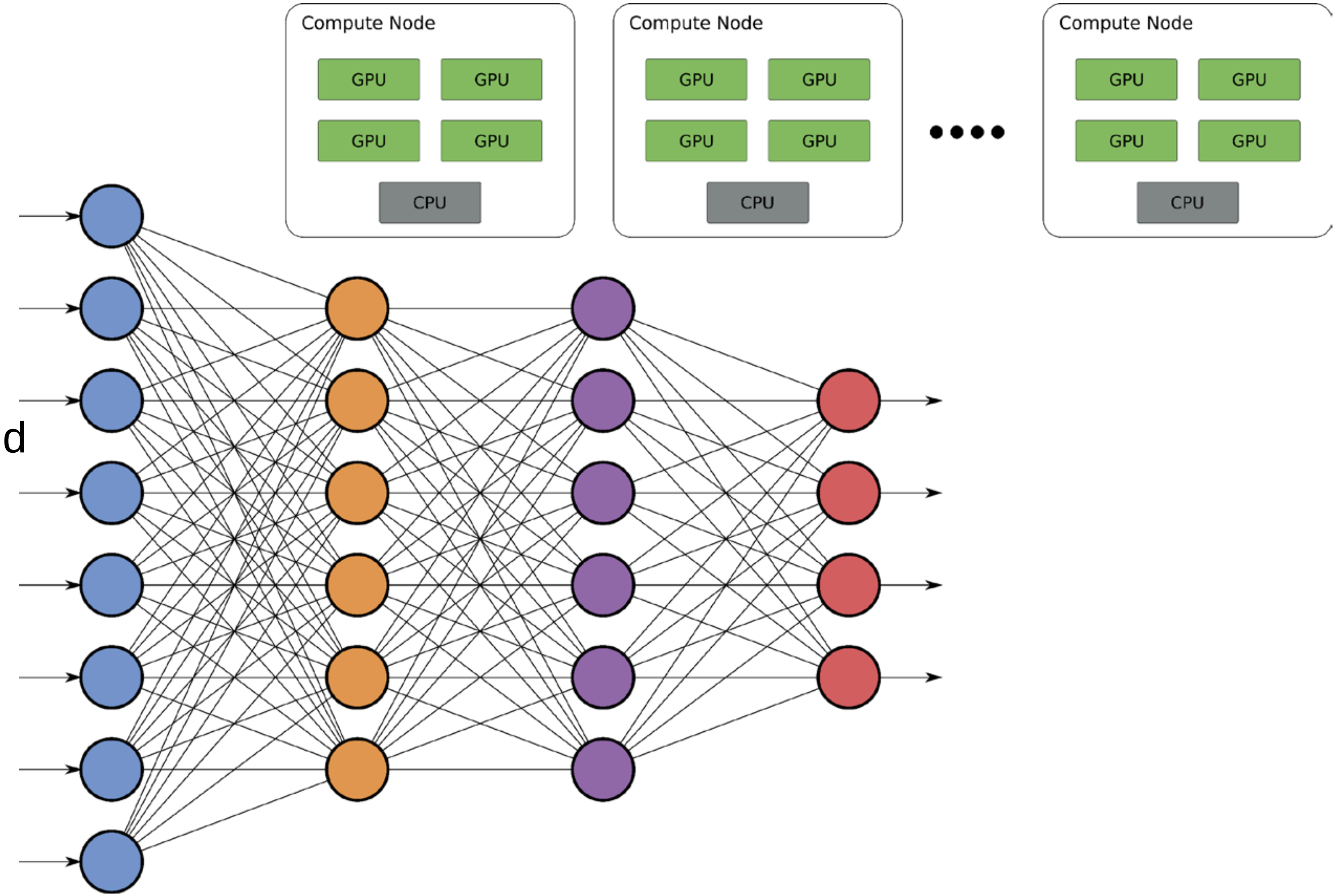
Distributed Deep Learning

Feroz Zahid, Michael Riegler, and Tor Skeie

Simula Research Laboratory and University of Oslo



simula

UiO : University of Oslo

Myself

- Full professor, Department of informatics, University of Oslo, Norway
- Adjunct research scientist, Simula Research Laboratory, Norway
- Fabriscale Technologies, CTO and co-founder
- Areas of expertise
  - HPC networking and management/middleware systems
  - Cloud computing and virtualization
  - Industrial Ethernet and wireless networking

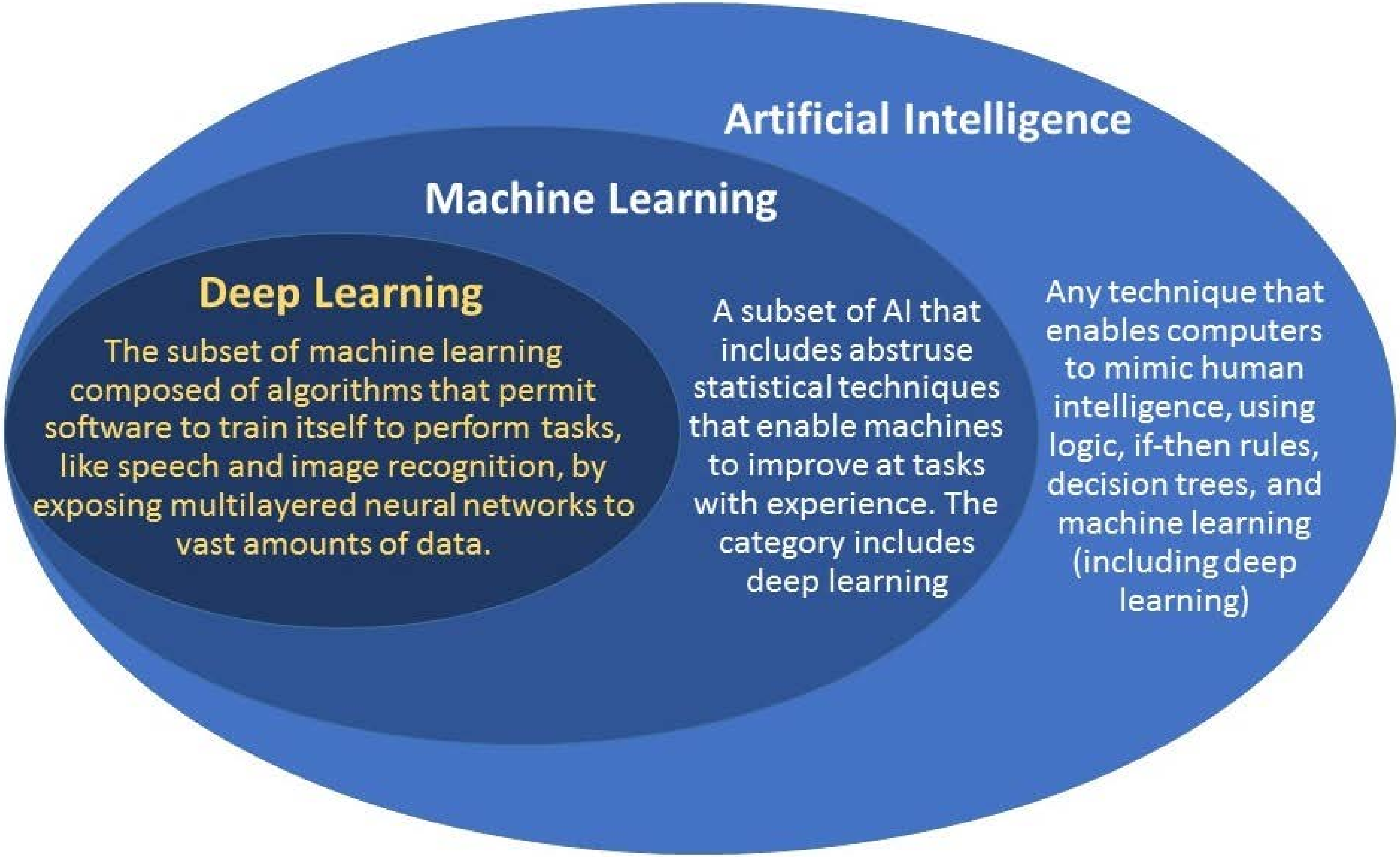
This presentation will introduce distributed deep learning, walk through prominent techniques, and identify existing challenges and future directions

