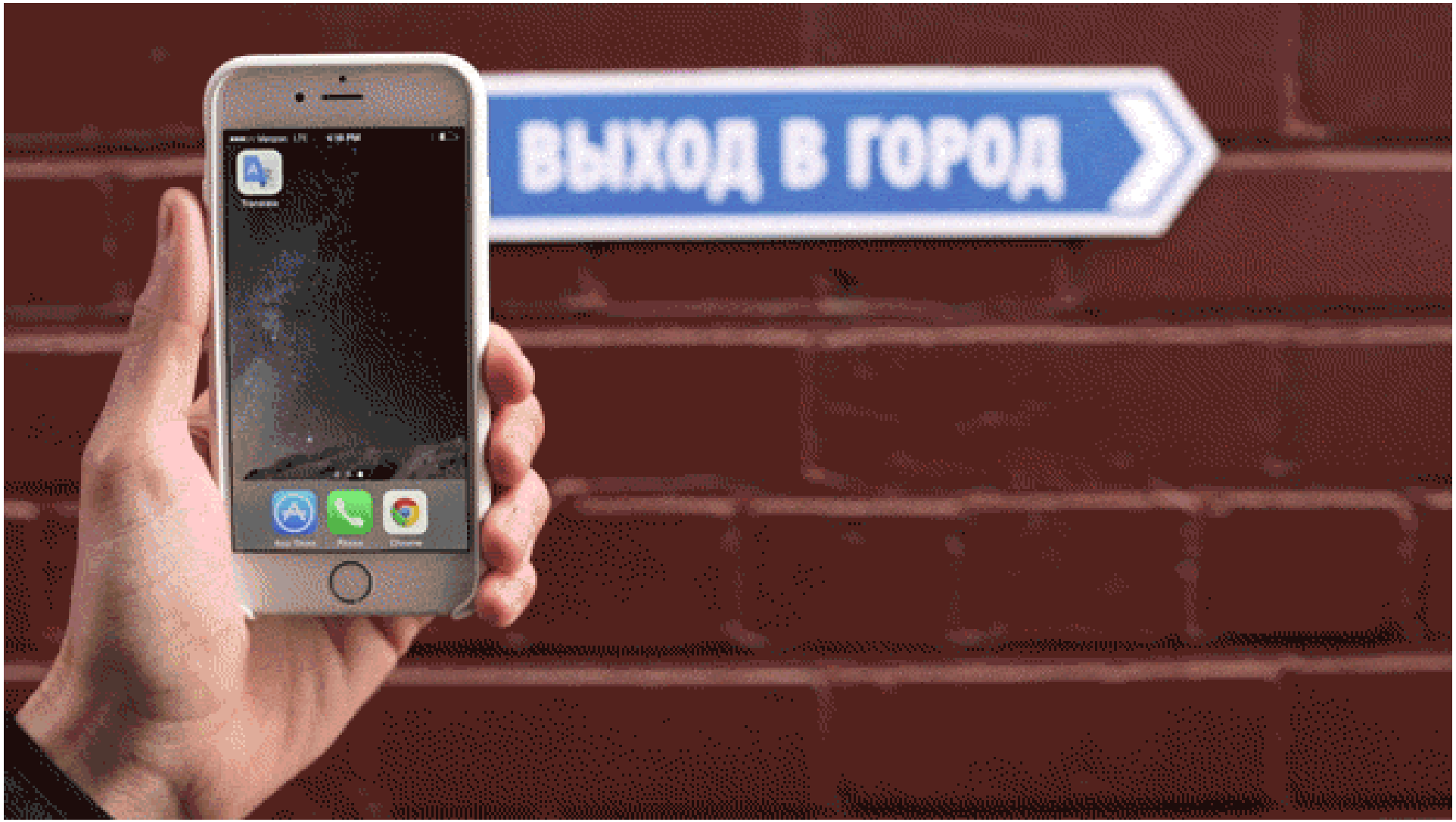


Courtesy: https://research.googleblog.com/2015/07/how-google-translate-squeezes-deep.html



Courtesy: https://arxiv.org/pdf/1808.02334.pdf

Google Translate



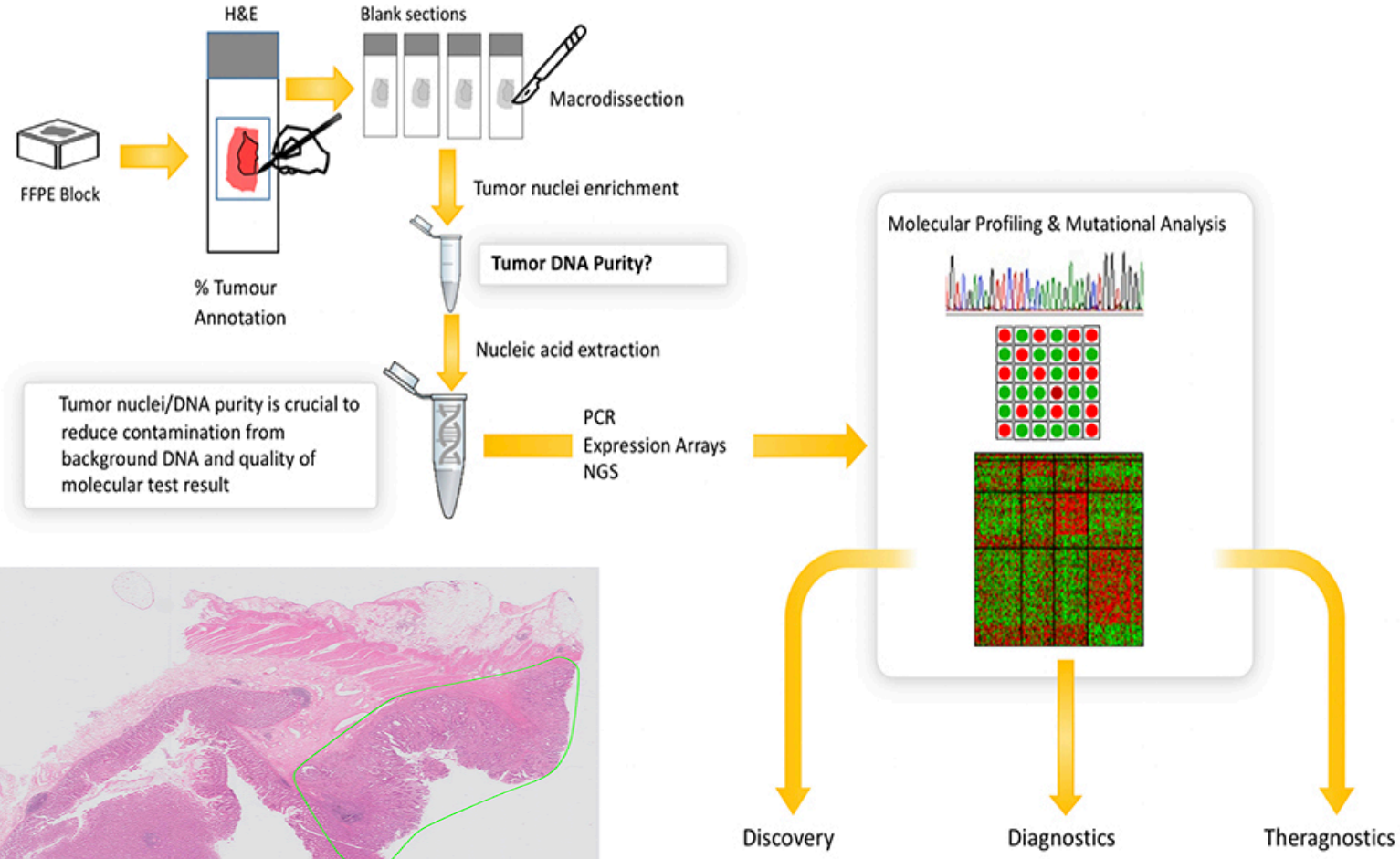
Courtesy: https://www.theverge.com/2015/1/14/7544919/google-translate-update-real-time-signs-conversations

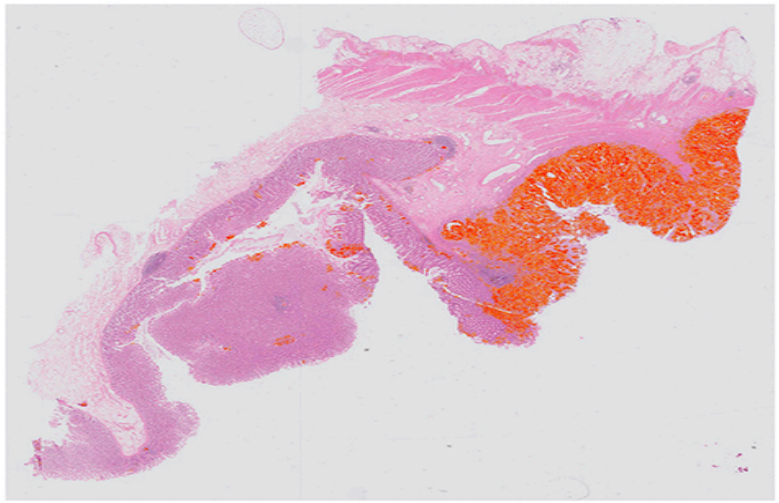
Self Driving Cars

Courtesy: http://www.teslarati.com/teslas-full-self-driving-capability-arrive-3-months-definitely-6-months-says-musk/

AI-Driven Digital Pathology

- Applications
  - Prostate Cancer Detection
  - Metastasis Detection in Breast Cancer
  - Genetic Mutation Prediction
  - Tumor Detection for Molecular Analysis

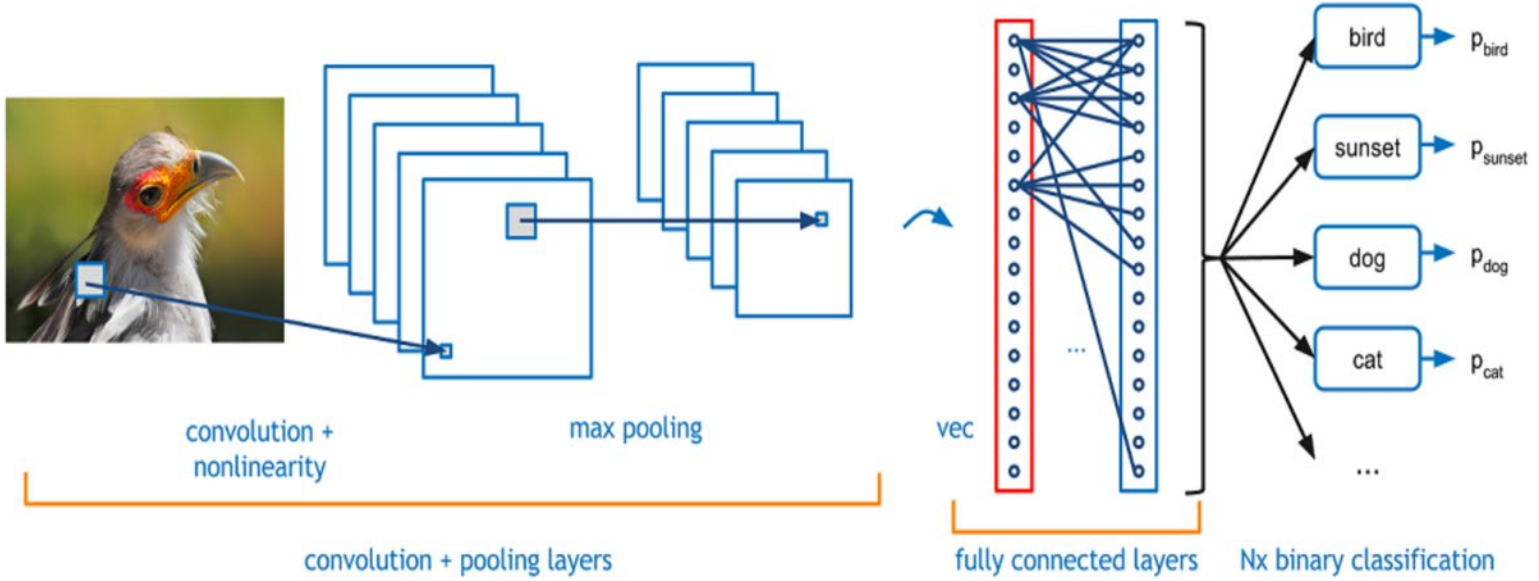




Courtesy: https://www.frontiersin.org/articles/10.3389/fmed.2019.00185/full

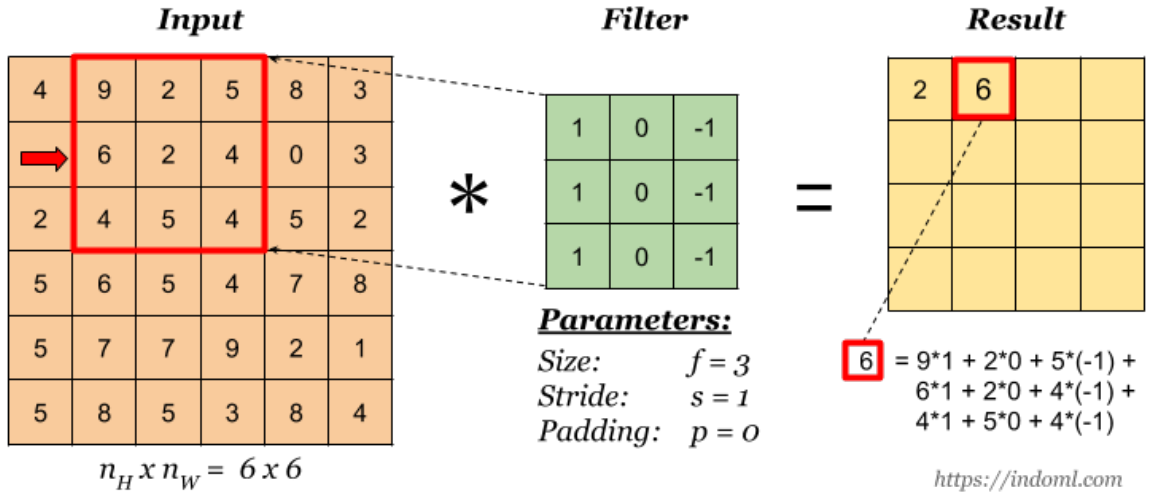
Most Well Known Application Area: Computer Vision

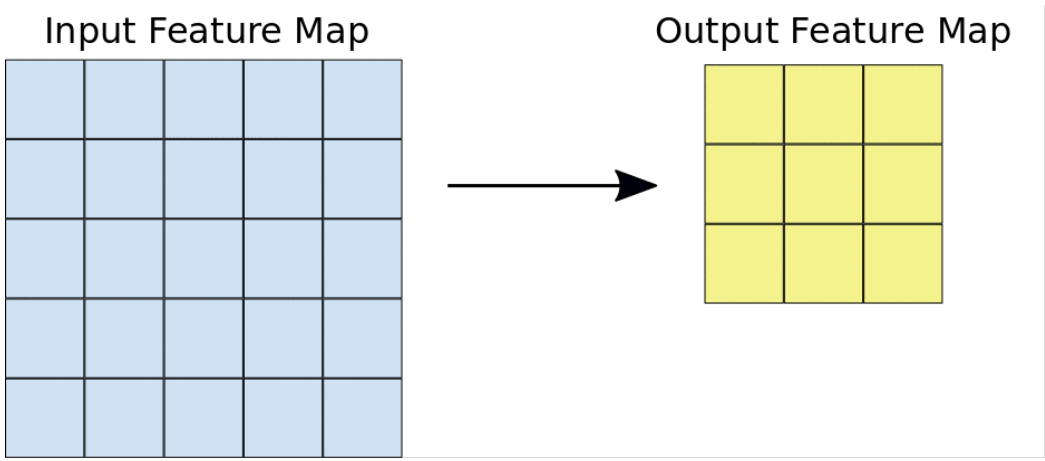
- Computer Vision Applications (image classification, object detection ….)
  - **For many, the default answer is Convolutional Neural Networks (CNNs)**
- Convolutional Neural Network
  - Dense Layers (used a classifier)
  - **Convolution Layer (used as Feature Extraction layer)**
    - Convolution operation
    - Activation function
    - Pooling



Courtesy: https://adeshpande3.github.io/A-Beginner%27s-Guide-To-Understanding-Convolutional-Neural-Networks/

What is a Convolution Operation? Why do we need it?





Found Vertical Edge

10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0

*

1	0	-1
1	0	-1
1	0	-1

=

0	30	30	0
0	30	30	0
0	30	30	0
0	30	30	0

Different Filter will give a different feature

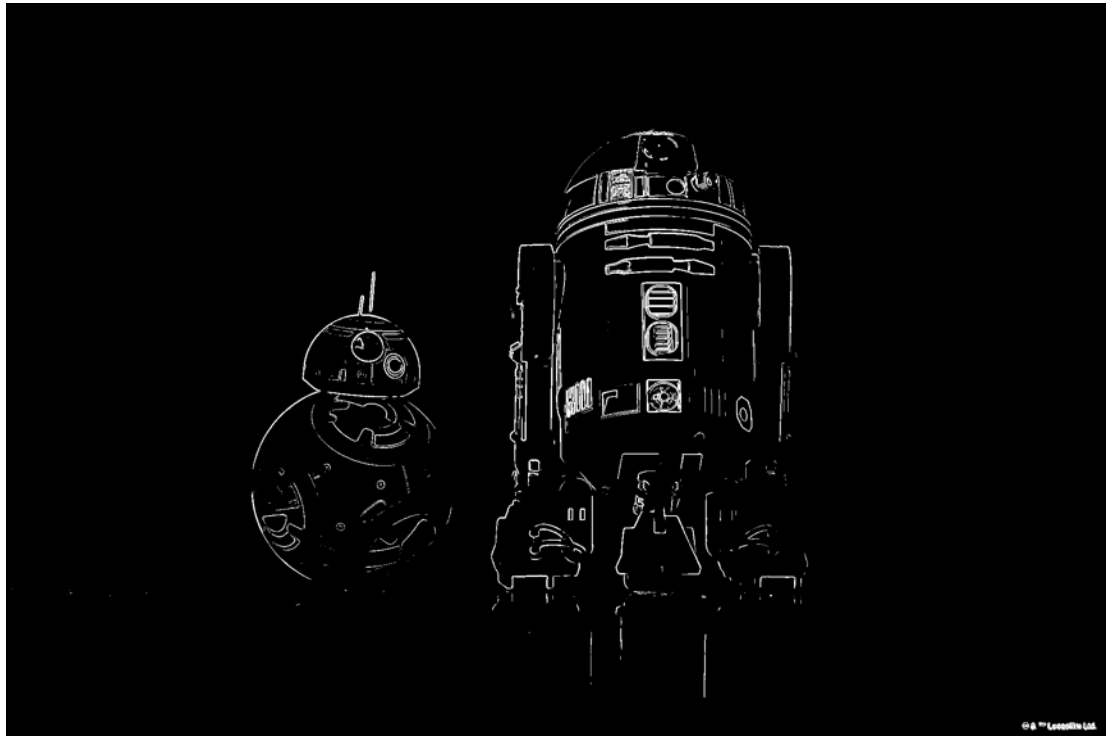
Courtesy: https://www.analyticsvidhya.com/blog/2018/12/guide-convolutional-neural-network-cnn/

Example of a Convolution Filter

Sobel Filter

-1	-2	-1
0	0	0
1	2	-1

-1	0	1
-2	0	2
-1	0	1

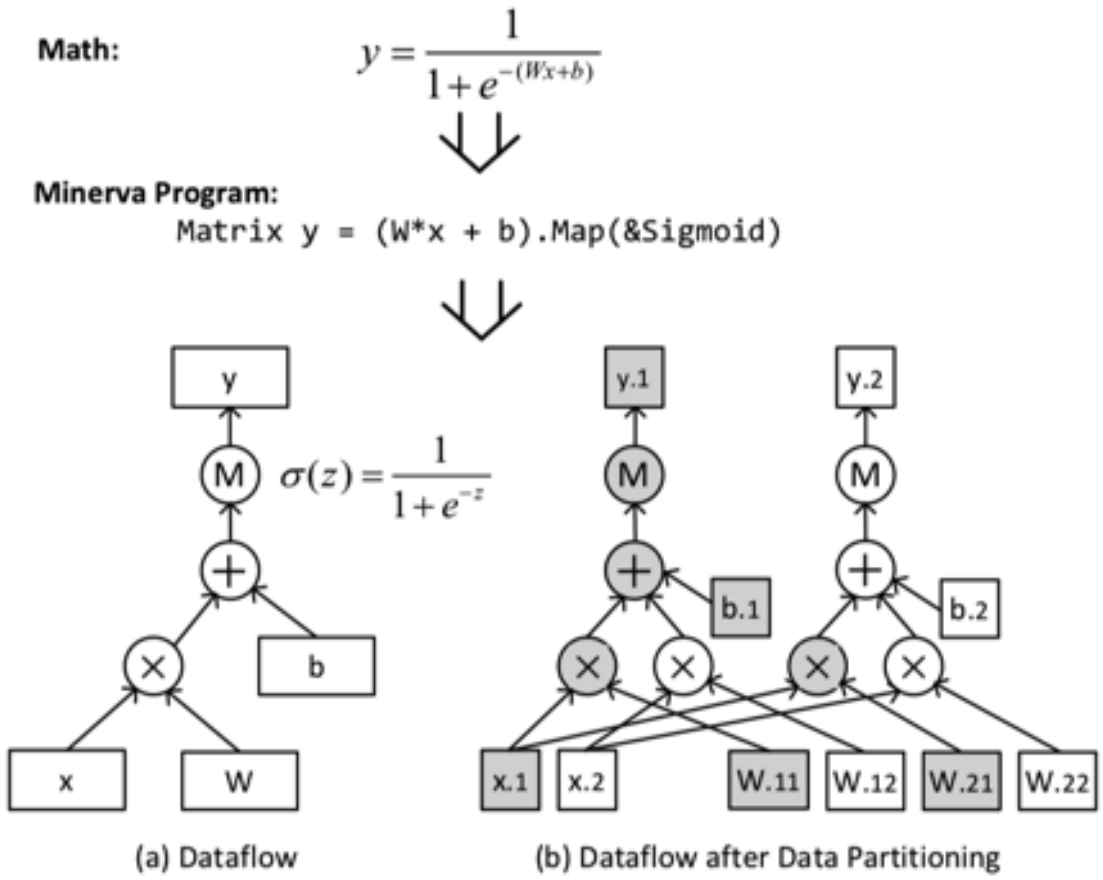


Outline

- **Introduction**
  - The Past, Present, and Future of Deep Learning
  - What are Deep Neural Networks?
  - Diverse Applications of Deep Learning
  - **Deep Learning Frameworks**
- Overview of Execution Environments
- Parallel and Distributed DNN Training
- Latest Trends in High-Performance Computing Architectures
- Challenges in Exploiting HPC Technologies for Deep Learning
- Solutions and Case Studies
- Open Issues and Challenges
- Hands-on Exercises
- Conclusion

DL Frameworks, Hardware Architectures, and Distributed Training

- Deep Learning frameworks have emerged
  - hide most of the *complicated mathematics*
  - focus on the *design* of neural networks
- We have saturated the peak potential of current-generation architectures
  - A single GPU or a many-core CPU is not enough!
- Two strategies to deal with current limitations
  - Parallel (multiple units in a single node) and/or Distributed (multiple nodes) training of DNNs
  - Dedicated hardware architectures for DNNs are being developed
- DL Frameworks will need to be enhanced for both strategies



Statement and its dataflow fragment. The data and computing vertexes with different colors reside on different processes.

Courtesy: https://web.stanford.edu/~rezab/nips2014workshop/submits/minerva.pdf

Deep Learning Frameworks

- Many Deep Learning frameworks!!
  - Google TensorFlow
  - Facebook Torch/PyTorch
  - Berkeley Caffe
  - Microsoft CNTK
  - Chainer/ChainerMN
  - Intel Neon/Nervana Graph
- Open Neural Net eXchange (ONNX) Format

Google TensorFlow (Most Popular)

- The most widely used framework open-sourced by Google
- Replaced Google's DistBelief framework
- Runs on almost all architectures (CPU, GPU, TPU, Mobile, etc.)
- Gone back and forth for APIs
  - TF 1.0 – Lazy Execution and Sessions/Estimators
  - TF 2.0 – Eager Execution and tf.keras
- https://github.com/tensorflow/tensorflow

Courtesy: https://www.tensorflow.org/

Martin Abadi et al., "TensorFlow: A system for large-scale machine learning" https://ai.google/research/pubs/pub45381

Facebook Torch/PyTorch - Catching up fast!

- Torch was written in Lua
  - Adoption wasn't wide-spread
- PyTorch is a Python adaptation of Torch
  - Gaining lot of attention
- Several contributors
  - Biggest support by Facebook

- PyTorch and Caffe2 have been merged now to PyTorch
- Key selling point is ease of expression and "define-by-run" approach

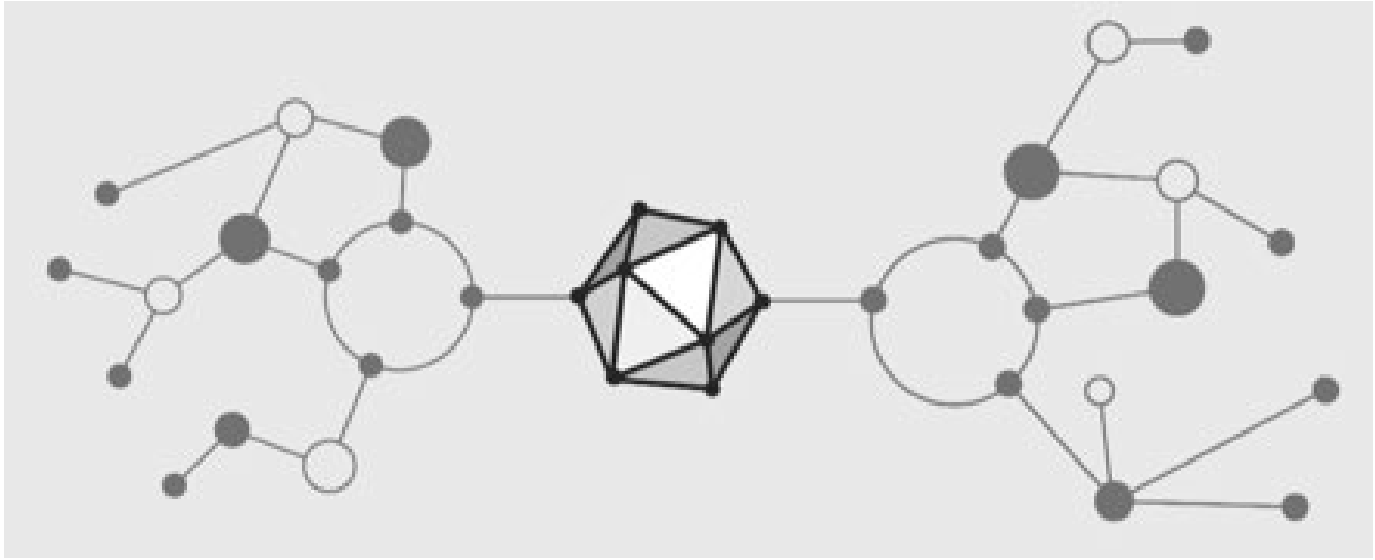
MXNet

- MXNet
  - An Apache incubator project
  - Strongly supported by Amazon now
- D2L.ai – A deep learning book with
  - Interactive jupyter notebooks, math formula, and a forum
- MXNet -- can work as a Keras backend
- Key selling point: Rich and flexible ecosystem with Gluon
  - GluonCV – Computer Vision
  - GluonNLP – Natural Language Processing

Open Neural Network eXchange (ONNX) Format

- ONNX- Not a Deep Learning framework but an open format to exchange "**trained**" networks across different frameworks

- Currently supported
  - Frameworks: Caffe2, Chainer, CNTK, MXNet, PyTorch
  - Convertors: CoreML, TensorFlow
  - Runtimes: NVIDIA



- https://onnx.ai
- https://github.com/onnx

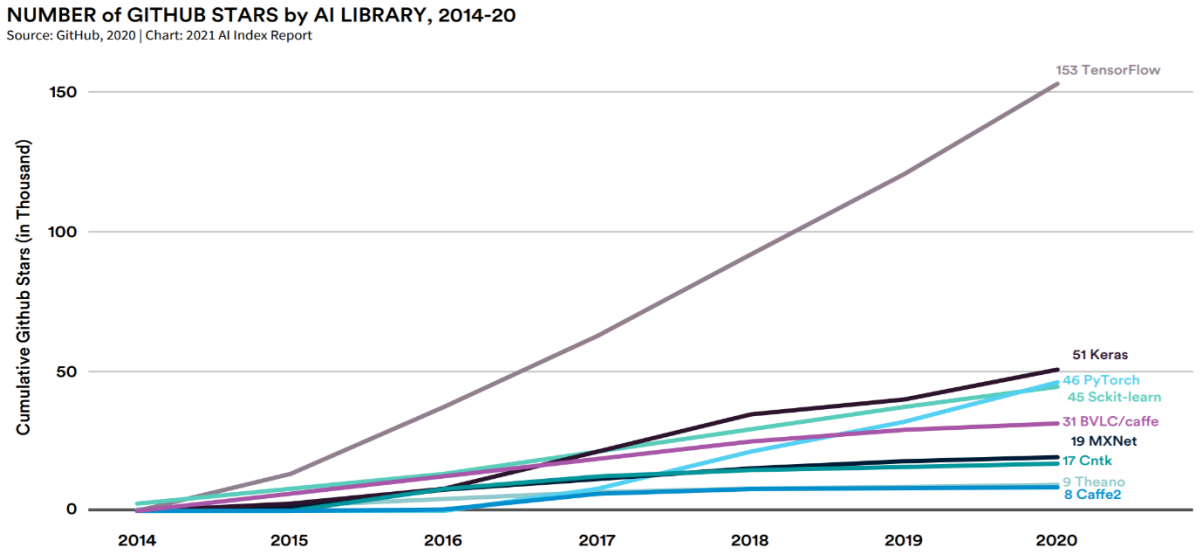
Many Other DL Frameworks…

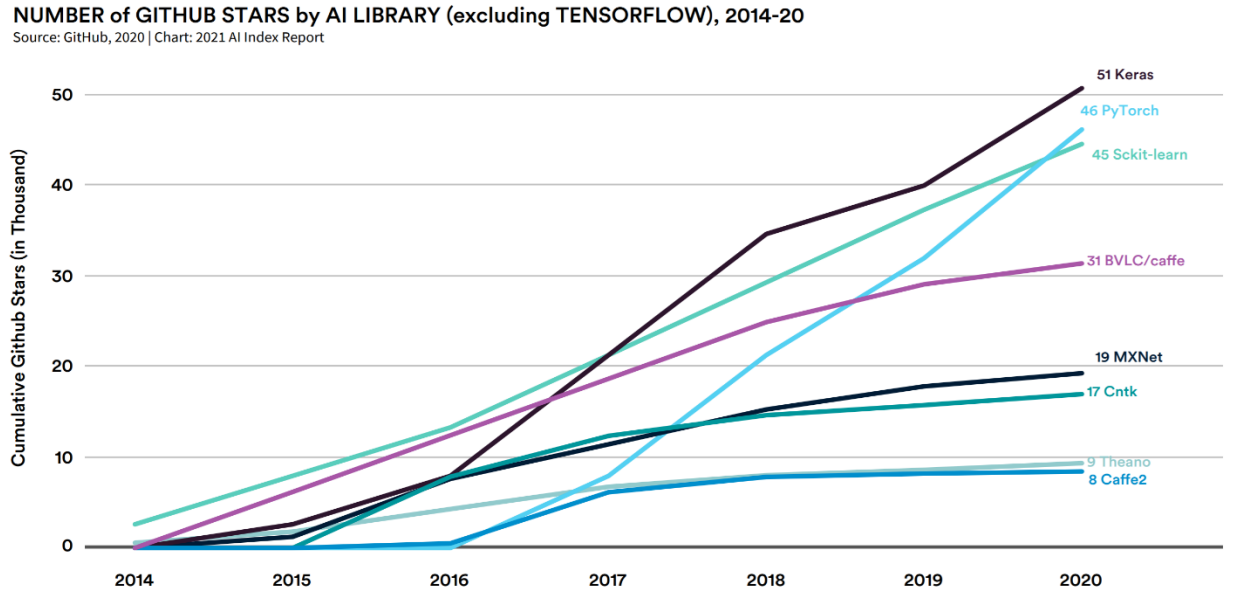
- Caffe – https://caffe.berkeleyvision.org
- Keras - https://keras.io
- Theano - http://deeplearning.net/software/theano/
- Blocks - https://blocks.readthedocs.io/en/latest/
- Intel BigDL - https://software.intel.com/en-us/articles/bigdl-distributed-deep-learning-on-apache-spark
- The list keeps growing and the names keep getting longer and weirder ;-)
  - Livermore Big Artificial Neural Network Toolkit (LBANN) - https://github.com/LLNL/lbann
  - Deep Scalable Sparse Tensor Network Engine (DSSTNE) - https://github.com/amzn/amazon-dsstne

Statistics about DL Frameworks

- AI Index report offers very detailed trends about AI and ML
  - Interesting stats. about DL frameworks

- TheGradient* has a latest article on *PyTorch winning over TensorFlow* in CVPR, ICML, ICLR and other conferences





Courtesy: https://aiindex.stanford.org

* https://thegradient.pub/state-of-ml-frameworks-2019-pytorch-dominates-research-tensorflow-dominates-industry/

Outline

- Introduction
- **Overview of Execution Environments**
- Parallel and Distributed DNN Training
- Latest Trends in High-Performance Computing Architectures
- Challenges in Exploiting HPC Technologies for Deep Learning
- Solutions and Case Studies
- Open Issues and Challenges
- Hands-on Exercises
- Conclusion

Architectures and Execution Environments for DL frameworks?

- Early (2014) frameworks used a single fast GPU
- Today, parallel training on multiple GPUs is being supported by most frameworks
- Distributed (multiple nodes) training is still upcoming
  - *A lot of fragmentation in the efforts (Horovod, MPI, NCCL, Gloo, gRPC, etc.)*

- Hardware and Software Architectures for DL are also emerging
  - Habana, Nervana, Google TPUs, and many more…
  - Smartphones - OK Google, Siri, Cortana, Alexa, etc.
  - DrivePX – the computer that drives NVIDIA's self-driving car
  - Deeplearn.js (a DL framework in a web-browser)
  - TensorFlow playground - http://playground.tensorflow.org/

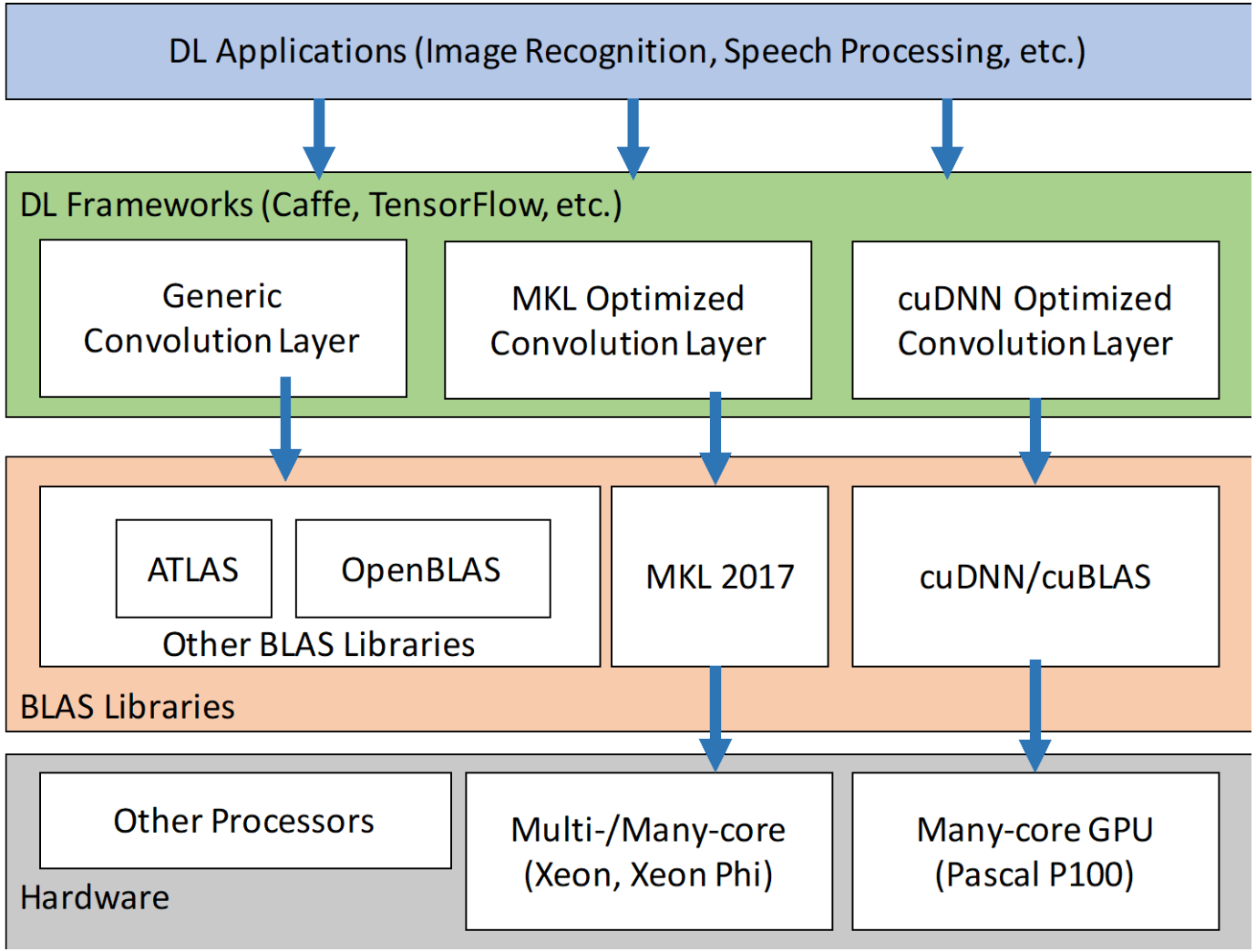
Conventional Execution on GPUs and CPUs

- We all have heard
  - *Our framework is faster than your framework!*
- This needs to be understood in a holistic way
- Performance
  - Depends on the entire execution environment (the full architectural stack)
  - Multiple helper libraries and systems have an impact
- Isolated view of performance is not helpful for ML/DL workloads
- Architecture-specific (CPU/GPU/TPU) optimizations need to be developed and properly used

A. A. Awan, H. Subramoni, and Dhabaleswar K. Panda. "An In-depth Performance Characterization of CPU- and GPU-based DNN Training on Modern Architectures", In Proceedings of the Machine Learning on HPC Environments (MLHPC'17). ACM, New York, NY, USA, Article 8.

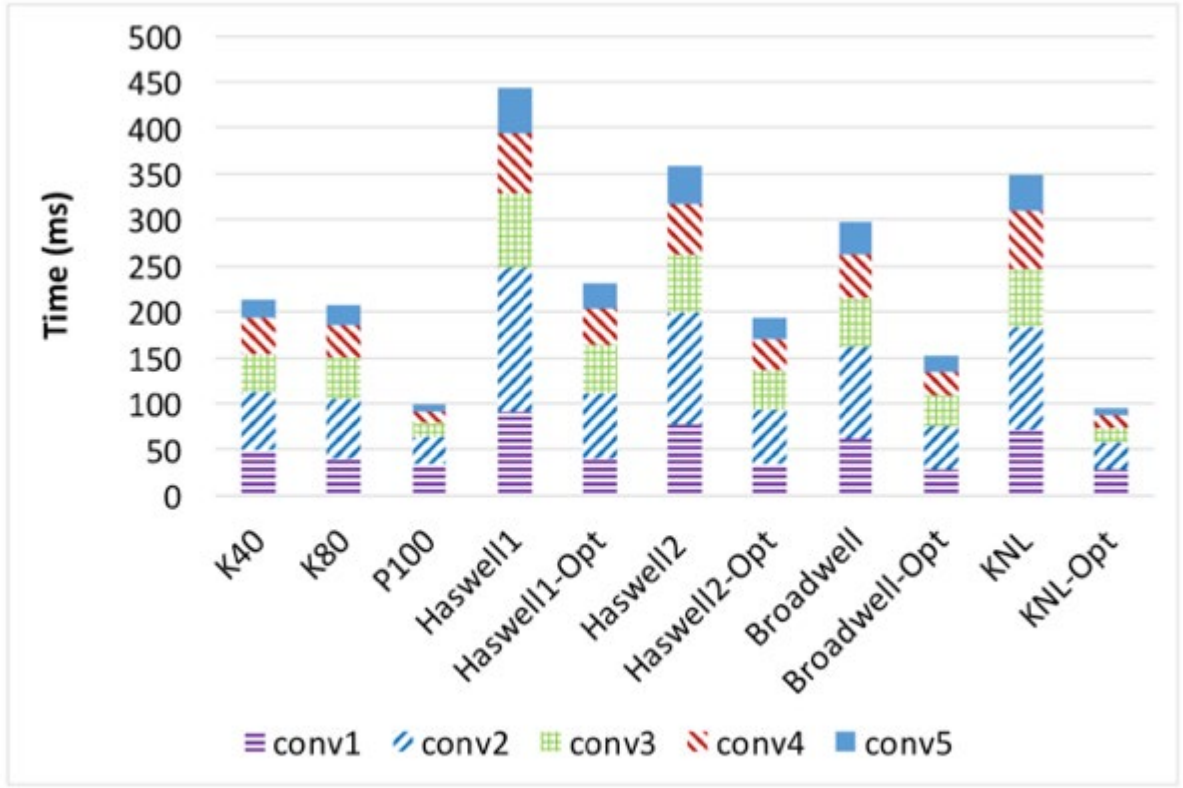
DL Frameworks and Underlying Libraries

- BLAS Libraries – the heart of math operations
  - Atlas/OpenBLAS
  - NVIDIA cuBlas
  - Intel Math Kernel Library (MKL)
- Most compute intensive layers are generally optimized for a specific hardware
  - E.g. Convolution Layer, Pooling Layer, etc.
- DNN Libraries – the heart of Convolutions!
  - NVIDIA cuDNN (already reached its 7th iteration – cudnn-v7)
  - Intel MKL-DNN – a promising development for CPU-based ML/DL training

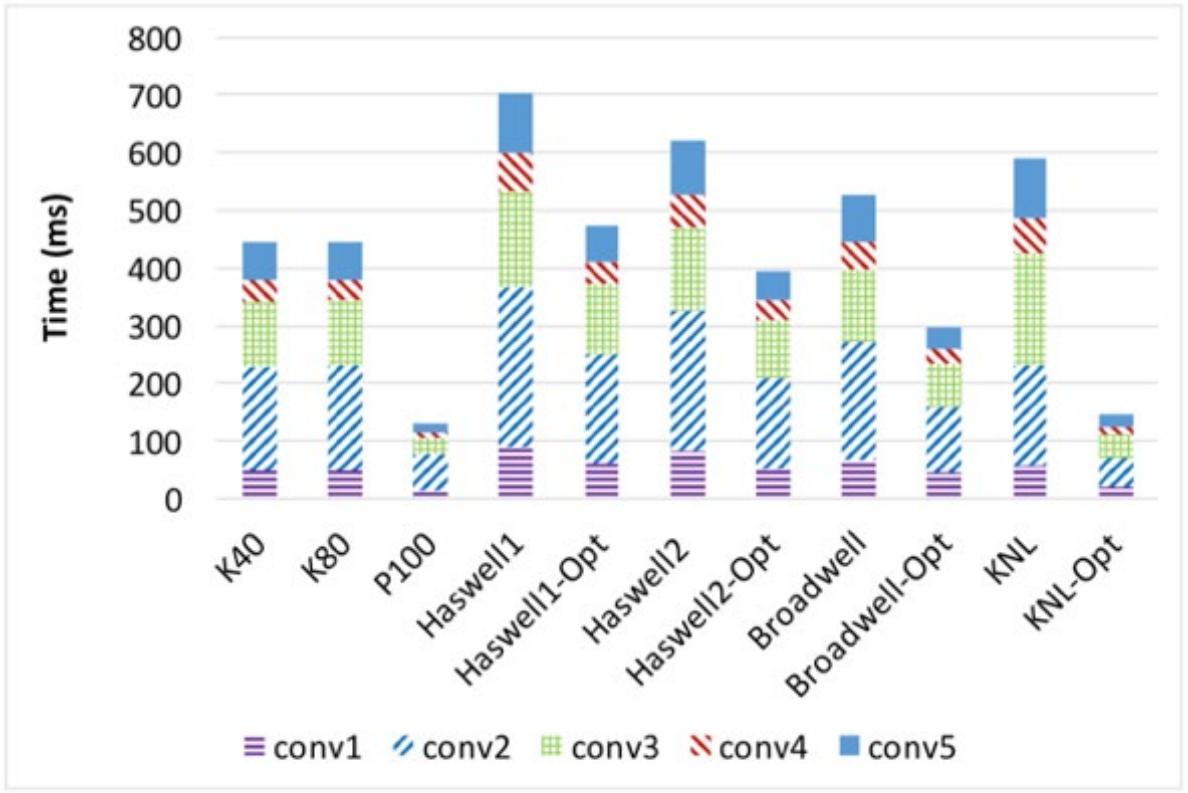


A. A. Awan, H. Subramoni, and Dhabaleswar K. Panda. "An In-depth Performance Characterization of CPU- and GPU-based DNN Training on Modern Architectures", In Proceedings of the Machine Learning on HPC Environments (MLHPC'17). ACM, New York, NY, USA, Article 8.