Introduction and Motivation

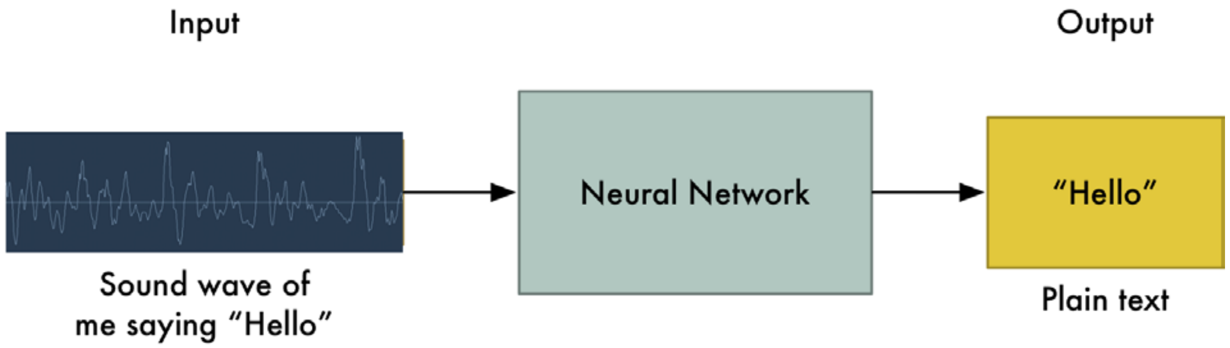
Existing Techniques and Toolsets

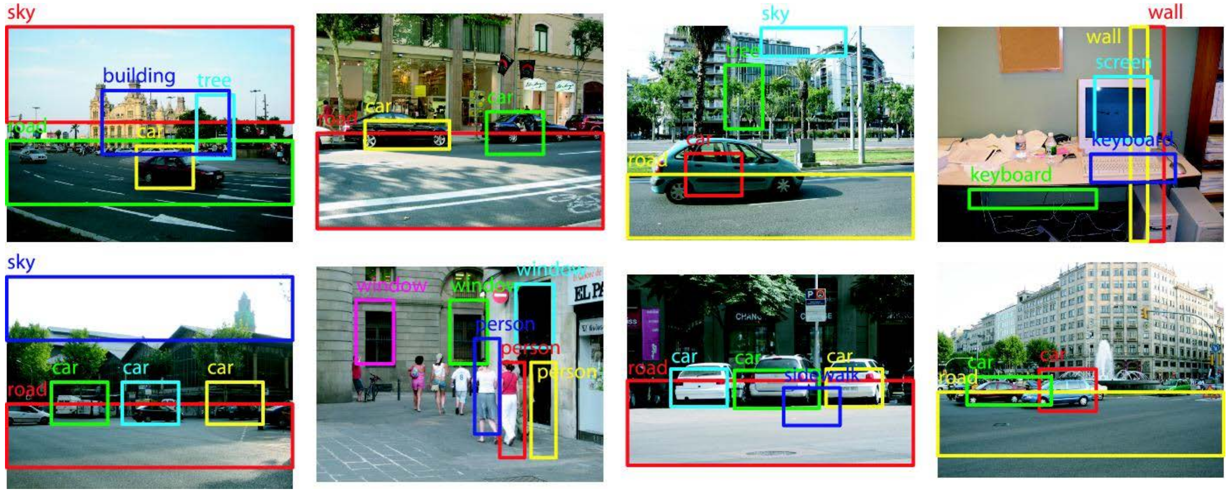
Future Directions

AI, machine learning, deep learning…



What you can do…



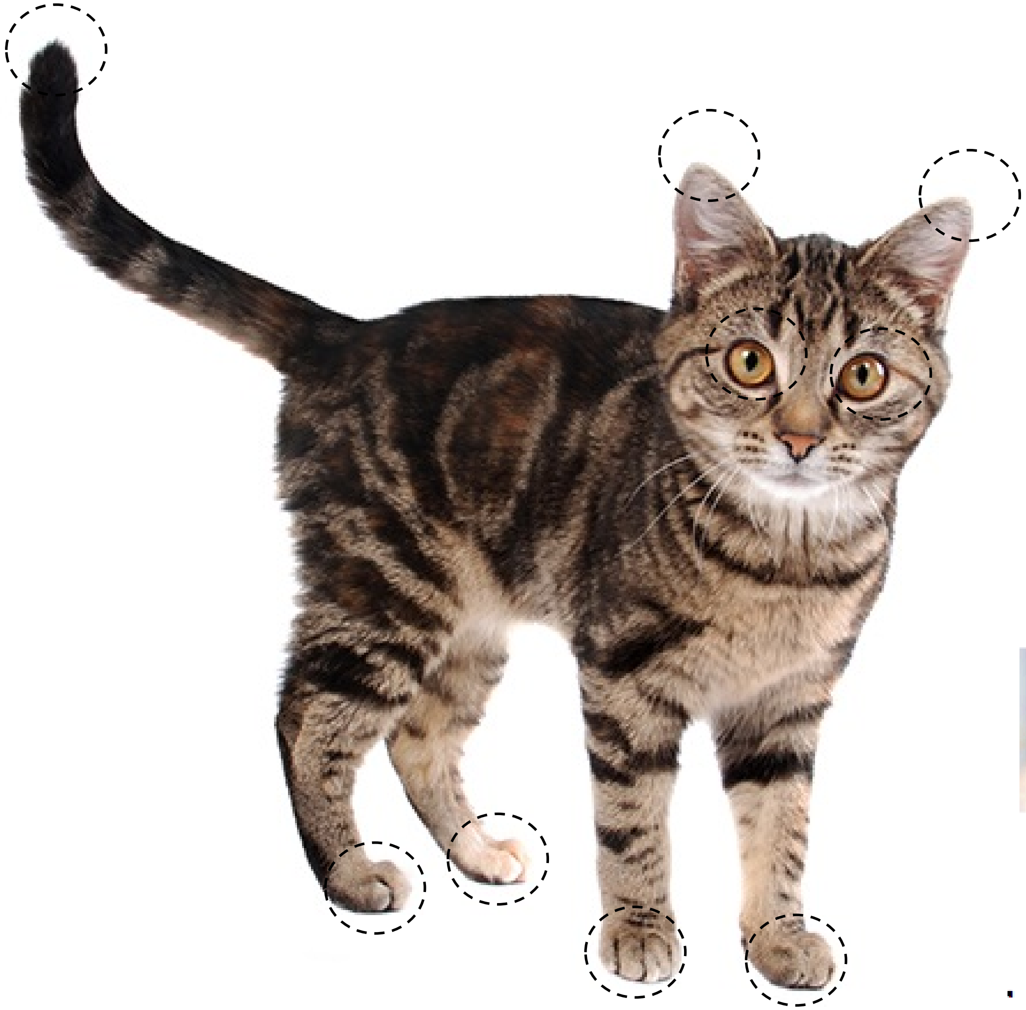


Deep learning in Computer Vision, Fei-Fei Li et al., 2016

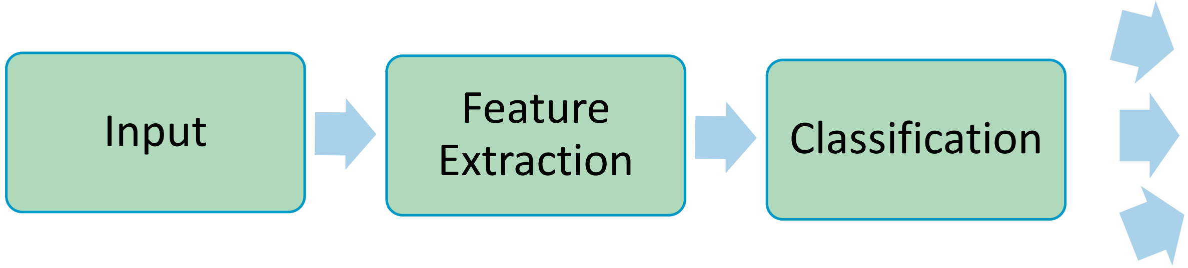
- Image classification
- Text processing
- Traffic classification
- Predictions of characteristics
- Climate prediction
- Health care
- many more….

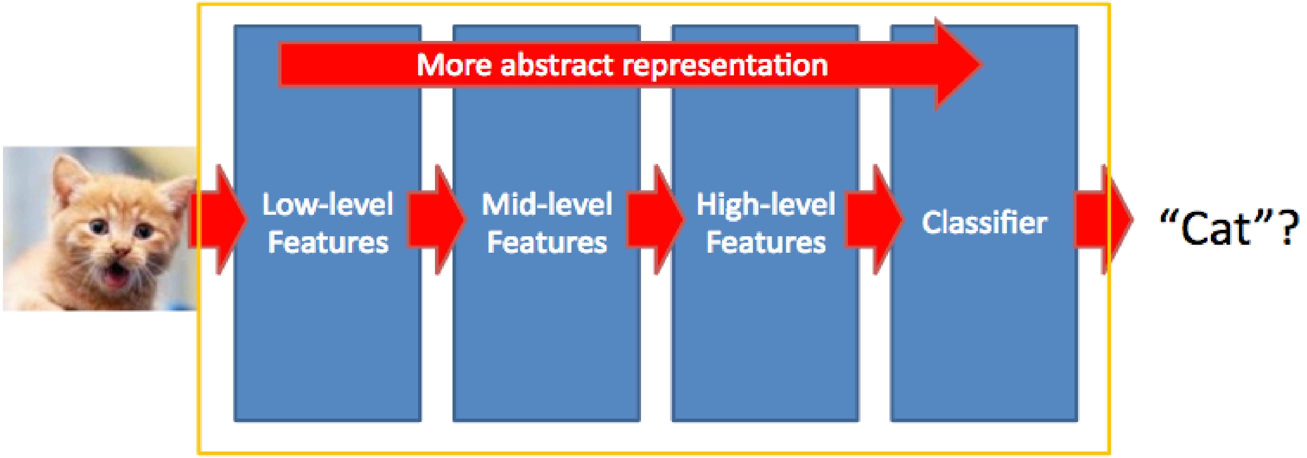
Classification refers to identifying the category to which a new observation belong based on previous examples

Classification refers to identifying the category to which a new observation belong based on previous examples