Where does the Performance come from?



(a) AlexNet: Forward Propagation



(b) AlexNet: Backward Propagation

- The full landscape: Forward and Backward Pass -- ***Faster Convolutions → Faster Training***
- Performance of Intel KNL == NVIDIA P100 for AlexNet Training – ***Volta is in a different league!***
- Most performance gains are based on improvements in layer ***conv2*** and ***conv3*** for AlexNet

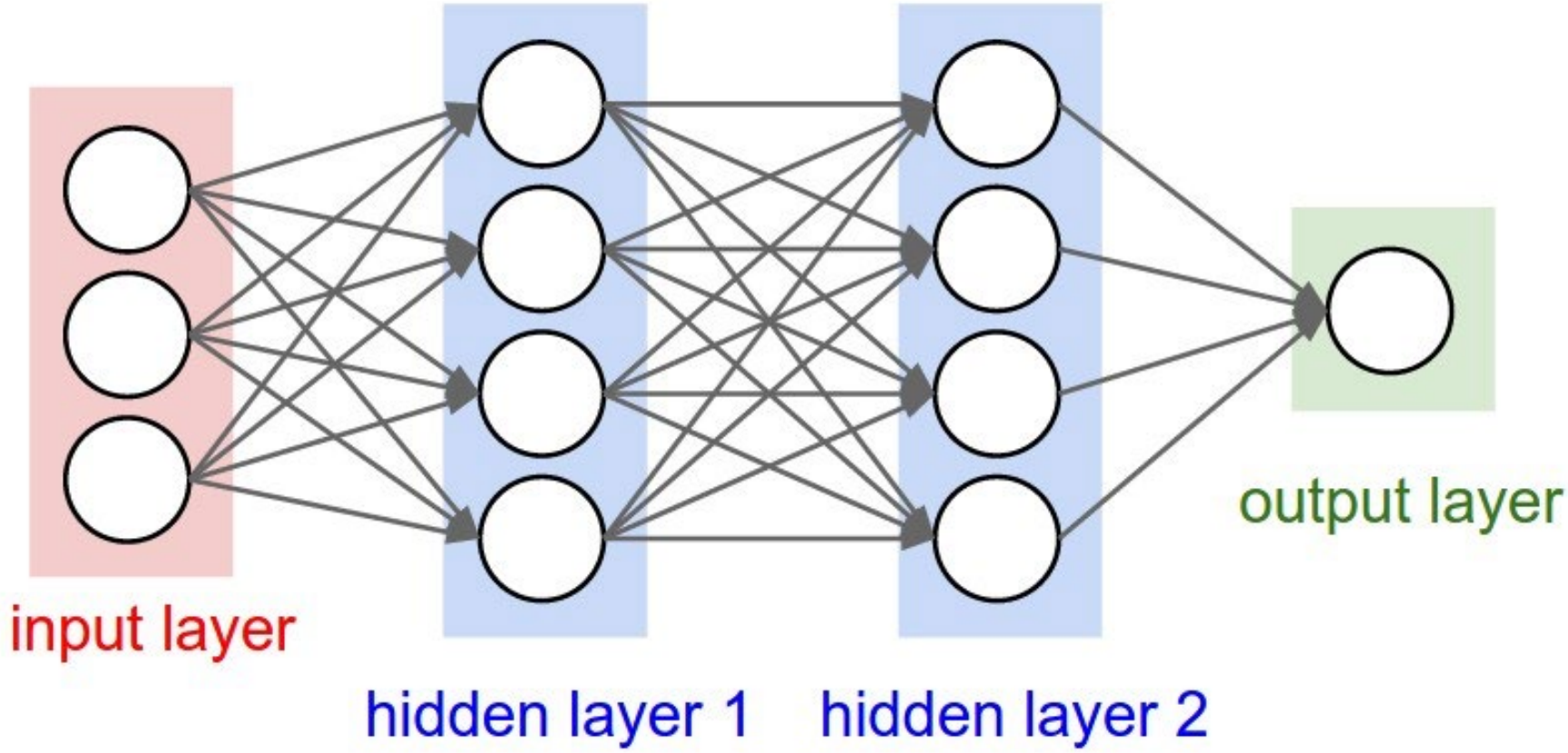
A. A. Awan, H. Subramoni, and Dhabaleswar K. Panda. "An In-depth Performance Characterization of CPU- and GPU-based DNN Training on Modern Architectures", In Proceedings of the Machine Learning on HPC Environments (MLHPC'17). ACM, New York, NY, USA, Article 8.

Outline

- Introduction
- Overview of Execution Environments
- **Parallel and Distributed DNN Training**
- Latest Trends in High-Performance Computing Architectures
- Challenges in Exploiting HPC Technologies for Deep Learning
- Solutions and Case Studies
- Open Issues and Challenges
- Hands-on Exercises
- Conclusion

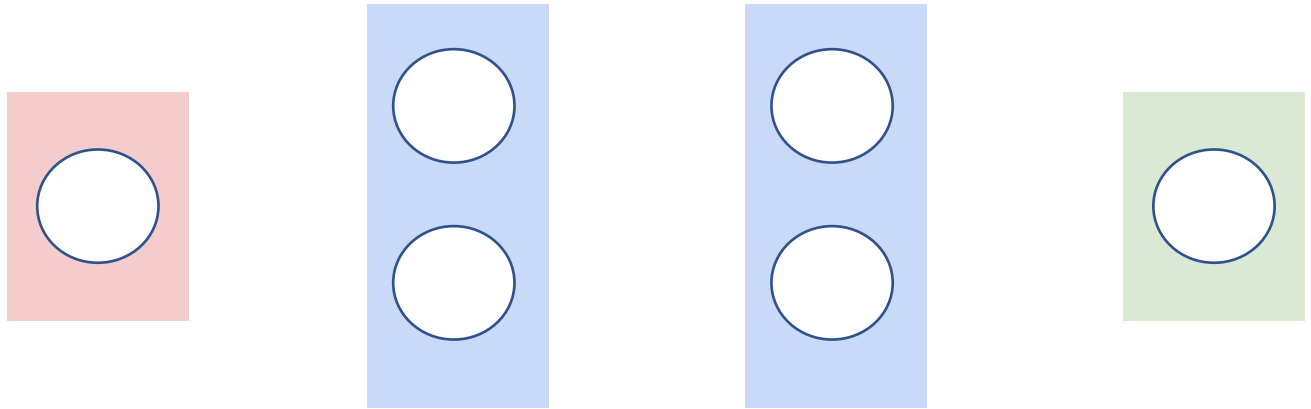
Understanding the Deep Neural Network Concepts

- Example of a 3-layer Deep Neural Network (DNN) – (input layer is not counted)

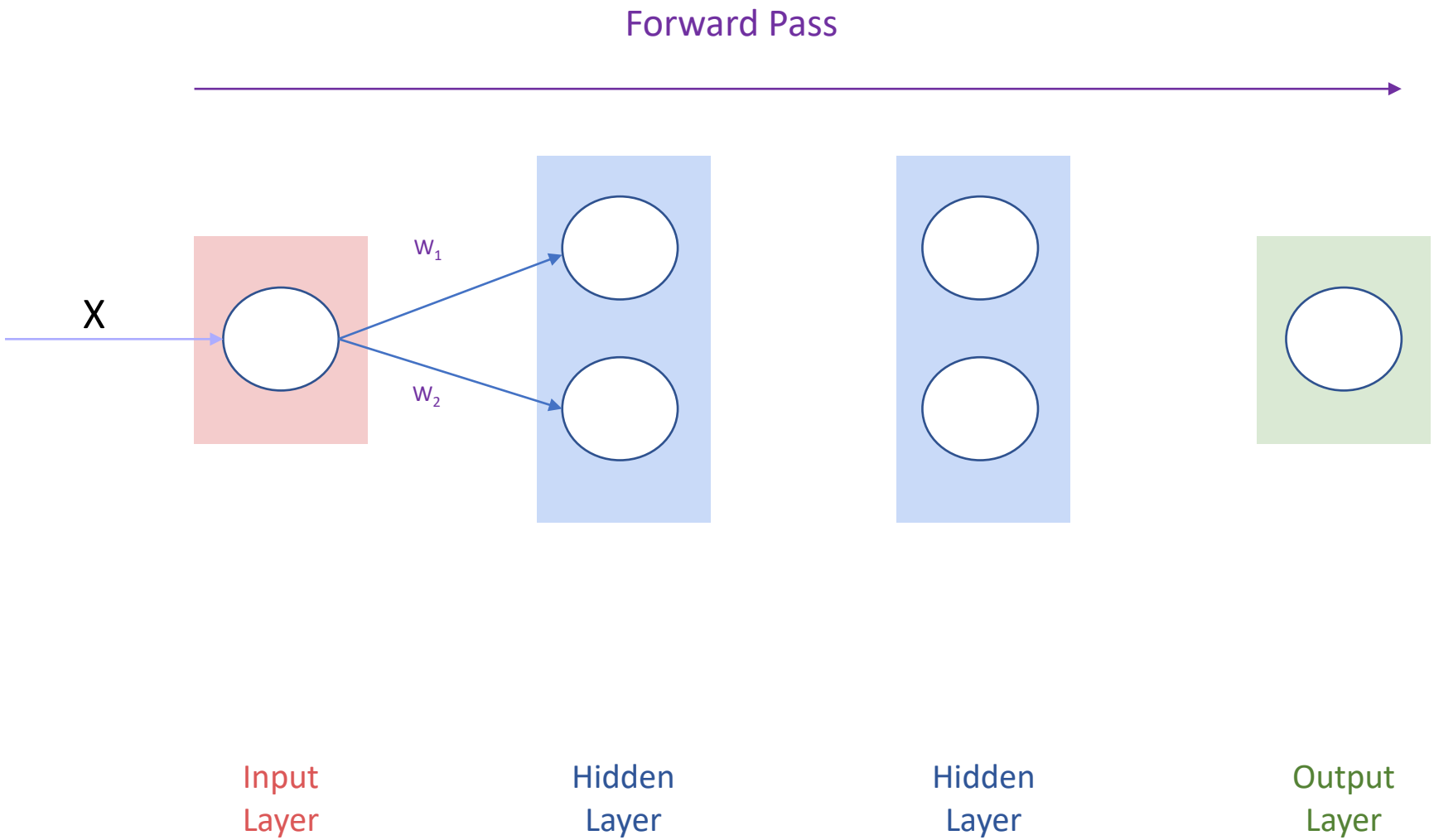


Courtesy: http://cs231n.github.io/neural-networks-1/

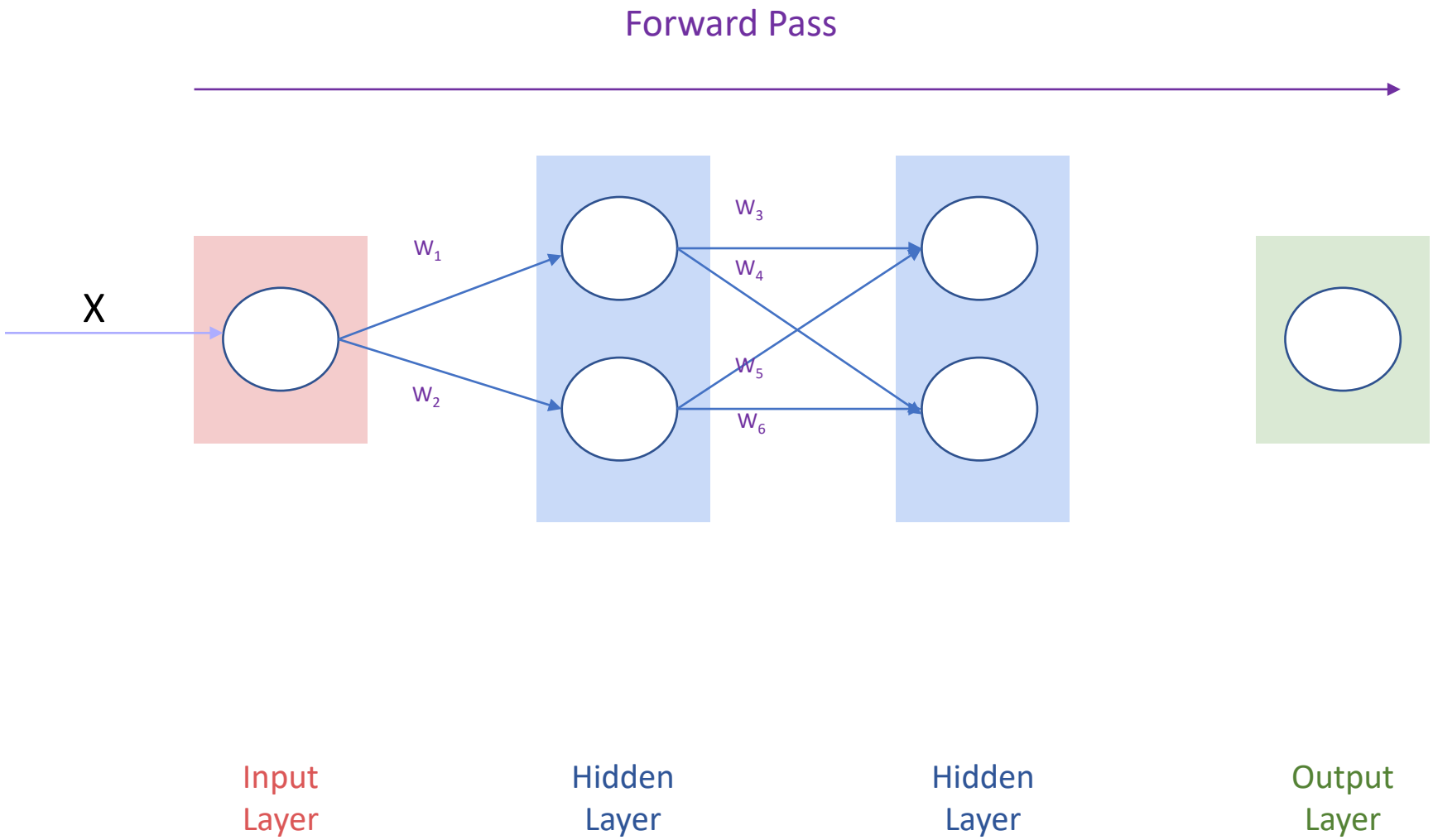
Deep Neural Network Concepts : Forward Pass



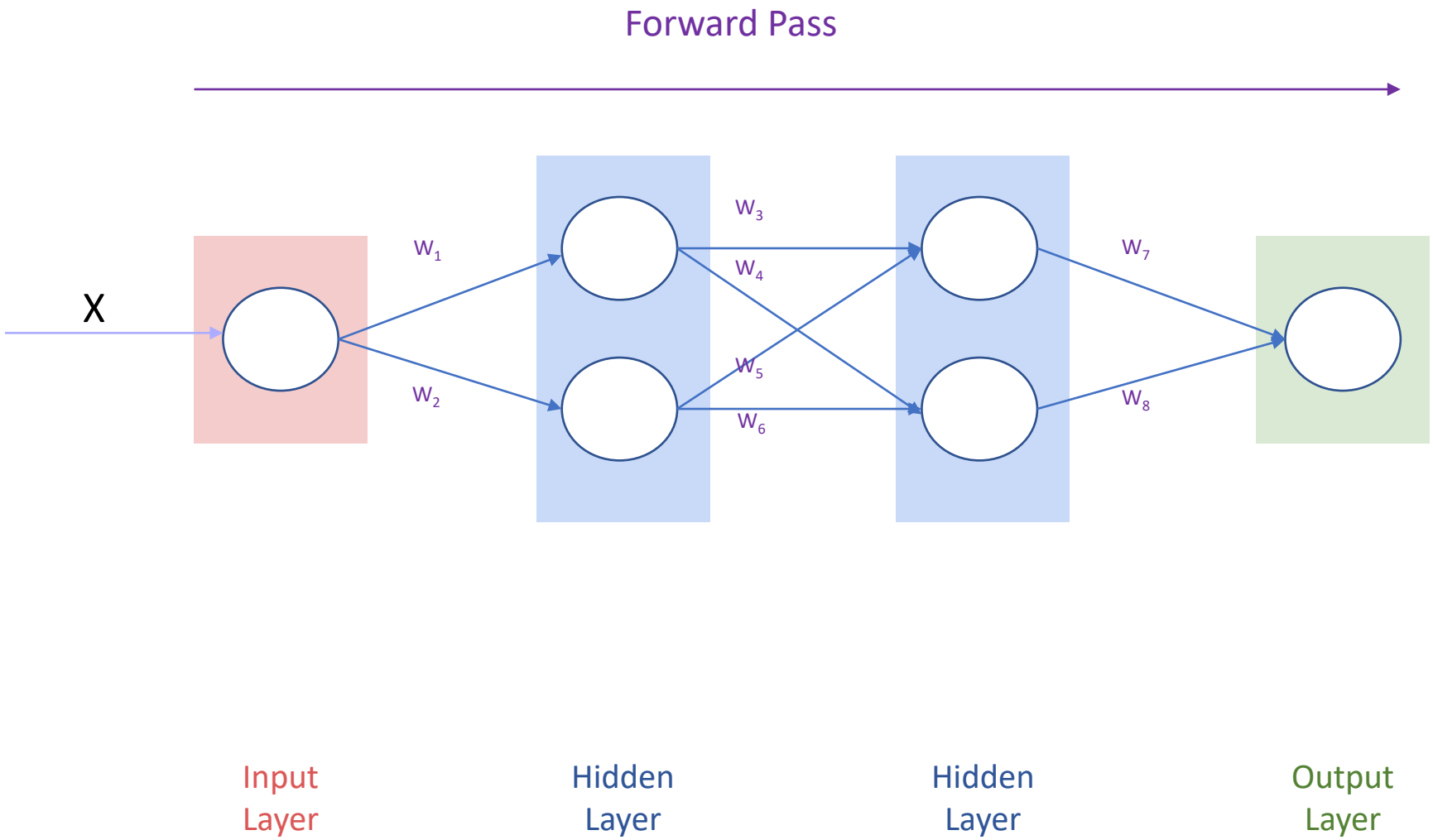
Deep Neural Network Concepts : Forward Pass



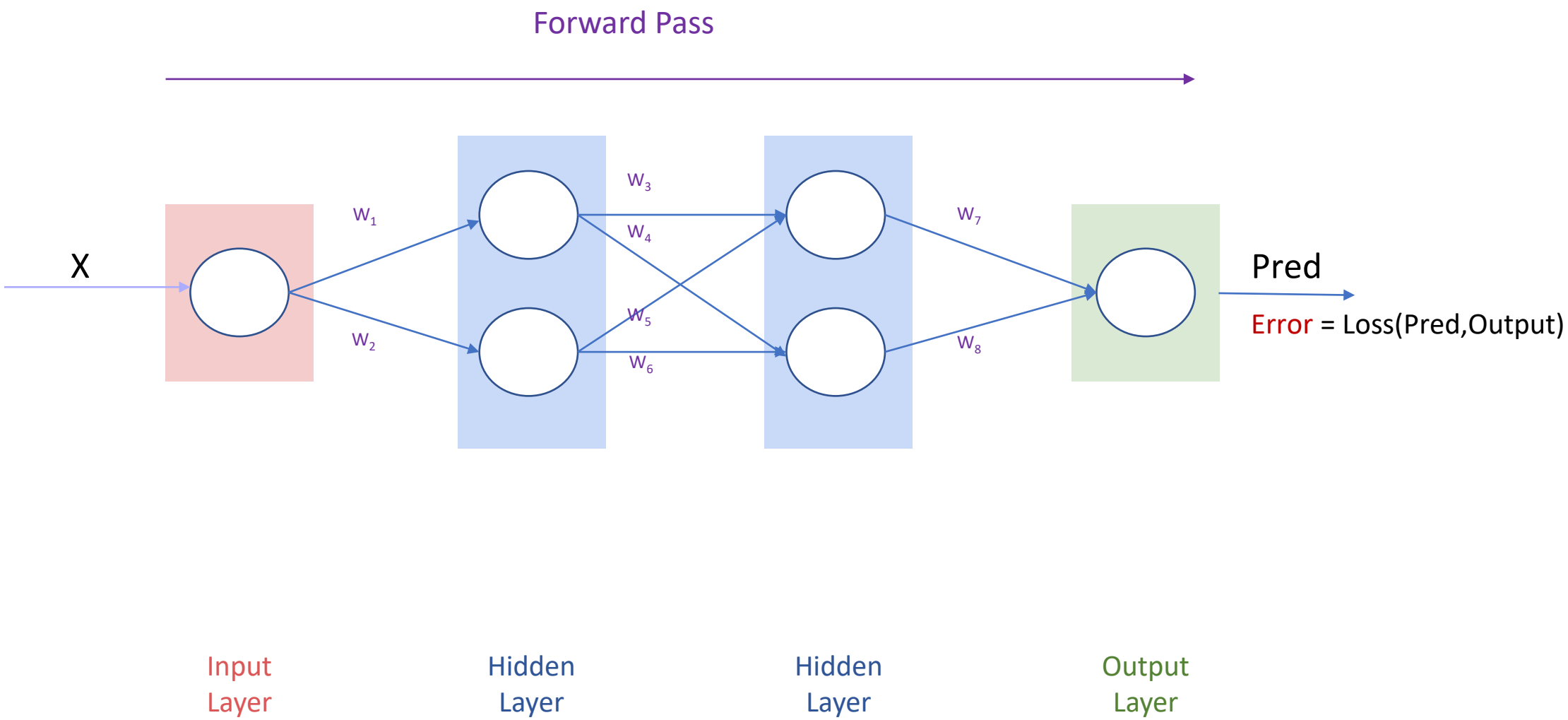
Deep Neural Network Concepts : Forward Pass



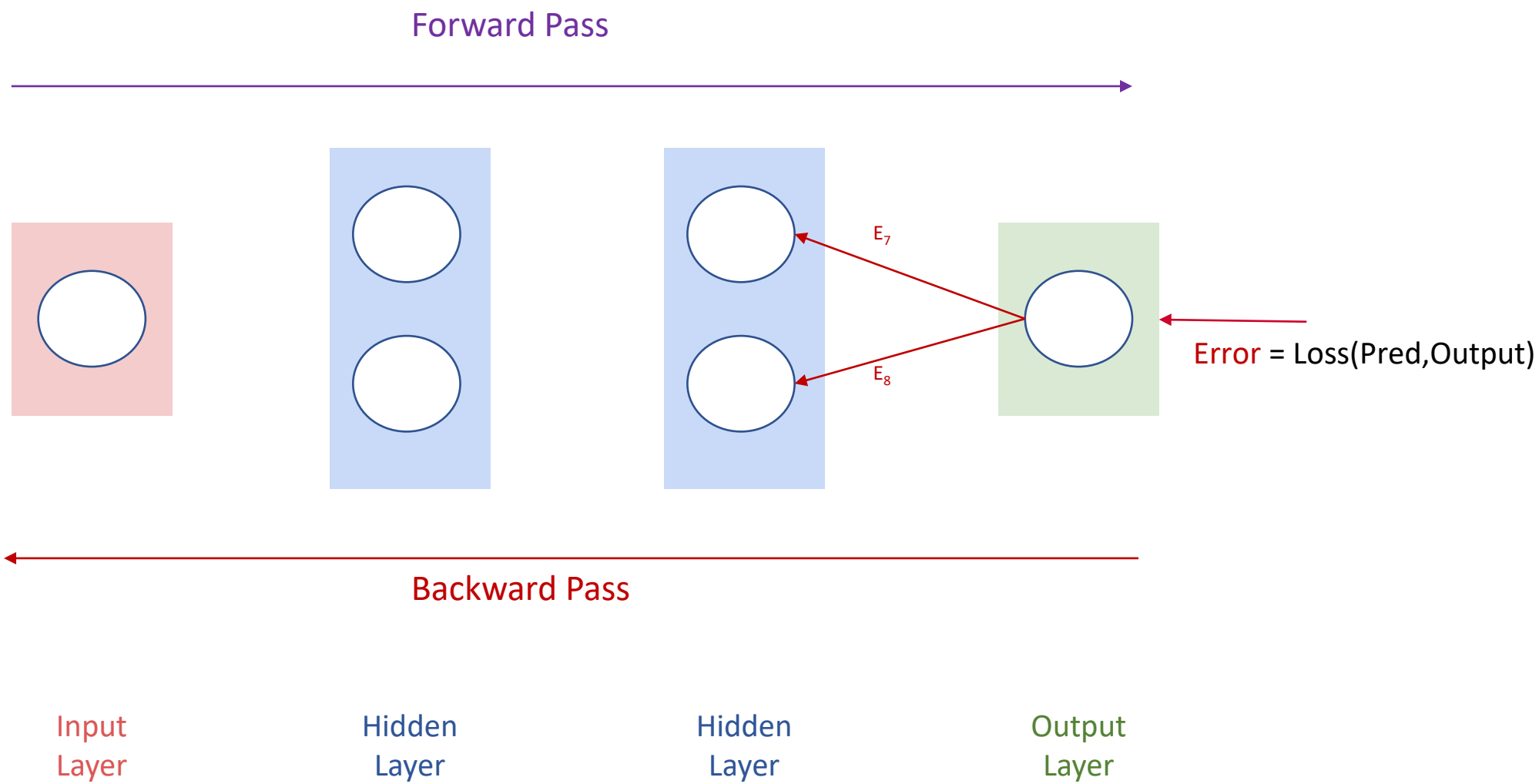
Deep Neural Network Concepts : Forward Pass



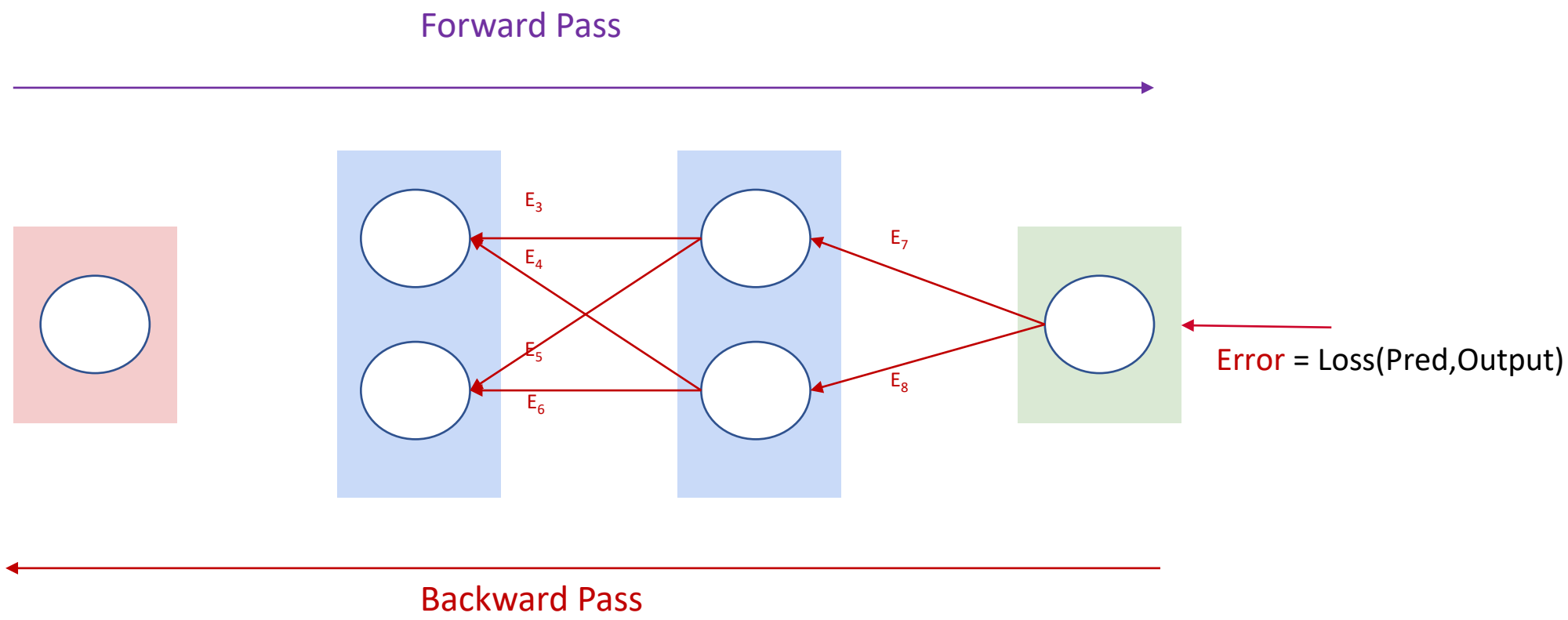
Deep Neural Network Concepts : Forward Pass



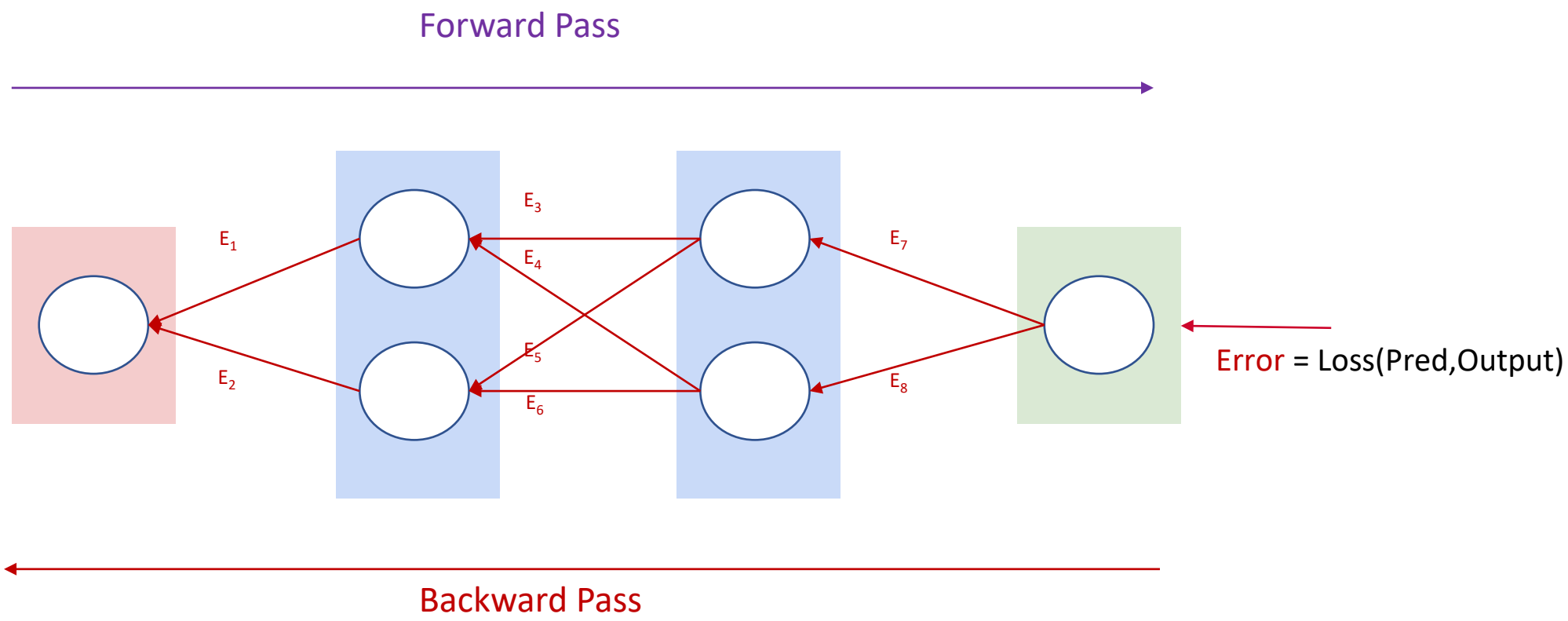
Deep Neural Network Concepts : Backward Pass



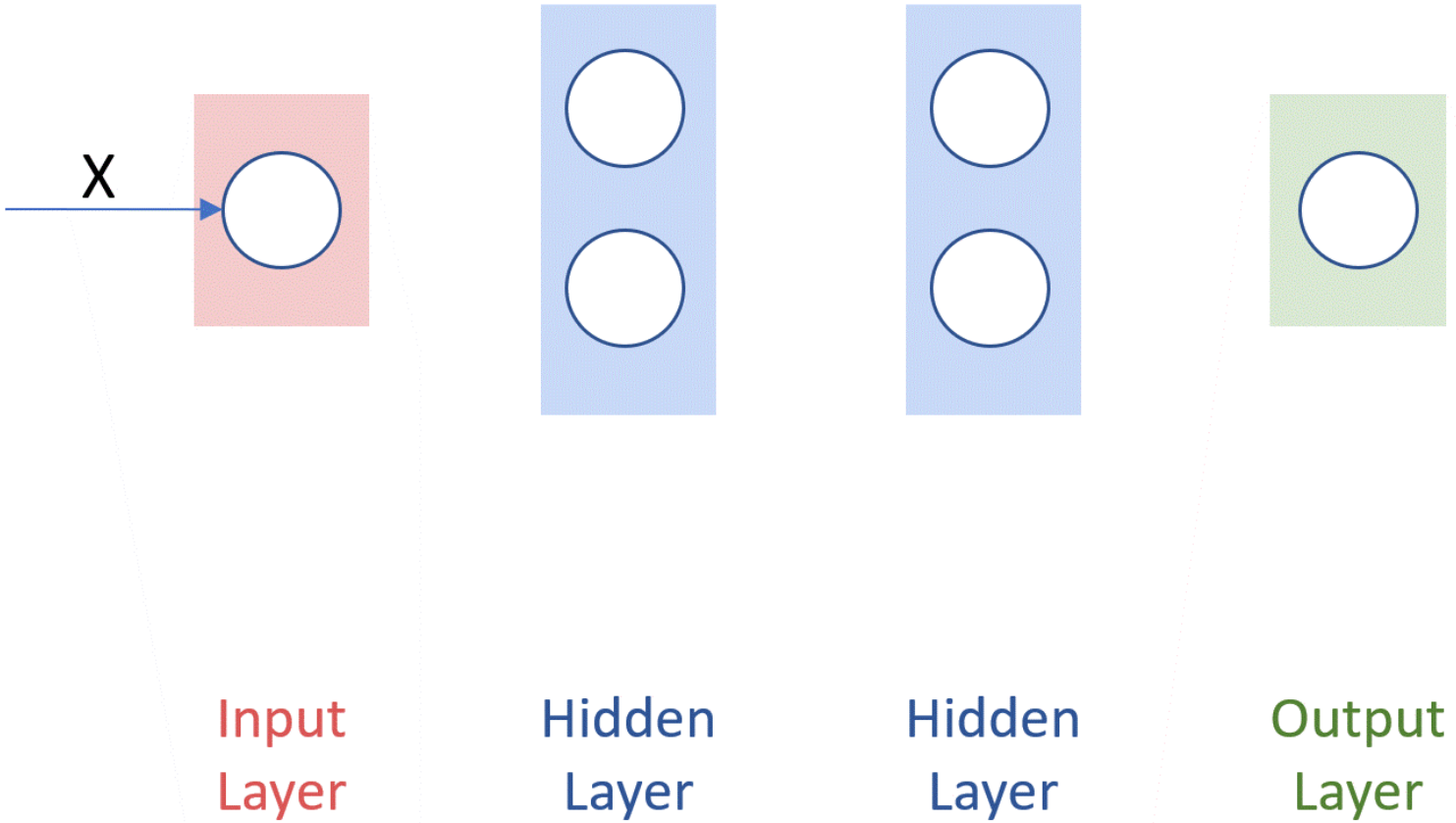
Deep Neural Network Concepts : Backward Pass



Deep Neural Network Concepts : Backward Pass

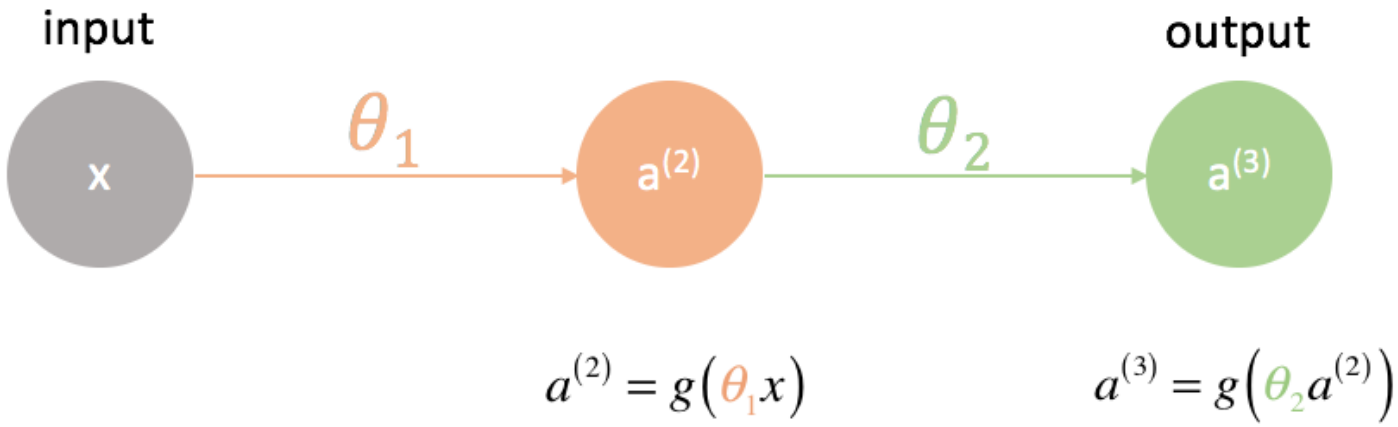


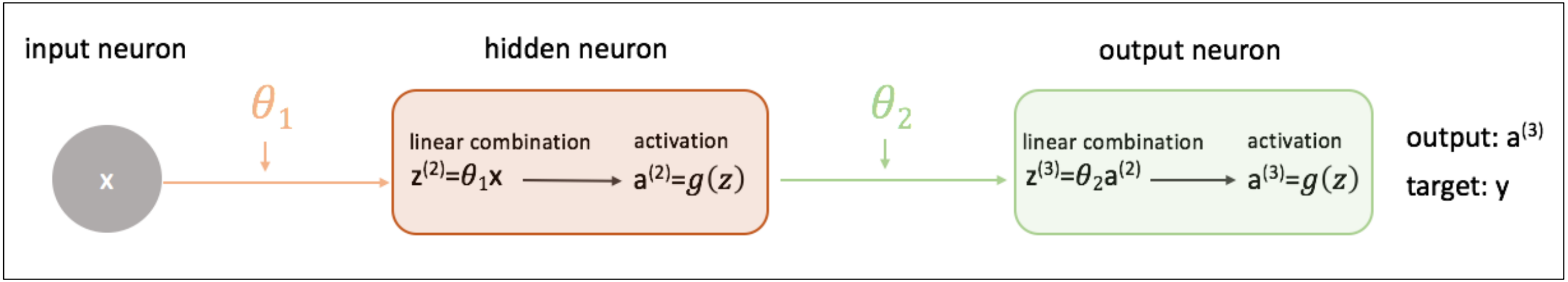
Deep Neural Network Concepts : DNN Training



Essential Concepts: Activation function and Back-propagation

- Back-propagation involves complicated mathematics.
  - Luckily, most DL Frameworks give you a one line implementation -- `model.backward()`



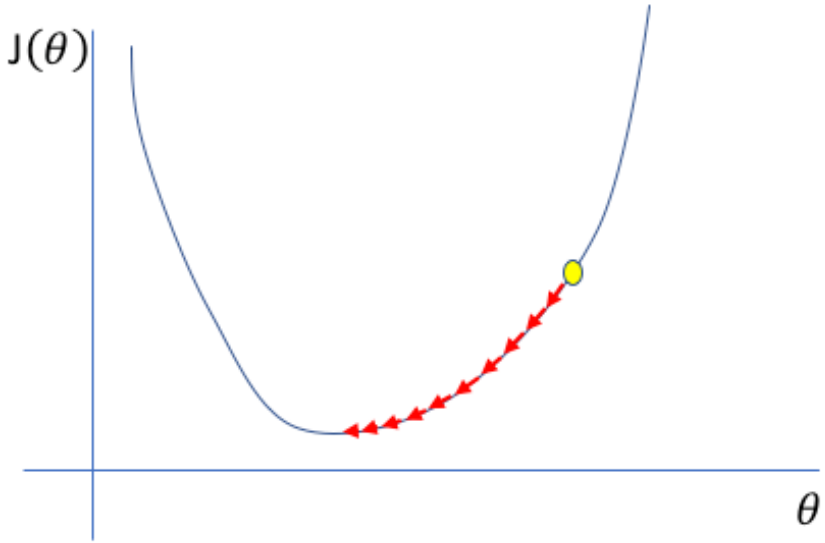


- What are Activation functions?
  - RELU (a Max fn.) is the most common activation fn.
  - Sigmoid, tanh, etc. are also used

Courtesy: https://www.jeremyjordan.me/neural-networks-training/

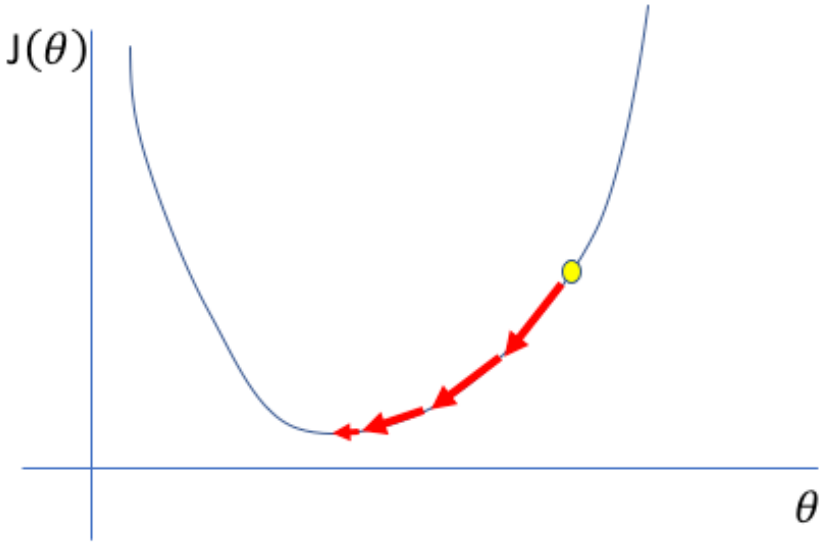
Essential Concepts: Learning Rate (α)

Too low



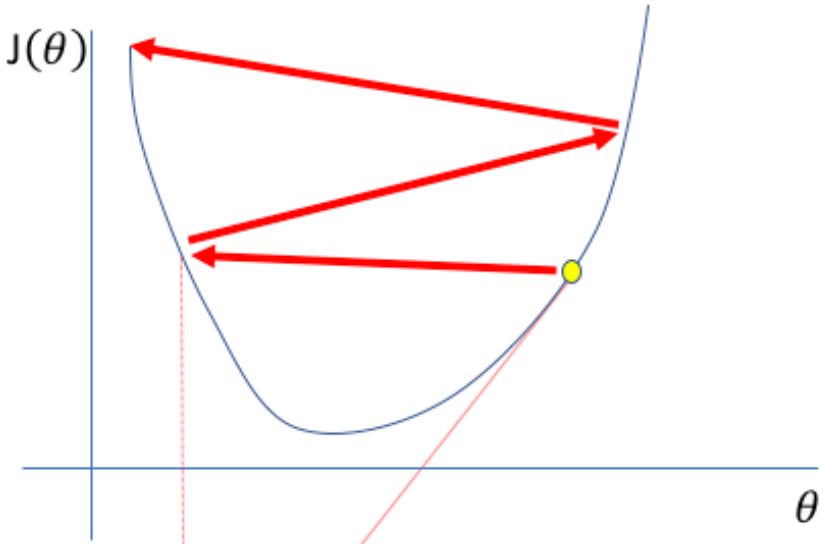
A small learning rate requires many updates before reaching the minimum point

Just right



The optimal learning rate swiftly reaches the minimum point

Too high

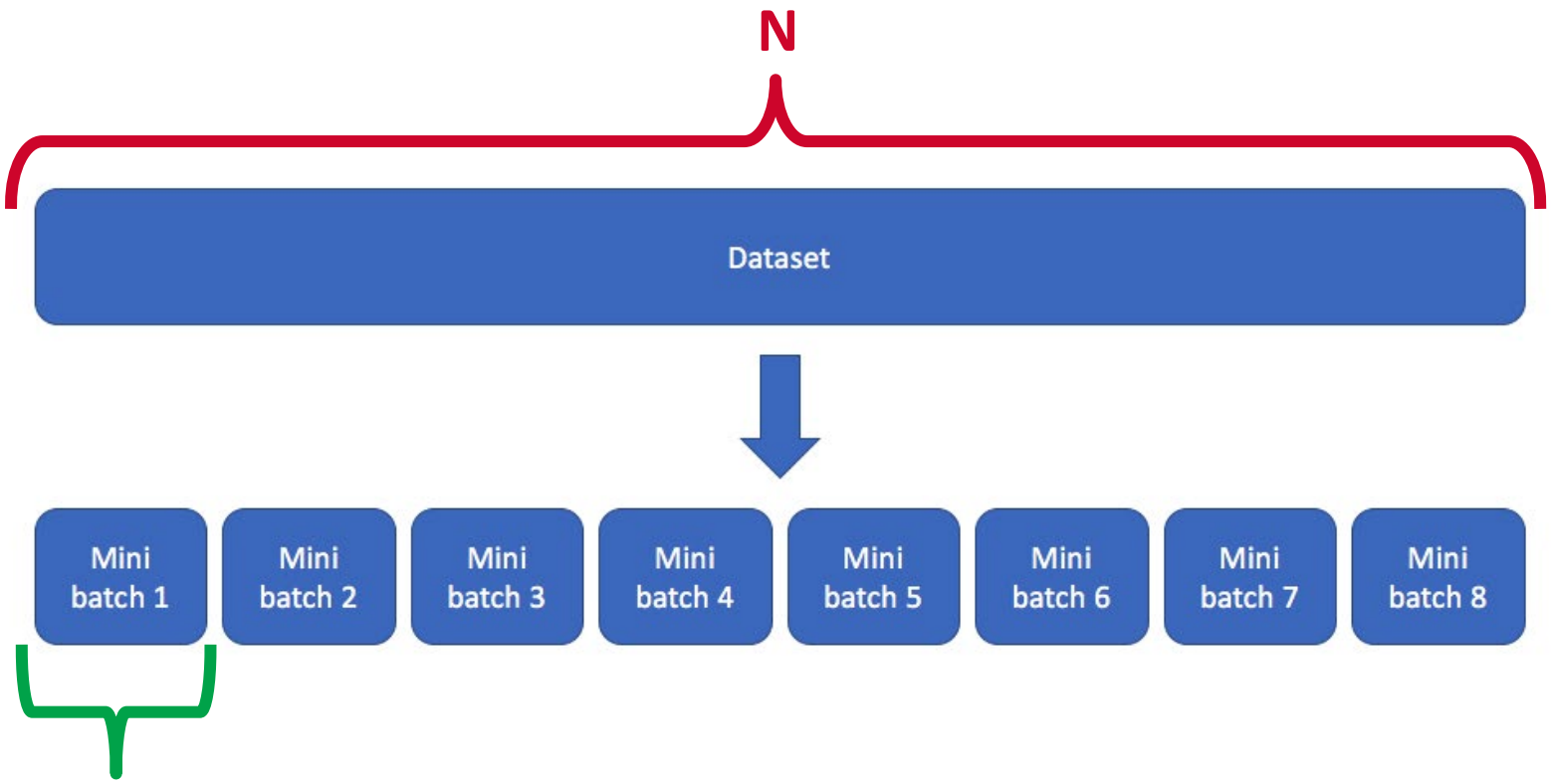


Too large of a learning rate causes drastic updates which lead to divergent behaviors

Courtesy: https://www.jeremyjordan.me/nn-learning-rate/

Essential Concepts: Batch Size

- Batched Gradient Descent
  - Batch Size = **N**
- Stochastic Gradient Descent
  - Batch Size = **1**
- Mini-batch Gradient Descent
  - Somewhere in the middle
  - Common:
    - **Batch Size** = 64, 128, 256, etc.
- Finding the optimal batch size will yield the fastest learning.



*One full pass over **N** is called an **epoch** of training*

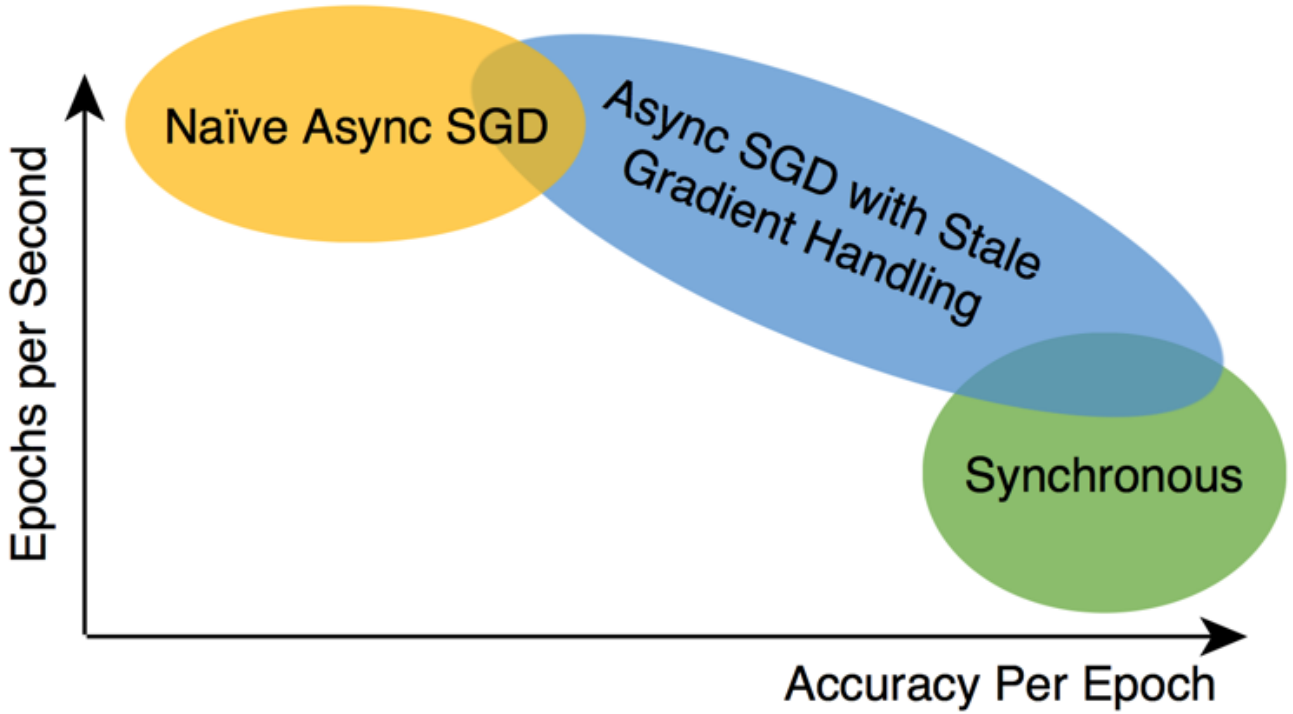
Courtesy: https://www.jeremyjordan.me/gradient-descent/

The Need for Parallel and Distributed Training

- Why do we need Parallel Training?
- Larger and Deeper models are being proposed
  - **AlexNet** -> **ResNet** -> **NASNet – AmoebaNet**
  - DNNs require a lot of memory and a lot of computation
  - Larger models cannot fit a GPU's memory
- Single GPU training cannot keep up with ever-larger models
- Community has moved to multi-GPU training
- Multi-GPU in one node is good but there is a limit to Scale-up (8-16 GPUs)
- **Multi-node (Distributed or Parallel) Training is necessary!!**

Synchronous vs. Asynchronous Training

- Epochs per second (EPS)?
  - A variant of images/second
  - Basically, what is the speed of training the model
- Accuracy per Epoch (APE)?
  - E.g. 60% in one full pass over the dataset



- Async → Higher EPS but lower APE
- Sync → Higher APE but lower EPS

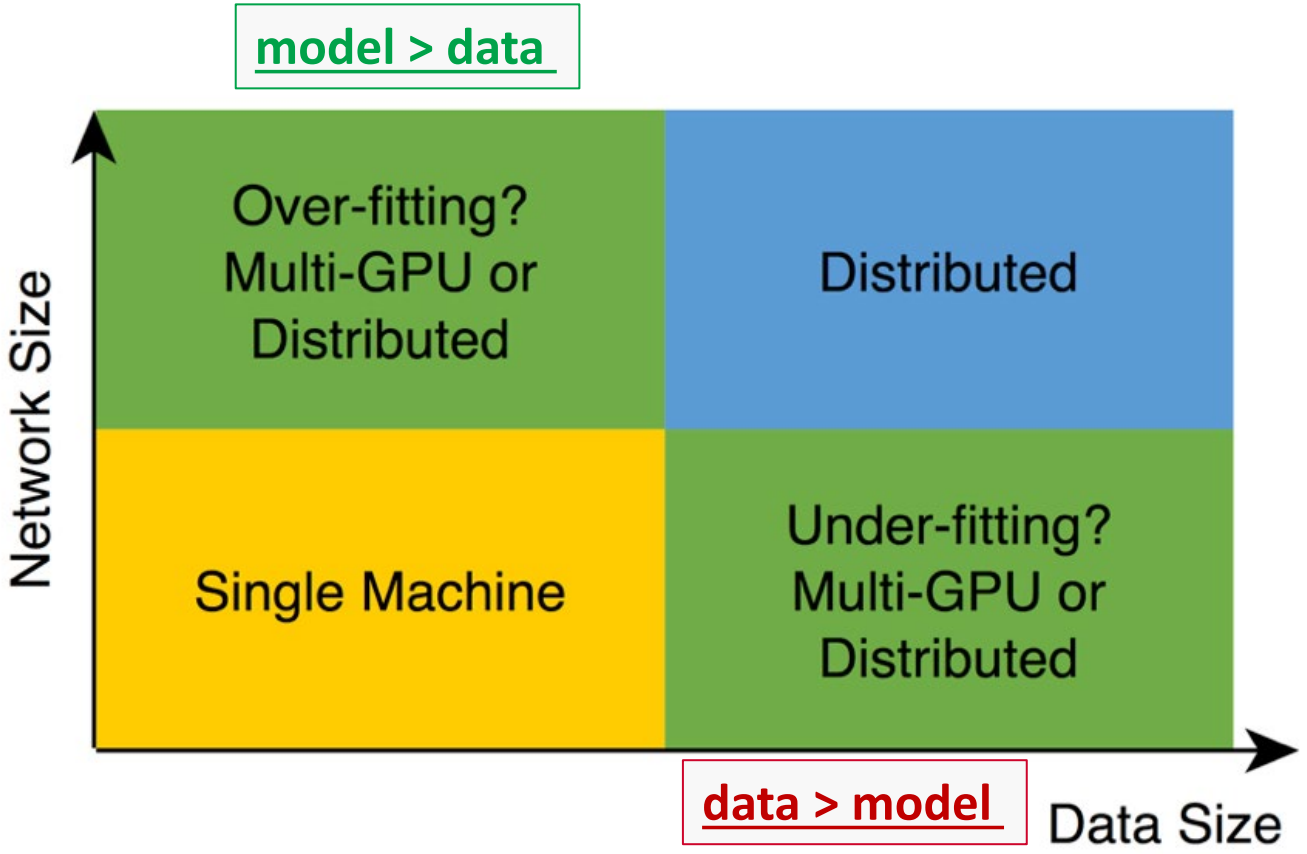
Courtesy: http://engineering.skymind.io/distributed-deep-learning-part-1-an-introduction-to-distributed-training-of-neural-networks

Impact of Model Size and Dataset Size

- *Large models → better accuracy*
- *More data → better accuracy*

- Single-node Training; good for
  - Small model and small dataset

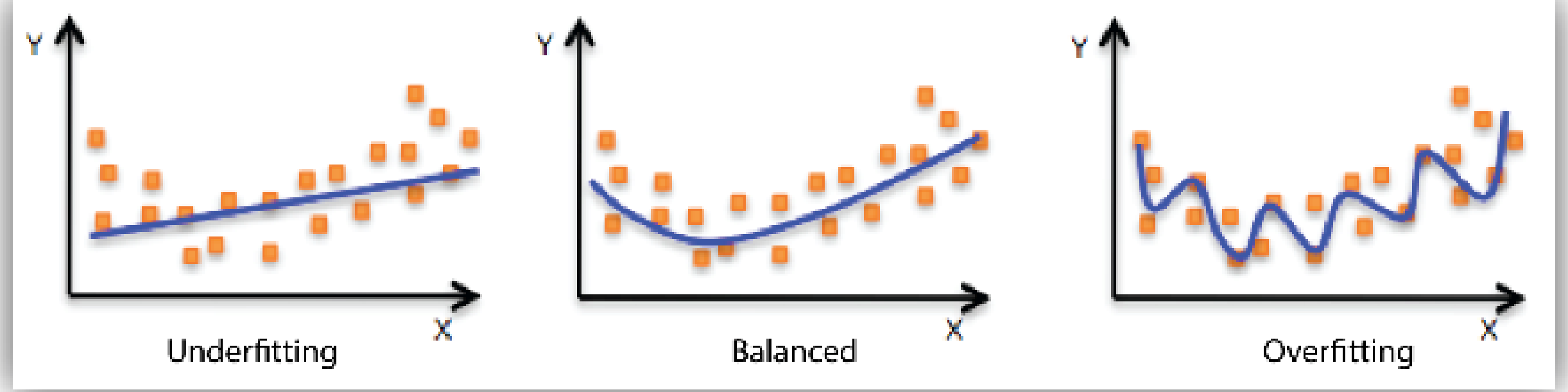
- Distributed Training; good for:
  - Large models and large datasets



Courtesy: http://engineering.skymind.io/distributed-deep-learning-part-1-an-introduction-to-distributed-training-of-neural-networks

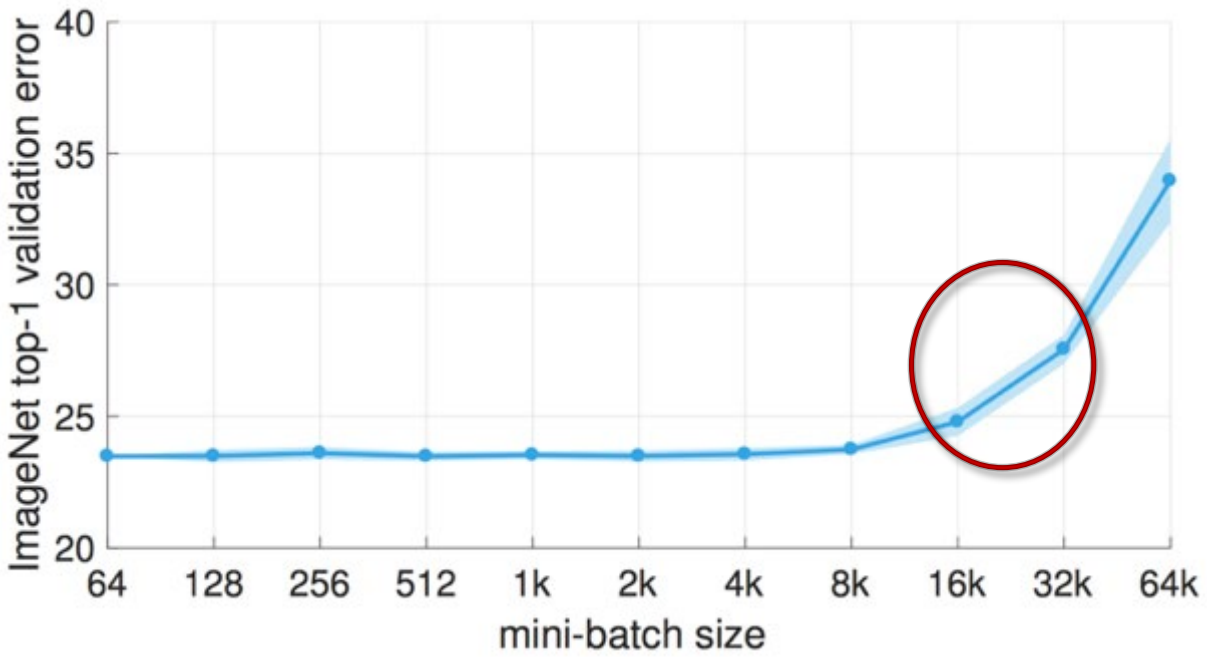
Overfitting and Underfitting

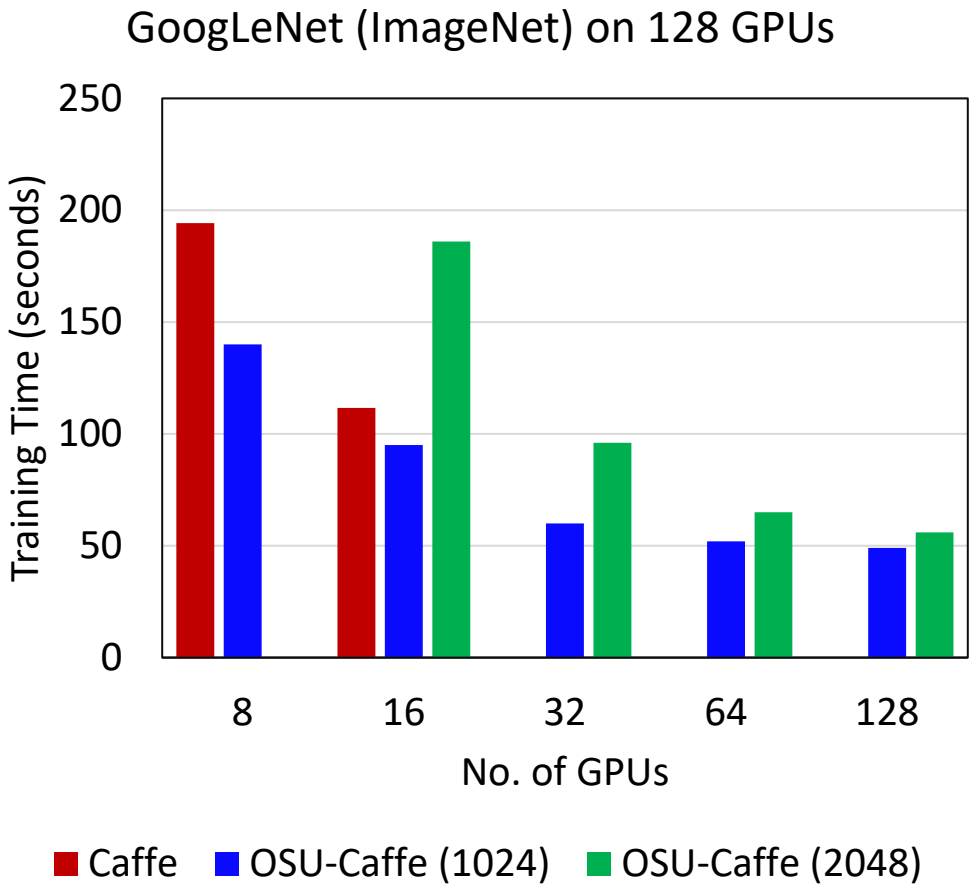
- Overfitting – **model > data** → so model is not learning but memorizing your data
- Underfitting – **data > model** → so model is not learning because it cannot capture the complexity of your data



Courtesy: https://docs.aws.amazon.com/machine-learning/latest/dg/model-fit-underfitting-vs-overfitting.html

Impact of Large Batch Size





*Large Batch Size is **bad** for Accuracy*

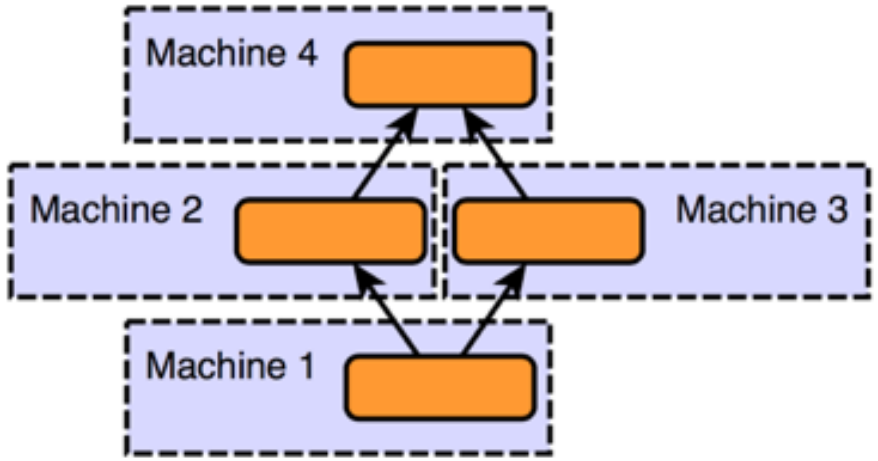
*But **good** for speed and scalability*

Courtesy: https://research.fb.com/publications/imagenet1kin1h/

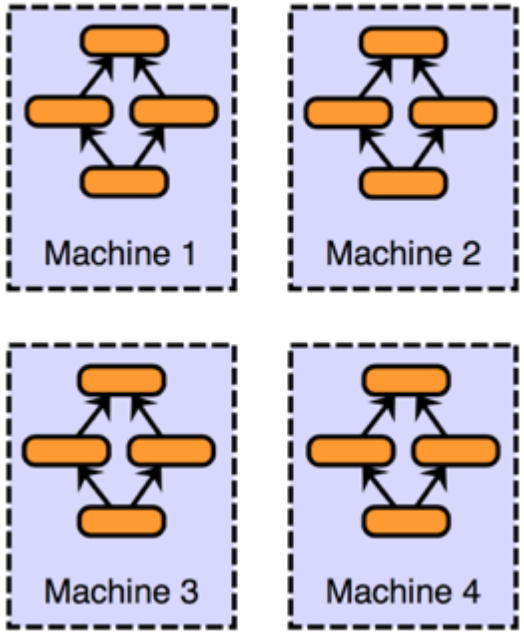
A. A. Awan et al., S-Caffe: Co-designing MPI Runtimes and Caffe for Scalable Deep Learning on Modern GPU Clusters. PPoPP '17

Parallelization Strategies

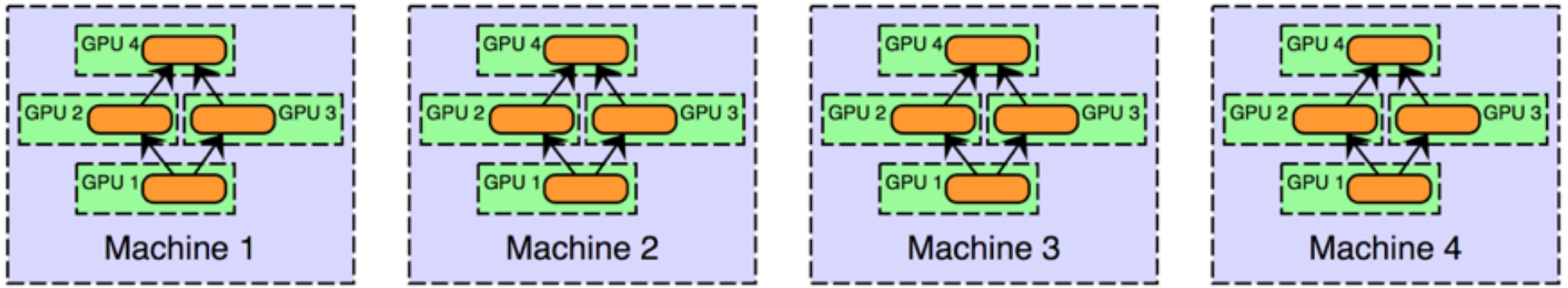
- Some parallelization strategies..
  - Data Parallelism or Model Parallelism
  - Hybrid Parallelism



Model Parallelism



Data Parallelism

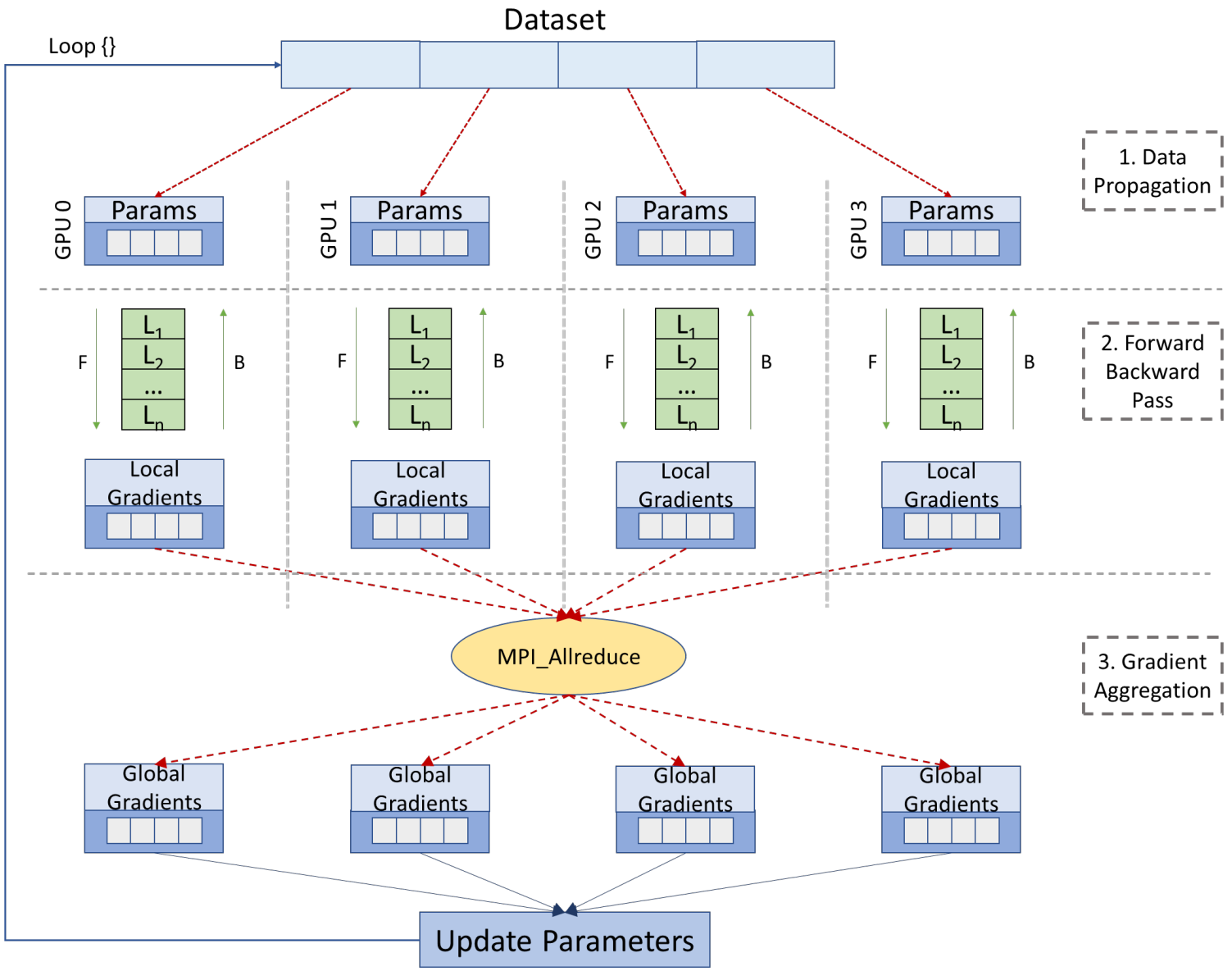


Hybrid (Model and Data) Parallelism

Courtesy: http://engineering.skymind.io/distributed-deep-learning-part-1-an-introduction-to-distributed-training-of-neural-networks

Data Parallelism and MPI Collectives

- Step1: Data Propagation
  - Distribute the Data among GPUs
- Step2: Forward Backward Pass
  - Perform forward pass and calculate the prediction
  - Calculate Error by comparing prediction with actual output
  - Perform backward pass and calculate gradients
- Step3: Gradient Aggregation
  - Call MPI_Allreduce to reduce the local gradients
  - Update parameters locally using global gradients



Benefits of Data Parallel Training: Using OSU-Caffe

- Strong scaling CIFAR10 Training with OSU-Caffe (1 –> 4 GPUs) – **Batch Size 2K**
- Large batch size is needed for scalability.
- Adding more GPUs will degrade the scaling efficient

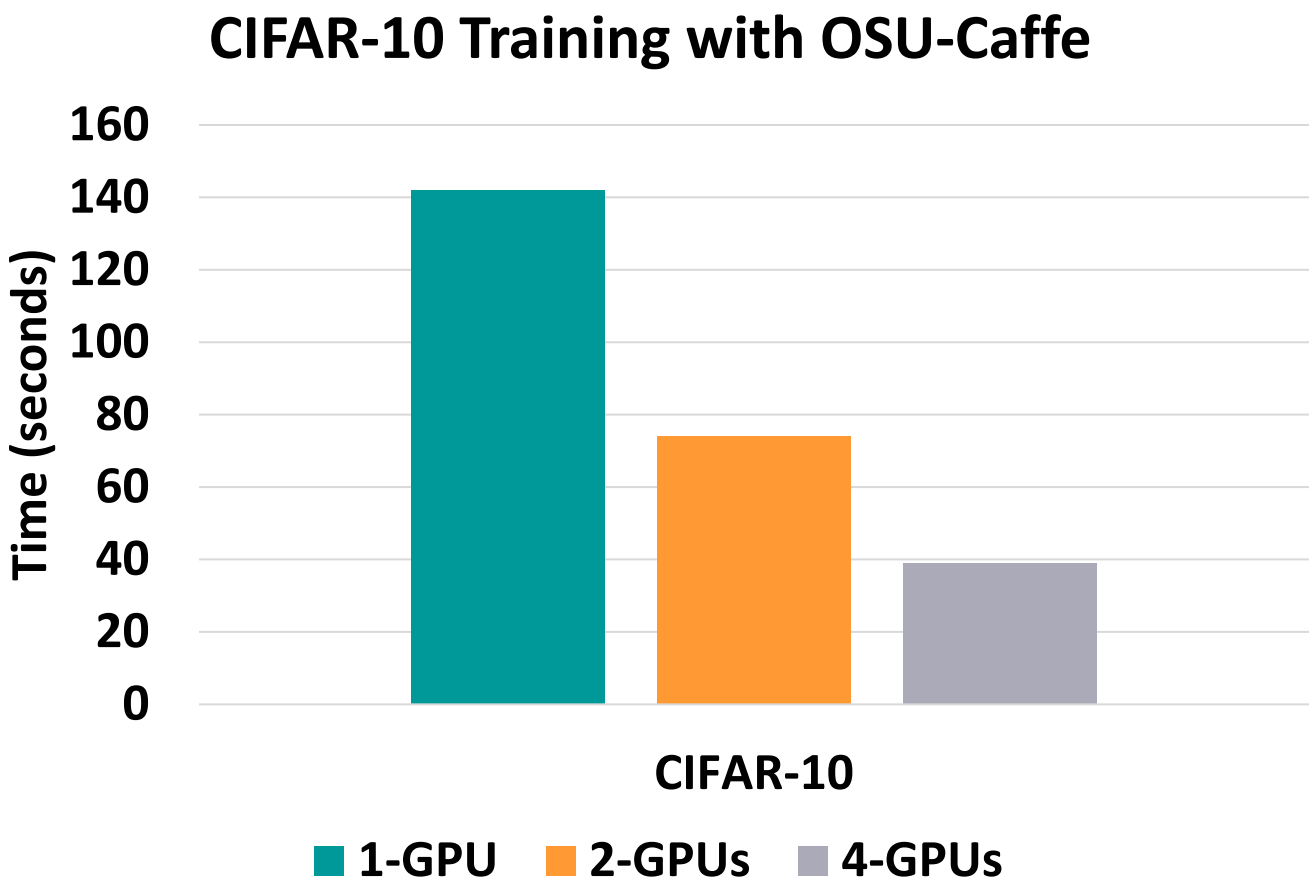
Run Command - (change $np from 1—4)

mpirun_rsh -np **$np** ./build/tools/caffe train -solver examples/cifar10/cifar10_quick_solver.prototxt -scal **strong**

Output: I0123 21:49:24.289763 75582 caffe.cpp:351] Avg. Time Taken: 142.101

Output: I0123 21:54:03.449211 97694 caffe.cpp:351] Avg. Time Taken: 74.6679

Output: I0123 22:02:46.858219 20659 caffe.cpp:351] Avg. Time Taken: 39.8109



OSU-Caffe is available from the HiDL project page
http://hidl.cse.ohio-state.edu

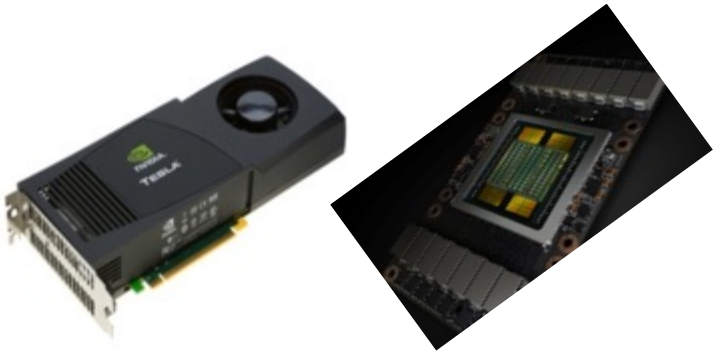
Outline

- Introduction
- Overview of Execution Environments
- Parallel and Distributed DNN Training
- **Latest Trends in High-Performance Computing Architectures**
- Challenges in Exploiting HPC Technologies for Deep Learning
- Solutions and Case Studies
- Open Issues and Challenges
- Hands-on Exercises
- Conclusion

High-Performance Architectures: Drivers of Modern HPC Clusters

Multi-/Many-core Processors

High Performance Interconnects - InfiniBand
<1usec latency, 200Gbps Bandwidth>



Accelerators
high compute density, high performance/watt
>1 TFlop DP on a chip

SSD, NVMe-SSD, NVRAM

- Multi-core/many-core technologies
- Remote Direct Memory Access (RDMA)-enabled networking (InfiniBand and RoCE)
- Solid State Drives (SSDs), Non-Volatile Random-Access Memory (NVRAM), NVMe-SSD
- Accelerators (NVIDIA GPGPUs)
- Available on HPC Clouds, e.g., Amazon EC2, NSF Chameleon, Microsoft Azure, etc.