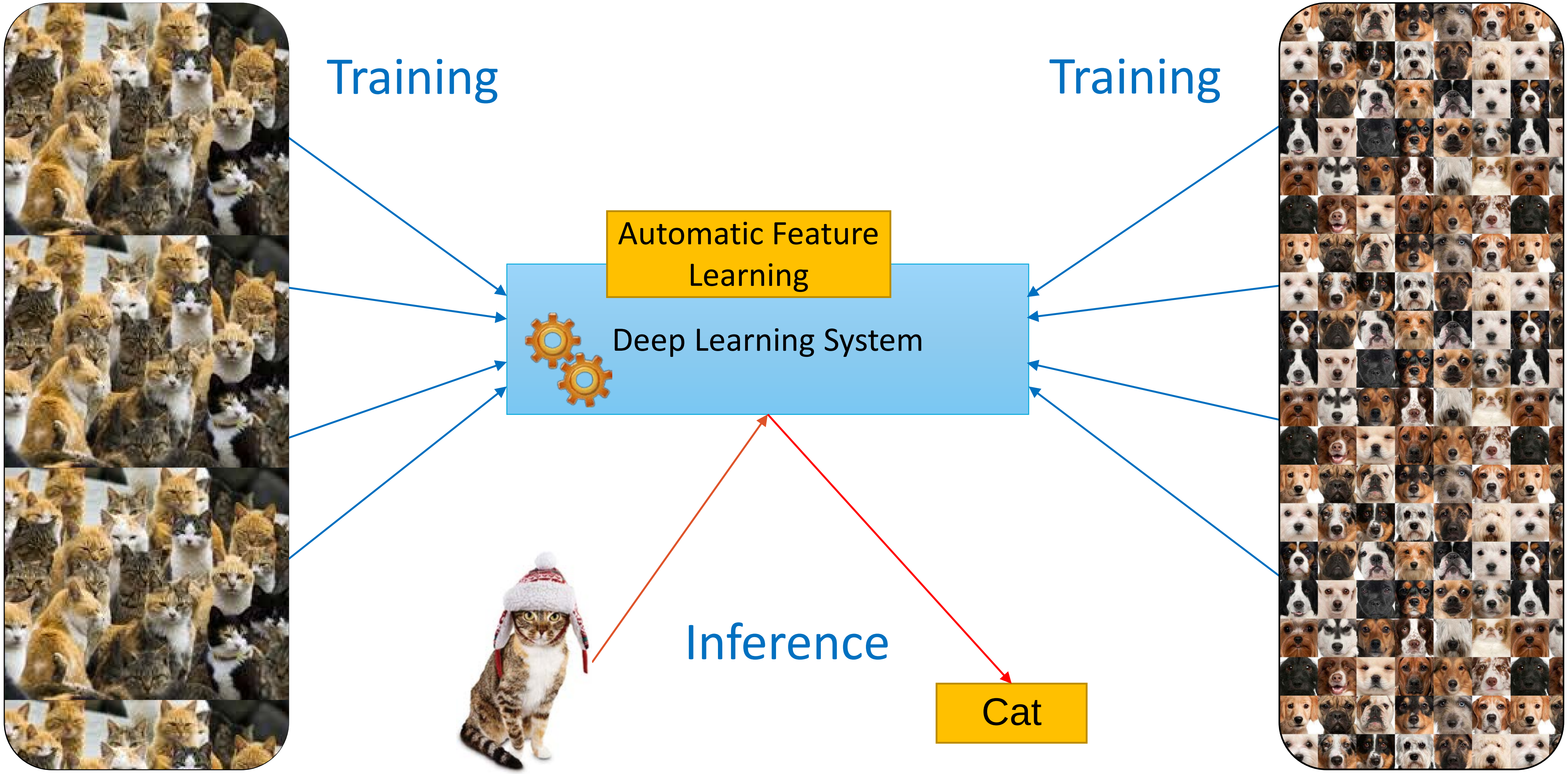
Typical ML Flow





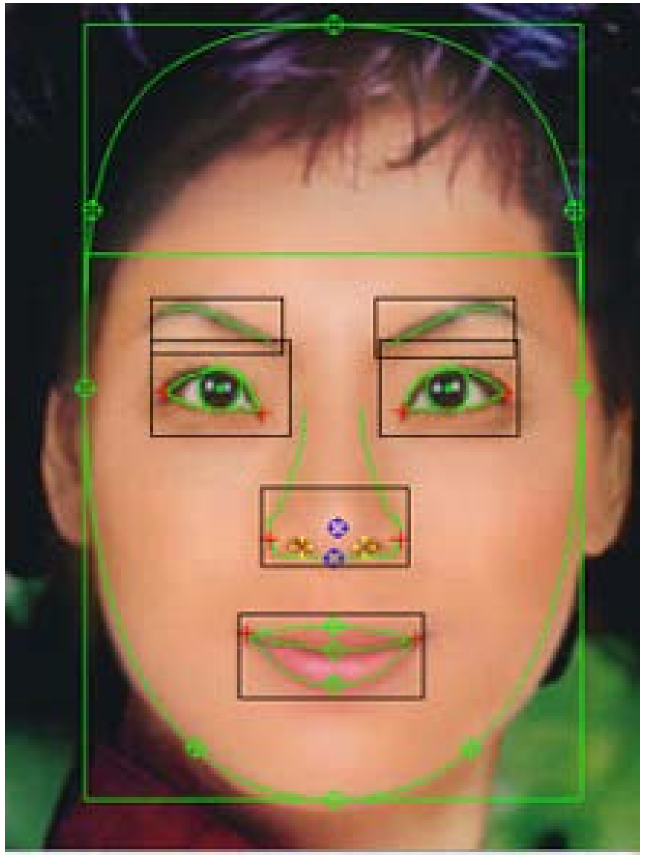
Deep Learning: train layers of features so that classifier works well.

Deep learning is a machine learning technique in which identification and extraction of features is directly done from the dataset



Feature creation

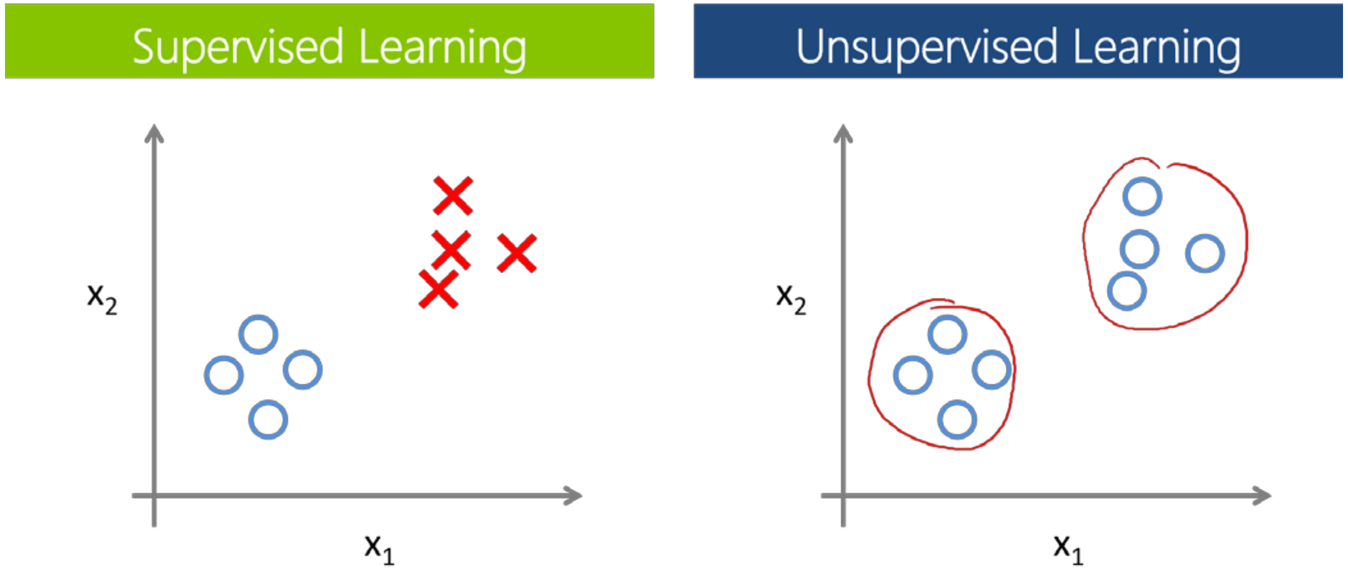
- Deep learning to create automatically features from data
- Features describe the data content in an abstract representation
- For example face features, edges, color distribution, time series, etc.