Summit

Sunway TaihuLight

Sierra

K - Computer

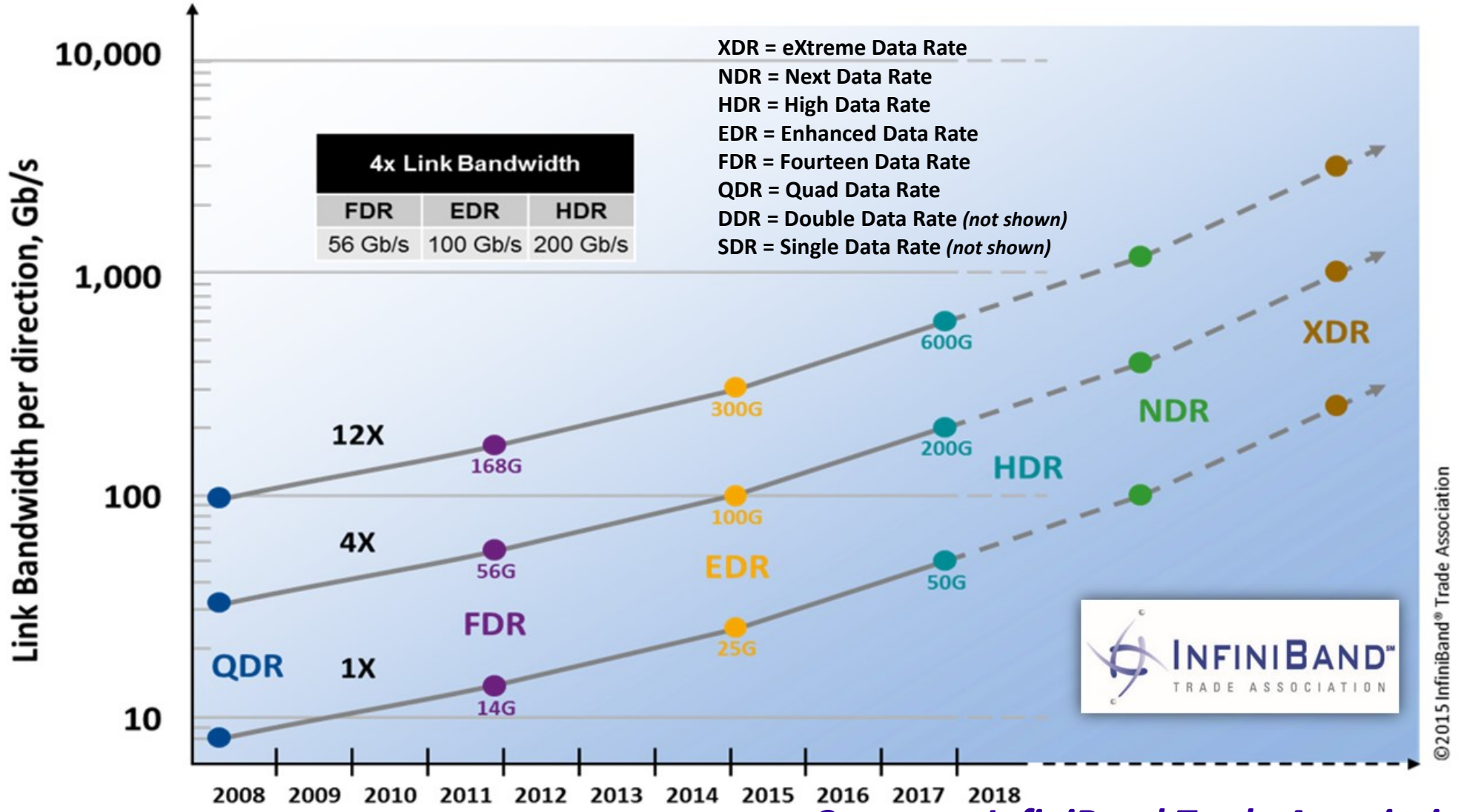
High-Performance Architectures for Distributed DL

- **Hardware Architectures**
  - **Interconnects**
    - **InfiniBand, RoCE, Omni-Path, Slingshot, etc.**
  - **Processors**
    - **GPUs, Multi-/Many-core CPUs, Tensor Processing Unit (TPU), FPGAs, etc.**
- Communication Middleware
  - Message Passing Interface (MPI)
    - CUDA-Aware MPI
  - NVIDIA NCCL

Overview of High Performance Interconnects

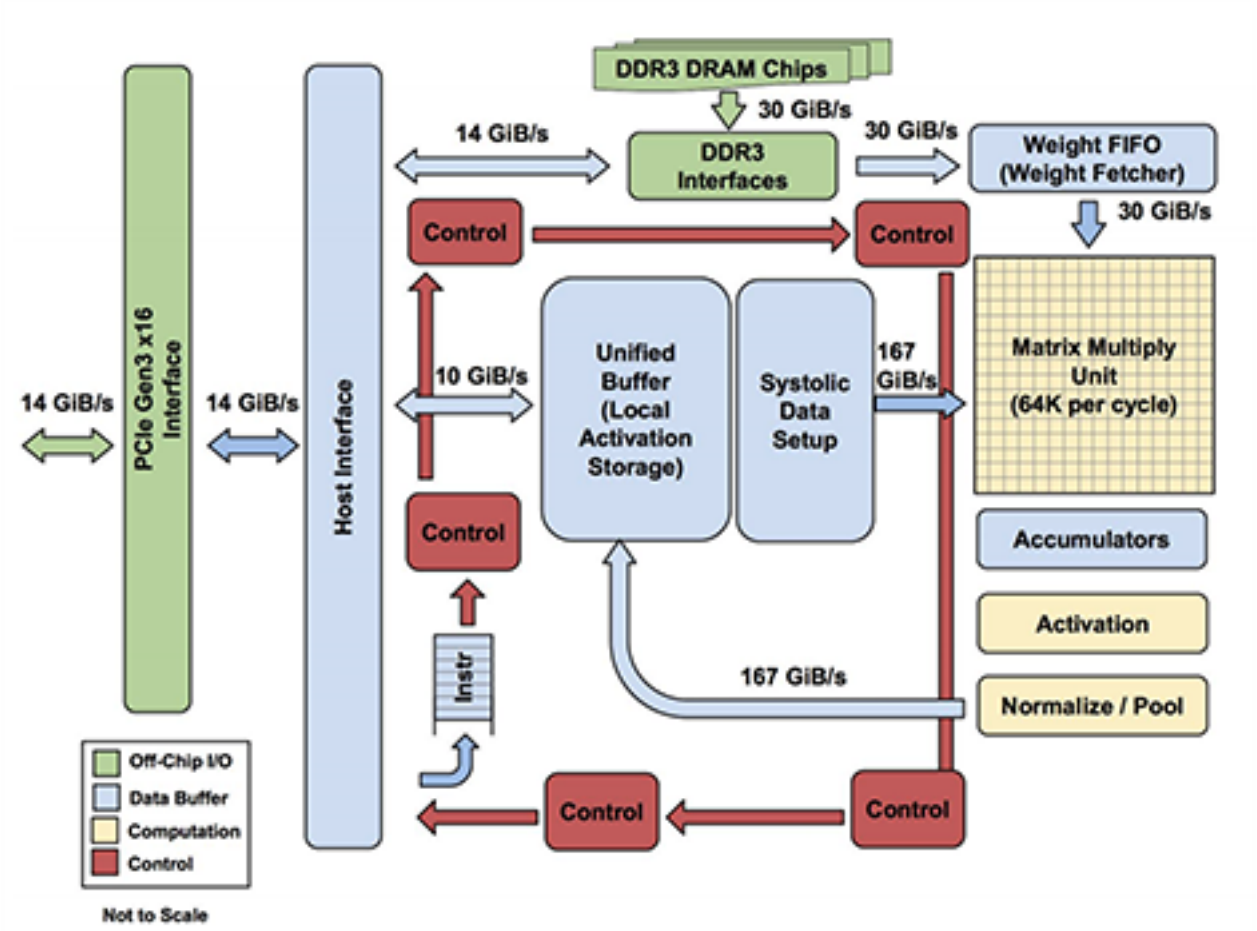
- High-Performance Computing (HPC) has adopted advanced interconnects and protocols
  - InfiniBand (IB)
  - Omni-Path
  - High Speed Ethernet 10/25/40/50/100/200 Gigabit Ethernet/iWARP
  - RDMA over Converged Enhanced Ethernet (RoCE)
- Very Good Performance
  - Low latency (few micro seconds)
  - High Bandwidth (200 Gb/s with HDR InfiniBand)
  - Low CPU overhead (5-10%)
- OpenFabrics software stack with IB, Omni-Path, iWARP and RoCE interfaces are driving HPC systems
- Many such systems in Top500 list

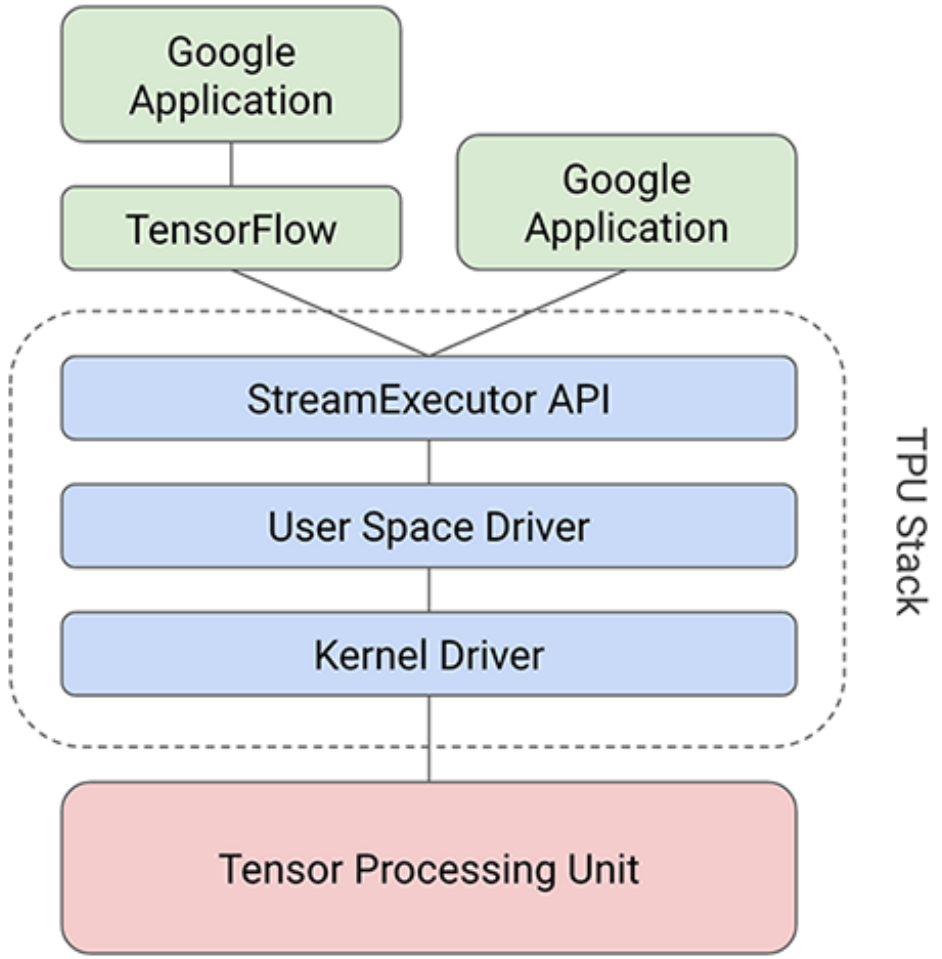
InfiniBand Link Speed Standardization Roadmap



Courtesy: InfiniBand Trade Association

Hardware for DNN Training and Inference: TPUs





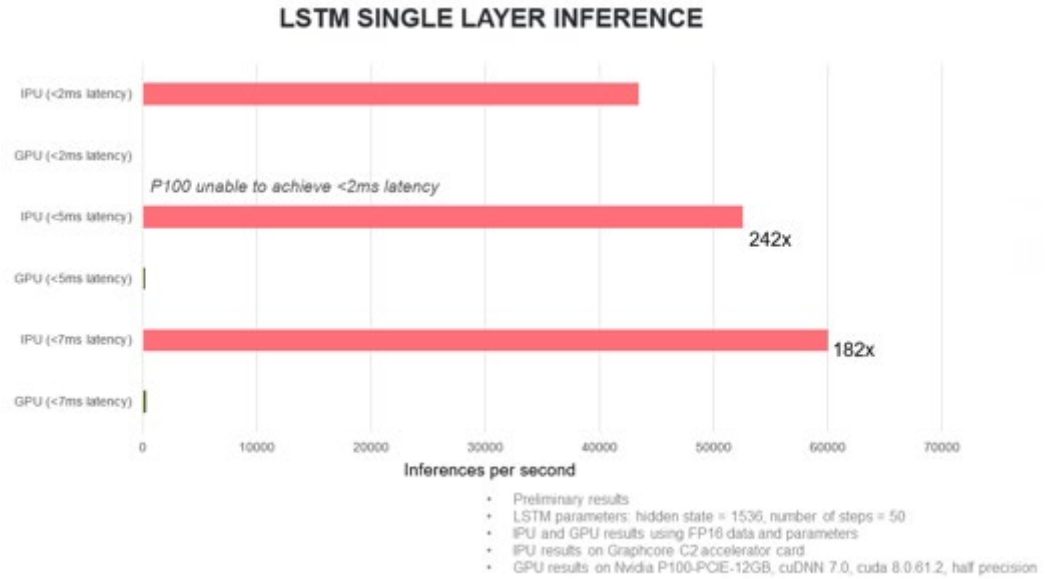
- CISC style instruction set
- Uses systolic arrays as the heart of multiply unit

Courtesy: https://cloud.google.com/blog/big-data/2017/05/an-in-depth-look-at-googles-first-tensor-processing-unit-tpu

: https://www.nextplatform.com/2017/04/05/first-depth-look-googles-tpu-architecture/

Hardware for DNN Training and Inference: IPUs

- Specifically designed for AI workloads – an Intelligence Processing Unit (IPU)
  - Massively parallel
  - Low-precision floating-point compute
  - Higher compute density
- Early benchmarks show 10-100x speedup over GPUs
  - Presented at NIPS 2017
- HPC Wire: Microsoft Azure IPU instances
  https://www.hpcwire.com/2019/11/15/microsoft-azure-adds-graphcores-ipu/



Courtesy: https://www.graphcore.ai/posts/preliminary-ipu-benchmarks-providing-previously-unseen-performance-for-a-range-of-machine-learning-applications

Hardware for DNN Training: Habana Gaudi

- Habana Labs – Training Accelerator called Gaudi – (HotChips '19)
- Gaudi – AI processor with RoCE integrated
- Gaudi software – Enables high-level frameworks
- ***Intel has acquired Habana for $2 billion!***

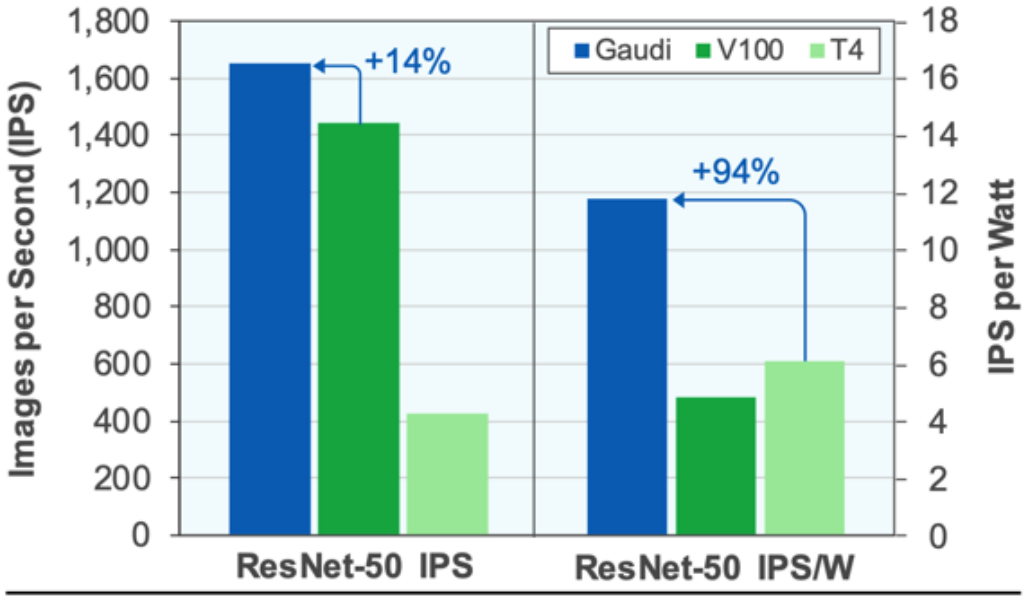
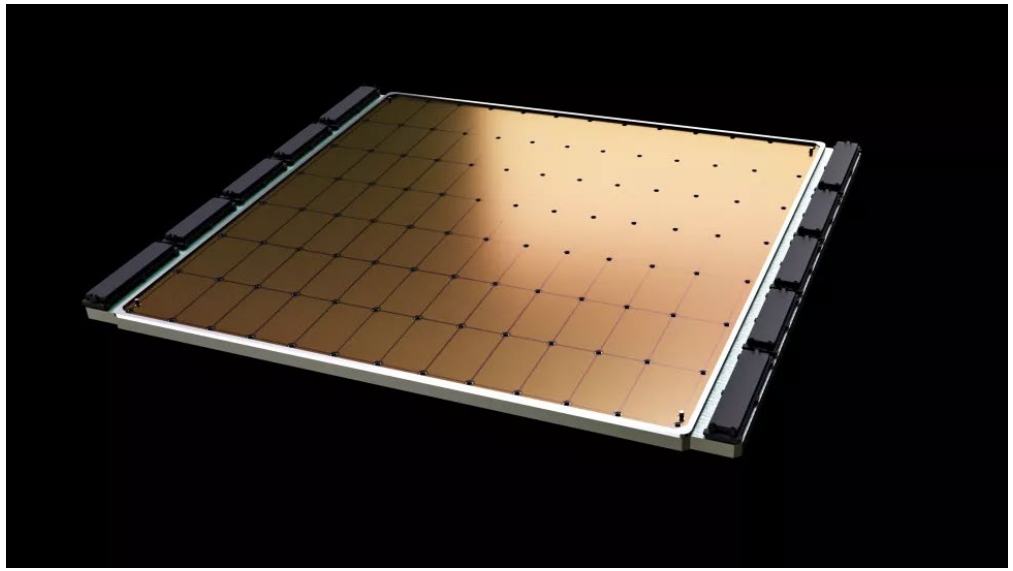


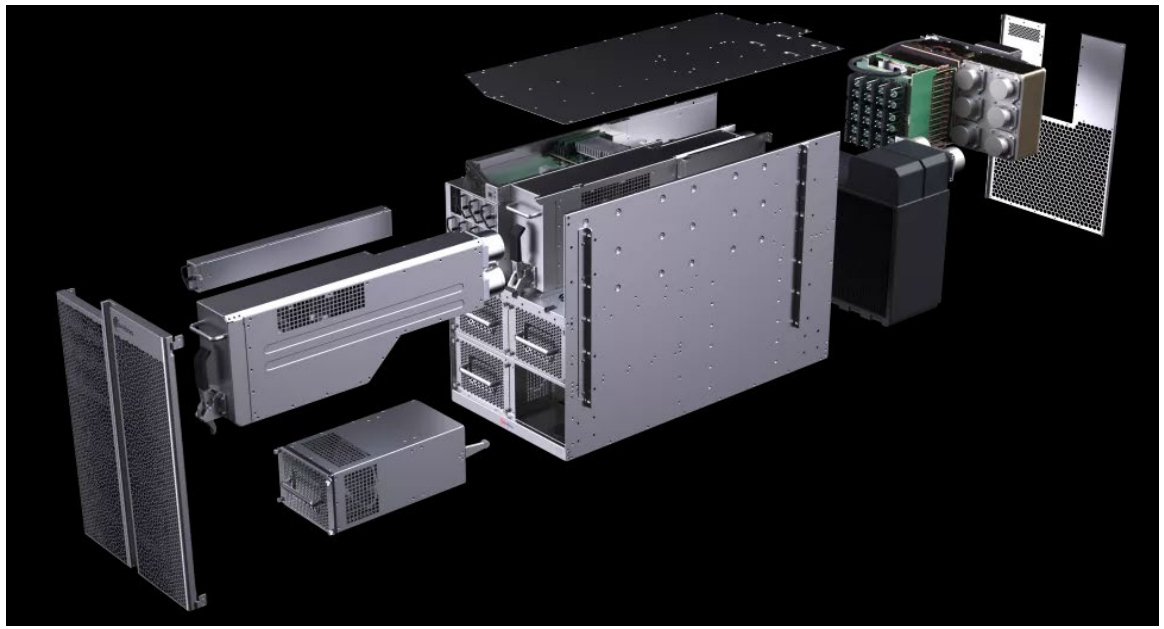
Figure 1. Gaudi emulated performance. For training the simple ResNet-50 model, Habana's Gaudi card offers throughput similar to that of Nvidia's high-end V100 GPU at half the power. It also beats Nvidia's Tesla T4 card in performance per watt.

Courtesy: https://habana.ai/wp-content/uploads/2019/06/Habana-Offers-Gaudi-for-AI-Training.pdf

Hardware for DNN Training: Cerebras

- Cerebras: First-Gen Wafer-Scale Engine (WSE) contains 400,000 Sparse Linear Algebra Compute (SLAC) Cores
- Swarm Communication fabric in a 2D mesh with 100 Pb/s of bandwidth
- Teased World's Largest Chip with 2.6 Trillion 7nm Transistors and 850000 Cores (HotChips '20)



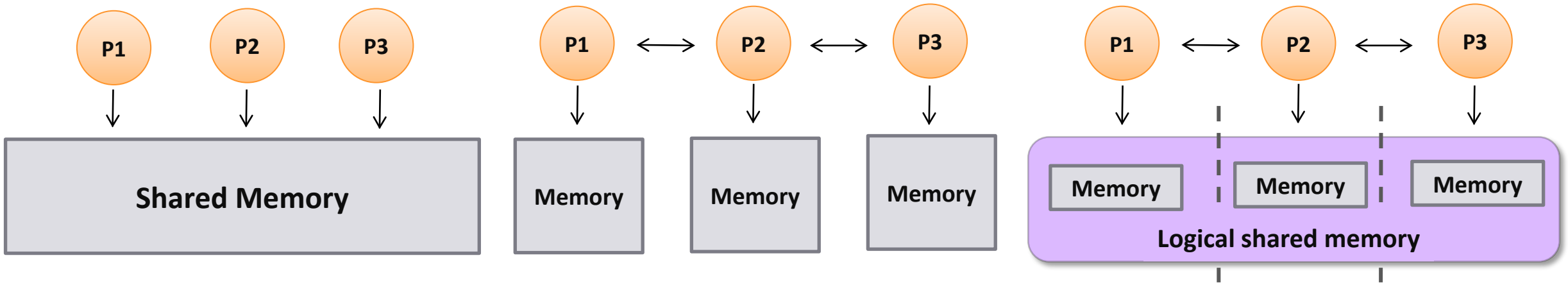


Courtesy: https://www.cerebras.net/product/#chip, https://www.tomshardware.com/news/worlds-biggest-chip-cerebras-7nm-26-trillion-transistors-850000-cores-wafer-scale-engine

High-Performance Architectures for Distributed DL

- Hardware Architectures
  - Interconnects
    - InfiniBand, RoCE, Omni-Path, etc.
  - Processors
    - GPUs, Multi-/Many-core CPUs, Tensor Processing Unit (TPU), FPGAs, etc.
- **Communication Middleware**
  - **Message Passing Interface (MPI)**
    - **CUDA-Aware MPI**
  - **NVIDIA NCCL**

Parallel Programming Models Overview



Shared Memory Model

SHMEM, DSM

Distributed Memory Model

MPI (Message Passing Interface)

Partitioned Global Address Space (PGAS)

OpenSHMEM, UPC, Chapel, X10, CAF, …

- Programming models provide abstract machine models
- Models can be mapped on different types of systems
  - e.g. Distributed Shared Memory (DSM), MPI within a node, etc.
- PGAS models and Hybrid MPI+PGAS models are gradually receiving importance

Allreduce Collective Communication Pattern

- Element-wise Sum data from all processes and sends to all processes

int MPI_Allreduce (const void *sendbuf, void * recvbuf, int count, MPI_Datatype datatype, MPI_Op operation, MPI_Comm comm)

Input-only Parameters	
Parameter	**Description**
sendbuf	Starting address of send buffer
recvbuf	Starting address of recv buffer
type	Data type of buffer elements
count	Number of elements in the buffers
operation	Reduction operation to be performed (e.g. sum)
comm	Communicator handle

Input/Output Parameters	
Parameter	**Description**
recvbuf	Starting address of receive buffer

Sendbuf (Before)

Recvbuf (After)

Overview of the MVAPICH2 Project

- High Performance open-source MPI Library
- Support for multiple interconnects
  - InfiniBand, Omni-Path, Ethernet/iWARP, RDMA over Converged Ethernet (RoCE), and AWS EFA
- Support for multiple platforms
  - x86, OpenPOWER, ARM, Xeon-Phi, GPGPUs (NVIDIA and AMD)
- Started in 2001, first open-source version demonstrated at SC '02
- Supports the latest MPI-3.1 standard
- http://mvapich.cse.ohio-state.edu
- Additional optimized versions for different systems/environments:
  - MVAPICH2-X (Advanced MPI + PGAS), since 2011
  - MVAPICH2-GDR with support for NVIDIA GPGPUs, since 2014
  - MVAPICH2-MIC with support for Intel Xeon-Phi, since 2014
  - MVAPICH2-Virt with virtualization support, since 2015
  - MVAPICH2-EA with support for Energy-Awareness, since 2015
  - MVAPICH2-Azure for Azure HPC IB instances, since 2019
  - MVAPICH2-X-AWS for AWS HPC+EFA instances, since 2019
- Tools:
  - OSU MPI Micro-Benchmarks (OMB), since 2003
  - OSU InfiniBand Network Analysis and Monitoring (INAM), since 2015

- **Used by more than 3,175 organizations in 89 countries**
- **More than 1.39 Million downloads from the OSU site directly**
- Empowering many TOP500 clusters (Nov '20 ranking)
  - 4th , 10,649,600-core (Sunway TaihuLight) at NSC, Wuxi, China
  - 9th, 448, 448 cores (Frontera) at TACC
  - 14th, 391,680 cores (ABCI) in Japan
  - 21st, 570,020 cores (Nurion) in South Korea and many others
- Available with software stacks of many vendors and Linux Distros (RedHat, SuSE, OpenHPC, and Spack)
- Partner in the 9th ranked TACC Frontera system
- **Empowering Top500 systems for more than 16 years**

GPU-Aware (CUDA-Aware) MPI Library: MVAPICH2-GDR

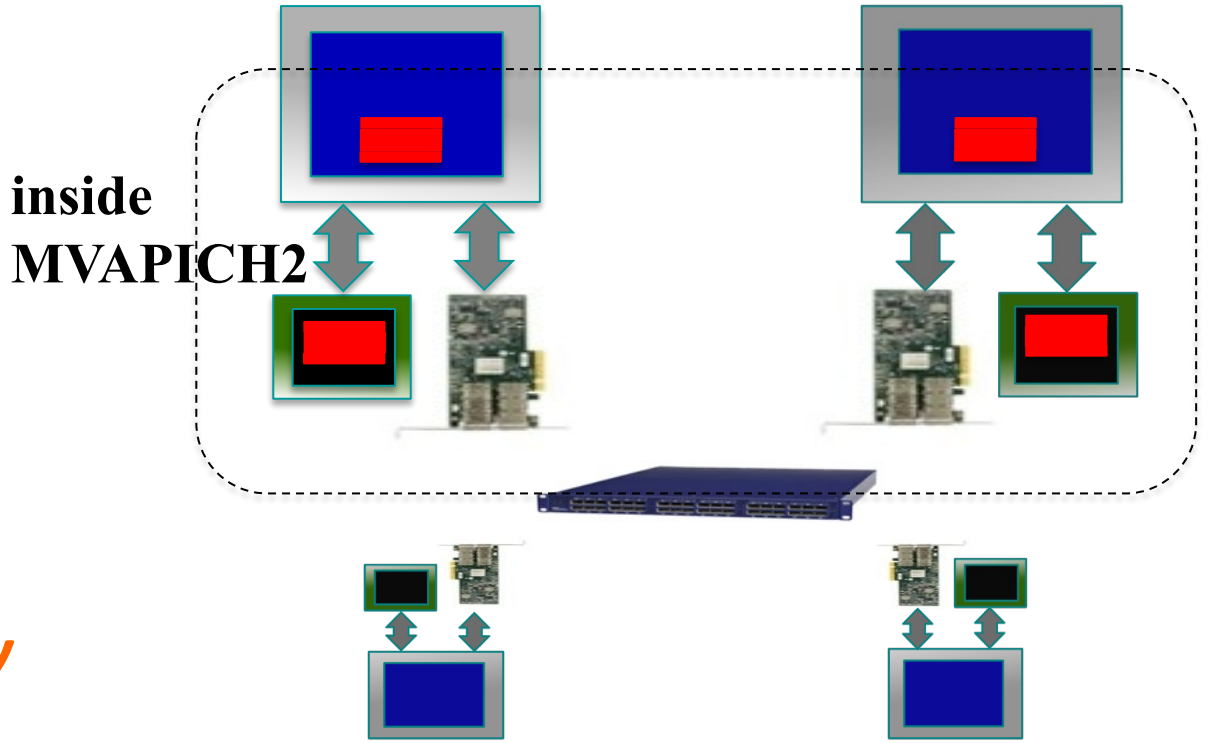
- Standard MPI interfaces used for unified data movement
- Takes advantage of Unified Virtual Addressing (>= CUDA 4.0)
- Overlaps data movement from GPU with RDMA transfers

At Sender:

```
MPI_Send(s_devbuf, size, …);
```

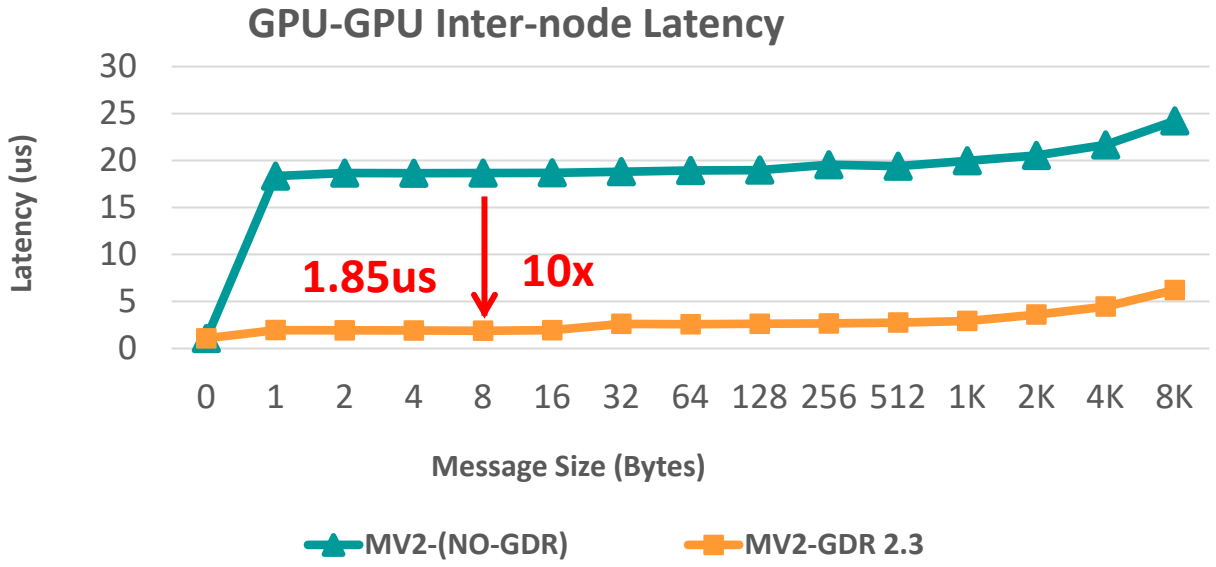
At Receiver:

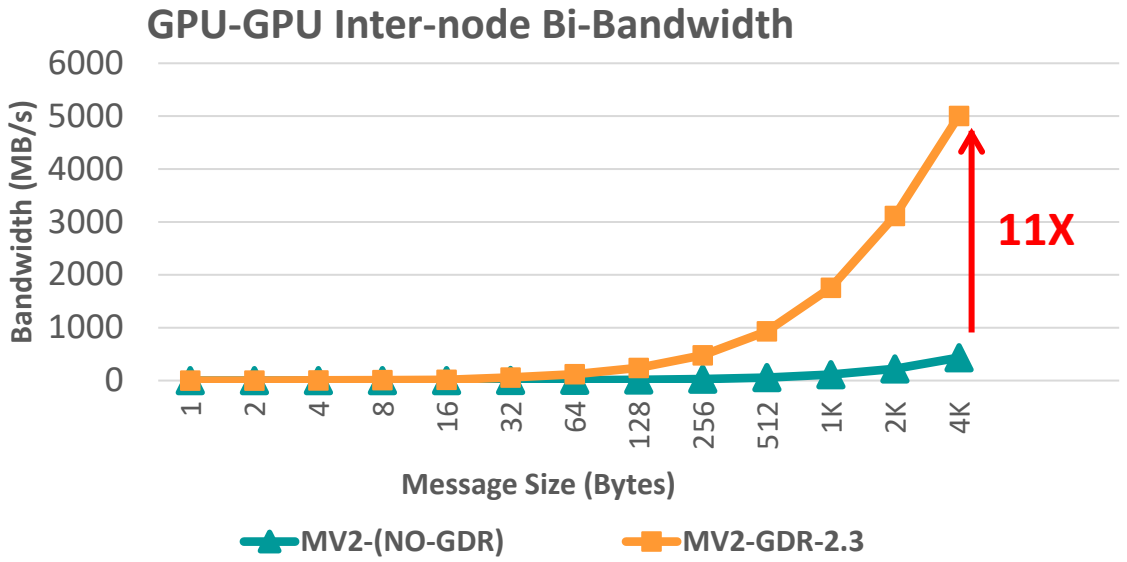
```
MPI_Recv(r_devbuf, size, …);
```

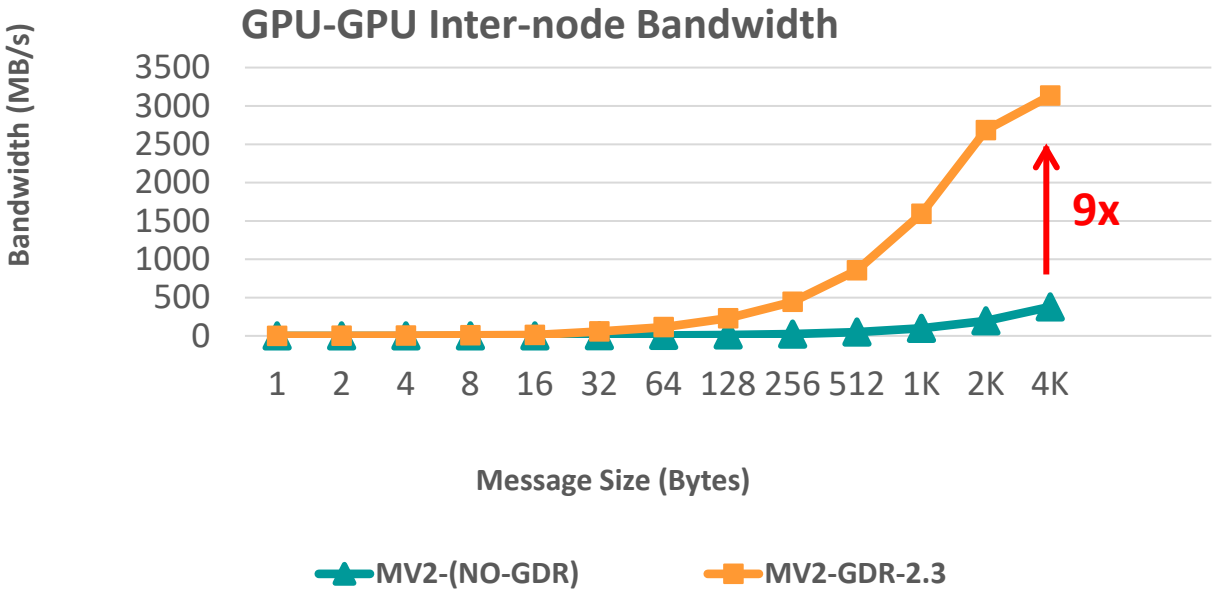


High Performance and High Productivity

Optimized MVAPICH2-GDR Design



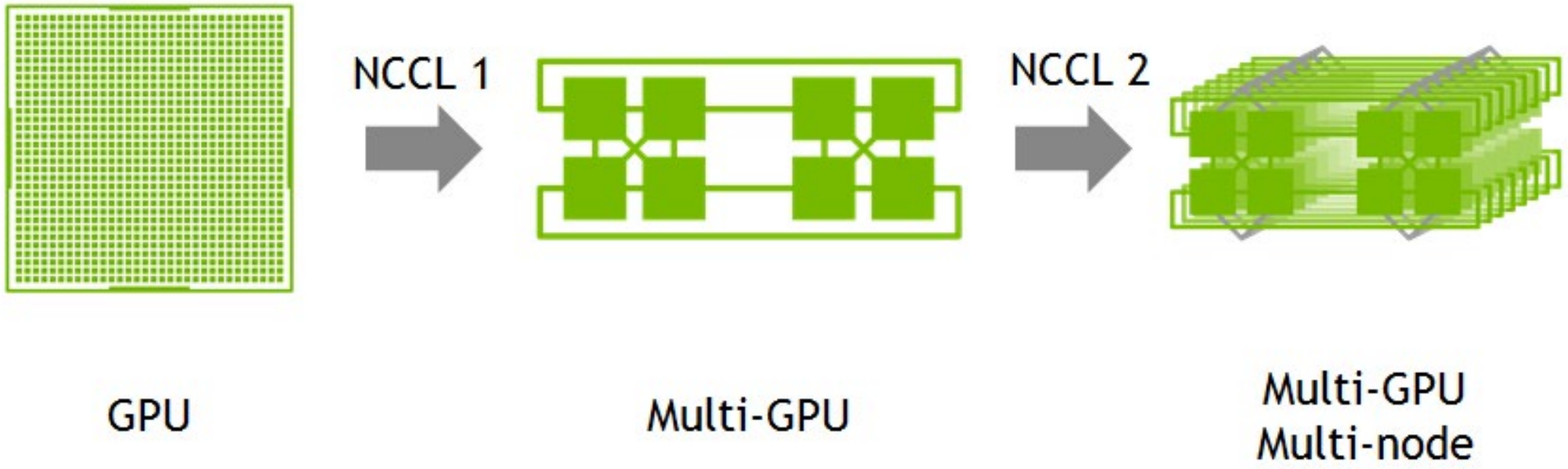




MVAPICH2-GDR-2.3.1
Intel Haswell (E5-2687W @ 3.10 GHz) node - 20 cores
NVIDIA Volta V100 GPU
Mellanox Connect-X4 EDR HCA
CUDA 9.0
Mellanox OFED 4.0 with GPU-Direct-RDMA

NCCL Communication Library

- NVIDIA Collective Communication Library (NCCL)
- Main Motivation: Deep Learning workloads
- NCCL1– efficient dense-GPU communication within the node
- NCCL2– multiple DGX systems connected to each other with InfiniBand systems



Courtesy: https://developer.nvidia.com/nccl

Outline

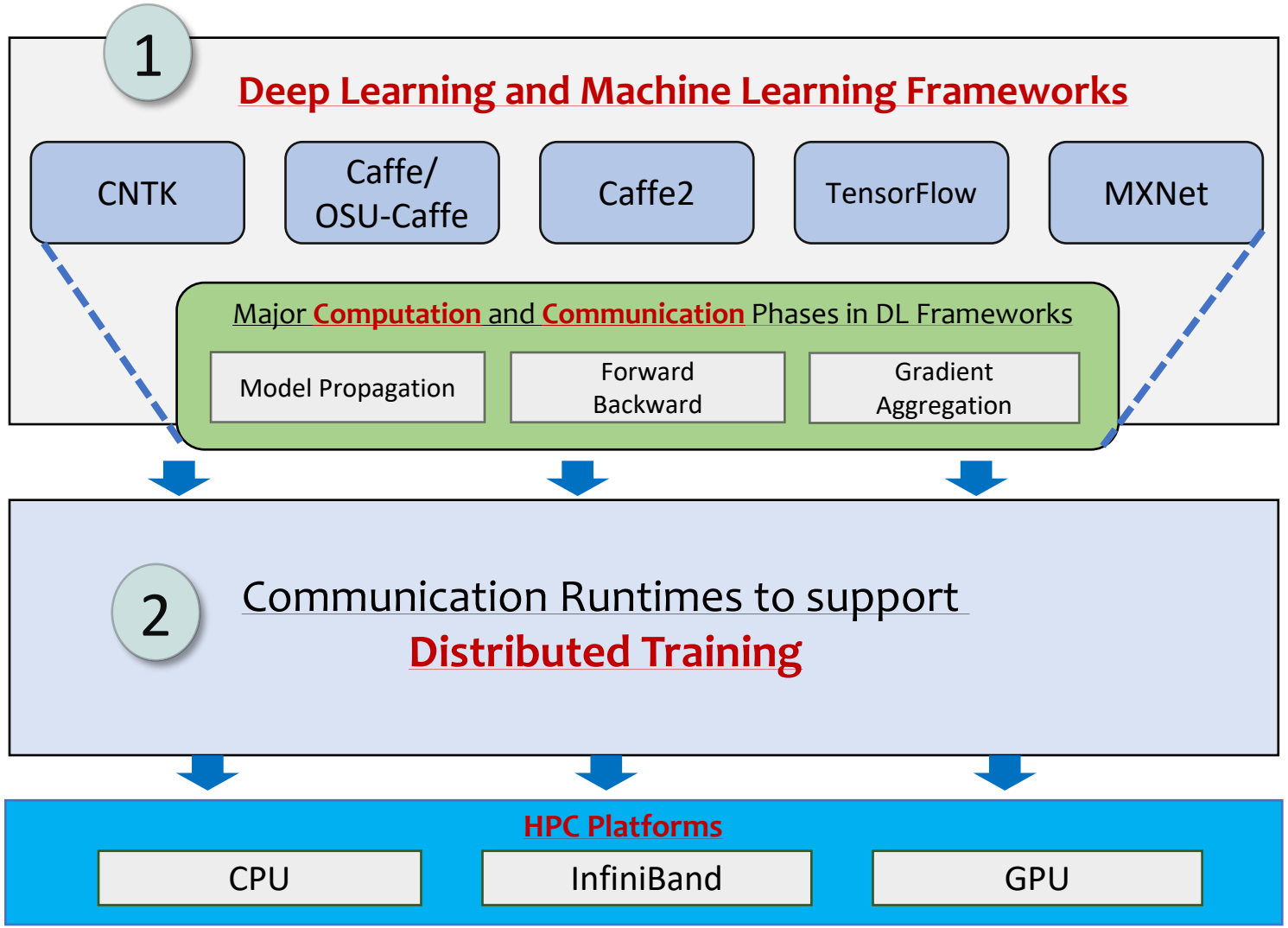
- Introduction
- Overview of Execution Environments
- Parallel and Distributed DNN Training
- Latest Trends in High-Performance Computing Architectures
- **Challenges in Exploiting HPC Technologies for Deep Learning**
- Solutions and Case Studies
- Open Issues and Challenges
- Hands-on Exercises
- Conclusion

Broad Challenge: Exploiting HPC for Deep Learning

How to efficiently scale-out a Deep Learning (DL) framework and take advantage of heterogeneous High Performance Computing (HPC) resources?

Research Challenges to Exploit HPC Technologies

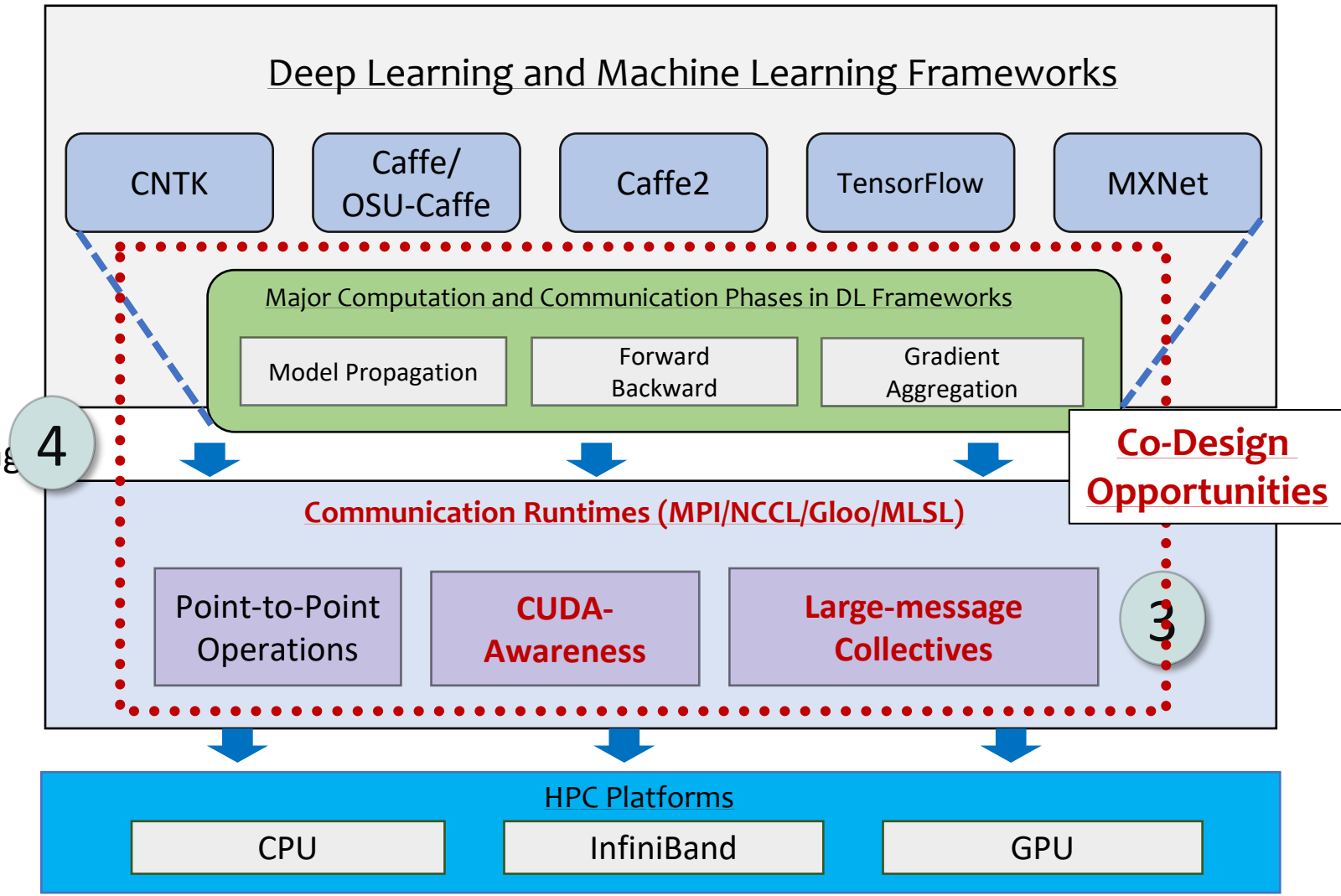
1. What are the fundamental issues in designing **DL frameworks**?
   - Memory Requirements
   - **Computation** Requirements
   - **Communication** Overhead
2. Why do we need to support **distributed training**?
   - To overcome the limits of single-node training
   - To better utilize hundreds of existing HPC Clusters



Research Challenges to Exploit HPC Technologies (Cont'd)

3. What are the **new design challenges** brought forward by DL frameworks for Communication runtimes?
   - Large Message **Collective Communication** and Reductions
   - GPU Buffers (**CUDA-Awareness**)

4. Can a **Co-design** approach help in achieving Scale-up and Scale-out efficiently?
   - **Co-Design** the support at **Runtime level** and Exploit it at the **DL Framework level**
   - What performance benefits can be observed?
   - What needs to be fixed at the **communication runtime** layer?

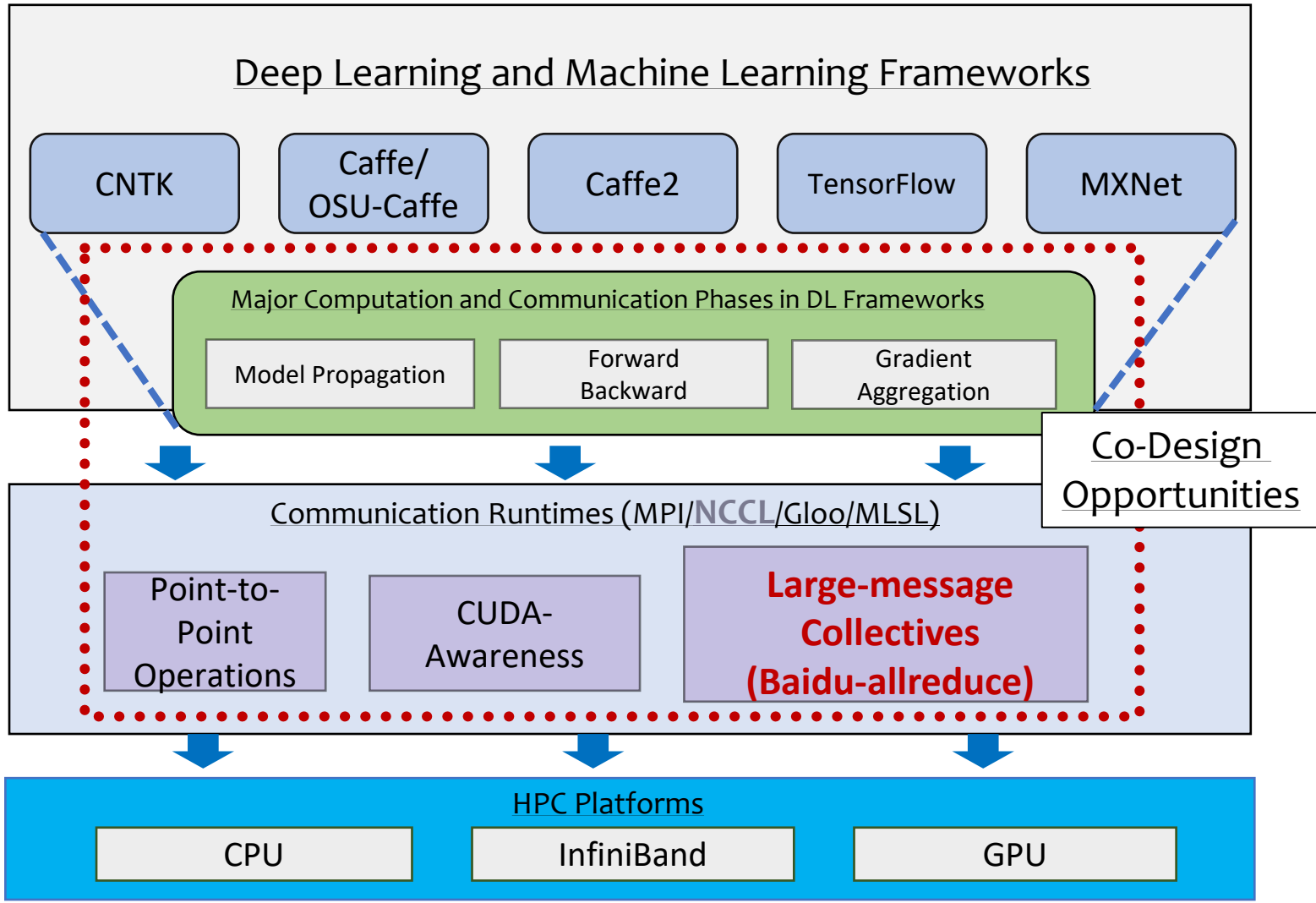


Outline

- Introduction
- Overview of Execution Environments
- Parallel and Distributed DNN Training
- Latest Trends in High-Performance Computing Architectures
- Challenges in Exploiting HPC Technologies for Deep Learning
- **Solutions and Case Studies**
- Open Issues and Challenges
- Hands-on Exercises
- Conclusion

Solutions and Case Studies: Exploiting HPC for DL

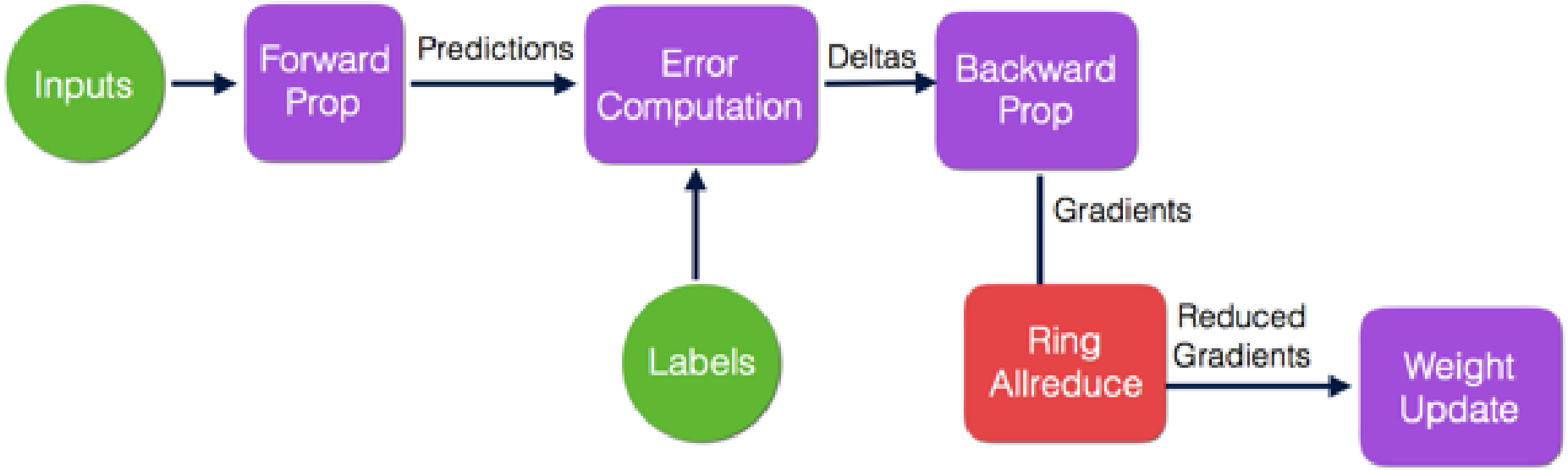
- **Data Parallelism**
  - **Baidu-allreduce**
  - NVIDIA NCCL/NCCL2
  - Co-design MPI runtimes and DL Frameworks
  - Distributed Training for TensorFlow
- Model and Hybrid Parallelism
  - GPipe
  - FlexFlow
  - HyPar-Flow
  - GEMS
  - SUPER



Baidu's Ring-Allreduce in TensorFlow

Scaling with TensorFlow

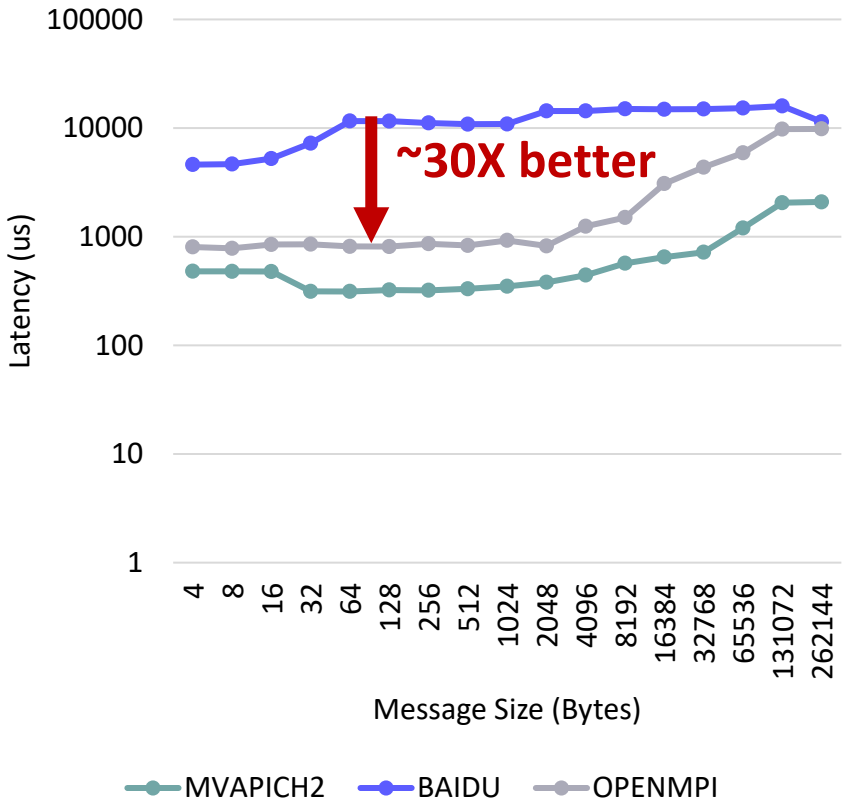
- Run many independent TensorFlow processes
- Insert allreduce as a node in the graph:

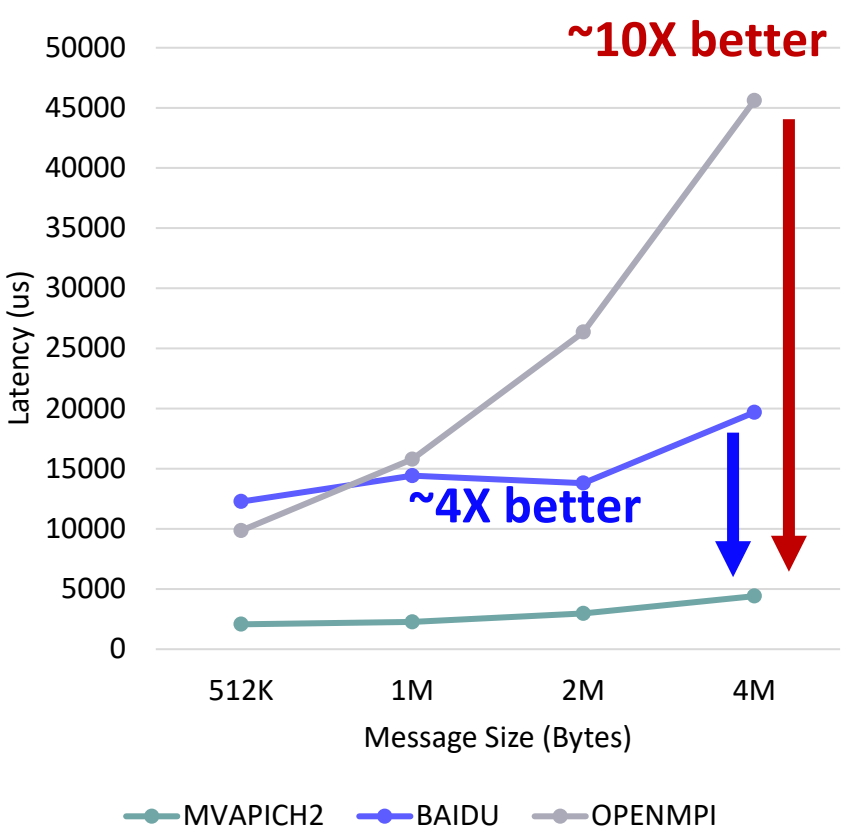


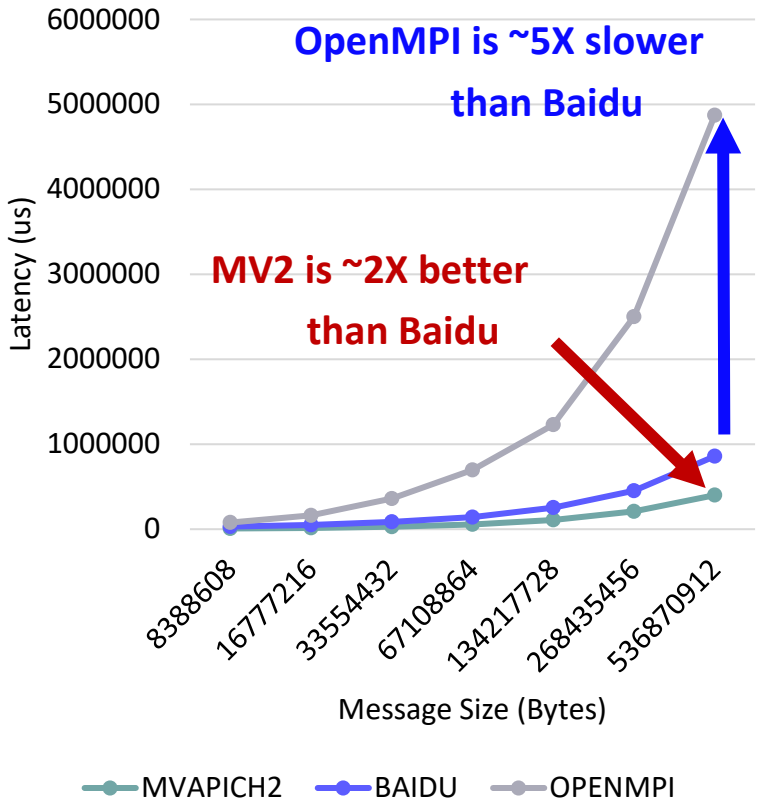
Courtesy: http://on-demand.gputechconf.com/gtc/2017/presentation/s7543-andrew-gibiansky-effectively-scakukbg-deep-learning-frameworks.pdf

MVAPICH2-GDR: Allreduce Comparison with Baidu and OpenMPI

- 16 GPUs (4 nodes) MVAPICH2-GDR vs. Baidu-Allreduce and OpenMPI 3.0



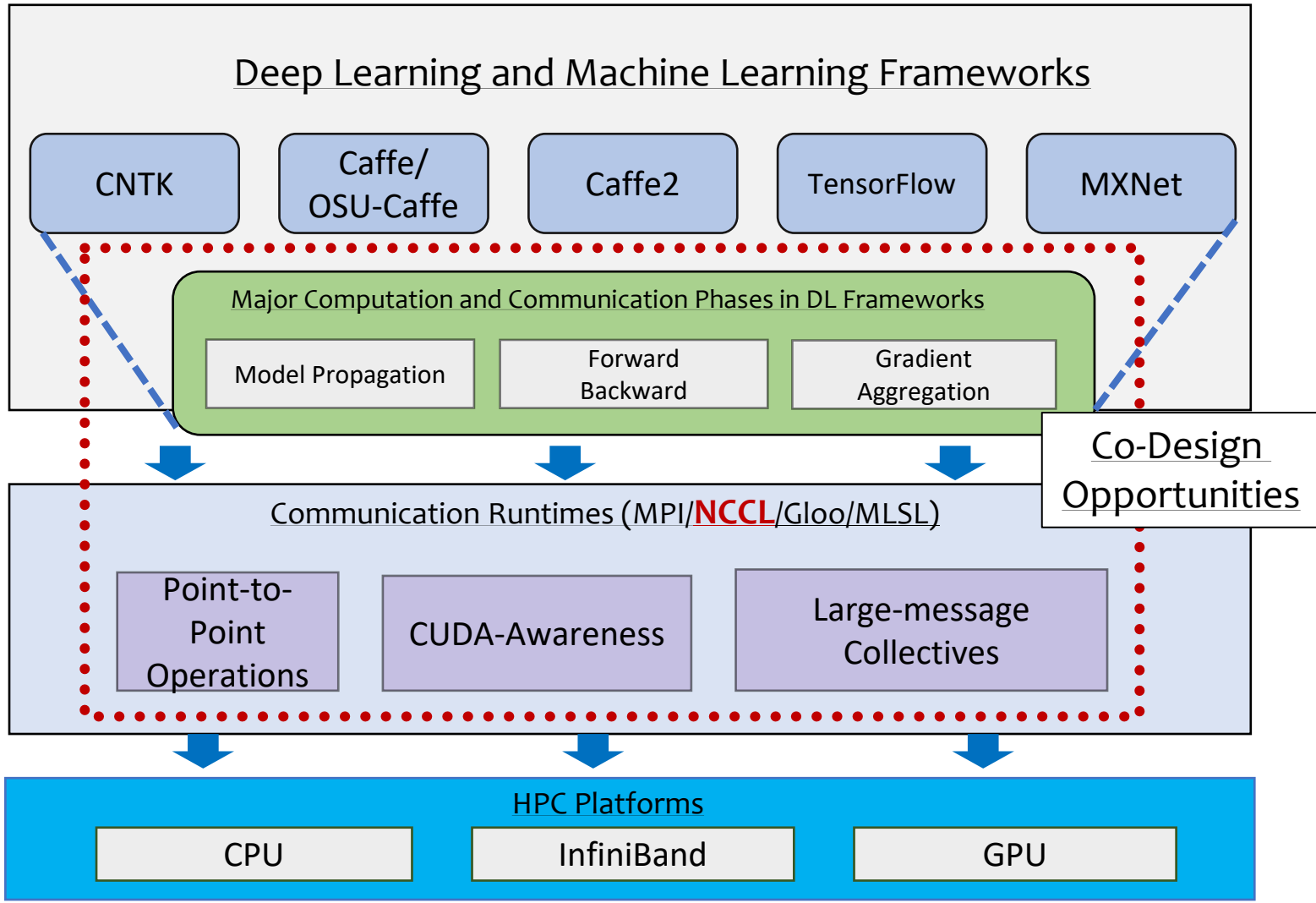




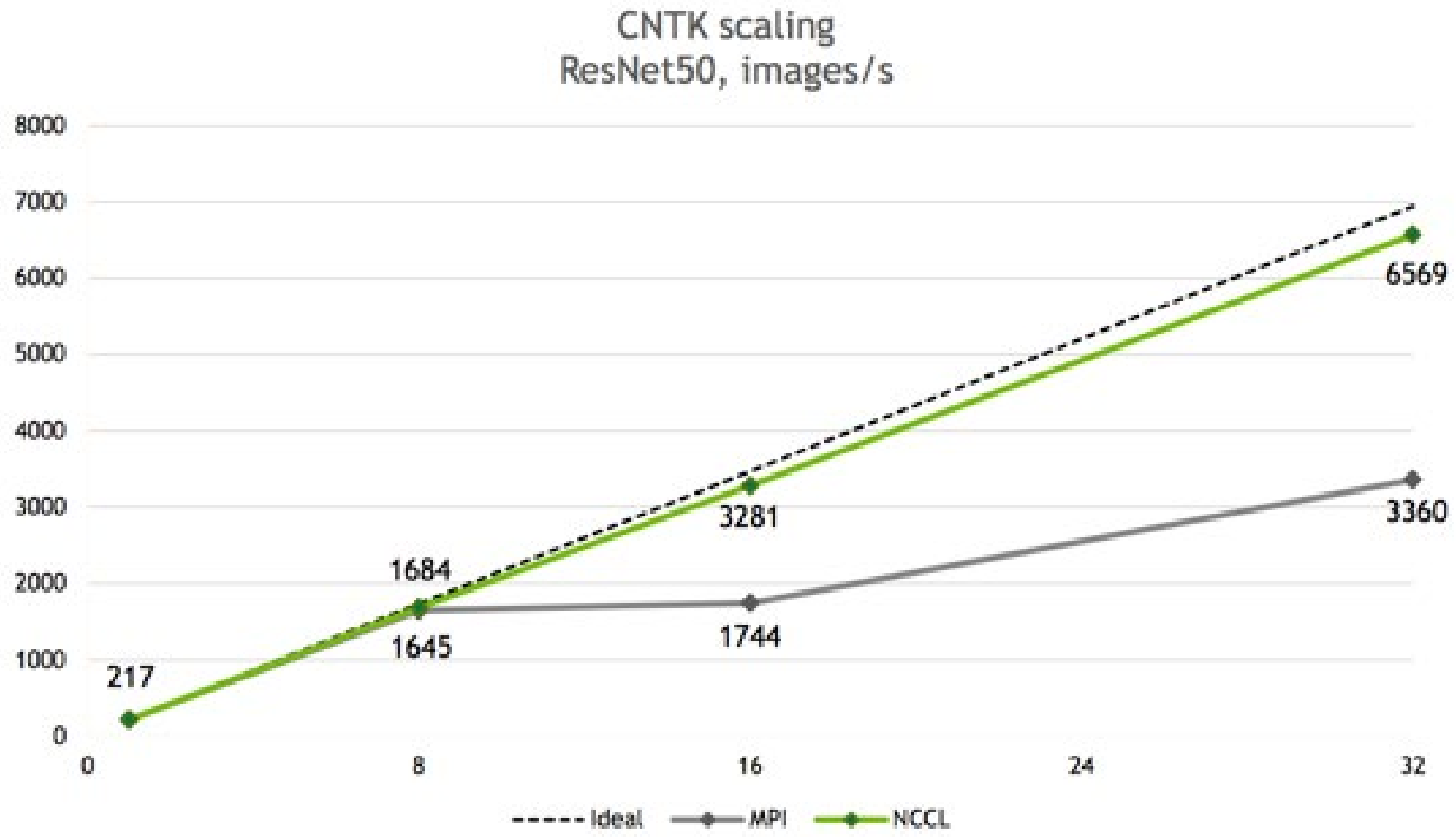
**Available since MVAPICH2-GDR 2.3a*

Solutions and Case Studies: Exploiting HPC for DL

- **Data Parallelism**
  - Baidu-allreduce
  - **NVIDIA NCCL/NCCL2**
  - Co-design MPI runtimes and DL Frameworks
  - Distributed Training for TensorFlow
- Model and Hybrid Parallelism
  - GPipe
  - FlexFlow
  - HyPar-Flow
  - GEMS
  - SUPER



NCCL2: Multi-node GPU Collectives

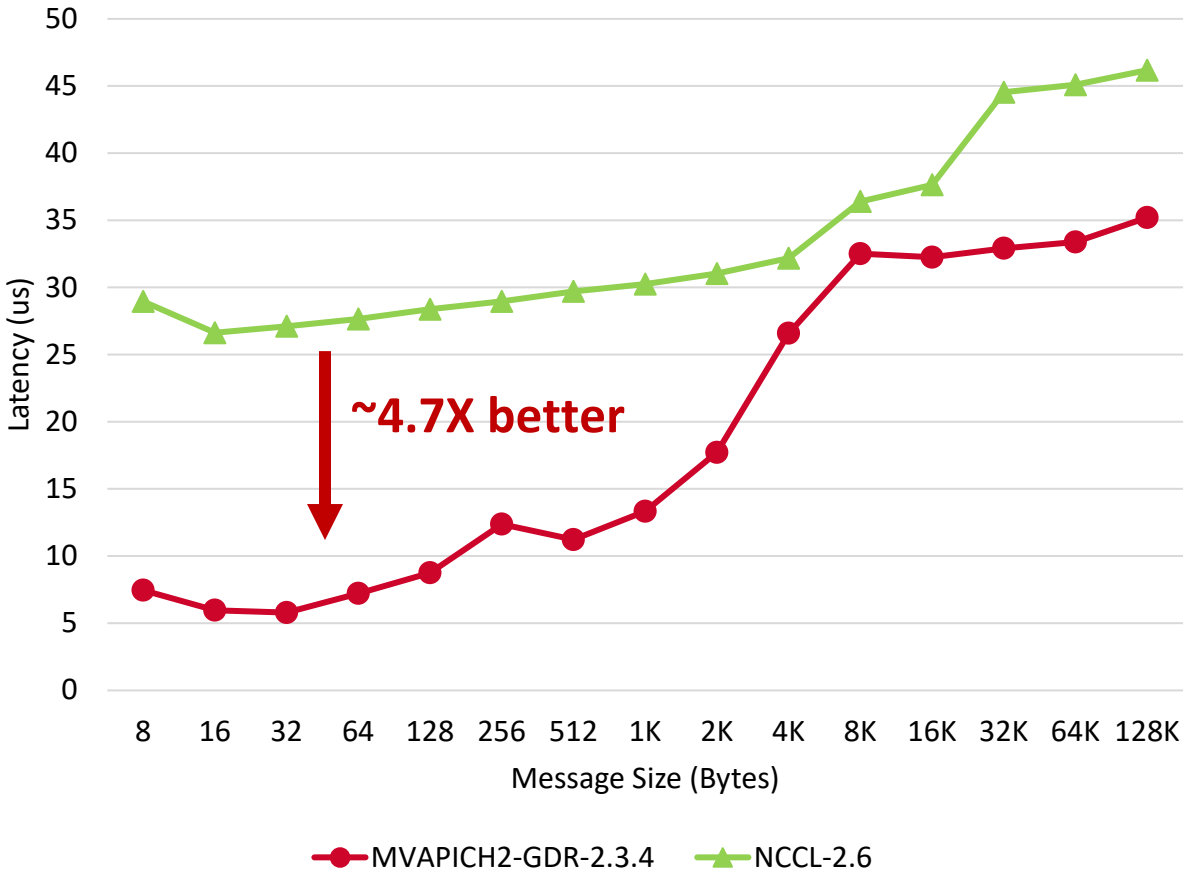


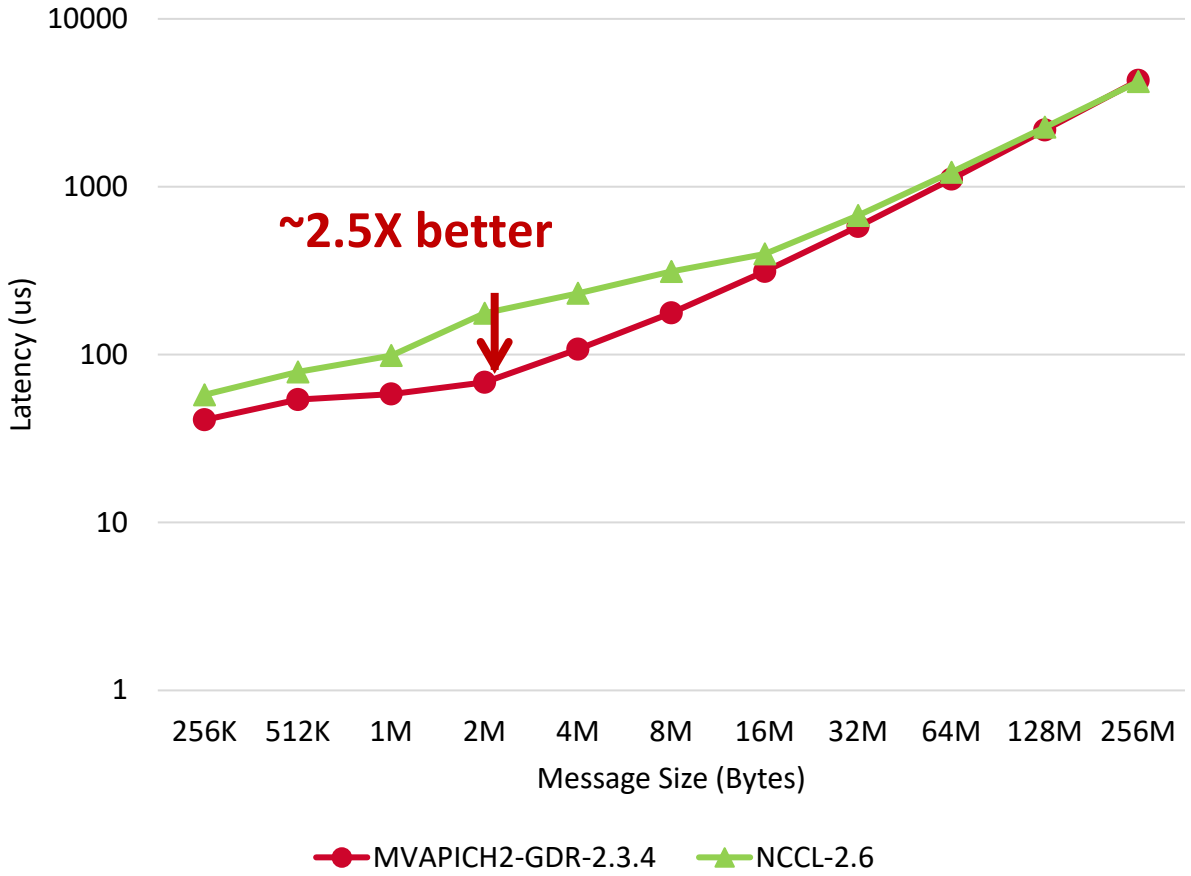
Courtesy: http://on-demand.gputechconf.com/gtc/2017/presentation/s7155-jeaugey-nccl.pdf

MVAPICH2-GDR vs. NCCL2 – Allreduce Operation (DGX-2)

- Optimized designs in MVAPICH2-GDR offer better/comparable performance for most cases
- MPI_Allreduce (MVAPICH2-GDR) vs. ncclAllreduce (NCCL2) on 1 DGX-2 node (16 Volta GPUs)

Platform: Nvidia DGX-2 system (16 Nvidia Volta GPUs connected with NVSwitch), CUDA 10.1



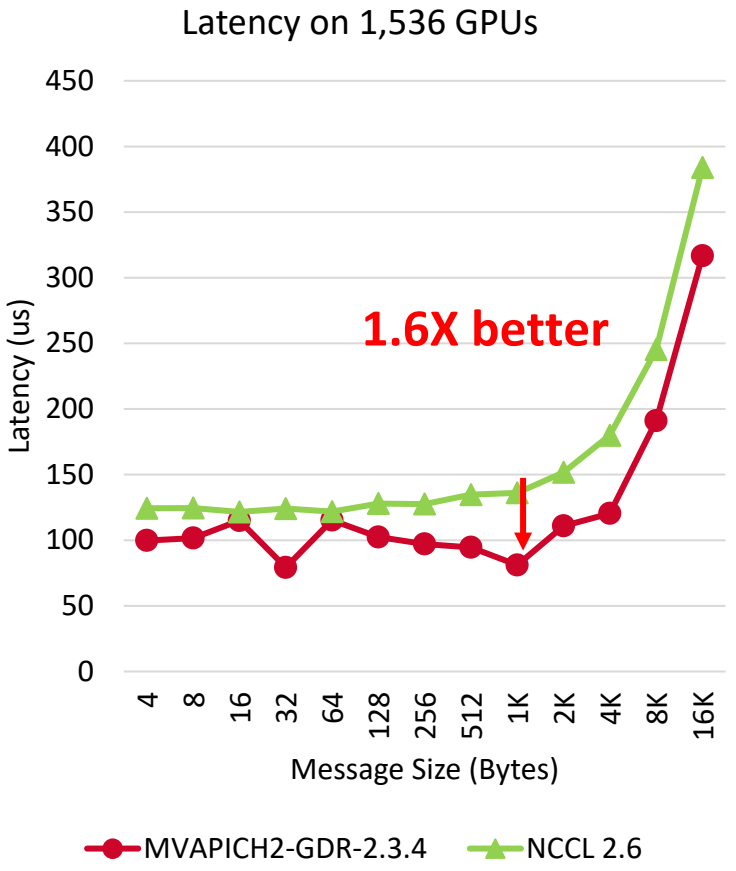


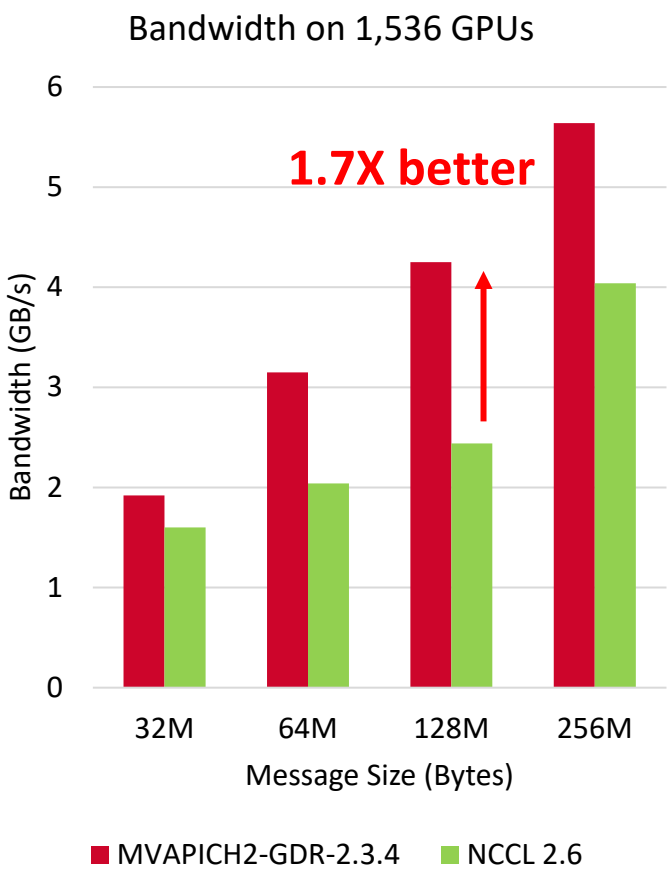
C.-H. Chu, P. Kousha, A. Awan, K. S. Khorassani, H. Subramoni and D. K. Panda, "NV-Group: Link-Efficient Reductions for Distributed Deep Learning on Modern Dense GPU Systems, " ICS-2020, June-July 2020.

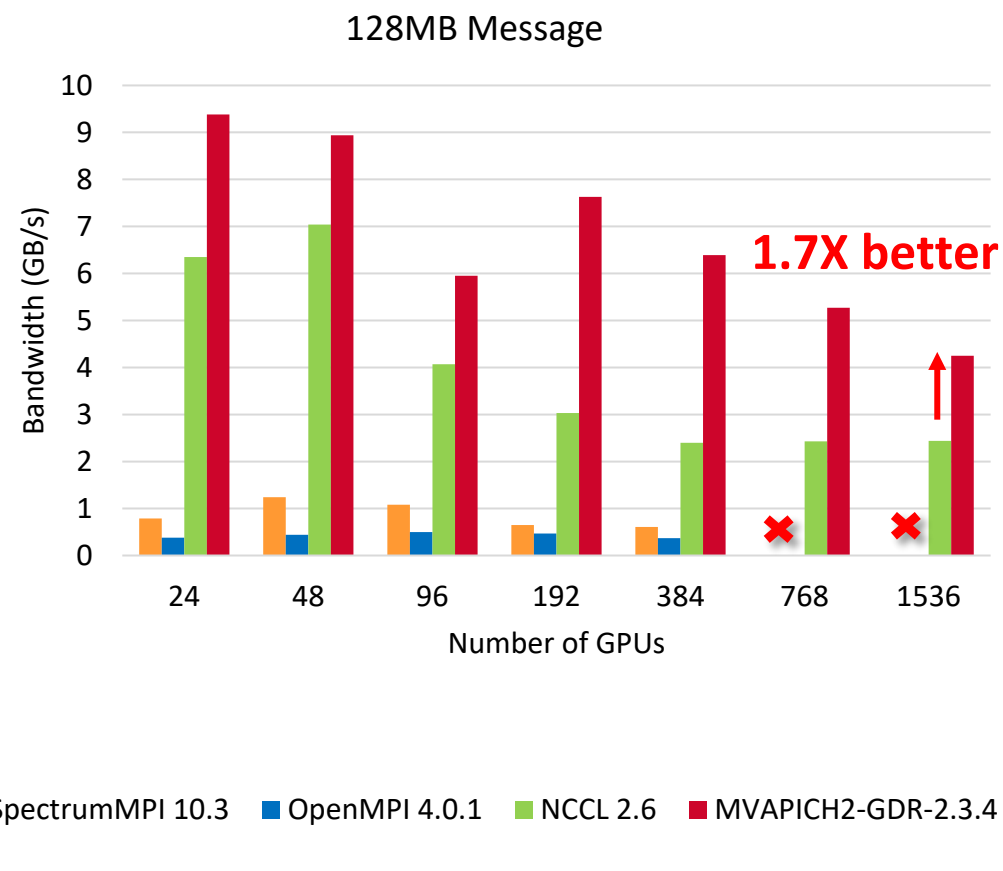
MVAPICH2-GDR: MPI_Allreduce at Scale (ORNL Summit)

- Optimized designs in MVAPICH2-GDR offer better performance for most cases
- MPI_Allreduce (MVAPICH2-GDR) vs. ncclAllreduce (NCCL2) up to 1,536 GPUs

Platform: Dual-socket IBM POWER9 CPU, 6 NVIDIA Volta V100 GPUs, and 2-port InfiniBand EDR Interconnect



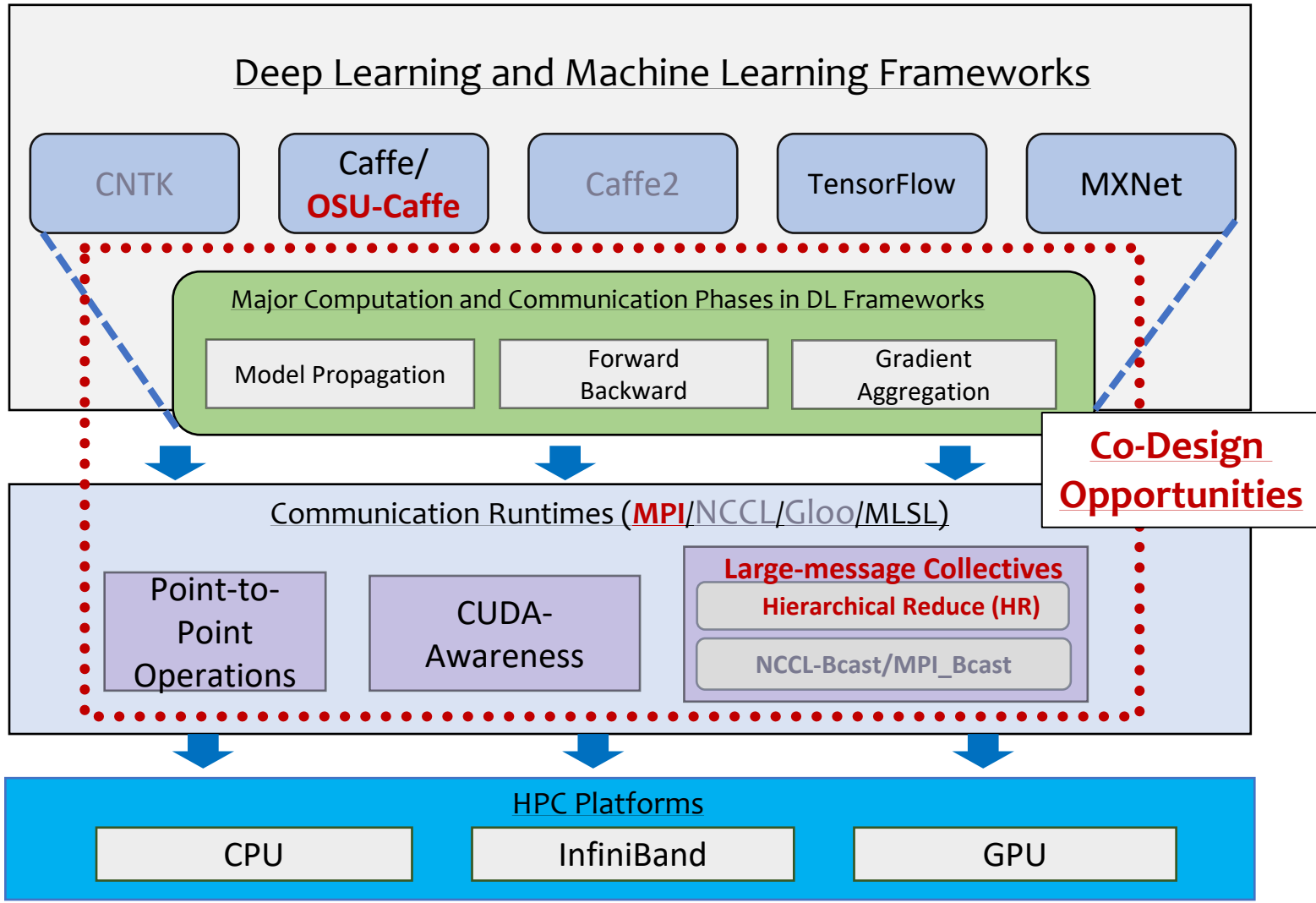




C.-H. Chu, P. Kousha, A. Awan, K. S. Khorassani, H. Subramoni and D. K. Panda, "NV-Group: Link-Efficient Reductions for Distributed Deep Learning on Modern Dense GPU Systems, " ICS-2020, June-July 2020.