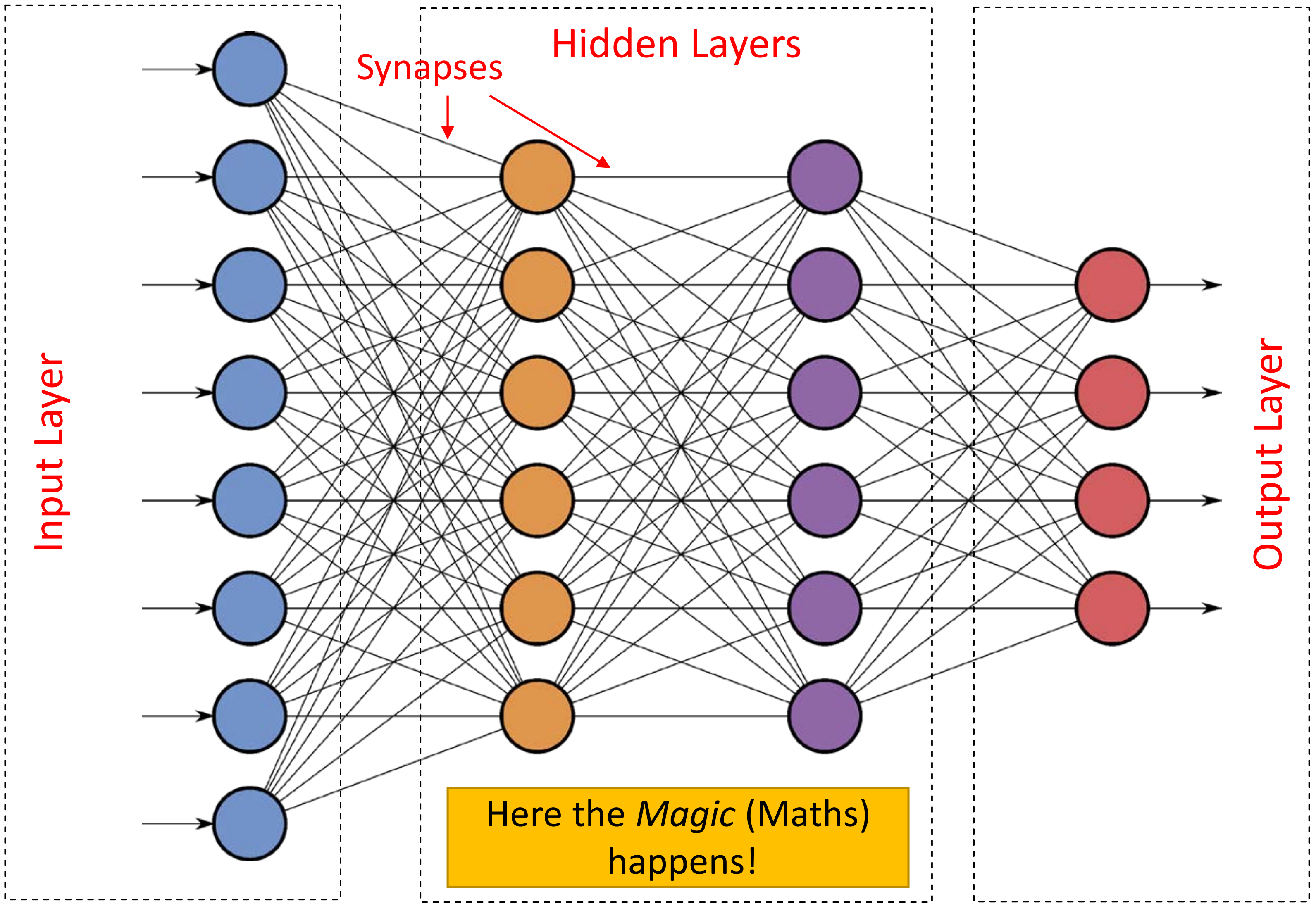
Three pillars of deep learning

- Algorithms
- Data
- Computational power

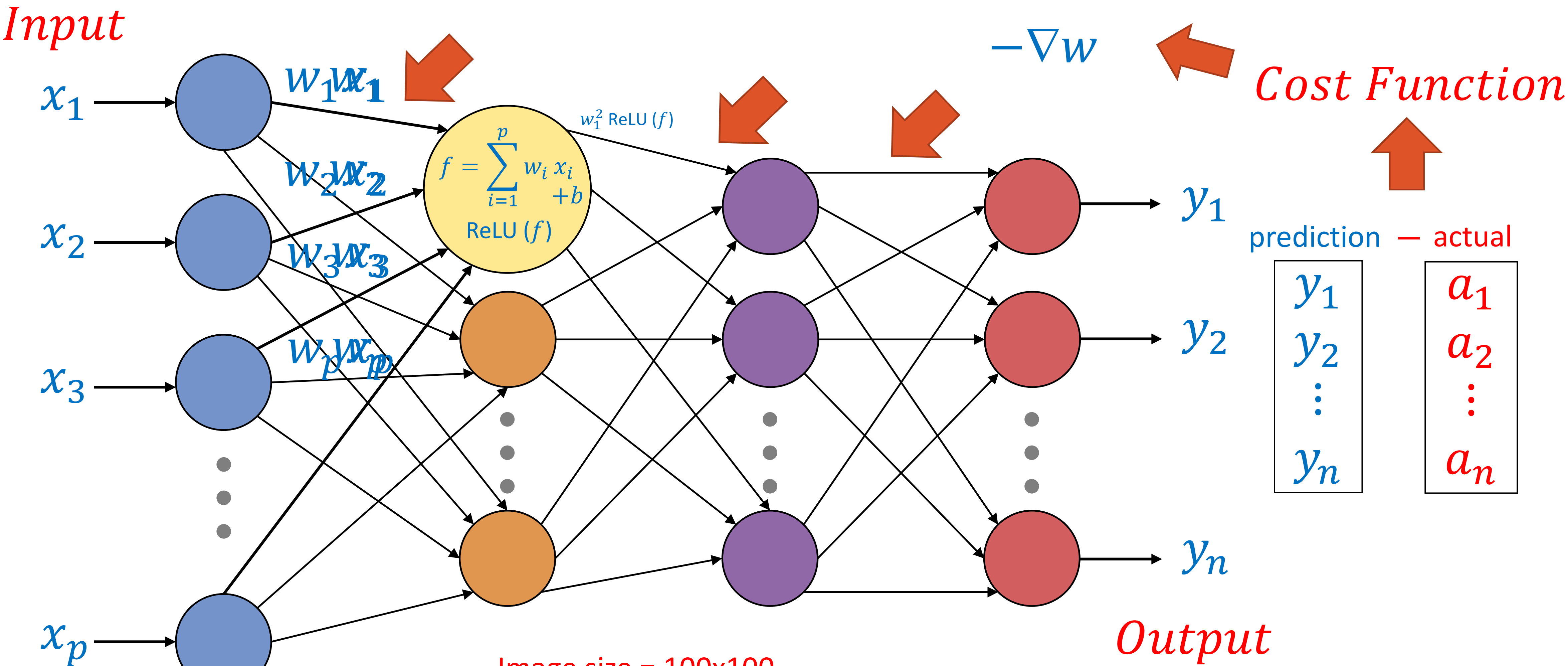
- Supervised
  - ➢ Labelled training data
  - ➢ Algorithm learns from data
- Unsupervised
  - ➢ No training data
  - ➢ Algorithm learns by itself



Deep learning is predominately based on artificial neural networks (ANNs). ANNs progressively *learn* using *neurons* arranged in many layers



In training neural networks, prediction errors are *backpropagated* for gradually tuning weights and biases to values that predict better



No. of Neurons

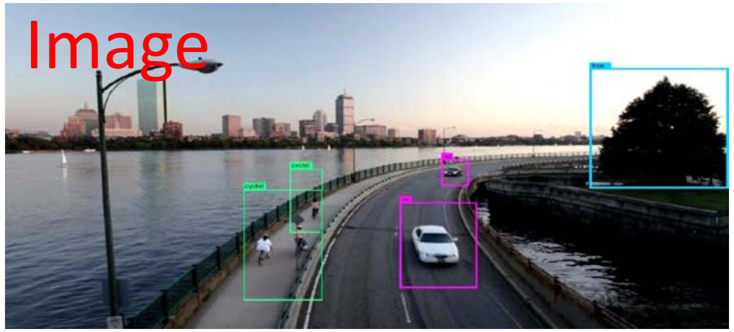
p

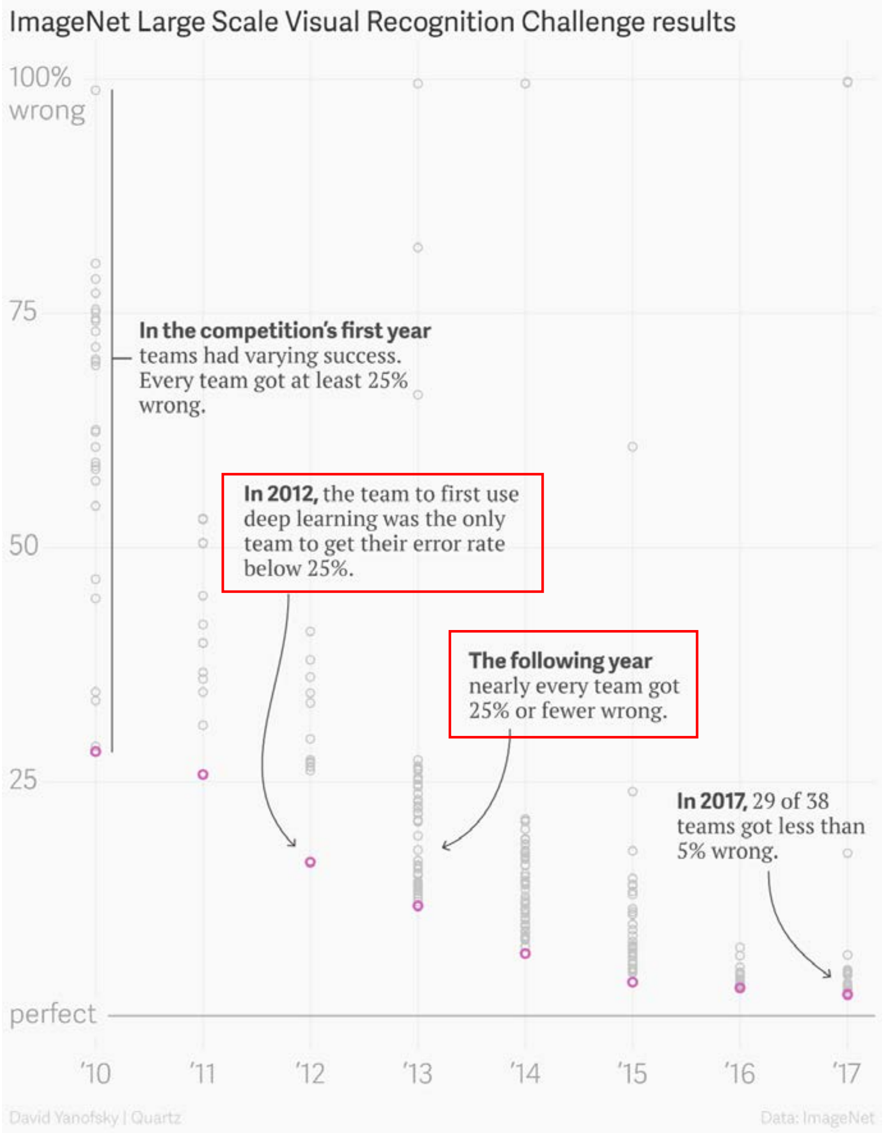
Image size = 100x100
Layers = 16x16, Output = 2
16322 parameters!

n

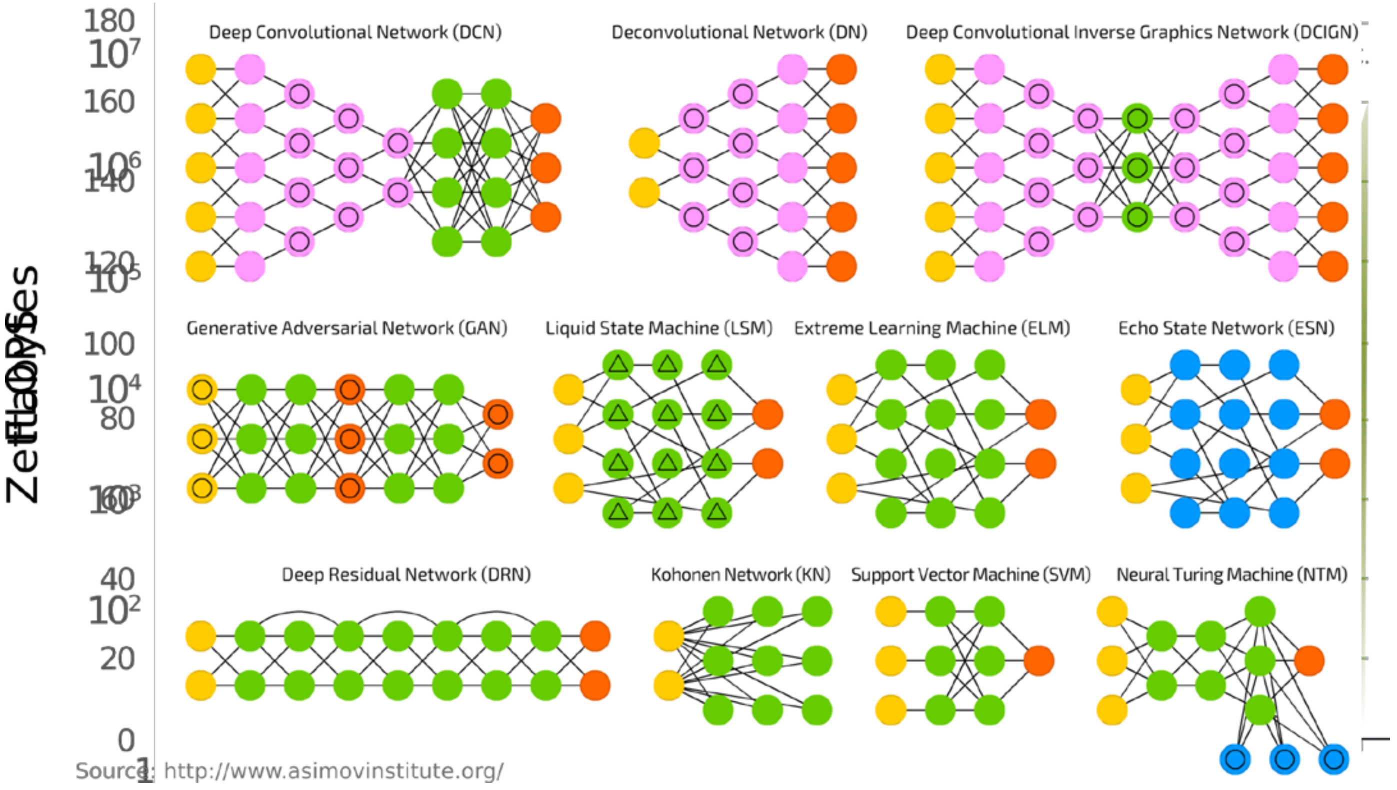
Deep learning has greatly improved prediction accuracies in various fields, and has become a backbone of artificial intelligence