Solutions and Case Studies: Exploiting HPC for DL

- **Data Parallelism**
  - Baidu-allreduce
  - NVIDIA NCCL/NCCL2
  - **Co-design MPI runtimes and DL Frameworks**
  - Distributed Training for TensorFlow
- Model and Hybrid Parallelism
  - GPipe
  - FlexFlow
  - HyPar-Flow
  - GEMS
  - SUPER



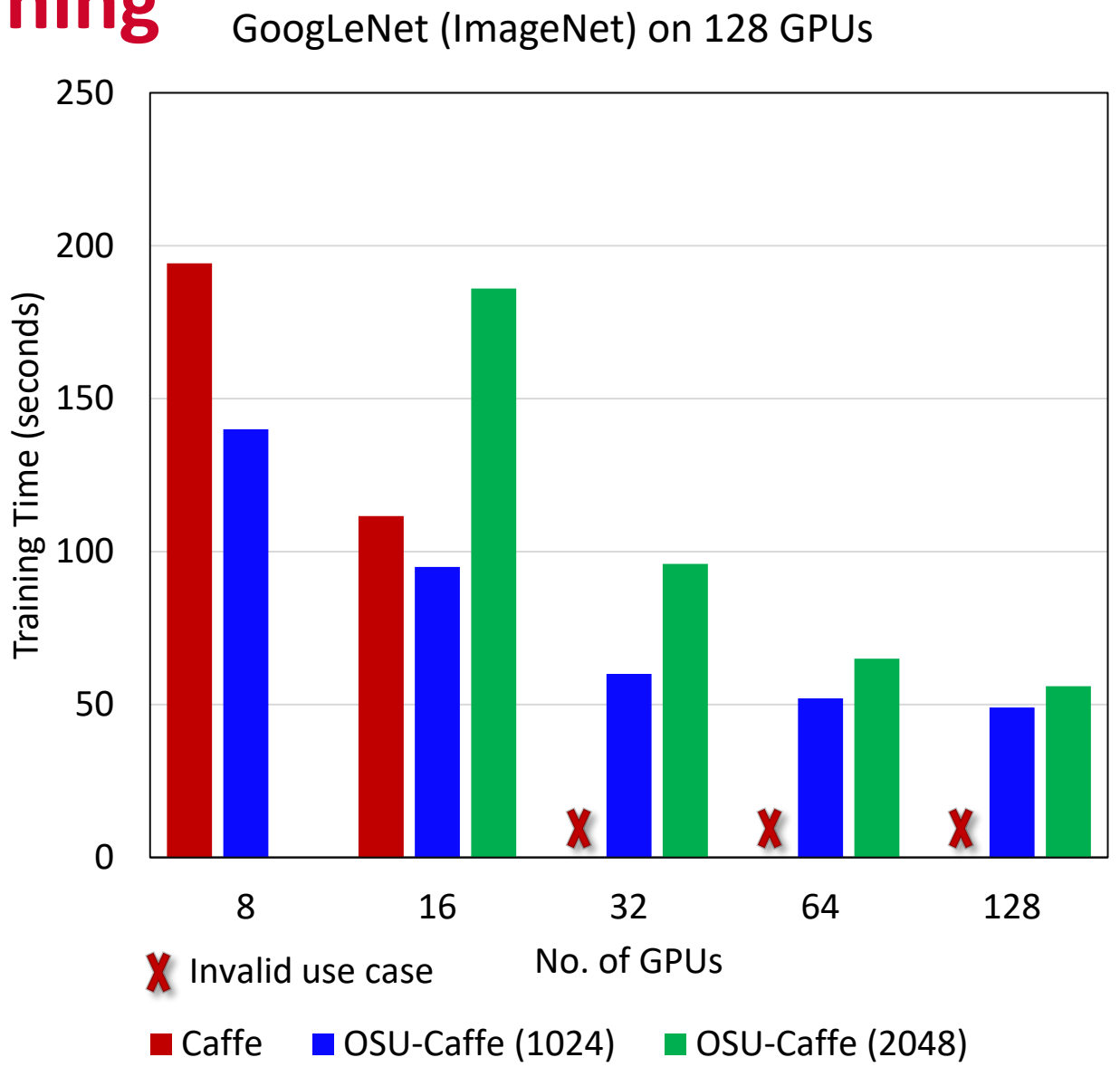
S-Caffe: Proposed Co-Design Overview

- To address the limitations of Caffe and existing MPI runtimes, we propose the **OSU-Caffe (S-Caffe)** framework
- At the application (DL framework) level
  - Develop a fine-grain workflow – i.e. layer-wise communication instead of communicating the entire model
- At the runtime (MPI) level
  - Develop support to perform reduction of very-large GPU buffers
  - Perform reduction using GPU kernels

OSU-Caffe is available from the HiDL project page (http://hidl.cse.ohio-state.edu)

OSU-Caffe: Scalable Deep Learning

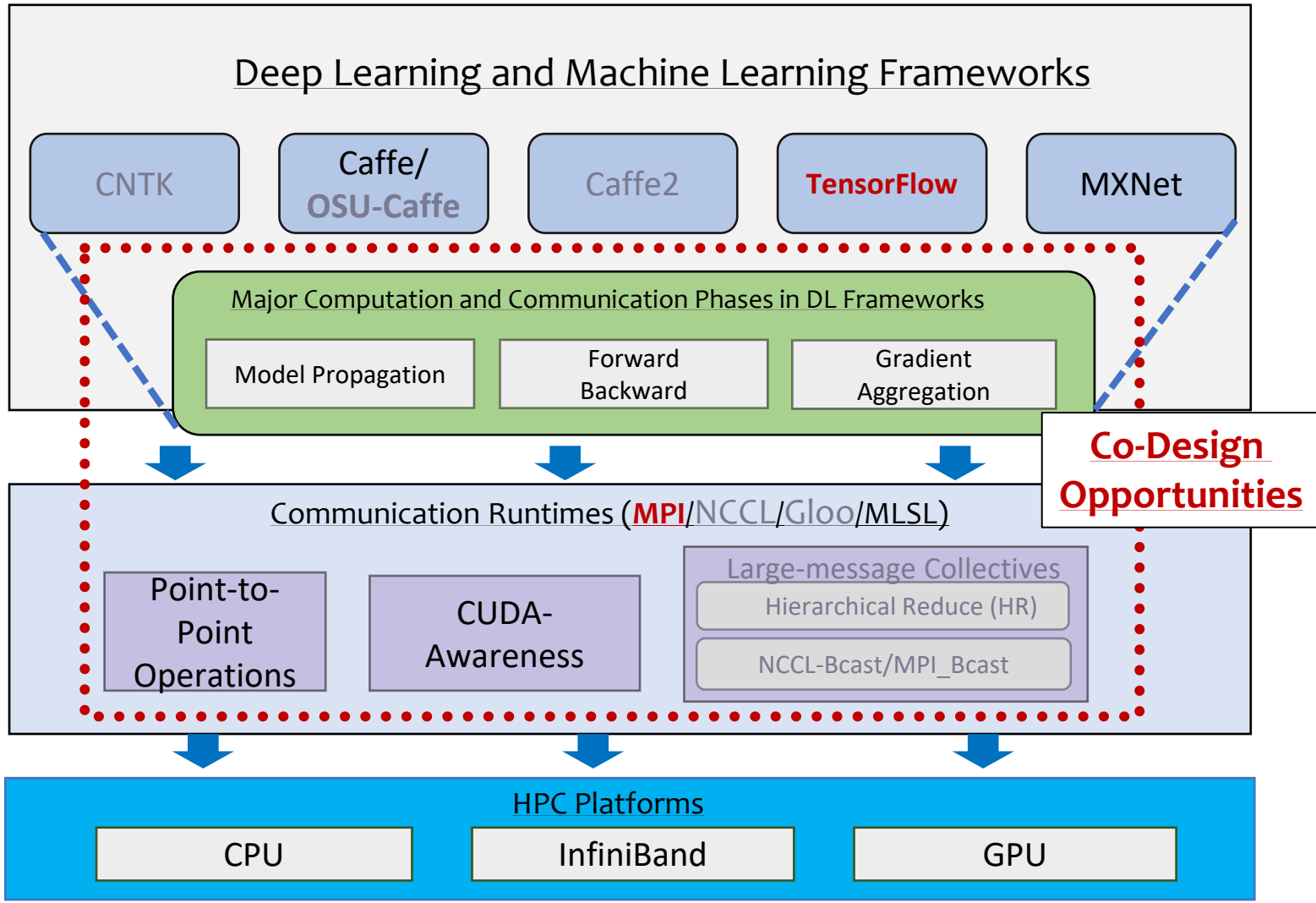
- Caffe : A flexible and layered Deep Learning framework.
- Benefits and Weaknesses
  - Multi-GPU Training within a single node
  - Performance degradation for GPUs across different sockets
  - Limited Scale-out
- OSU-Caffe: MPI-based Parallel Training
  - Enable Scale-up (within a node) and Scale-out (across multi-GPU nodes)
  - Scale-out on 64 GPUs for training CIFAR-10 network on CIFAR-10 dataset
  - Scale-out on 128 GPUs for training GoogLeNet network on ImageNet dataset



A. A. Awan, K. Hamidouche, J. M. Hashmi, and D. K. Panda, S-Caffe: Co-designing MPI Runtimes and Caffe for Scalable Deep Learning on Modern GPU Clusters. In *Proceedings of the 22nd ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming* (PPoPP '17)

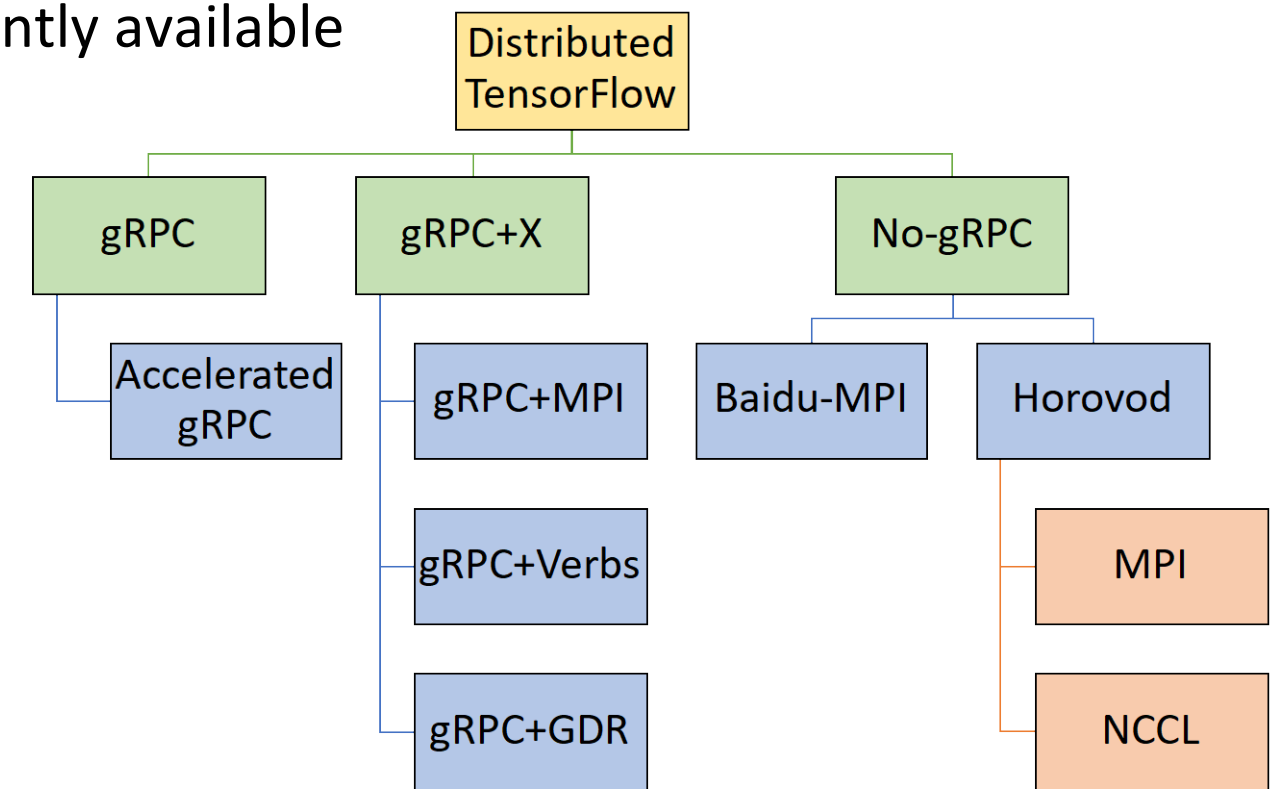
Solutions and Case Studies: Exploiting HPC for DL

- **Data Parallelism**
  - Baidu-allreduce
  - NVIDIA NCCL/NCCL2
  - Co-design MPI runtimes and DL Frameworks
  - **Distributed Training for TensorFlow**
- Model and Hybrid Parallelism
  - GPipe
  - FlexFlow
  - HyPar-Flow
  - GEMS
  - SUPER



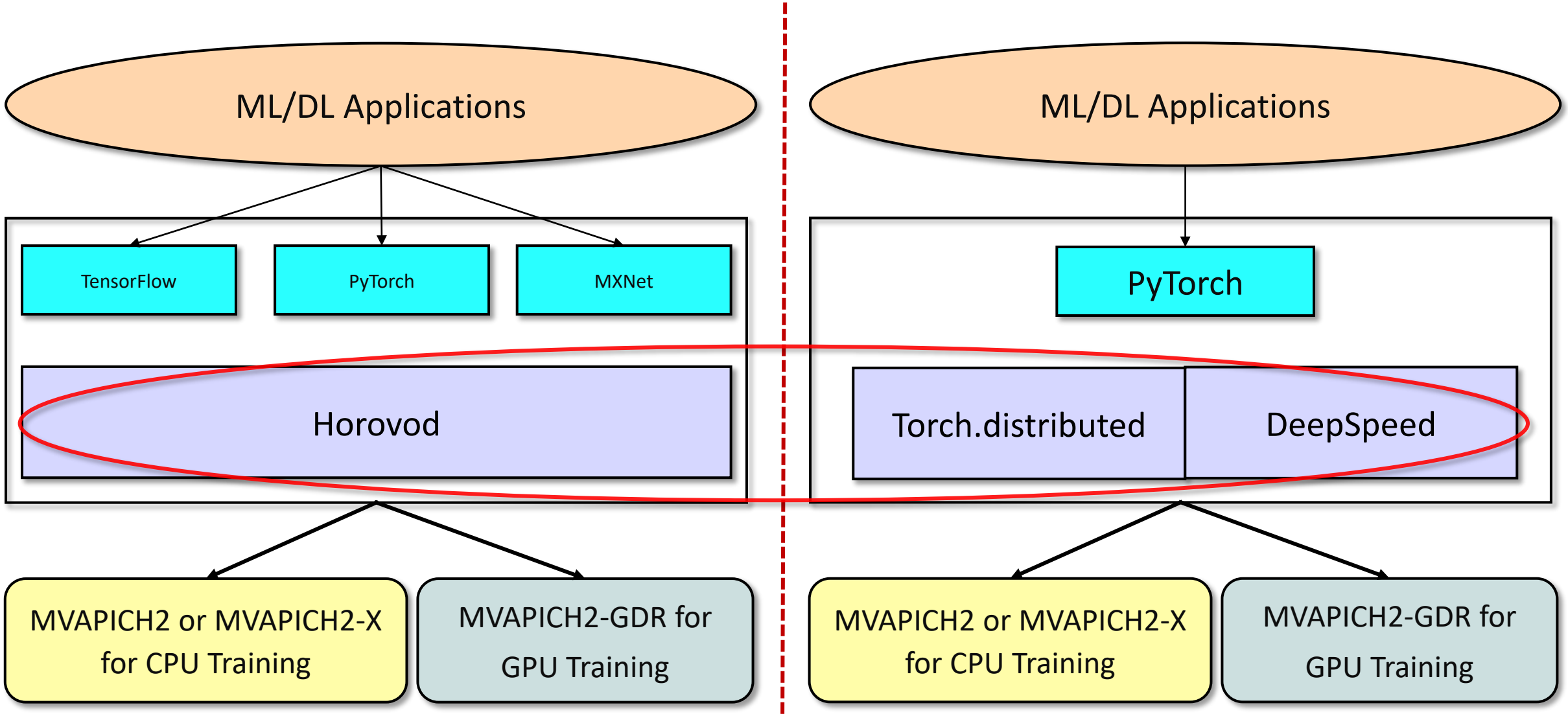
Data Parallel Training with TensorFlow (TF)

- Need to understand several options currently available
- gRPC (official support)
  - Open-source – can be enhanced by others
  - Accelerated gRPC (add RDMA to gRPC)
- gRPC+X
  - Use gRPC for bootstrap and rendezvous
  - ***Actual communication is in "X"***
  - X→ MPI, Verbs, GPUDirect RDMA (GDR), etc.
- No-gRPC
  - Baidu – the first one to use MPI Collectives for TF
  - Horovod – Use NCCL, or MPI, or any other future library (e.g. IBM DDL support recently added)



A. A. Awan, J. Bedorf, C.-H. Chu, H. Subramoni and D. K. Panda, "Scalable Distributed DNN Training using TensorFlow and CUDA-Aware MPI: Characterization, Designs, and Performance Evaluation", CCGrid '19. https://arxiv.org/abs/1810.11112

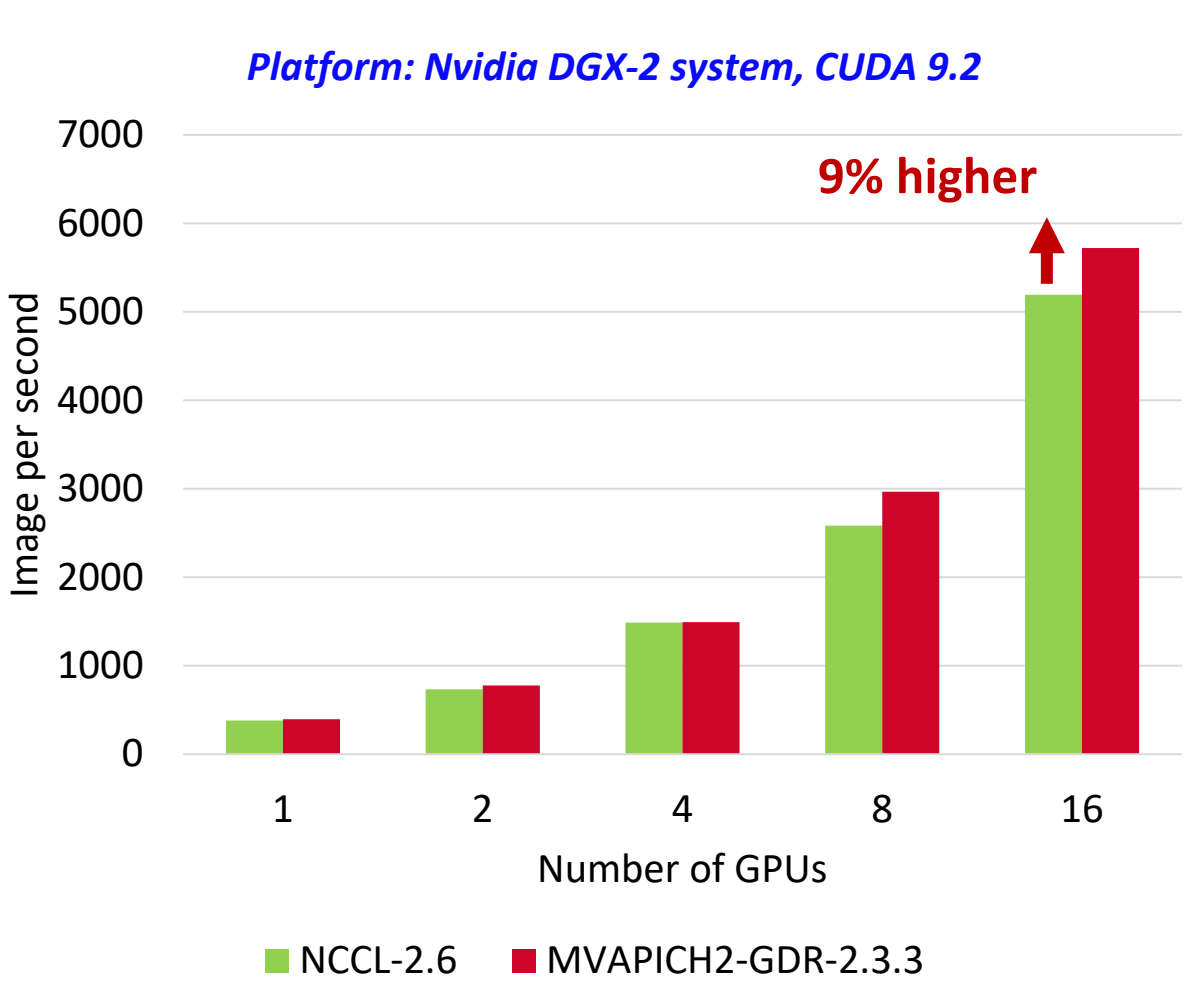
MVAPICH2 (MPI)-driven Infrastructure for ML/DL Training

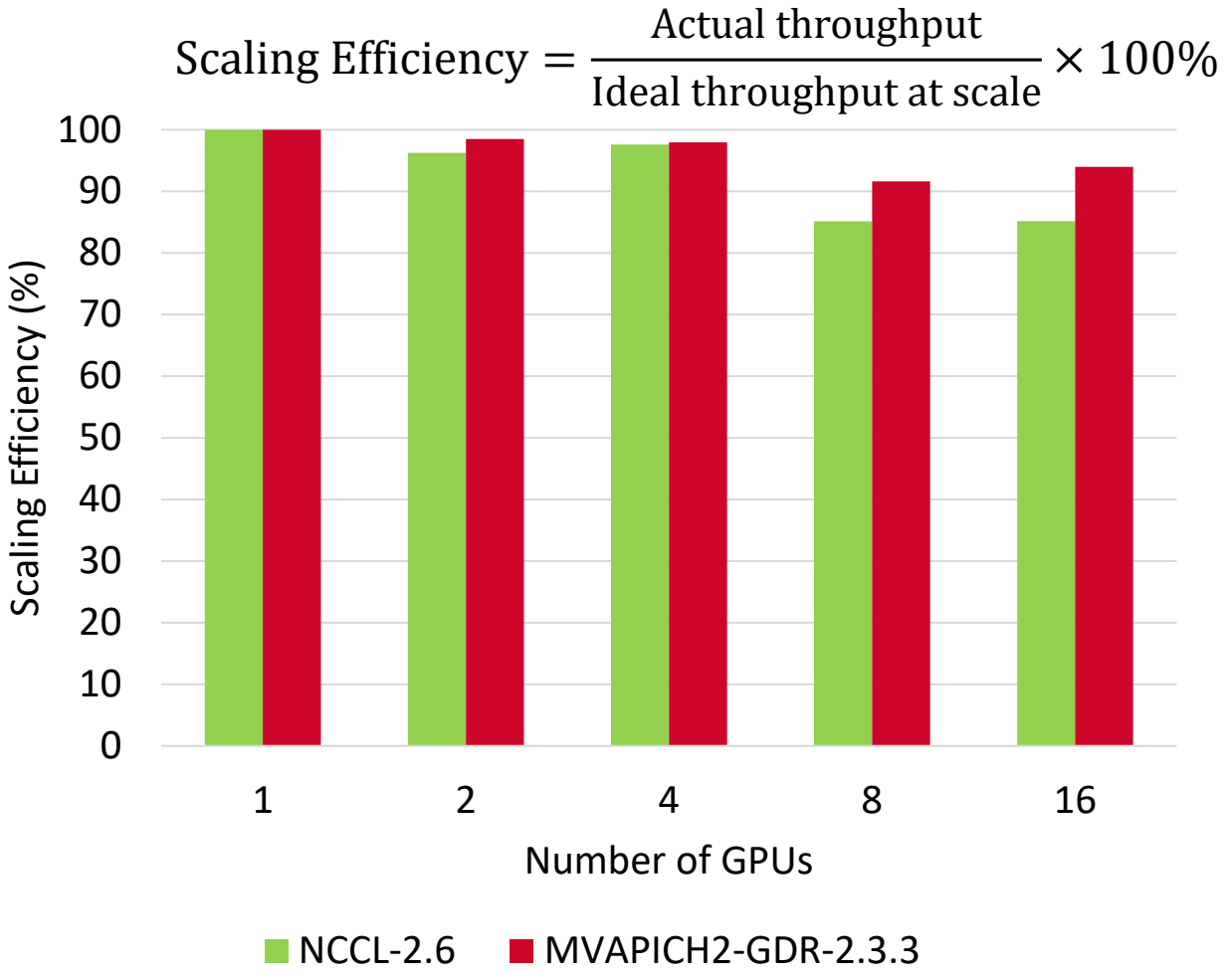


More details available from: http://hidl.cse.ohio-state.edu

Scalable TensorFlow using Horovod and MVAPICH2-GDR

- **ResNet-50 Training using TensorFlow benchmark on 1 DGX-2 node (16 Volta GPUs)**

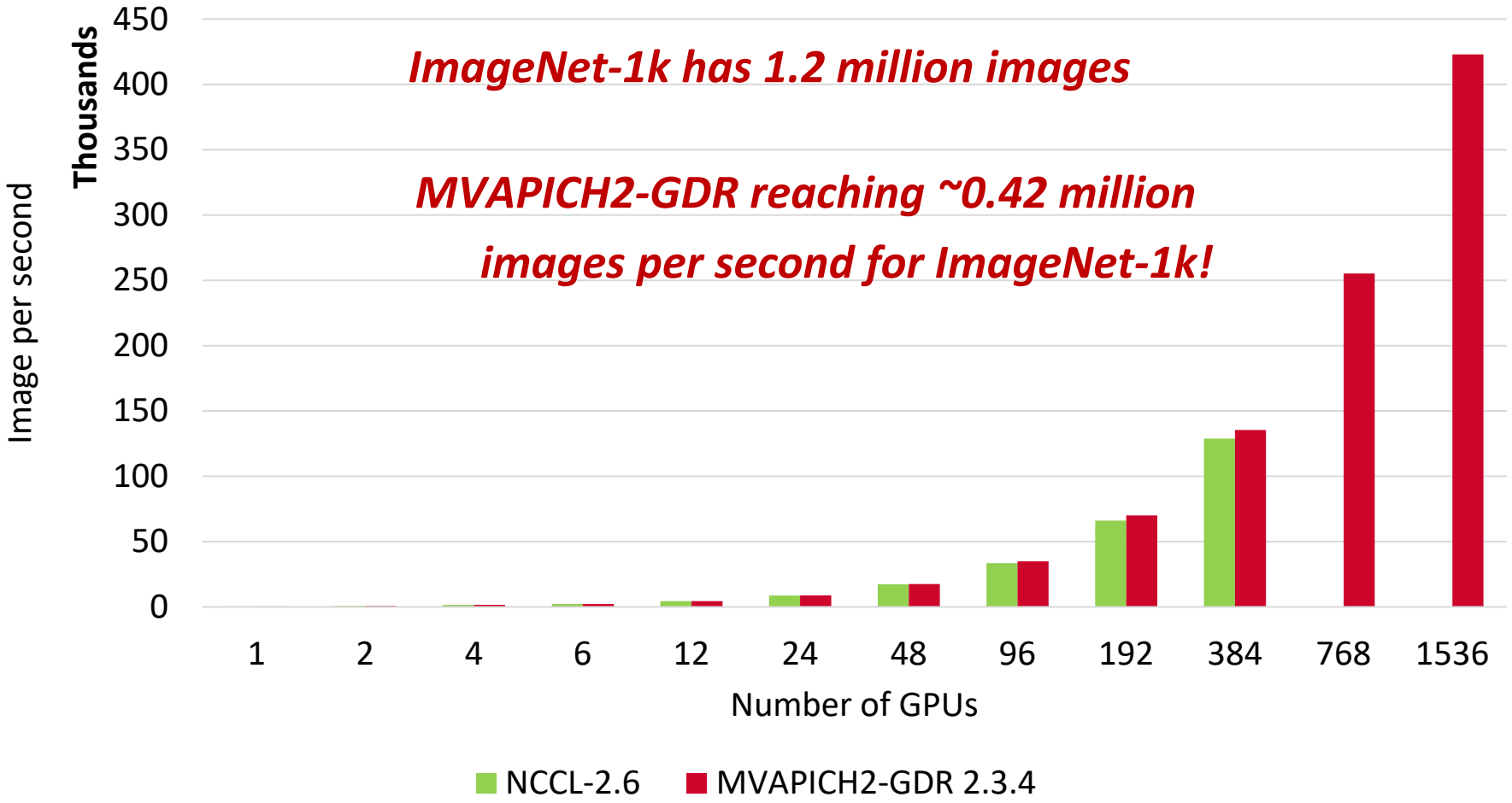




C.-H. Chu, P. Kousha, A. Awan, K. S. Khorassani, H. Subramoni and D. K. Panda, "NV-Group: Link-Efficient Reductions for Distributed Deep Learning on Modern Dense GPU Systems, " ICS-2020.

Distributed TensorFlow on ORNL Summit (1,536 GPUs)

- ResNet-50 Training using TensorFlow benchmark on SUMMIT -- 1536 Volta GPUs!
- 1,281,167 (1.2 mil.) images
- Time/epoch = 3 seconds
- Total Time (90 epochs) = 3 x 90 = 270 seconds = **4.5 minutes!**

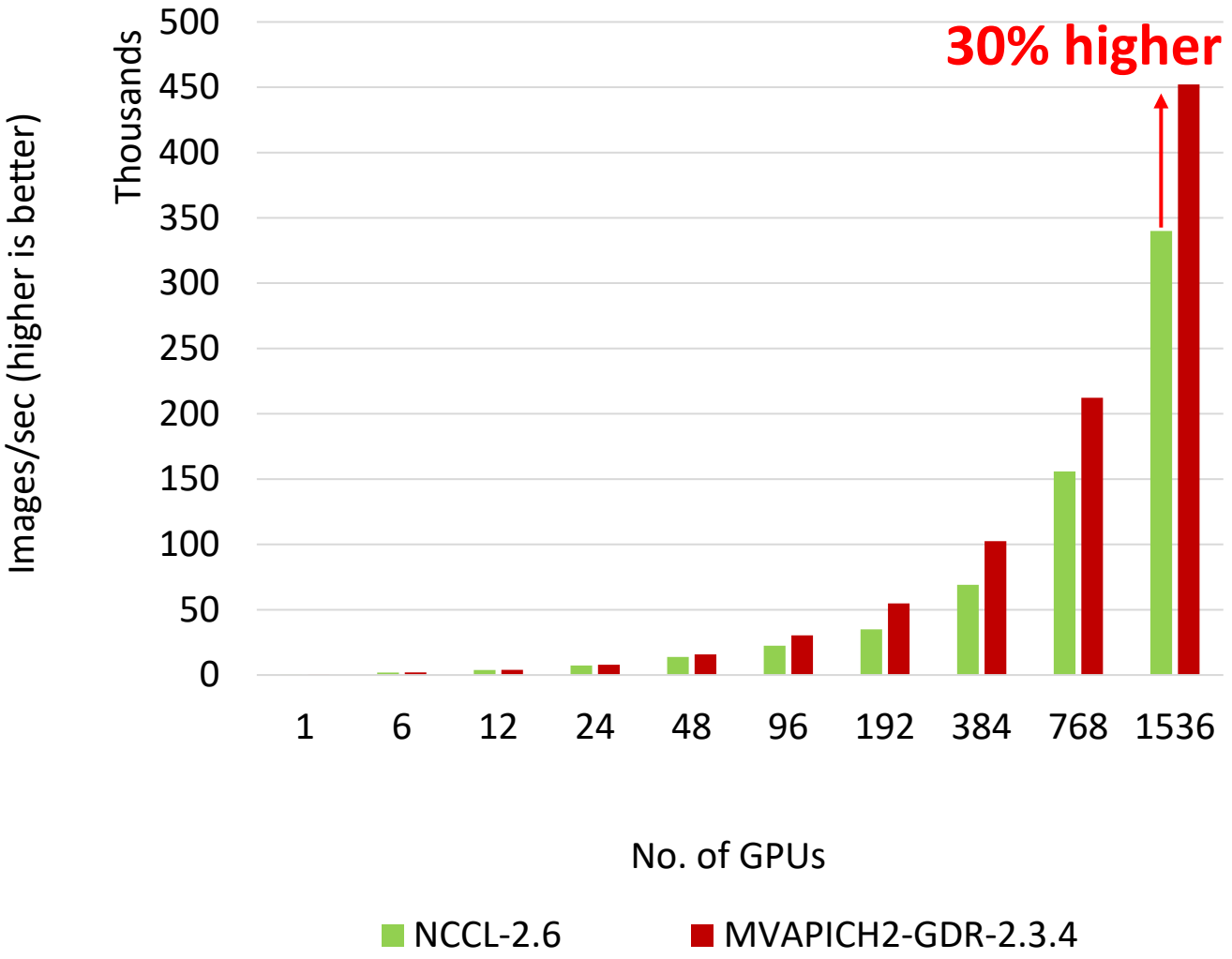


*We observed issues for NCCL2 beyond 384 GPUs

Platform: The Summit Supercomputer (#2 on Top500.org) – 6 NVIDIA Volta GPUs per node connected with NVLink, CUDA 10.1

Scaling PyTorch on ORNL Summit using MVAPICH2-GDR

- ResNet-50 training using PyTorch + Horovod on Summit
  - Synthetic ImageNet dataset
  - Up to 256 nodes, 1536 GPUs
- MVAPICH2-GDR can outperform NCCL2
  - **Up to 30% higher throughput**
- CUDA 10.1 cuDNN 7.6.5
  PyTorch v1.5.0 Horovod v0.19.1



C.-H. Chu, P. Kousha, A. Awan, K. S. Khorassani, H. Subramoni and D. K. Panda, "NV-Group: Link-Efficient Reductions for Distributed Deep Learning on Modern Dense GPU Systems, " ICS-2020, June-July 2020.

Platform: The Summit Supercomputer (#2 on Top500.org) – 6 NVIDIA Volta GPUs per node connected with NVLink, CUDA 10.1

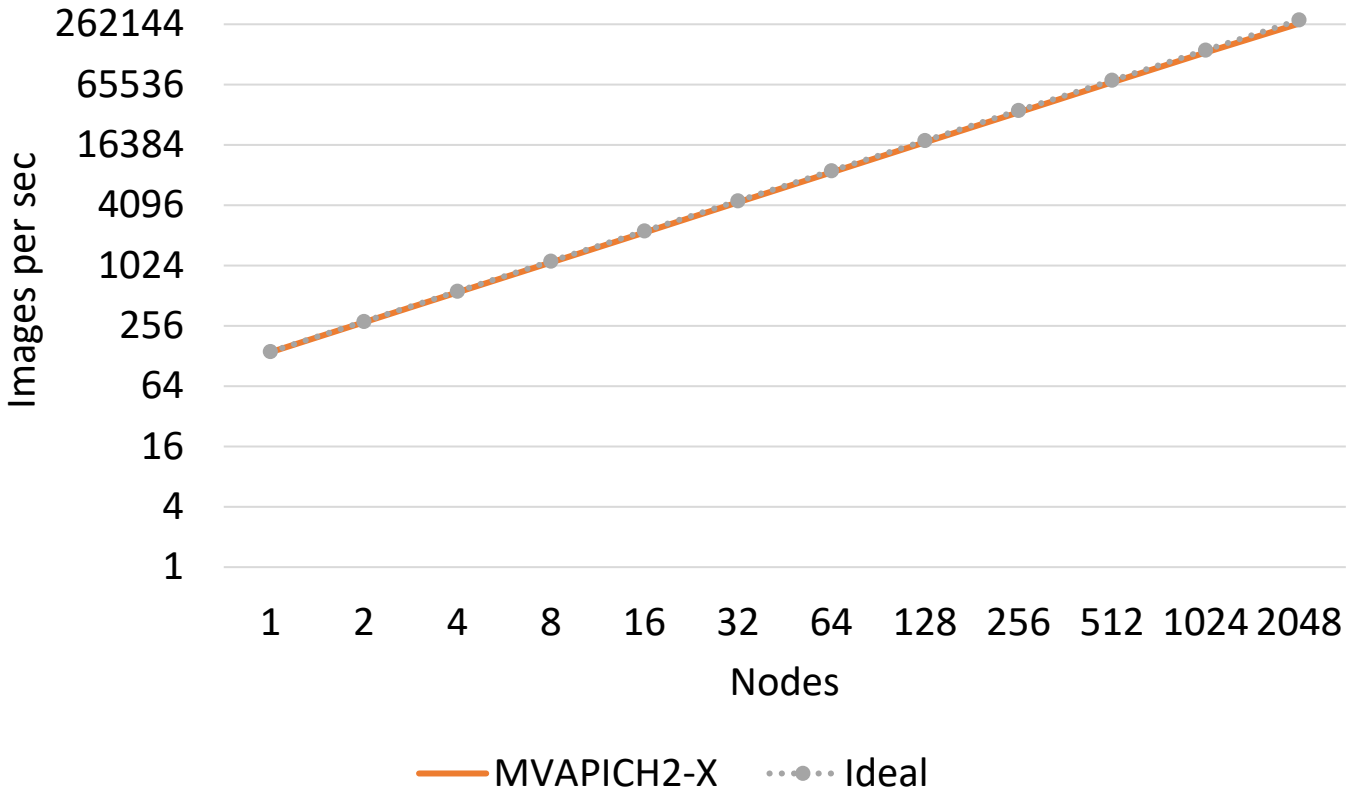
PyTorch at Scale: Training ResNet-50 on 256 V100 GPUs

- Training performance for 256 V100 GPUs on LLNL Lassen
  - **~10,000 Images/sec faster** than NCCL training!

Distributed Framework	Torch.distributed		Horovod		DeepSpeed	
Images/sec on 256 GPUs	61,794	**72,120**	74,063	**84,659**	80,217	**88,873**
Communication Backend	NCCL	MVAPICH2-GDR	NCCL	MVAPICH2-GDR	NCCL	MVAPICH2-GDR

Distributed TensorFlow on TACC Frontera (2,048 CPU nodes)

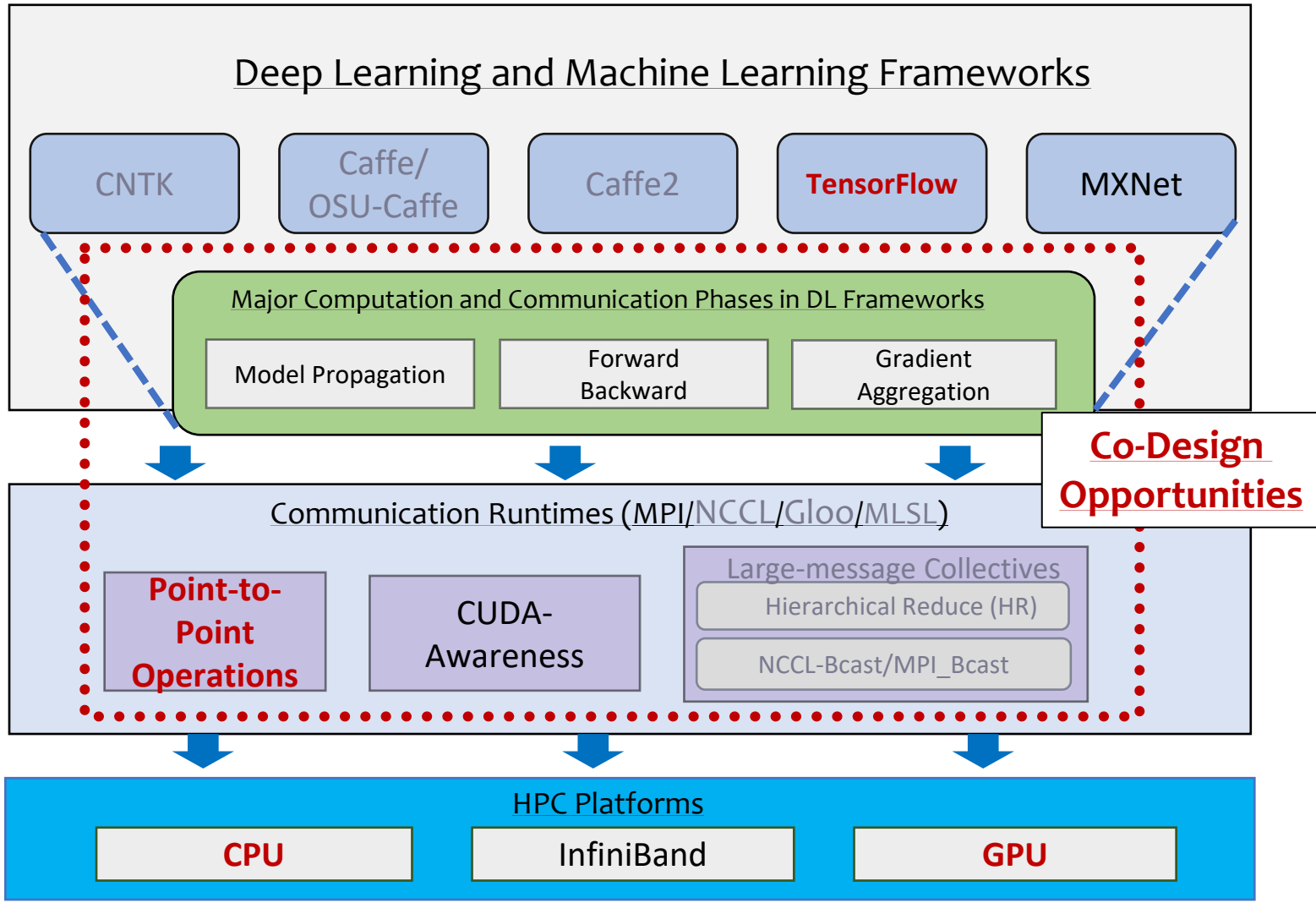
- Scaled TensorFlow to 2,048 nodes on Frontera using MVAPICH2 and IntelMPI
- MVAPICH2 and IntelMPI give similar performance for DNN training
- Report a peak of 260,000 images/sec on 2,048 nodes
- On 2,048 nodes, ResNet-50 can be trained in 7 minutes!