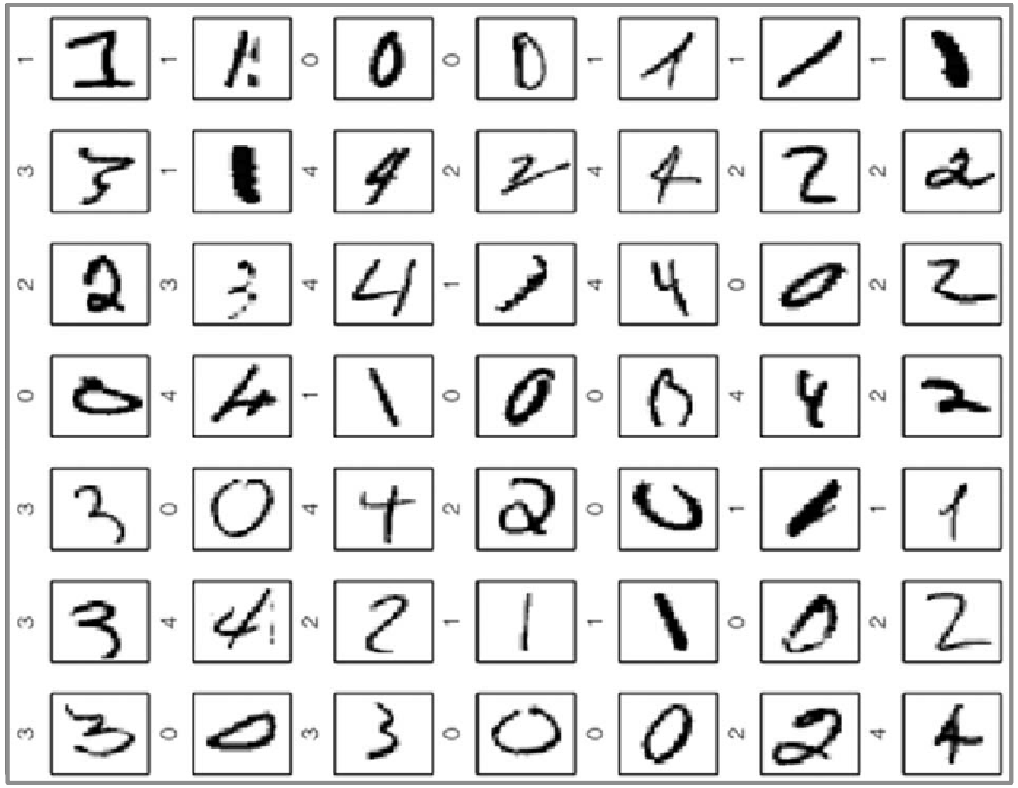
- Improving prediction accuracies
  - Important in real-world applications such as disease identification, self-driving cars
- Fueling new applications
  - Science, Finance, Healthcare, Automotive





Availability of large amount of data, improved processing capabilities, and advances in neural networks contributed to the deep learning success





ImageNet is a large database of (URLs) of about 14 million images with more than 20000 different classes.

MNIST contains >60k hand-written digits with labels.

- Advancement in neural networks
  - Convolutional networks, Recurrent Neural Networks, Long Term-short memory (LSTM)
  - ResNet50, Xception, VGG19, InceptionV3, InceptionResNetV2
- Development of hardware capabilities made it possible to execute previously infeasible computational tasks
  - Tesla V100, capable of 15 TFLOPS of single-precision operations
- Availability of large dataset
  - Digital universe expected to grow to 163 ZB in 2025
  - ImageNet, MNIST

Large, complex models and training using large datasets improves classification accuracy of deep learning algorithms

- State-of-the-art accuracies are achieved using very large models
  - Millions / Billions of parameters, Large number of layers and depth
    - Neural network of Google Brain[1] has 137 billion parameters!

Size and depth of the model, Top-1 accuracy, ImageNet dataset[2]

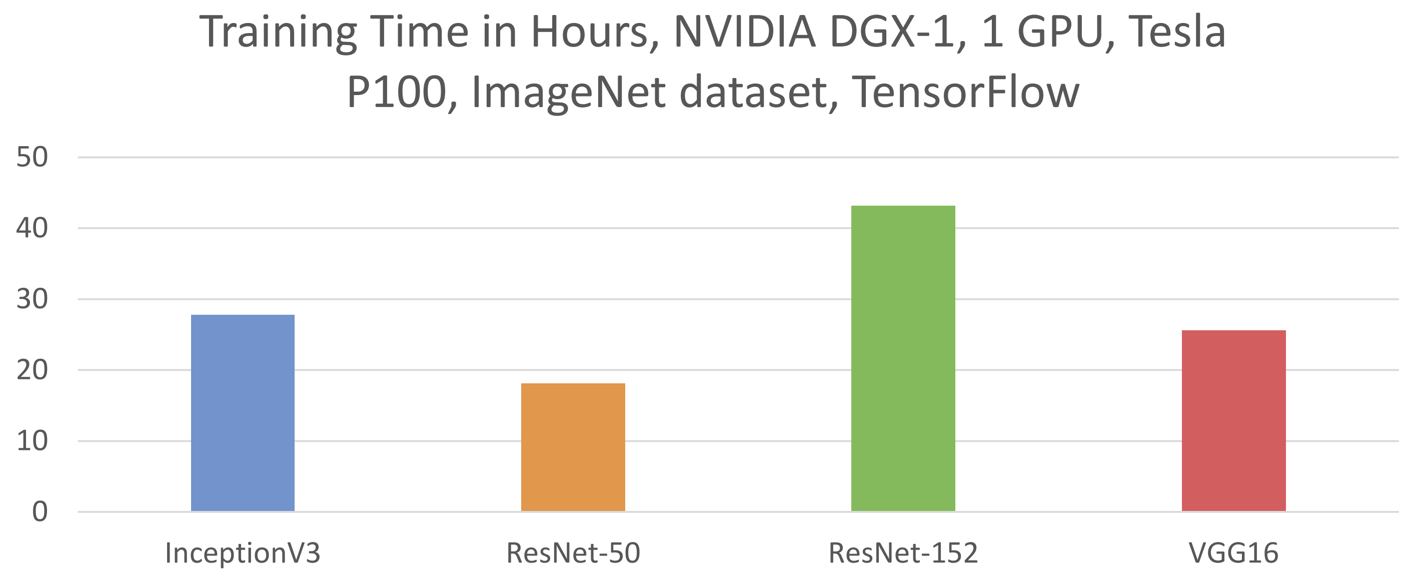
Model	Top-1 Accuracy %	No. of Parameters	Network Depth
MobileNet	66.5	4,253,964	88
ResNet50	75.9	25,636,712	168
Xception	79.0	22,910,480	126
VGG16	71.5	138,357,544	23
VGG19	72.7	143,667,240	26
InceptionV3	78.8	23,851,784	159
InceptionResNetV2	80.4	55,873,736	572

Note: Recently, Google's NASNet[3] reached accuracy of 82.7% with 22.6 Million parameters.

[1]Shazeer, Noam, et al. "Outrageously large neural networks: The sparsely-gated mixture-of-experts layer." arXiv preprint arXiv:1701.06538, 2017.
[2]https://keras.io/applications/ [3]Zoph, Barret, et al. "Learning transferable architectures for scalable image recognition." arXiv preprint arXiv:1707.07012, 2017.

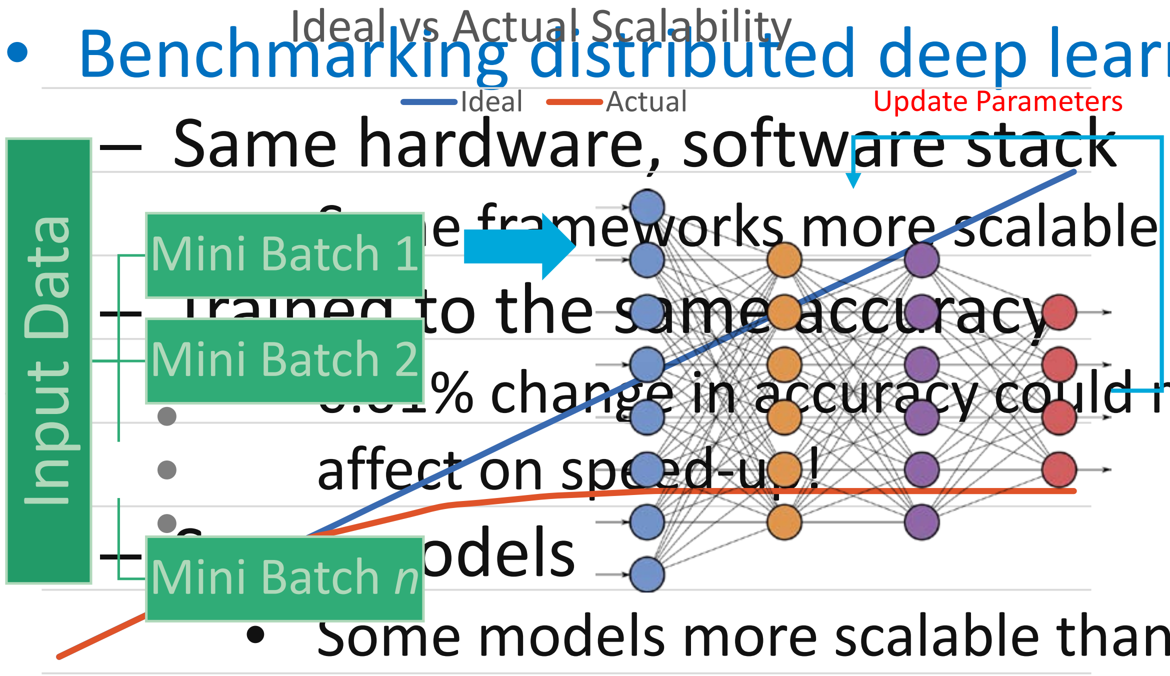
Efficient distributed deep learning across a large number of nodes is critical for modern deep learning applications

- It takes hours, and even days to train today's models on a single node
  - The models will only become more bigger and complex
    - The amount of training data will only increase
  - Increase in GPU processing power cannot cope up
    - Billions of parameters will not fit in GPU memory



Data: https://www.tensorflow.org/performance/benchmarks

Distributed deep learning should be scalable, should not affect the accuracy of the classification, and should fully utilize available resources



- Benchmarking distributed deep learning systems
  - Same hardware, software stack
    - Some frameworks more scalable than others
  - Trained to the same accuracy
    - 0.01% change in accuracy could mean a substantial affect on speed-up!
  - Same models
    - Some models more scalable than others
      - Due to computation / communication ratio

- Accuracy
  - Can be affected by the batch size
    - A larger batch size is usually proved beneficial for distributed deep learning
    - Reporting affected by n nodes with 1x GPU vs $\frac{n}{8}$ nodes with 8x GPUs

- Scaling Efficiency and Communication Overhead
  - Ratio between training time of one iteration on 1 nodes to total training time when distributed over n nodes
  - Number of epochs required for convergence
  - Learning rate adaptation, trade-off between runtime and accuracy

Goyal, Priya, et al. "Accurate, Large Minibatch SGD: Training ImageNet in 1 Hour." arXiv preprint arXiv:1706.02677, 2017.