A. Jain, A. A. Awan, H. Subramoni, DK Panda, "Scaling TensorFlow, PyTorch, and MXNet using MVAPICH2 for High-Performance Deep Learning on Frontera", DLS '19 (SC '19 Workshop).

Solutions and Case Studies: Exploiting HPC for DL

- Data Parallelism
  - Baidu-allreduce
  - NVIDIA NCCL/NCCL2
  - Co-design MPI runtimes and DL Frameworks
  - Distributed Training for TensorFlow
- **Model and Hybrid Parallelism**
  - **GPipe**
  - FlexFlow
  - HyPar-Flow
  - GEMS
  - SUPER



GPipe: Pipeline Parallelism

- GPipe allows training of very large models
- Split the model across layers
- Trained AmoebaNet with 600 million parameters
- AmoebaNet cannot be trained on a single GPU or TPU
- 2.7x speedup on 8 GPUs
  - Why?

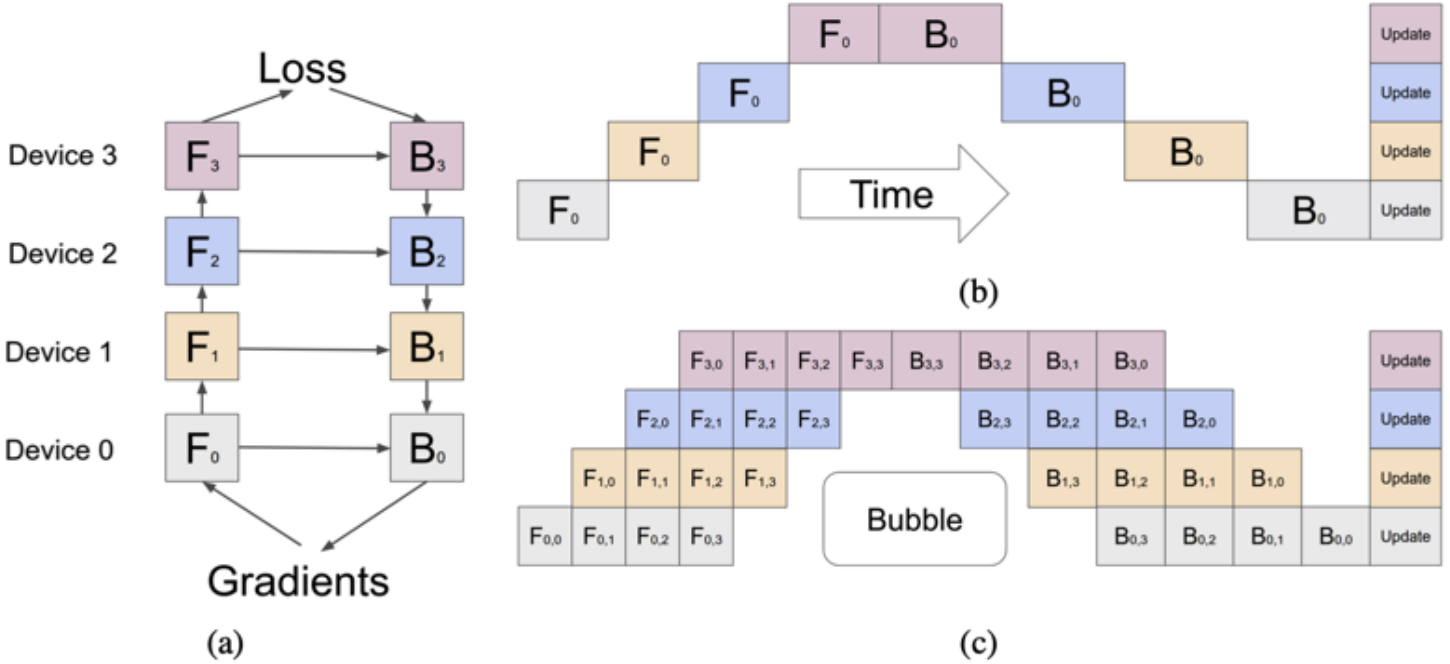


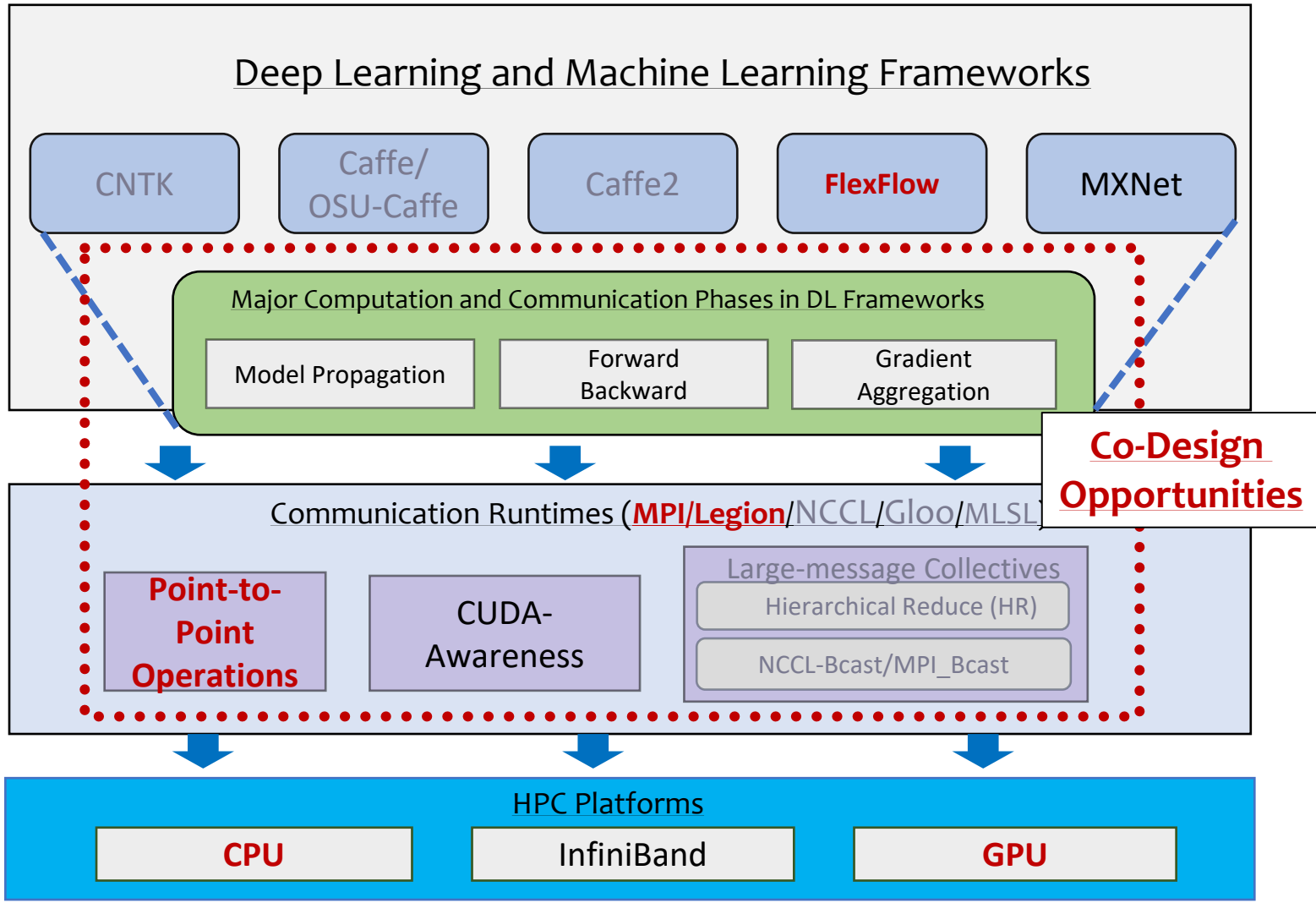
Table 3: Normalized training throughput using GPipe on GPUs without high-speed interconnect.

GPU	AmoebaNet			Transformer		
$K =$	2	4	8	2	4	8
$M = 32$	1	1.7	2.7	1	1.8	3.3

GPipe: https://arxiv.org/pdf/1811.06965.pdf

Solutions and Case Studies: Exploiting HPC for DL

- Data Parallelism
  - Baidu-allreduce
  - NVIDIA NCCL/NCCL2
  - Co-design MPI runtimes and DL Frameworks
  - Distributed Training for TensorFlow
- **Model and Hybrid Parallelism**
  - GPipe
  - **FlexFlow**
  - HyPar-Flow
  - GEMS
  - SUPER



FlexFlow: Beyond Data and Model Parallelism

- FlexFlow: identify more dimensions of parallelism in DNNs
- Develop strategies and graphs for parallel training
- Search algorithms for finding the best parallelization strategy
- Uses Legion tasks for distributed training on GPUs

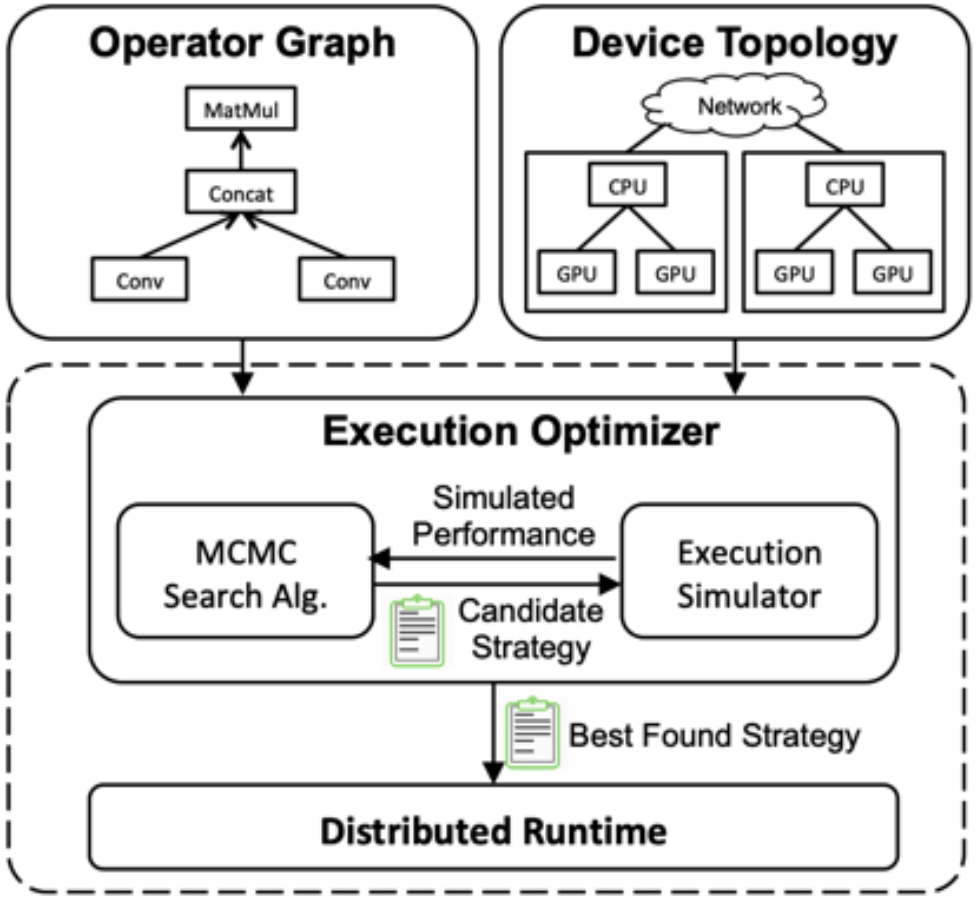
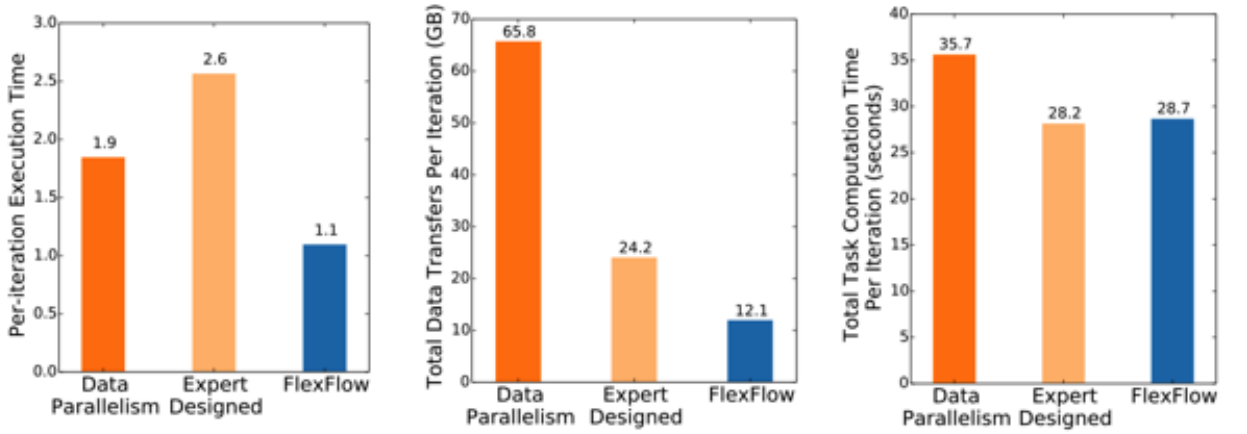


Figure 2: FlexFlow overview.

Courtesy: https://arxiv.org/pdf/1807.05358.pdf

FlexFlow: Performance Benefits

- FlexFlow: identify more dimensions of parallelism in DNNs
- Develop strategies and graphs for parallel training
- Search algorithms for finding the best parallelization strategy
- Uses Legion tasks for distributed training on GPUs



(a) Per-iteration execution time.

(b) Overall data transfers per iteration.

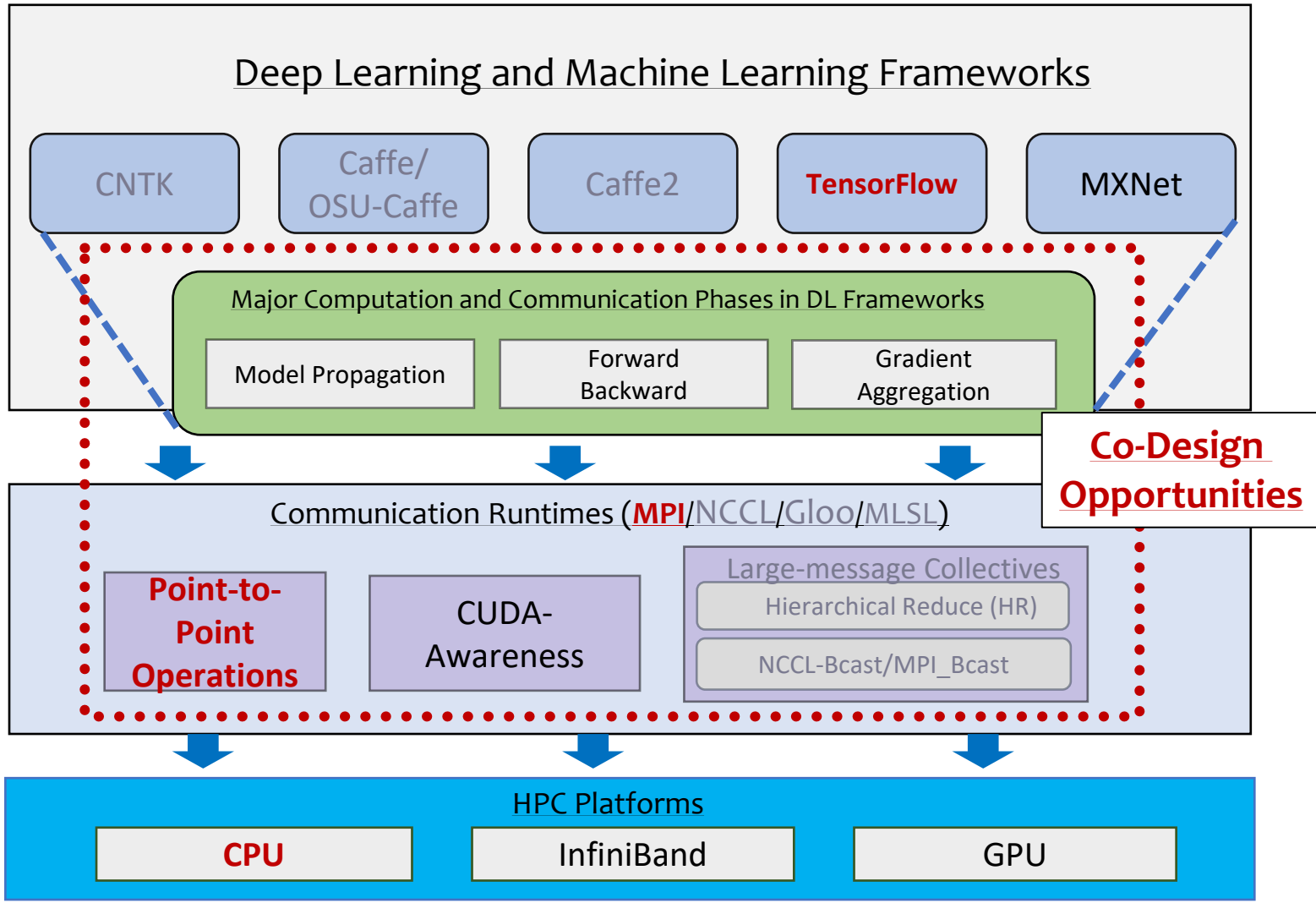
(c) Overall task computation time per iteration.

Figure 8: Parallelization performance for the NMT model on 64 K80 GPUs (16 nodes). FlexFlow reduces per-iteration execution time by 1.7-2.4× and data transfers by 2-5.5× compared to other approaches. FlexFlow achieves similar overall task computation time as expert-designed strategy, which is 20% fewer than data parallelism.

Courtesy: https://arxiv.org/pdf/1807.05358.pdf

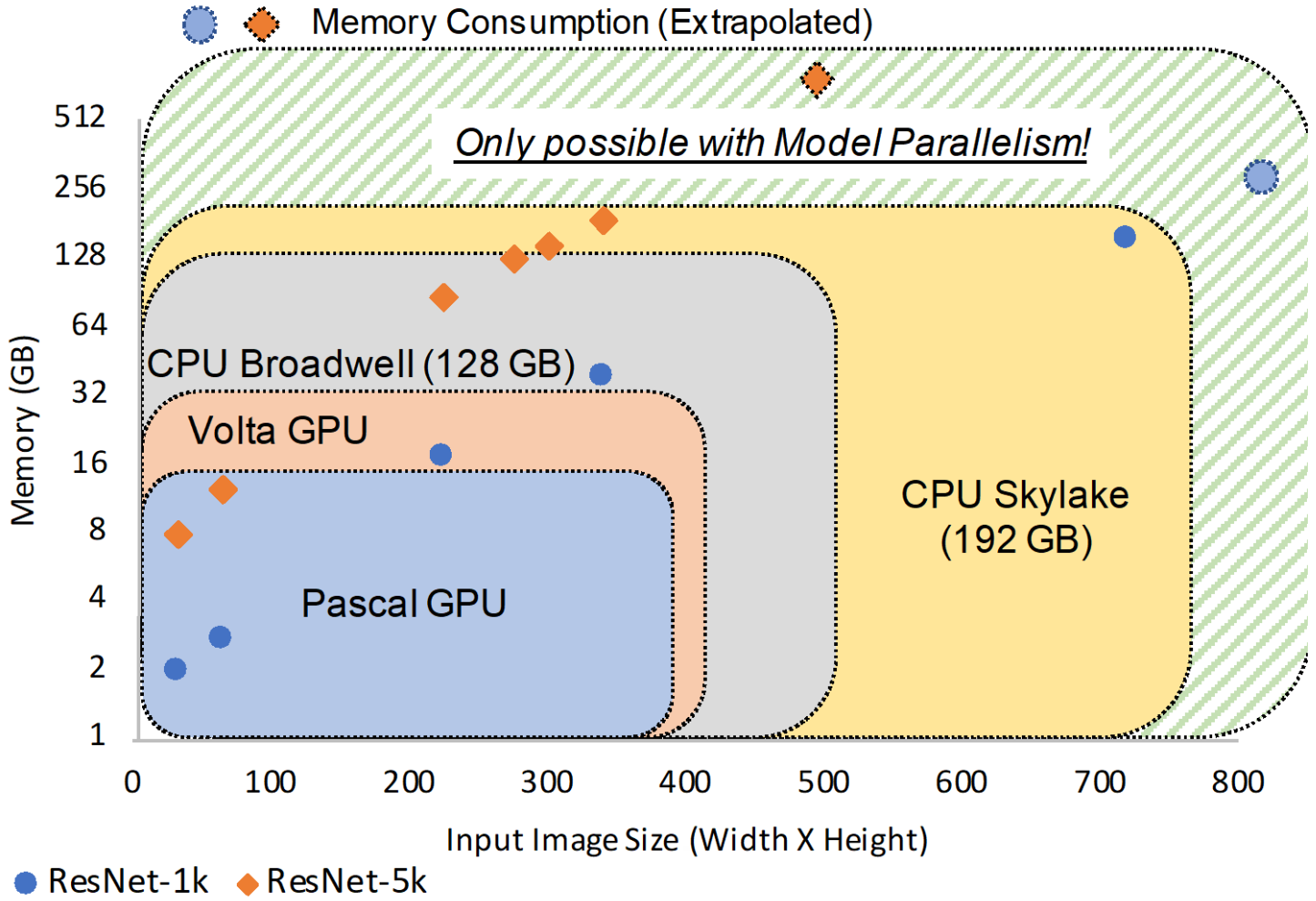
Solutions and Case Studies: Exploiting HPC for DL

- Data Parallelism
  - Baidu-allreduce
  - NVIDIA NCCL/NCCL2
  - Co-design MPI runtimes and DL Frameworks
  - Distributed Training for TensorFlow
- **Model and Hybrid Parallelism**
  - GPipe
  - FlexFlow
  - **HyPar-Flow**
  - GEMS
  - SUPER



HyPar-Flow: Hybrid Parallelism for TensorFlow

- Why Hybrid parallelism?
  - Data Parallel training has limits! →
- We propose HyPar-Flow
  - An easy to use Hybrid parallel training framework
    - Hybrid = Data + Model
  - Supports Keras models and exploits TF 2.0 Eager Execution
  - Exploits MPI for Point-to-point and Collectives

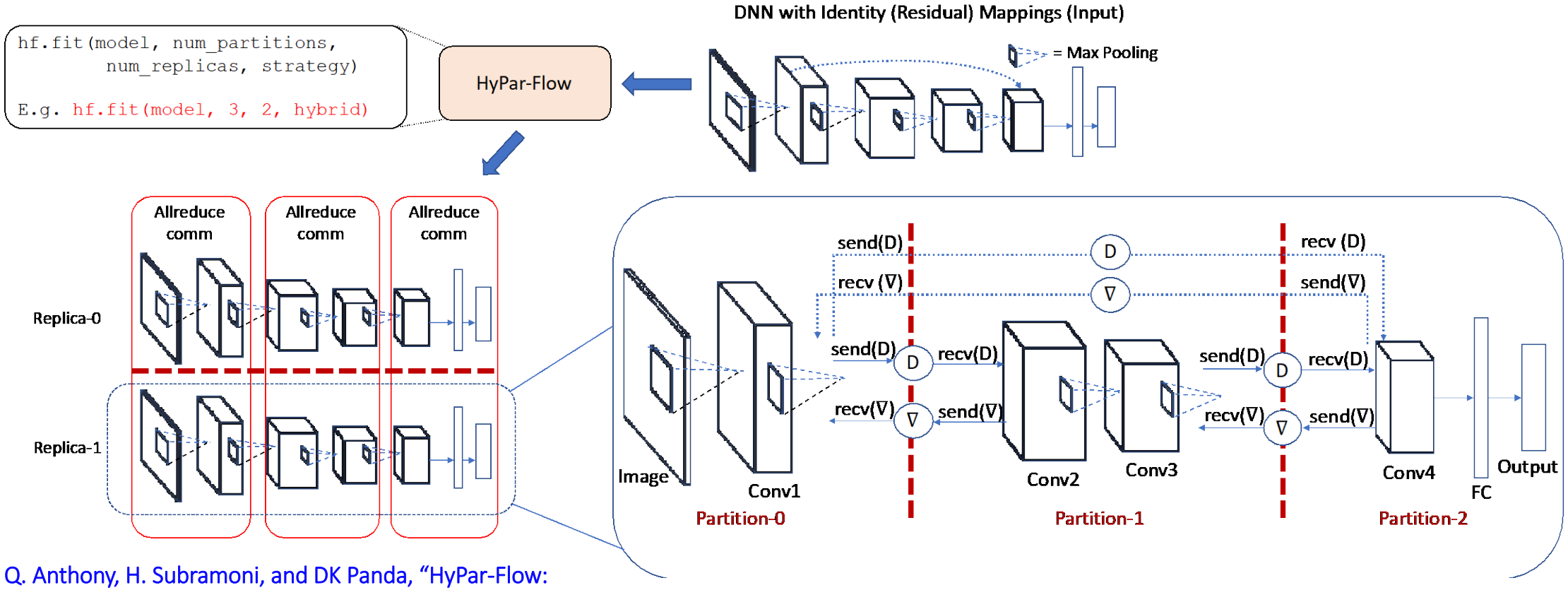


Benchmarking large-models lead to better insights and ability to develop new approaches!

A. A. Awan, A. Jain, Q. Anthony, H. Subramoni, and DK Panda, "HyPar-Flow: Exploiting MPI and Keras for Hybrid Parallel Training of TensorFlow models", ISC '20, https://arxiv.org/pdf/1911.05146.pdf

Model/Hybrid Parallelism and MPI Collectives

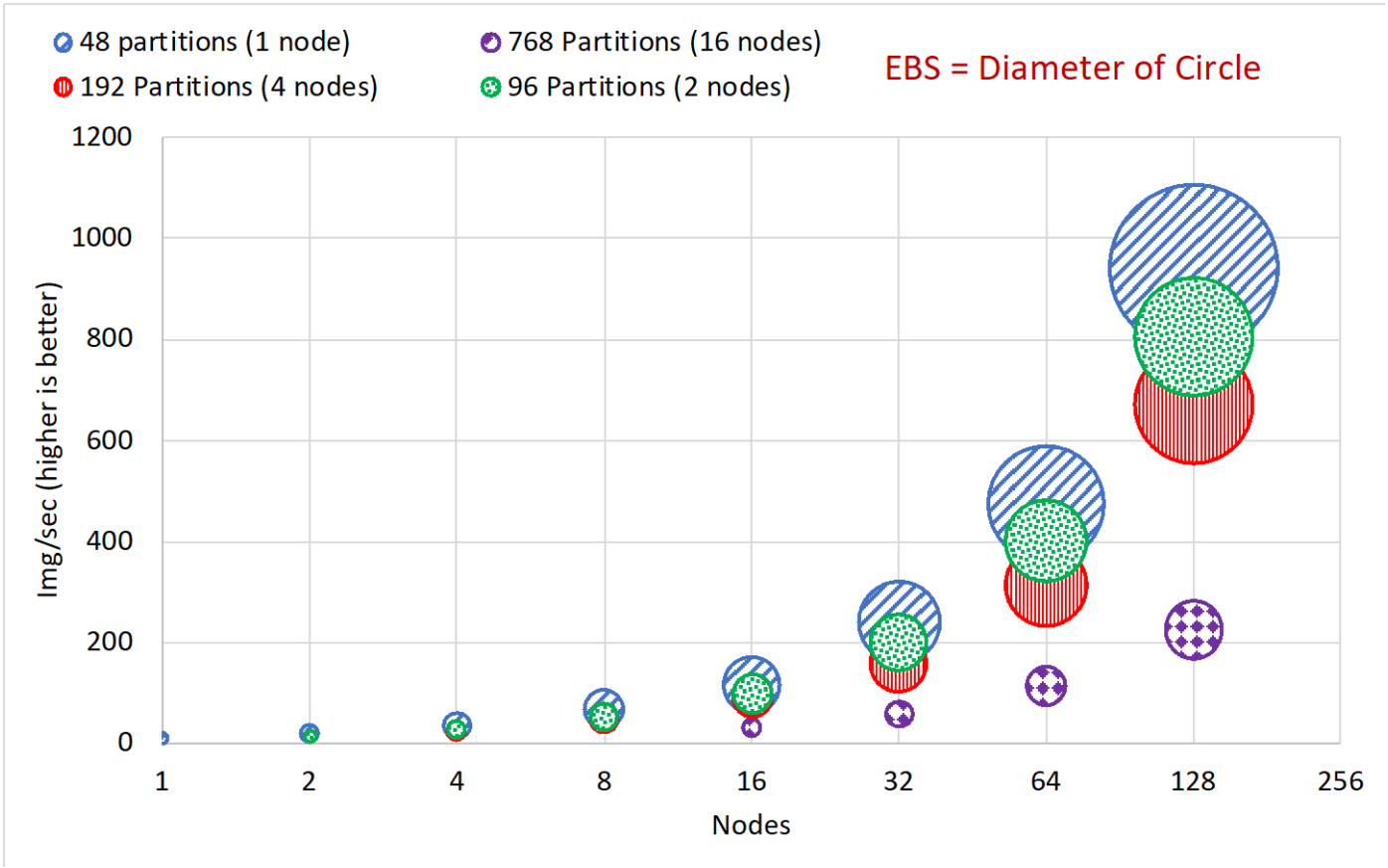
- HyPar-Flow is practical (easy-to-use) and high-performance (uses MPI)
  - Based on Keras models and exploits TF 2.0 Eager Execution
  - Leverages MPI Pt-to-pt. and Collectives for communication



A. A. Awan, A. Jain, Q. Anthony, H. Subramoni, and DK Panda, "HyPar-Flow: Exploiting MPI and Keras for Hybrid Parallel Training of TensorFlow models", ISC '20, https://arxiv.org/pdf/1911.05146.pdf

Benefits of Model/Hybrid Parallel Training: Using HyPar-Flow

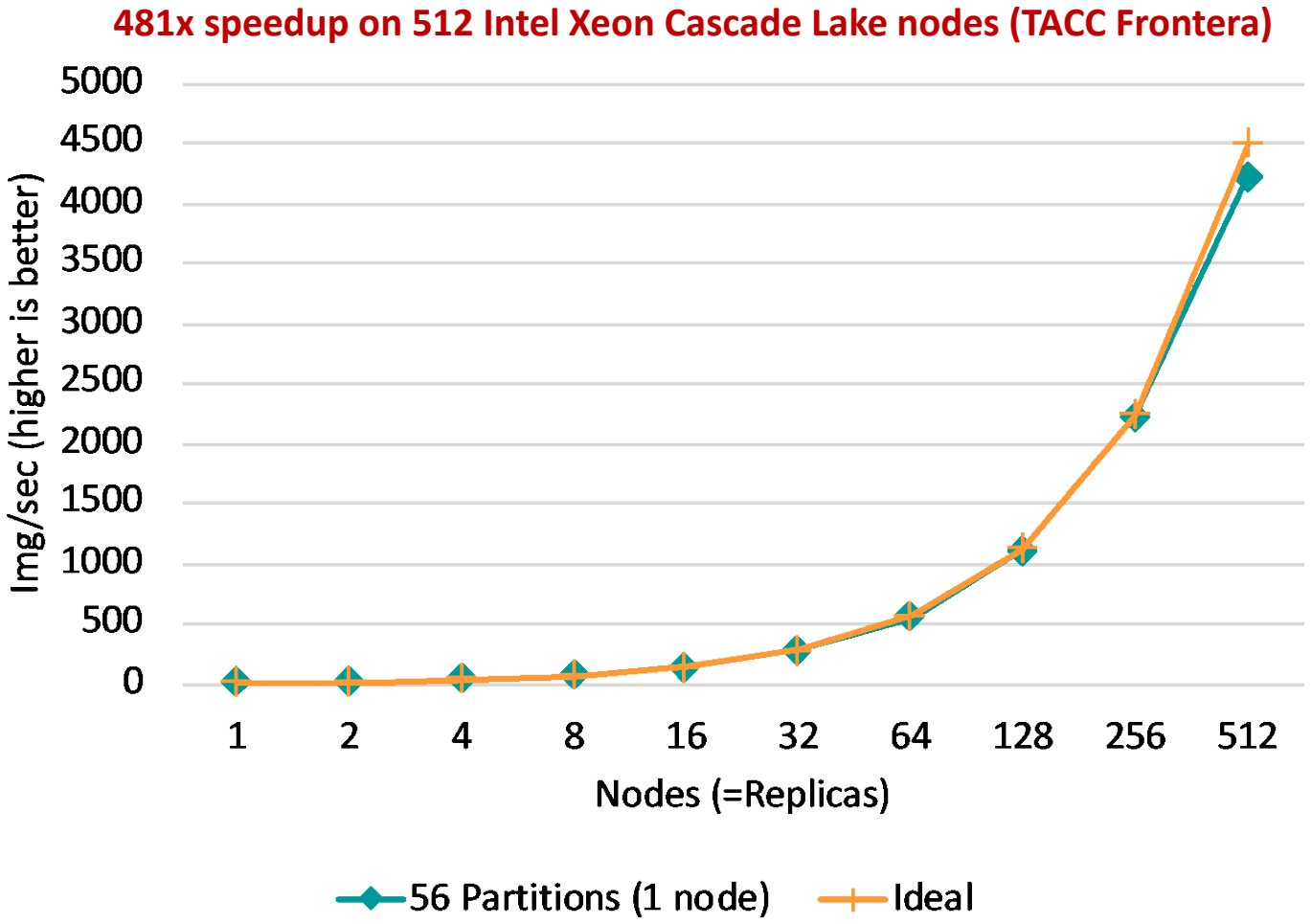
- HyPar-Flow*: Hybrid Parallel training of TensorFlow models
- Exploit MPI for communication and Keras for model definitions
- Hybrid→ combination of Model and Data parallelism
- Speedup over one node: **110x on 128 nodes**
- EBS = Effective batch size



A. A. Awan, A. Jain, Q. Anthony, H. Subramoni, and DK Panda, "HyPar-Flow: Exploiting MPI and Keras for Hybrid Parallel Training of TensorFlow models", ISC '20, https://arxiv.org/pdf/1911.05146.pdf

HyPar-Flow at Scale (512 nodes on TACC Frontera)

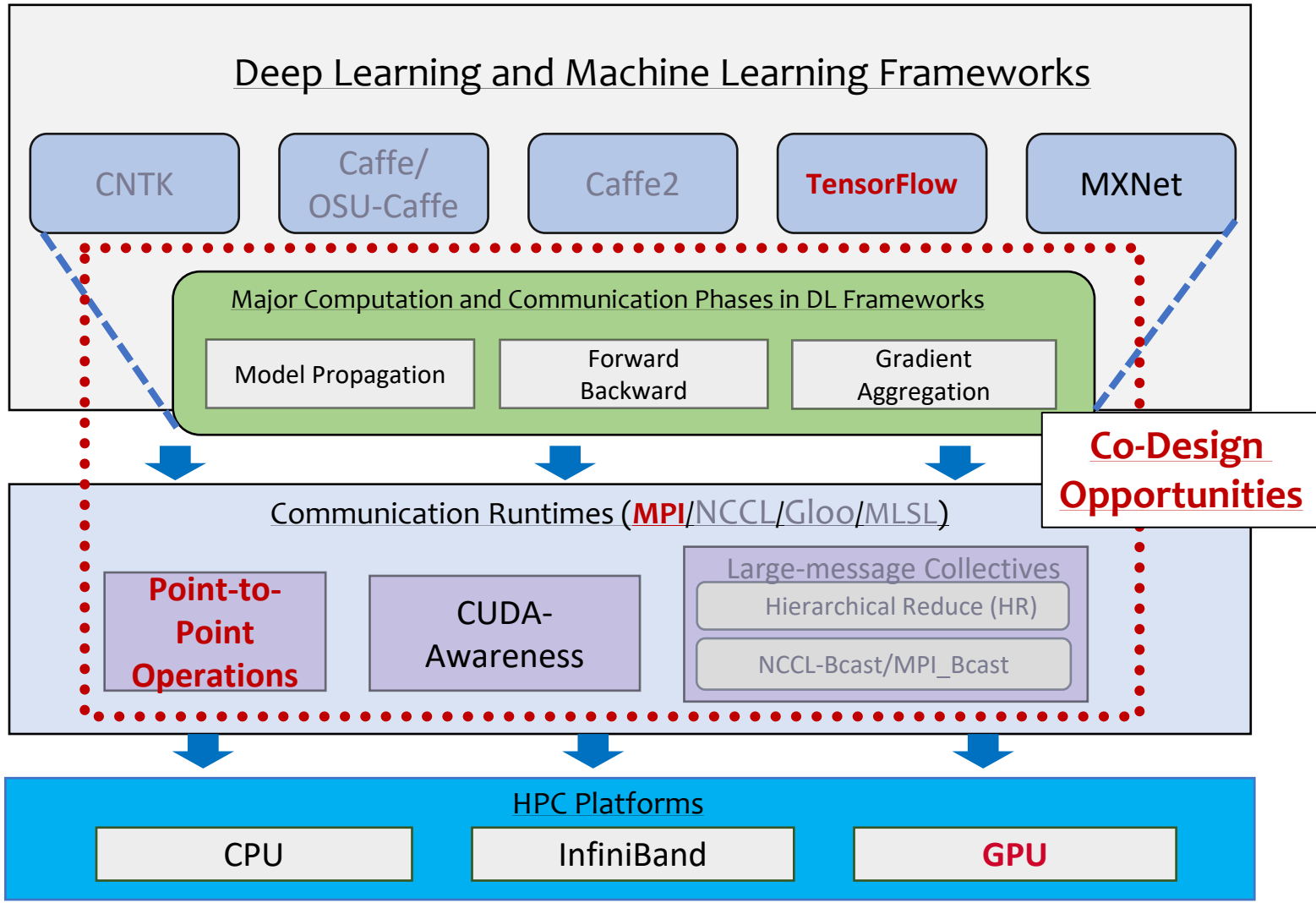
- ResNet-1001 with variable batch size
- Approach:
  - 48 model-partitions for 56 cores
  - 512 model-replicas for 512 nodes
  - Total cores: 56 x 512 = 28,672
- Speedup
  - **253X** on 256 nodes
  - **481X** on 512 nodes
- Scaling Efficiency
  - **98%** up to 256 nodes
  - **93.9%** for 512 nodes



A. A. Awan, A. Jain, Q. Anthony, H. Subramoni, and DK Panda, "HyPar-Flow: Exploiting MPI and Keras for Hybrid Parallel Training of TensorFlow models", ISC '20, https://arxiv.org/pdf/1911.05146.pdf

Solutions and Case Studies: Exploiting HPC for DL

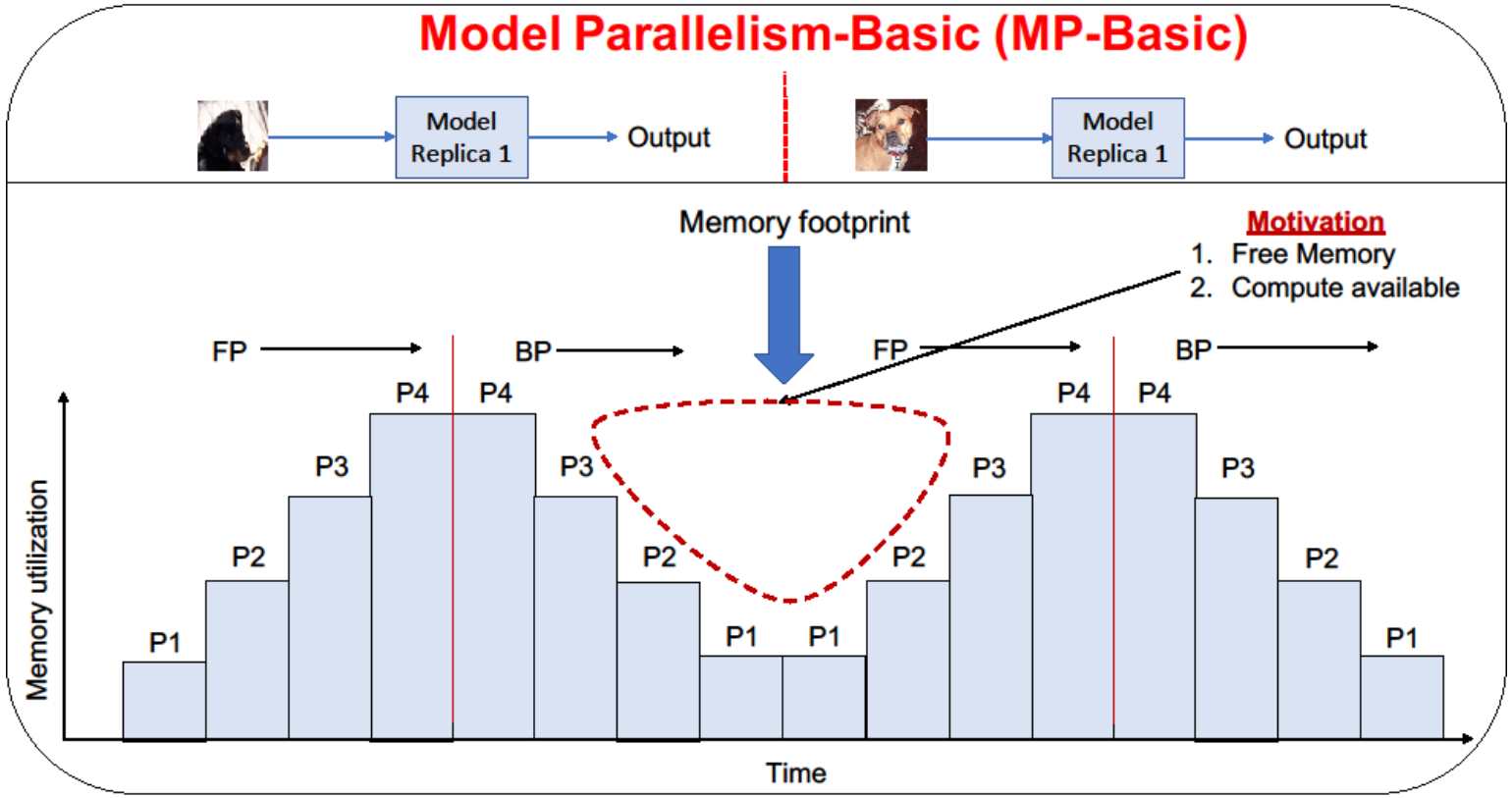
- Data Parallelism
  - Baidu-allreduce
  - NVIDIA NCCL/NCCL2
  - Co-design MPI runtimes and DL Frameworks
  - Distributed Training for TensorFlow
- **Model and Hybrid Parallelism**
  - GPipe
  - FlexFlow
  - HyPar-Flow
  - **GEMS**
  - SUPER



GEMS: GPU Enabled Memory Aware Model Parallelism Systems

Why do we need Memory aware designs?