This presentation will introduce distributed deep learning, walk through prominent techniques, and identify existing challenges and future directions

Introduction and Motivation

Existing Techniques and Toolsets

Future Directions

Scaling efficiency of distributed deep learning can be improved at different levels, and a co-design approach is needed

Apps

- Over multiple GPUs on a single node
- Over mu... nodes

Distribute Across Nodes

A

- Computationally light models
- Models that scale better
- Models

Design Better Models

B

- Hardware with more processing capabilities
- Interc... with r...

Improve Hardw...

C

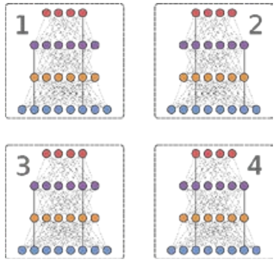
Deep Learning Frameworks: TensorFlow, Microsoft CNTK, theano, Caffe, Keras, torch

Deep Learning Acceleration: NVIDIA cuDNN, MKL-DNN, Single Node Communication, NCCL, Gloo, Multi Node Communication, Open MPI, MVAPICH

Heterogeneous Hardware: CPUs, GPUs, FPGAs, Interconnect Technologies

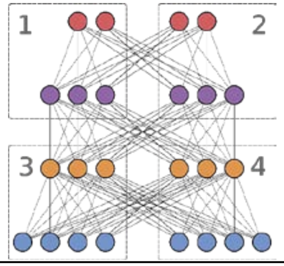
Distribute Across Nodes

There are three parallelization methods employed in distributed implementation of deep learning applications



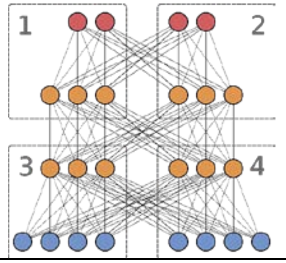
Data Parallelism

- Data partitioned across machines, each machine holds local copy of the model, synchronization required



Model Parallelism

- Neural network partitioned across machines, Parallelizing mathematical operations across machines

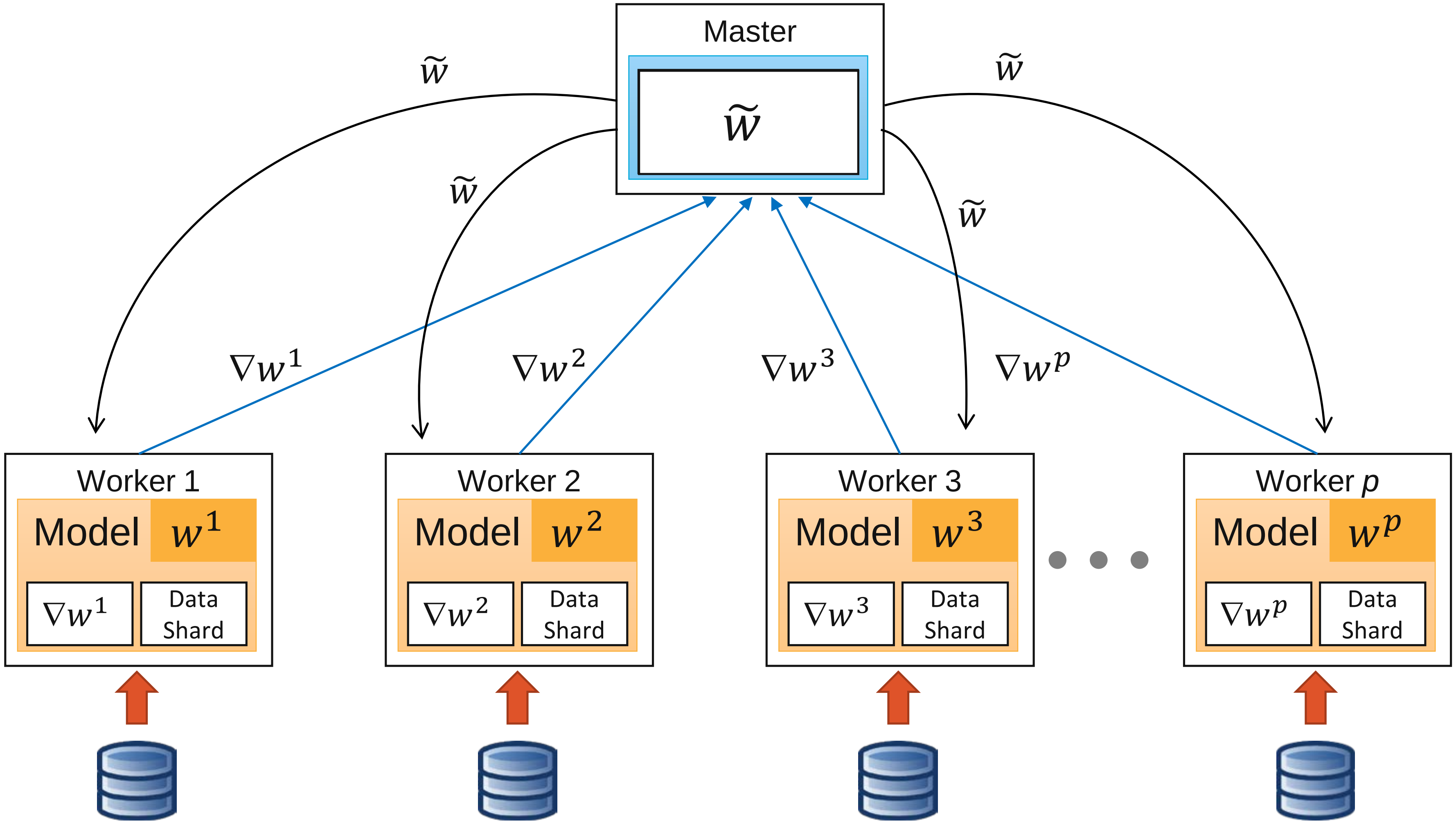


Hybrid Approaches

- Data partitioning for some parts of neural network, model partitioning for correctness on some parts, Automatic selection

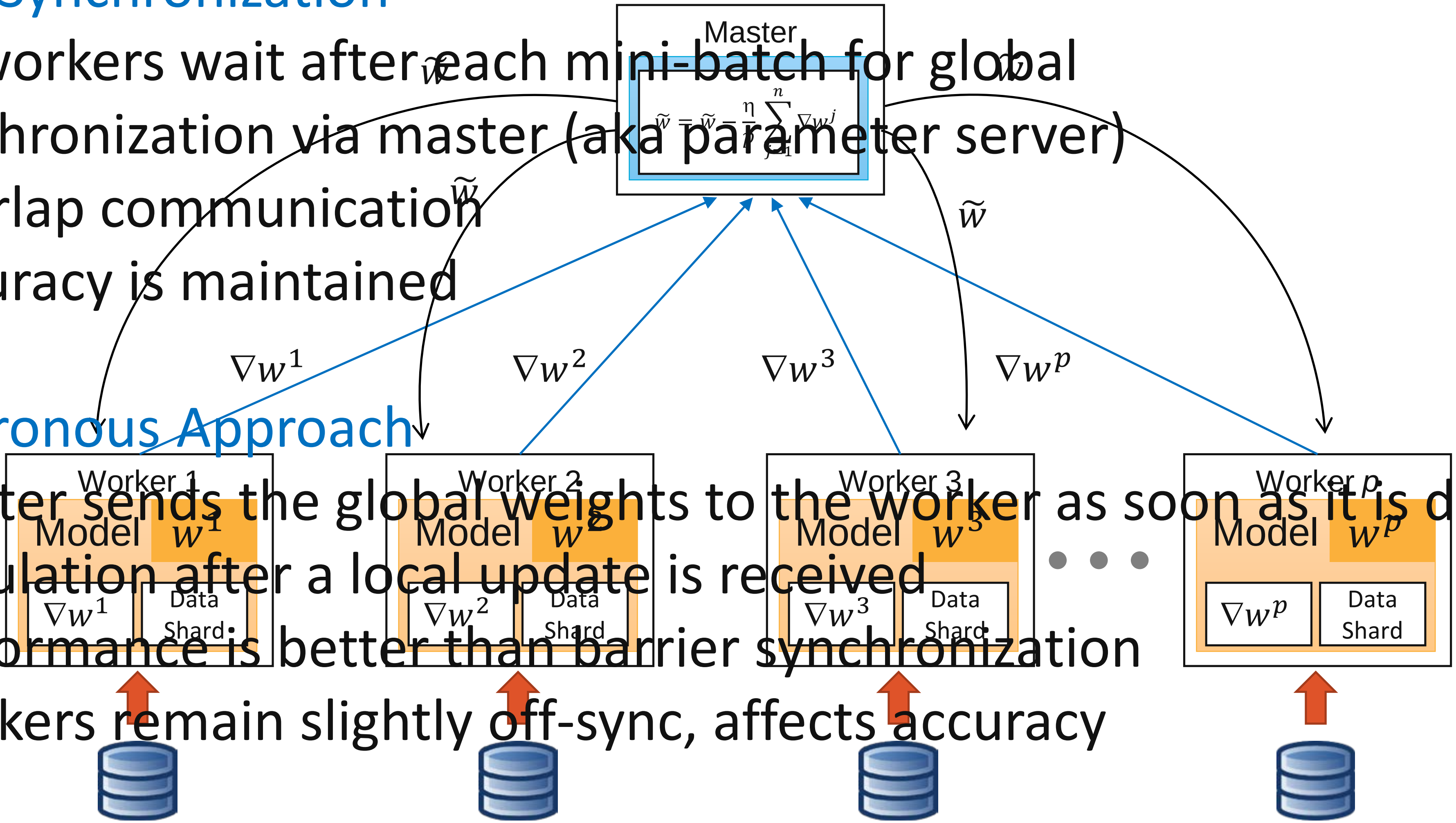
Distribute Across Nodes A

Data Parallelism: In Synchronized data parallelism, the algorithm includes two parts: sum of local gradients and broadcast of global weight to workers



Data Parallelism: Barrier before each update, overlapping communication and computation, or asynchronous approaches are generally employed

- Barrier Synchronization
  - All workers wait after each mini-batch for global synchronization via master (aka parameter server)
  - Overlap communication
  - Accuracy is maintained

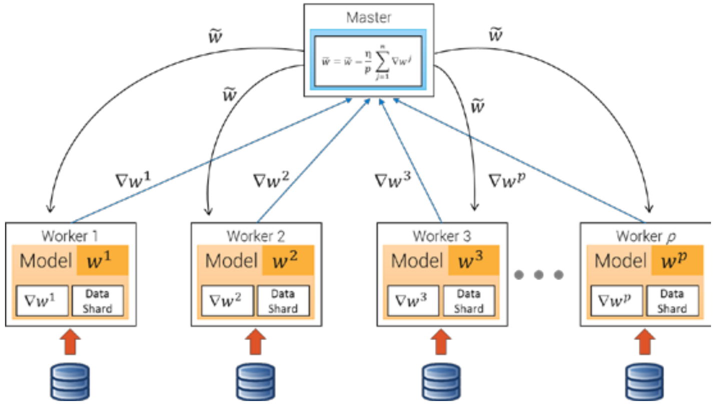


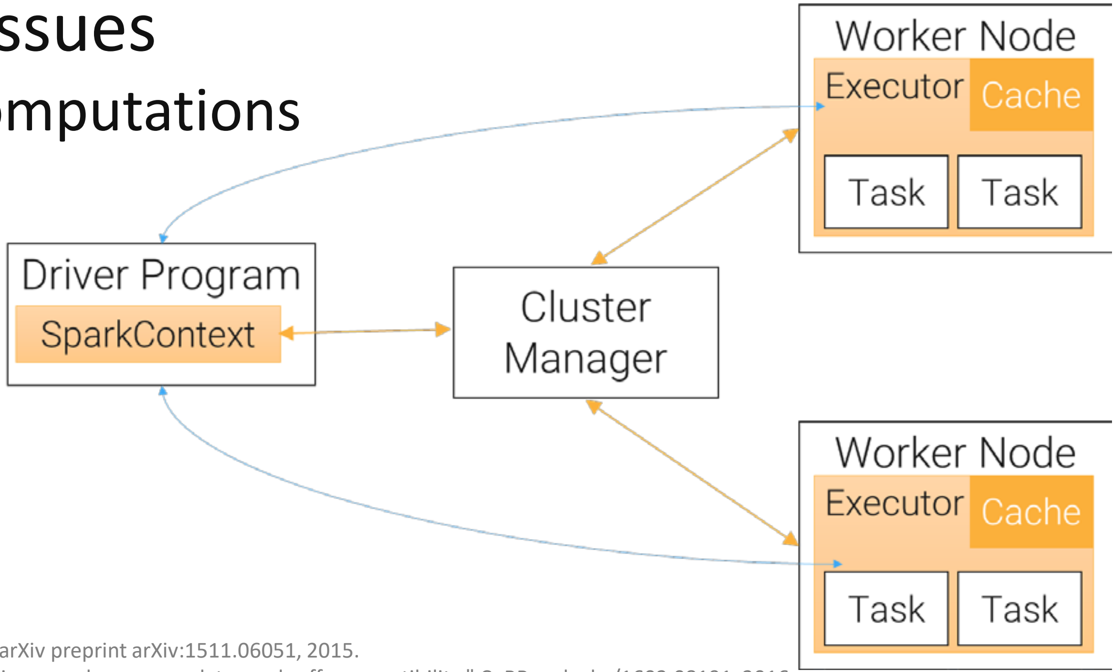
- Asynchronous Approach
  - Master sends the global weights to the worker as soon as it is done calculation after a local update is received
  - Performance is better than barrier synchronization
  - Workers remain slightly off-sync, affects accuracy

Dean, Jeffrey, et al. "Large scale distributed deep networks." Advances in neural information processing systems. 2012.

Data Parallelism: Big data processing frameworks and resource management systems also make good candidates for distributed deep learning

- Spark-based Solutions
  - Integration of deep learning frameworks, e.g. TensorFlow with Spark
    - ClusterManager takes role of parameter server
    - Mesos used as resource manager
  - Performance issues
    - Iterative computations

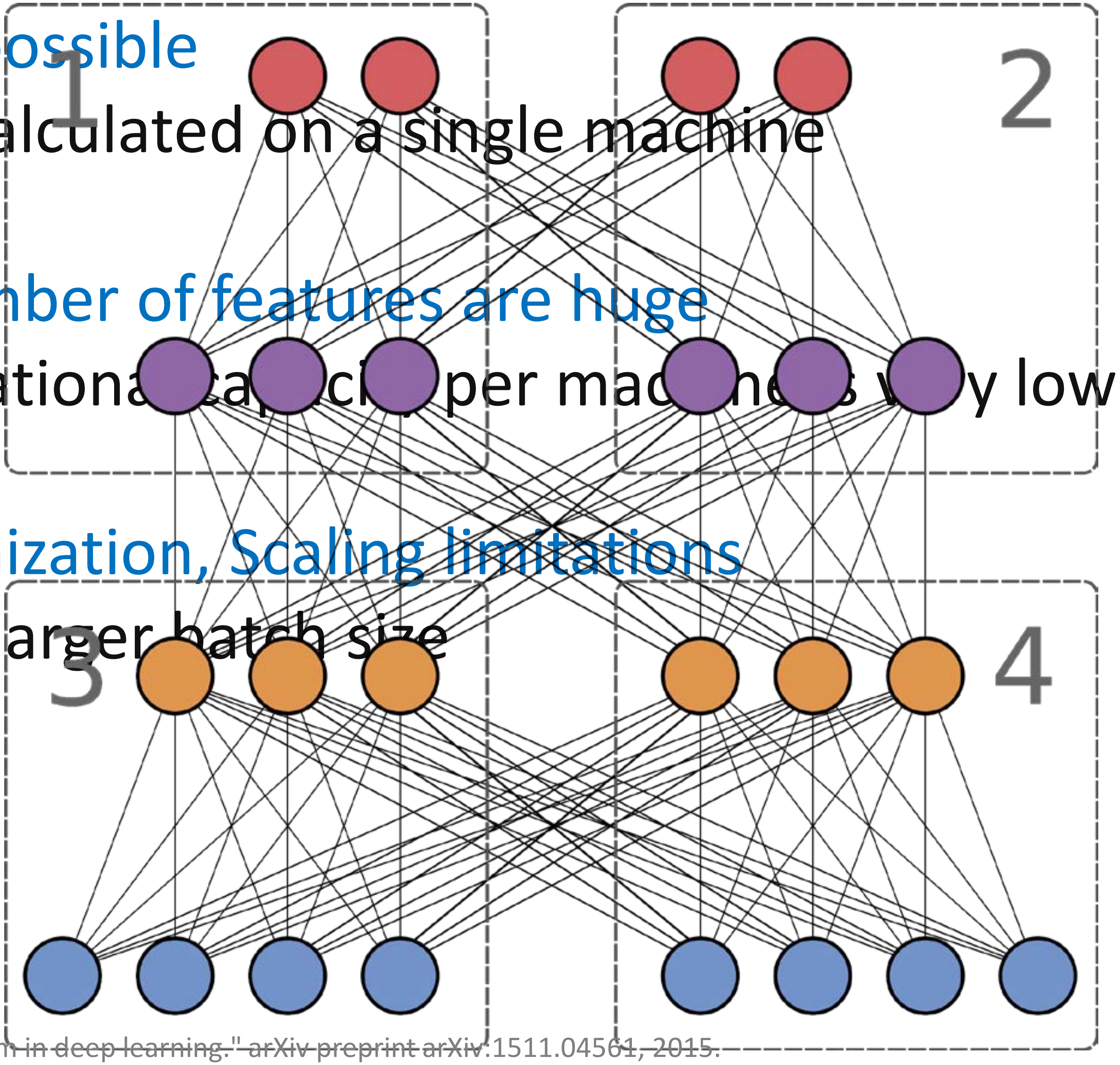




Moritz, Philipp, et al. "Sparknet: Training deep networks in spark." arXiv preprint arXiv:1511.06051, 2015.
Kim, Hanjoo, et al. "Deepspark: Spark-based deep learning supporting asynchronous updates and caffe compatibility." CoRR, vol. abs/1602.08191, 2016.