- Data and Model Parallel training has limitation!
- Maximum Batch Size depends on the memory.
- Basic Model Parallelism suffers from underutilization of memory and compute →



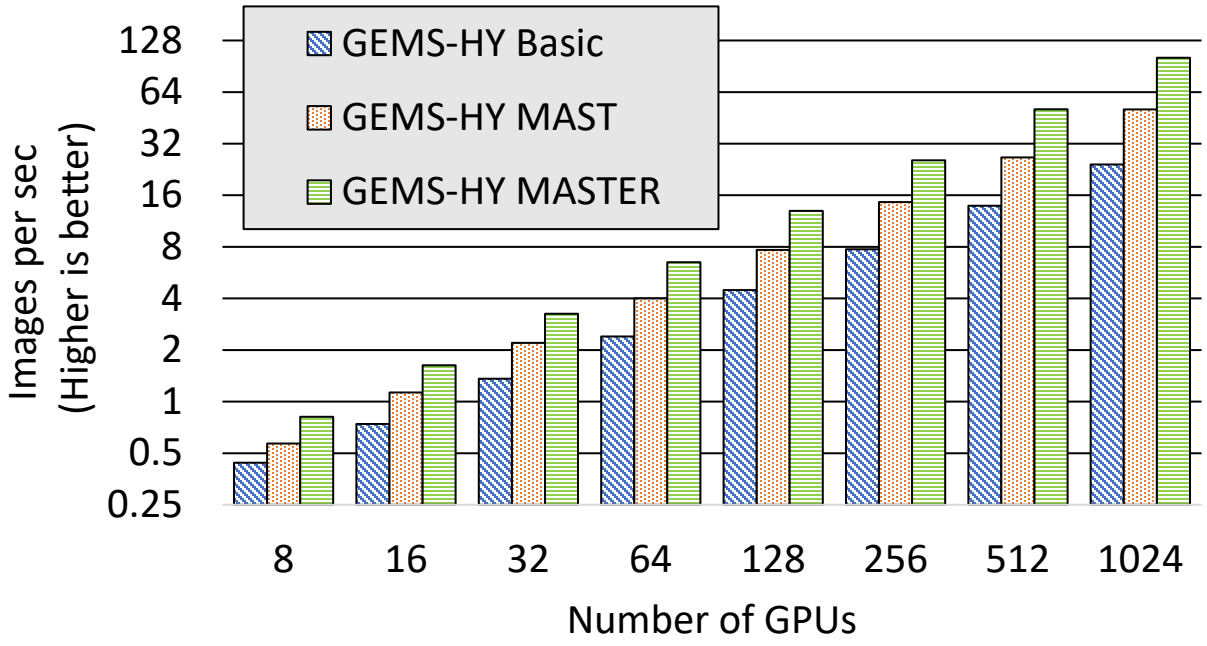
Memory requirement increases with the increase in image size!

A. Jain, A. Awan, A. Aljuhani, J. Hashmi, Q. Anthony, H. Subramoni, D. Panda, R. Machiraju, A. Parwani, "GEMS: GPU Enabled Memory Aware Model Parallelism System for Distributed DNN", SC '20

GEMS at Scale (1,024 V100 GPUs on LLNL Lassen)

- We propose memory aware designs to accelerate the training of DNN on large image sizes.
- Approaches:
  - Memory Aware Synchronized Training (MAST)
  - Memory Aware Synchronized Training with Enhanced Replications (MASTER)
- Setup
  - ResNet-1k on 512 X 512 images
  - 128 Replications on 1024 GPUs
- Scaling Efficiency
  - **97.32%** on 1024 nodes

97.32% scaling efficiency on 1024 V100 GPUs (LLNL Lassen)

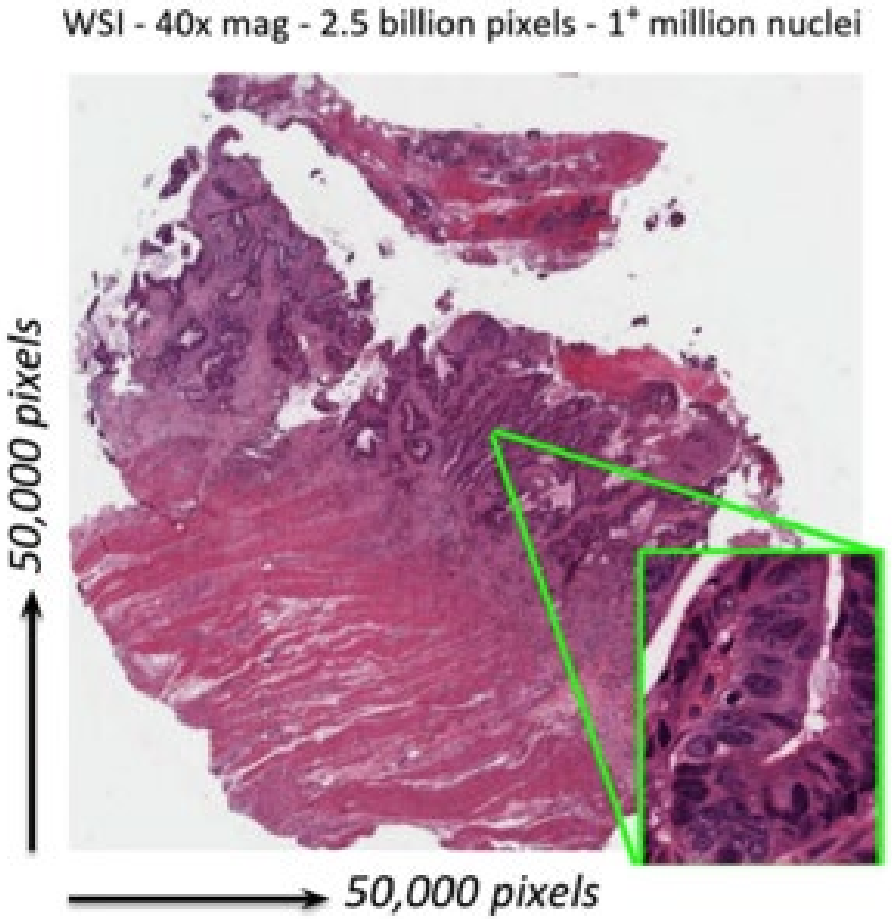


A. Jain, A. Awan, A. Aljuhani, J. Hashmi, Q. Anthony, H. Subramoni, D. Panda, R. Machiraju, A. Parwani, "GEMS: GPU Enabled Memory Aware Model Parallelism System for Distributed DNN", SC '20

Exploiting Model Parallelism in AI-Driven Digital Pathology

- Pathology whole slide image (WSI)
  - Each WSI = 100,000 x 100,000 pixels
  - Can not fit in a single GPU memory
  - Tiles are extracted to make training possible
- Two main problems with tiles
  - Restricted tile size because of GPU memory limitation
  - Smaller tiles loose structural information
- Can we use Model Parallelism to train on larger tiles to get better accuracy and diagnosis?
- Reduced training time significantly
  - **32 hours (1 node, 1 GPU) -> 7.25 hours (1 node, 4 GPUs) ->**

    27 mins (32 nodes, 128 GPUs)

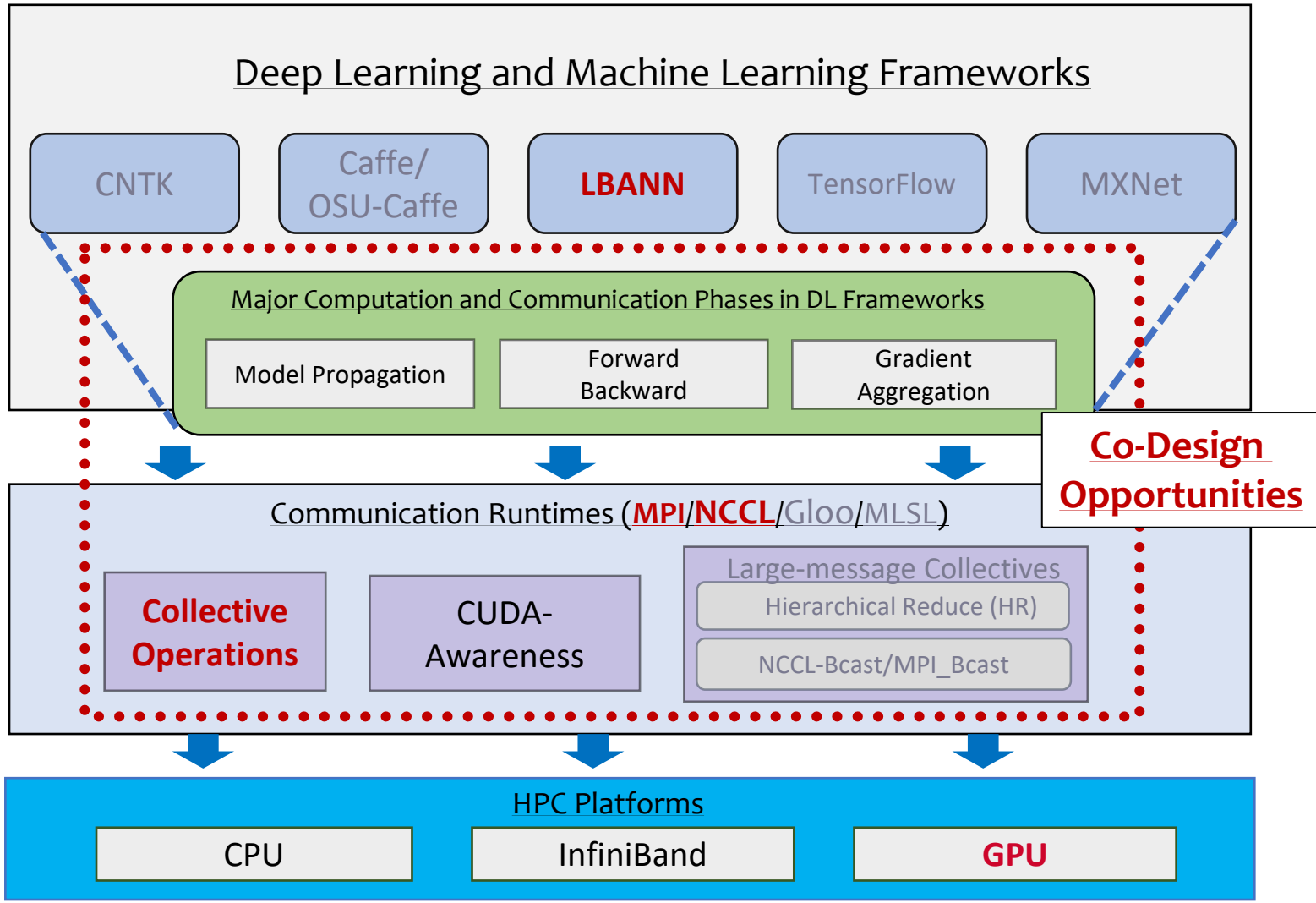


Courtesy: https://blog.kitware.com/digital-slide-archive-large-image-and-histomicstk-open-source-informatics-tools-for-management-visualization-and-analysis-of-digital-histopathology-data/

A. Jain, A. Awan, A. Aljuhani, J. Hashmi, Q. Anthony, H. Subramoni, D. Panda, R. Machiraju, A. Parwani, "GEMS: GPU Enabled Memory Aware Model Parallelism System for Distributed DNN", SC '20

Solutions and Case Studies: Exploiting HPC for DL

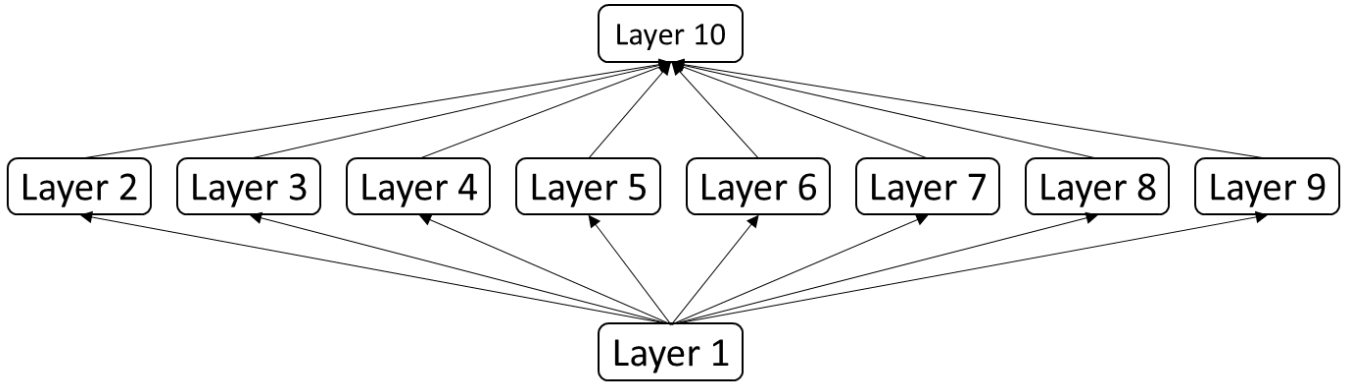
- Data Parallelism
  - Baidu-allreduce
  - NVIDIA NCCL/NCCL2
  - Co-design MPI runtimes and DL Frameworks
  - Distributed Training for TensorFlow
- **Model and Hybrid Parallelism**
  - GPipe
  - FlexFlow
  - HyPar-Flow
  - GEMS
  - **SUPER**



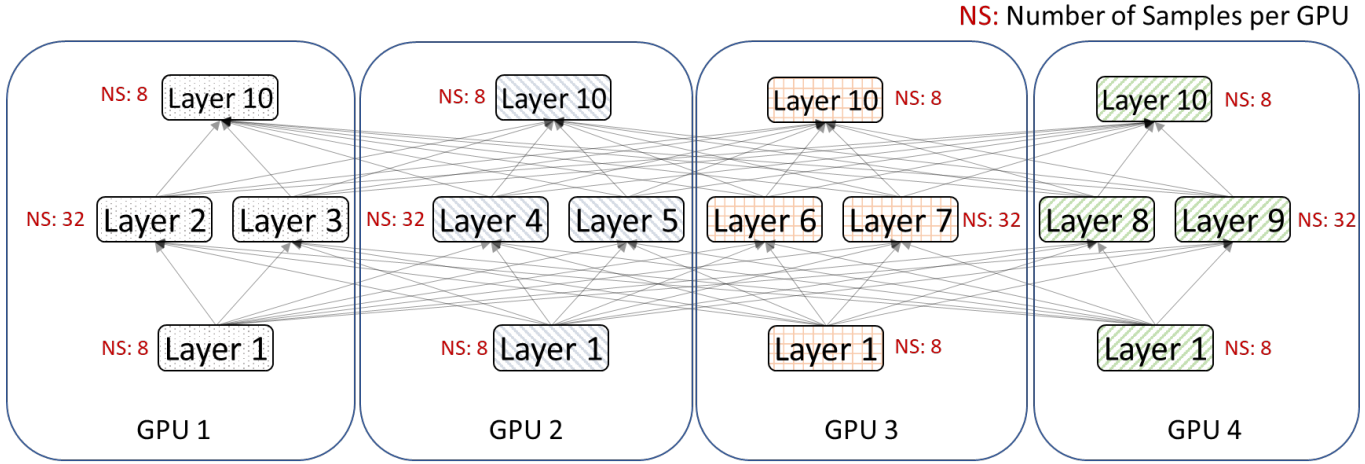
SUPER: SUb-Graph Parallelism for TransformERs

Sub-Graph Parallelism

- Exploits inherent parallelism in modern DNN architectures
- Improves the Performance of multi-branch DNN architectures →
- Can be used to accelerate the training of state-of-the-art Transformer models
- Provides better than Data-Parallelism for in-core models



Simple example of a multi-branch DNN architecture



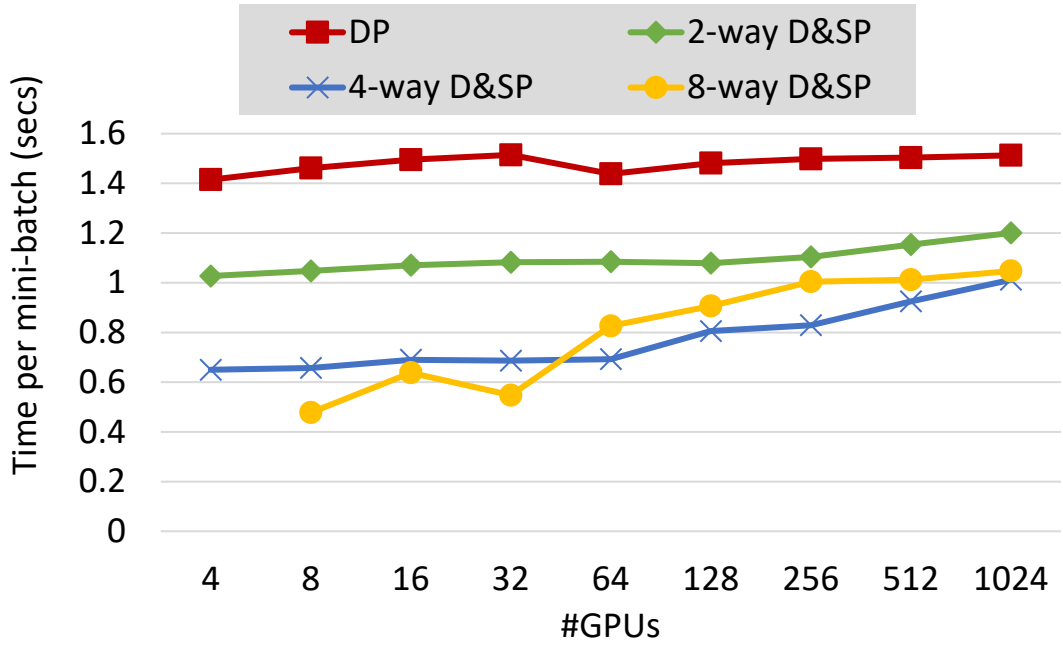
4-way Sub-Graph Parallelism combined with Data-Parallelism (D&SP)

A. Jain, T. moon, T. Benson, H. Subramoni, S. Jacobs, D. Panda, B. Essen, "SUPER: SUb-Graph Parallelism for TransformERs", IPDPS '21

Accelerating Transformers using SUPER

- We propose sub-graph parallelism integrated with data parallelism to accelerate the training of Transformers.
- Approach
  - Data and Sub-Graph Parallelism (D&SP)
    - #-way D&SP (#: number of sub-graphs)
- Setup
  - T5-Large-Mod on WMT Dataset
  - 1024 NVIDIA V100 GPUs
- Speedup
  - Up to **3.05X** over Data Parallelism (DP)

Up to 3.05X speedup over Data Parallel designs (LLNL Lassen)



A. Jain, T. moon, T. Benson, H. Subramoni, S. Jacobs, D. Panda, B. Essen, "SUPER: SUb-Graph Parallelism for TransformERs", IPDPS '21

Outline

- Introduction
- Overview of Execution Environments
- Parallel and Distributed DNN Training
- Latest Trends in High-Performance Computing Architectures
- Challenges in Exploiting HPC Technologies for Deep Learning
- Solutions and Case Studies
- **Open Issues and Challenges**
- Hands-on Exercises
- Conclusion

Open Issues and Challenges

- Convergence of DL and HPC
- Scalability and Large batch-size training?
- DL Benchmarks and Thoughts on Standardization
- Open Exchange and Making AI accessible?

Convergence of DL and HPC

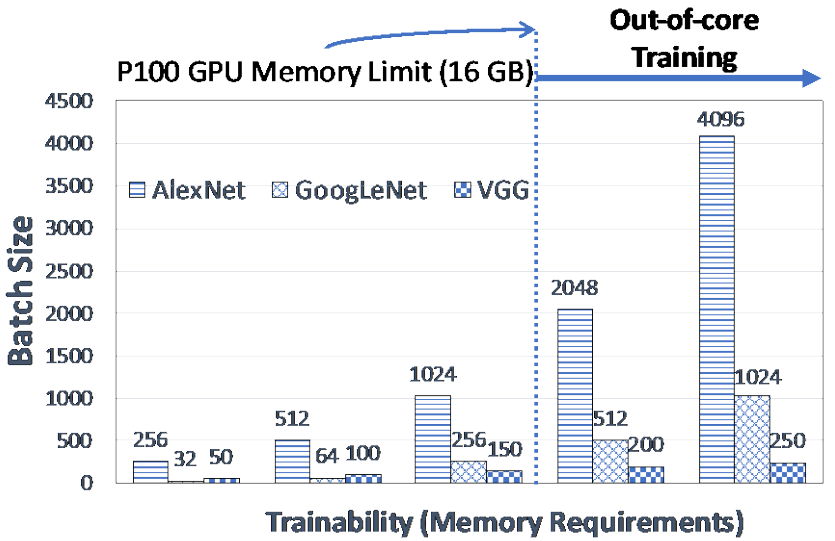
- Is Deep Learning an HPC Problem?
  - Distributed DNN Training is definitely an HPC problem
  - Inference – not yet an HPC problem
- Why HPC can help?
  - Decades of research for communication models and performance optimizations
  - MPI, PGAS, and other upcoming programming models and communication runtimes can help for “data-parallel” training
- Some of the needs for DNN training are an exact match
  - Compute intensive problem
- Some needs are new for distributed/parallel communication runtimes
  - Large Message Communication
  - CUDA-Aware Communication

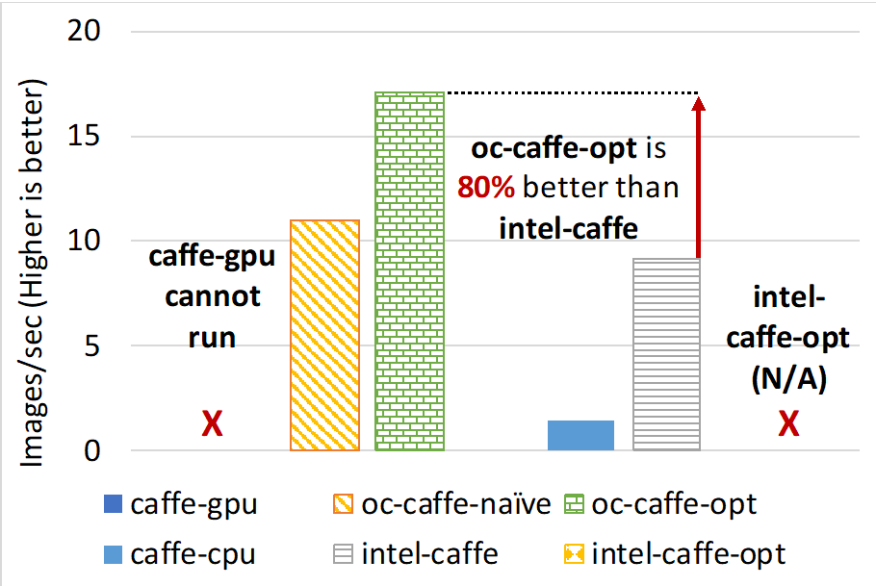
Scalability and Large batch-size training?

- Large batch-size helps improve the scalability
  - Lesser communication and more compute before synchronization
  - Limits to large batch-size
    - DL community is actively exploring this area
    - HPC community can also investigate overlap and latency-hiding techniques
- Is there a limit to DNN size?
  - Noam Shazeer's Outrageously Large Model (137 Billion Parameters)
  - https://arxiv.org/pdf/1701.06538.pdf
- Out-of-core Training for GPUs?
  - NVIDIA's vDNN - https://arxiv.org/pdf/1602.08124.pdf
  - Prune the network or selectively allocate/de-allocate memory on GPUs
  - OC-DNN and OC-Caffe

Scalability and Large (Out-of-core) Models?

- Large DNNs cannot be trained on GPUs due to memory limitation!
  - ResNet-50 for Image Recognition but current frameworks can only go up to a small batch size of 45
  - Next generation models like Neural Machine Translation (NMT) are ridiculously large, consists of billions of parameters, and require even more memory
  - Can we design Out-of-core DNN training support using new software features in CUDA 8/9 and hardware mechanisms in Pascal/Volta GPUs?
- General intuition is that managed allocations "will be" slow!
  - The proposed framework called **OC-Caffe (Out-of-Core Caffe)** shows the potential of managed memory designs that can provide performance with negligible/no overhead.
- OC-Caffe-Opt: up to **80% better** than Intel-optimized CPU Caffe for ResNet-50 training on the Volta V100 GPU with CUDA9 and CUDNN7





A. A. Awan, C.-H. Chu, H. Subramoni, X. Lu, and D. K. Panda, OC-DNN: Exploiting Advanced Unified Memory Capabilities in CUDA 9 and Volta GPUs for Out-of-Core DNN Training, HiPC '18

DL Benchmarks and Thoughts on Standardization

- Can we have a standardized interface?
  - Are we there yet?
  - Deep Learning Interface (DLI)? Inspired by Message Passing Interface (MPI)
    - What can be a good starting point?
    - Will it come from the HPC community or the DL community?
    - Can there be a collaboration across communities?
- What about standard benchmarks? Is there a need?
  - State-of-the-art
    - HKBU benchmarks - http://dlbench.comp.hkbu.edu.hk
    - Soumith Chintala's benchmarks - https://github.com/soumith/convnet-benchmarks
    - DAWN Bench – https://dawn.cs.stanford.edu/benchmark/
    - MLPerf – https://www.mlperf.org -- Latest and Widely Promoted now!

Open Exchange and Making AI accessible?

- OpenAI – a company focused towards making AI accessible and open
  - Backed up by several industry partners and individuals
    - Amazon, Microsoft, Infosys, Elon Musk, Peter Thiel, and others..
  - *Latest News: Microsoft will invest $1 Billion in OpenAI R&D*
  - https://www.hpcwire.com/2019/07/22/microsoft-investing-1b-in-openai-artificial-general-intelligence-rd/

- ONNX format
  - An open format to exchange trained models
  - Cross-framework compatibility
  - Created by Facebook and Microsoft
  - TensorFlow and CoreML (Apple) are also supported (Convertor only)

Outline

- Introduction
- Overview of Execution Environments
- Parallel and Distributed DNN Training
- Latest Trends in High-Performance Computing Architectures
- Challenges in Exploiting HPC Technologies for Deep Learning
- Solutions and Case Studies
- Open Issues and Challenges
- **Hands-on Exercises**
- Conclusion

Getting Set-up for the Hands-on Exercises

- You will run the experiments on the OSU RI2 cluster
- **Please use the account name and password from http://go.osu.edu/dltutorial**
- E.g. ssh ri2tut01@ri2.cse.ohio-state.edu and *tutorial01* as password
- Once on the shell, go to /opt/tutorials/dl-tutorial-21/labs (copy/paste the following line)

```
cd /opt/tutorials/dl-tutorial-21/labs
```

There are two folders for exercises (lab 1 and lab2) and one for homework (hw)

- Take a look at the README.md file for all scripts
  - copy/paste the run commands from README.md and not the slide deck

Lab1 - Overview

- Objectives
  - How to train a PyTorch model on a single NVIDIA GPU?
  - How to perform distributed training of a PyTorch model on multiple GPUs using InfiniBand and NVIDIA GPUs?
- Tasks
  - Run PyTorch on a Single GPU
  - Run PyTorch on two nodes with 1 GPU/node (using MVAPICH2)
  - Run PyTorch on two nodes with 1 GPU/node (using MVAPICH2-GDR)

Distributed Training with PyTorch using Horovod

- Examples to run data-parallel training with PyTorch using Horovod
- Available from: https://github.com/horovod/horovod/tree/master/examples

- To run ResNet50 with synthetic data with a single GPU, run

```
python pytorch_synthetic_benchmark.py \
--batch_size=32 \
----num-iters=10 \
```

Lab1-Task1: Run PyTorch on a single GPU

```
$ cd /opt/tutorials/dl-tutorial-21/labs/lab1
$ srun -N 1 --reservation=dltutorial run_pytorch_bench_single.sh
```

+ /opt/tutorials/dl-tutorial-21/miniconda3/envs/pytorch_mv2/bin/python /opt/tutorials/dl-tutorial-21/horovod/examples/pytorch/pytorch_synthetic_benchmark.py --batch-size 64 --num-iters=5

.

.

V100

```
Model: resnet50
Batch size: 64
Number of GPUs: 1
Running warmup...
Running benchmark...
Iter #0: 333.9 img/sec per GPU
Iter #1: 334.2 img/sec per GPU
Iter #2: 333.9 img/sec per GPU
Iter #3: 333.8 img/sec per GPU
Iter #4: 333.9 img/sec per GPU
Img/sec per GPU: 334.0 +-0.2
----------------------------------------
Total img/sec on 1 GPU(s): 334.0 +-0.2
----------------------------------------
```

Lab1-Task2: Run PyTorch on two nodes with 1 GPU/node (using MVAPICH2)

```
$ srun -N 2 --reservation=dltutorial run_pytorch_bench_multi_mv2.sh
```

+ /opt/tutorials/dl-tutorial-21/mv2/bin/mpirun_rsh -np 2 gpu11 gpu12 MV2_USE_CUDA=1 MV2_CPU_BINDING_POLICY=hybrid MV2_HYBRID_BINDING_POLICY=spread MV2_USE_RDMA_CM=0 /opt/tutorials/dl-tutorial-21/miniconda3/envs/pytorch_mv2/bin/python /opt/tutorials/dl-tutorial-21/horovod/examples/pytorch/pytorch_synthetic_benchmark.py --batch-size 64 --num-iters=5

.
.

V100

```
Model: resnet50
Batch size: 64
Number of GPUs: 2
Running warmup...
Running benchmark...
Iter #0: 247.4 img/sec per GPU
Iter #1: 254.6 img/sec per GPU
Iter #2: 255.8 img/sec per GPU
Iter #3: 261.9 img/sec per GPU
Iter #4: 261.0 img/sec per GPU
Img/sec per GPU: 256.1 +-10.3
-----------------------------------------
Total img/sec on 2 GPU(s): 512.3 +-20.6
-----------------------------------------
```

~1.53X on 2 GPUs

Lab1-Task3: Run PyTorch on two nodes with 1 GPU/node (using MVAPICH2-GDR)

```
$ srun -N 2 --reservation=dltutorial run_pytorch_bench_multi_mv2gdr.sh
```

+ /opt/tutorials/dl-tutorial-21/mv2-gdr/bin/mpirun_rsh -np 2 gpu11 gpu12 MV2_USE_CUDA=1 MV2_CPU_BINDING_POLICY=hybrid MV2_HYBRID_BINDING_POLICY=spread MV2_USE_RDMA_CM=0 MV2_GPUDIRECT_GDRCOPY_LIB=/opt/gdrcopy2.0/lib64/libgdrapi.so LD_PRELOAD=/opt/tutorials/dl-tutorial-21/mv2-gdr/lib/libmpi.so /opt/tutorials/dl-tutorial-21/miniconda3/envs/pytorch_gdr/bin/python /opt/tutorials/dl-tutorial-21/horovod/examples/pytorch/pytorch_synthetic_benchmark.py --batch-size 64 --num-iters=5

.

.

V100

```
Model: resnet50
Batch size: 64
Number of GPUs: 2
Running warmup...
Running benchmark...
Iter #0: 317.0 img/sec per GPU
Iter #1: 314.9 img/sec per GPU
Iter #2: 315.4 img/sec per GPU
Iter #3: 318.0 img/sec per GPU
Iter #4: 316.7 img/sec per GPU
Img/sec per GPU: 316.4 +-2.2
-----------------------------------------
Total img/sec on 2 GPU(s): 632.8 +-4.3
-----------------------------------------
```

~1.89X on 2 GPUs

Lab2 - Overview

- Objectives
  - How to train a TensorFlow model on a single NVIDIA GPU?
  - How to perform distributed training of a TensorFlow model on multiple GPUs using InfiniBand and NVIDIA GPUs?
- Tasks
  - Run TensorFlow on a Single GPU
  - Run TensorFlow on two nodes with 1 GPU/node (using MVAPICH2-GDR)

Distributed Training with TensorFlow using Horovod

- Examples to run data-parallel training with TensorFlow using Horovod
- Available from: https://github.com/horovod/horovod/tree/master/examples

- To run ResNet50 with synthetic data with a single GPU, run

```
python tensorflow2_synthetic_benchmark.py\
--batch_size=32 \
----num-iters=10 \
```

Lab2-Task1: Run TensorFlow on a Single GPU

```
$ cd /opt/tutorials/dl-tutorial-21/labs/lab2
$ srun -N 1 --reservation=dltutorial run_tf_bench_single.sh
```

+ /opt/tutorials/dl-tutorial-21/miniconda3/envs/tf_mv2_gdr/bin/python /opt/tutorials/dl-tutorial-21/horovod/examples/tensorflow2//tensorflow2_synthetic_benchmark.py –batch-size 64

```
.
Model: ResNet50
Batch size: 64
Number of GPUs: 1
Running warmup...
Running benchmark...
Iter #0: 339.6 img/sec per GPU
Iter #1: 337.9 img/sec per GPU
Iter #2: 337.8 img/sec per GPU
Iter #3: 337.8 img/sec per GPU
Iter #4: 337.8 img/sec per GPU
Img/sec per GPU: 338.2 +-1.4
-------------------------------------------------
Total img/sec on 1 GPU(s): 338.2 +-1.4
-------------------------------------------------
```

Lab2-Task2: Run TensorFlow on two nodes with 1 GPU/node (using MVAPICH2-GDR)

```
$ srun -N 2 --reservation=dltutorial run_tf_bench_multi_mv2gdr.sh
```

+ /opt/tutorials/dl-tutorial-21/mv2-gdr/bin/mpirun_rsh -np 2 gpu11 gpu12 MV2_USE_CUDA=1 MV2_SUPPORT_DL=1 MV2_CPU_BINDING_POLICY=hybrid MV2_HYBRID_BINDING_POLICY=spread MV2_USE_RDMA_CM=0 /opt/tutorials/dl-tutorial-21/miniconda3/envs/tf_mv2_gdr/bin/python /opt/tutorials/dl-tutorial-21/horovod/examples/tensorflow2/tensorflow2_synthetic_benchmark.py --batch-size 64 --num-iters=5.

V100

```
Model: ResNet50
Batch size: 64
Number of GPUs: 2
Running warmup...
Running benchmark...
Iter #0: 310.8 img/sec per GPU
Iter #1: 314.4 img/sec per GPU
Iter #2: 312.7 img/sec per GPU
Iter #3: 313.8 img/sec per GPU
Iter #4: 314.5 img/sec per GPU
Img/sec per GPU: 313.2 +-2.7
-----------------------------------------
Total img/sec on 2 GPU(s): 626.5 +-5.4
-----------------------------------------
```

1.85X on 2 GPUs

Hands-on Exercises: Key Takeaways

- Deep Learning models can be trained in multiple ways
  - Examples to run data-parallel training with Horovod are available at "https://github.com/horovod/horovod/tree/master/examples"
  - Single/Multiple GPU jobs -- similar
  - Horovod can be configured MPI, GLOO, NCCL, and oneCCL.
  - MVAPICH2-GDR offers near-linear speedup for multi-node training
  - MVAPICH2-GDR with CUDA-aware design delivers better performance
  - TensorFlow gives slightly better performance than PyTorch for ResNet50.

Homework - Overview

- Objectives
  - End-to-end training performance
  - How to train a deep learning model for MNIST dataset on a single NVIDIA GPU?
  - How to perform distributed training for MNIST dataset on multiple GPUs using InfiniBand and NVIDIA GPUs?
- Tasks
  - Train a deep learning model for MNIST on a Single GPU
  - Train a deep learning model for MNIST on two nodes with 1 GPU/node (using MVAPICH2-GDR)

Training Deep Learning model for MNIST

- Examples to run data-parallel training with PyTorch using Horovod
- Available from: https://github.com/horovod/horovod/tree/master/examples

- To train DNN for MNIST on a single GPU, run

python pytorch_mnist.py\

--batch_size=**64**

HW: Train a DL model for MNIST Dataset

```
$ cd /opt/tutorials/dl-tutorial-21/labs/hw
$ time srun -N 1 --reservation=dltutorial
run_pytorch_mnist_single.sh
```

```
Test set: Average loss: 0.0553, Accuracy: 98.30%

real    5m56.449s
user    0m0.008s
sys     0m0.011s
```

```
$ time srun -N 2 --reservation=dltutorial
run_pytorch_mnist_multi_mv2gdr.sh
```

```
Test set: Average loss: 0.0537, Accuracy: 98.33%

real    1m43.860s
user    0m0.007s
sys     0m0.013s
```

Outline

- Introduction
- Overview of Execution Environments
- Parallel and Distributed DNN Training
- Latest Trends in High-Performance Computing Architectures
- Challenges in Exploiting HPC Technologies for Deep Learning
- Solutions and Case Studies
- Open Issues and Challenges
- Hands-on Exercises
- **Conclusion**

Conclusion

- Exponential growth in Deep Learning frameworks
- Provided an overview of issues, challenges, and opportunities for communication runtimes
  - Efficient, scalable, and hierarchical designs are crucial for DL frameworks
  - Co-design of communication runtimes and DL frameworks will be essential
    - OSU-Caffe
    - TensorFlow (Baidu, Uber's Horovod, etc.)
    - Neon and Nervana Graph
- Need collaborative efforts to achieve the full potential
- Standardization may help remove fragmentation in DL frameworks

Funding Acknowledgments

Funding Support by

Equipment Support by

Acknowledgments to all the Heroes (Past/Current Students and Staffs)

Current Students (Graduate)

- N. Alnaasan (Ph.D.)
- Q. Anthony (Ph.D.)
- C.-C. Chun (Ph.D.)
- A. Jain (Ph.D.)
- K. S. Khorassani (Ph.D.)
- P. Kousha (Ph.D.)
- B. Michalowicz (Ph.D.)
- B. Ramesh (Ph.D.)
- N. Sarkauskas (Ph.D.)
- K. K. Suresh (Ph.D.)
- A. H. Tu (Ph.D.)
- S. Xu (Ph.D.)
- Q. Zhou (Ph.D.)

Current Research Scientists

- A. Shafi
- H. Subramoni

Current Software Engineers

- N. Shineman

Current Research Specialist

- J. Smith

Past Students

- A. Awan (Ph.D.)
- A. Augustine (M.S.)
- P. Balaji (Ph.D.)
- M. Bayatpour (Ph.D.)
- R. Biswas (M.S.)
- S. Bhagvat (M.S.)
- A. Bhat (M.S.)
- D. Buntinas (Ph.D.)
- L. Chai (Ph.D.)
- B. Chandrasekharan (M.S.)
- S. Chakraborthy (Ph.D.)
- N. Dandapanthula (M.S.)
- V. Dhanraj (M.S.)
- C.-H. Chu (Ph.D.)
- T. Gangadharappa (M.S.)
- K. Gopalakrishnan (M.S.)
- J. Hashmi (Ph.D.)
- W. Huang (Ph.D.)
- W. Jiang (M.S.)
- J. Jose (Ph.D.)
- M. Kedia (M.S.)
- S. Kini (M.S.)
- M. Koop (Ph.D.)
- K. Kulkarni (M.S.)
- R. Kumar (M.S.)
- S. Krishnamoorthy (M.S.)
- K. Kandalla (Ph.D.)
- M. Li (Ph.D.)
- P. Lai (M.S.)
- J. Liu (Ph.D.)
- M. Luo (Ph.D.)
- A. Mamidala (Ph.D.)
- G. Marsh (M.S.)
- V. Meshram (M.S.)
- A. Moody (M.S.)
- S. Naravula (Ph.D.)
- R. Noronha (Ph.D.)
- X. Ouyang (Ph.D.)
- S. Pai (M.S.)
- S. Potluri (Ph.D.)
- K. Raj (M.S.)
- R. Rajachandrasekar (Ph.D.)
- D. Shankar (Ph.D.)
- G. Santhanaraman (Ph.D.)
- N. Sarkauskas (B.S.)
- N. Senthil Kumar (M.S.)
- A. Singh (Ph.D.)
- J. Sridhar (M.S.)
- S. Srivastava (M.S.)
- S. Sur (Ph.D.)
- H. Subramoni (Ph.D.)
- K. Vaidyanathan (Ph.D.)
- A. Vishnu (Ph.D.)
- J. Wu (Ph.D.)
- W. Yu (Ph.D.)
- J. Zhang (Ph.D.)

Past Research Scientists

- K. Hamidouche
- S. Sur
- X. Lu

Past Senior Research Associate

- J. Hashmi

Past Programmers

- A. Reifsteck
- D. Bureddy
- J. Perkins

Past Research Specialist

- M. Arnold

Past Post-Docs

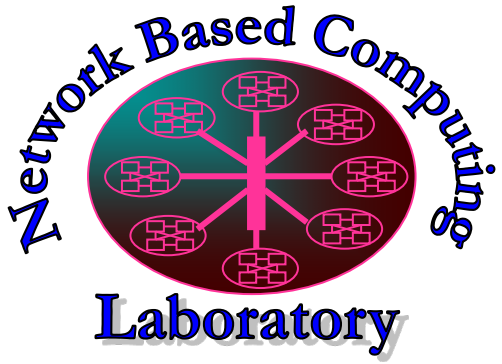
- D. Banerjee
- X. Besseron
- M. S. Ghazimeersaeed
- H.-W. Jin
- J. Lin
- M. Luo
- E. Mancini
- K. Manian
- S. Marcarelli
- A. Ruhela
- J. Vienne
- H. Wang

Thank You!

panda@cse.ohio-state.edu, subramon@cse.ohio-state.edu, jain.575@osu.edu

Follow us on

https://twitter.com/mvapich



Network-Based Computing Laboratory
http://nowlab.cse.ohio-state.edu/

MVAPICH
MPI, PGAS and Hybrid MPI+PGAS Library

The MVAPICH2 Project
http://mvapich.cse.ohio-state.edu/

The High-Performance Deep Learning Project
http://hidl.cse.ohio-state.edu/