Model Parallelism: Neural network is divided into several mini-models and distributed across multiple nodes

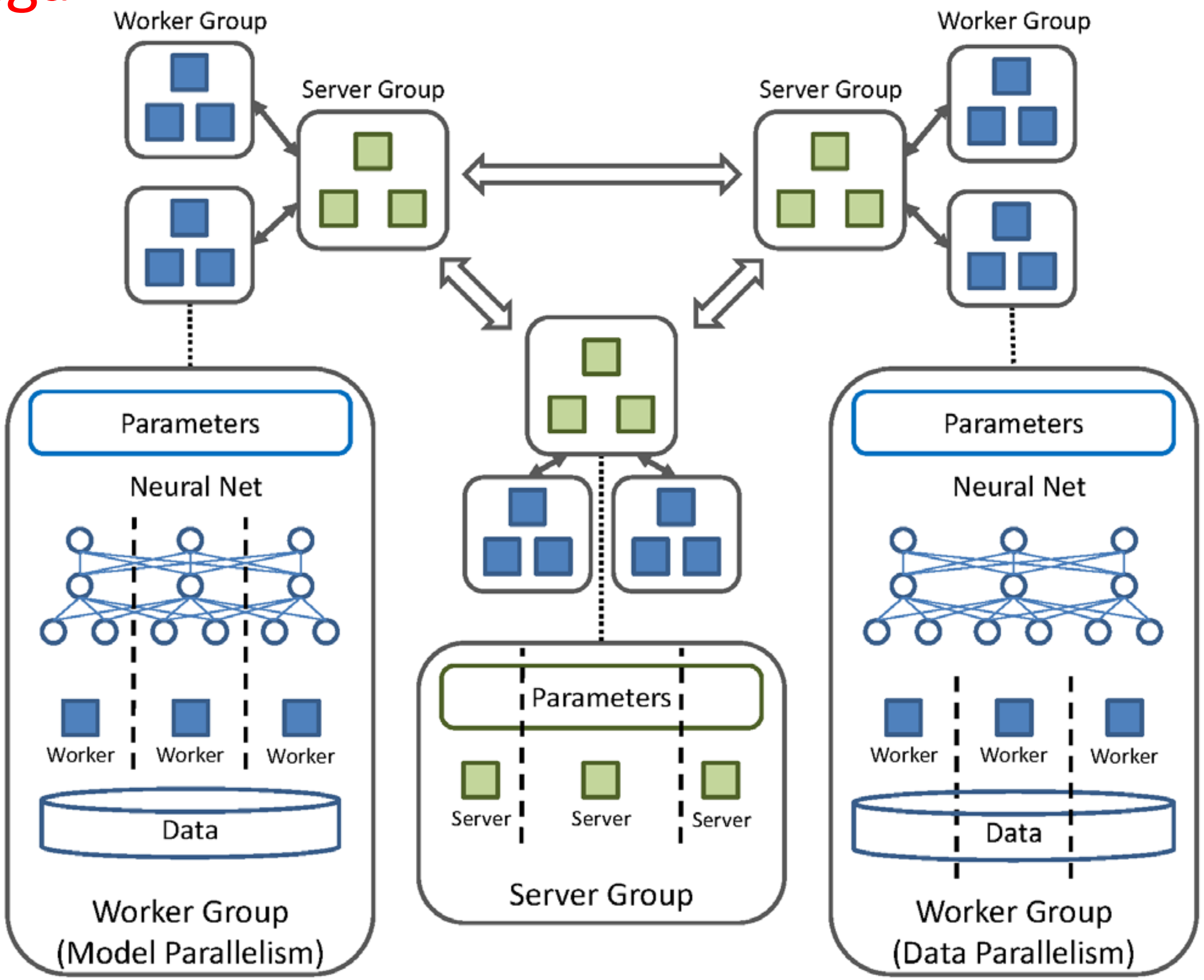
- Highest accuracy possible
  – As if model is calculated on a single machine
- Feasible when number of features are huge
  – Or the computational capacity per machine is very low
- Frequent synchronization, Scaling limitations
  – Problems with larger batch size



Dettmers, Tim. "8-bit approximations for parallelism in deep learning." arXiv preprint arXiv:1511.04561, 2015.

Hybrid Approach: Some parts of the neural network are divided using model parallelism while other parts employ data parallelism

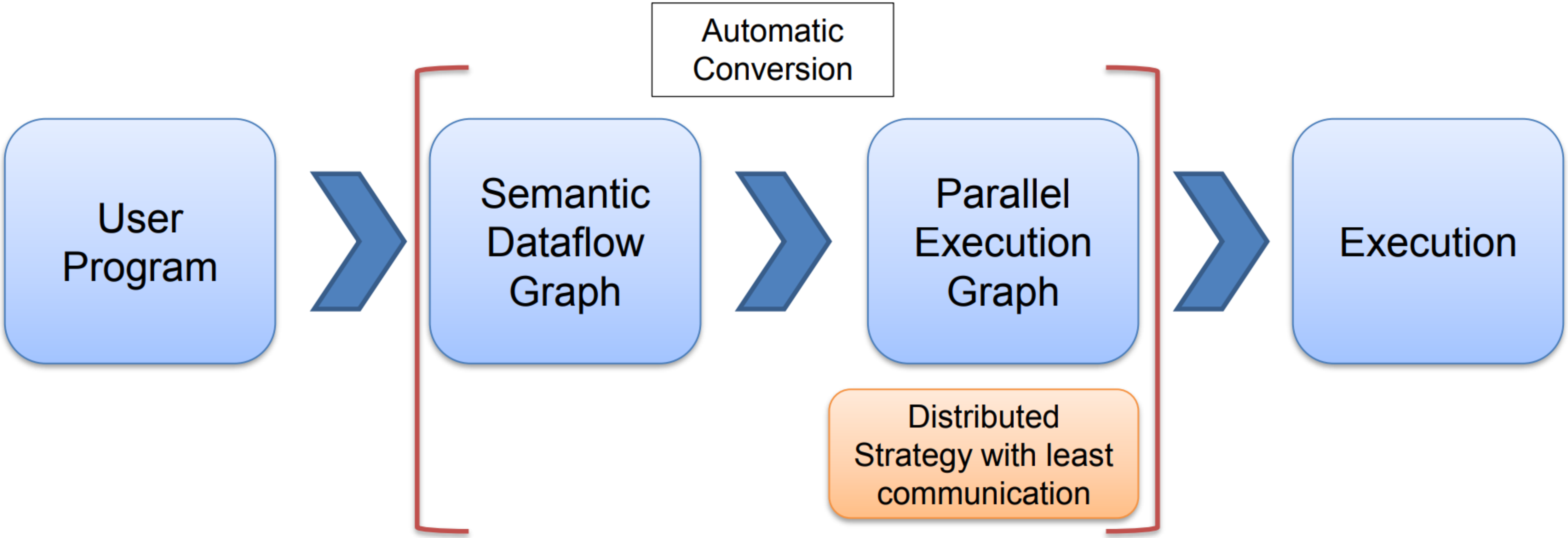
Example: Apache Singa



Source: http://singa.incubator.apache.org/

Hybrid Approach: Automatic selection of the best possible parallelism model has also been proposed in literature

Example: Tofu



Automatic generation of Neural networks using evolutionary algorithm, ORNL.

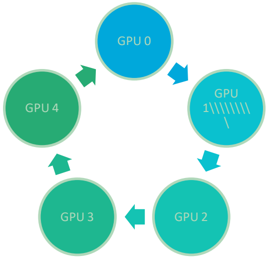
Source: Tofu – Parallelizing Deep Learning Systems with Automatic Tiling, Minjie Wang, 2017

Communication efficiency plays an important role in the scalability of the distributed deep learning algorithms

Communication overhead is the difference between runtime of one iteration when distributed over *n* processing units and runtime of one iteration on a single unit

Interconnect Bandwidth and Latency

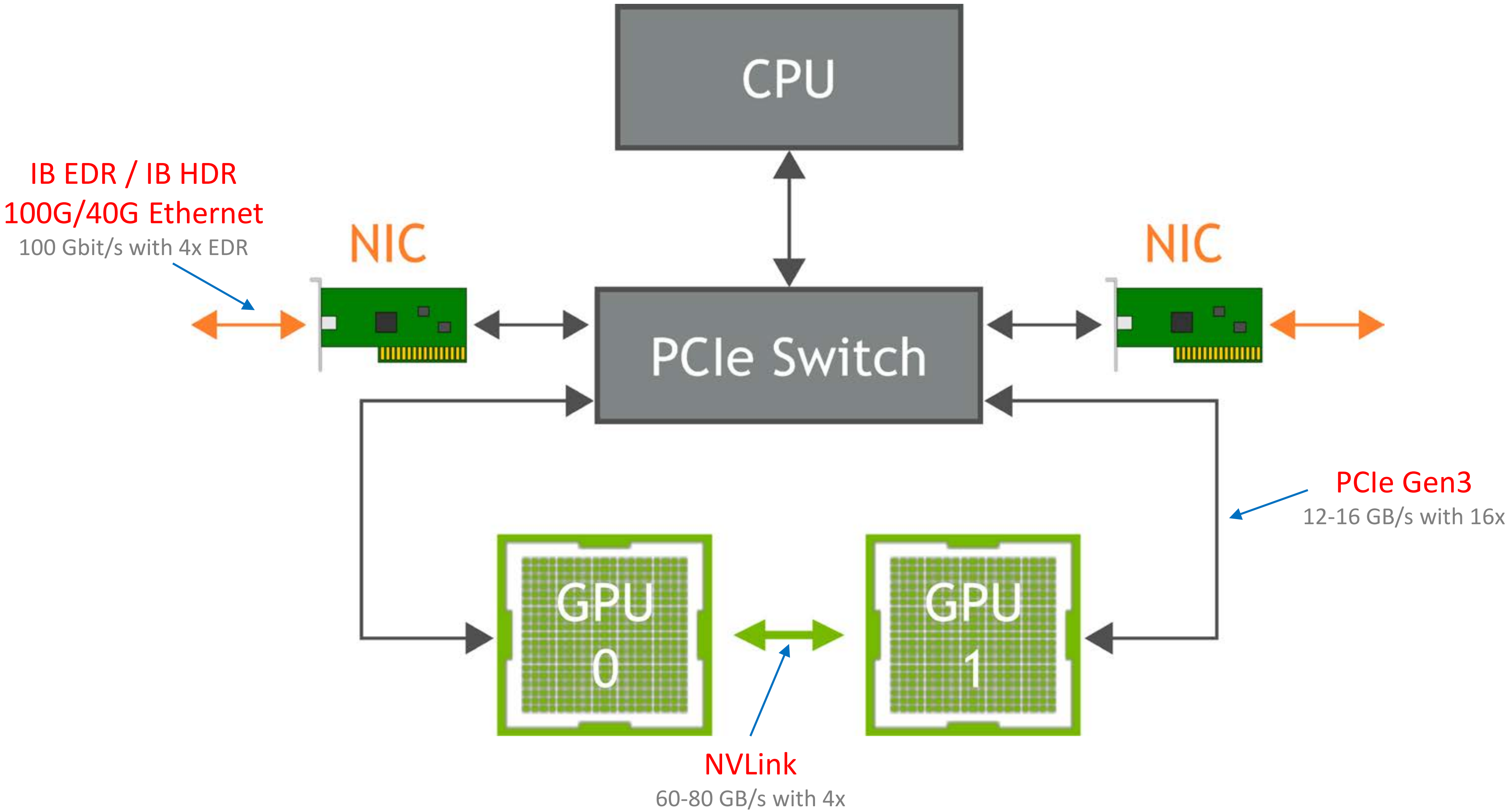
- Inter-node communication, Intra-node communication



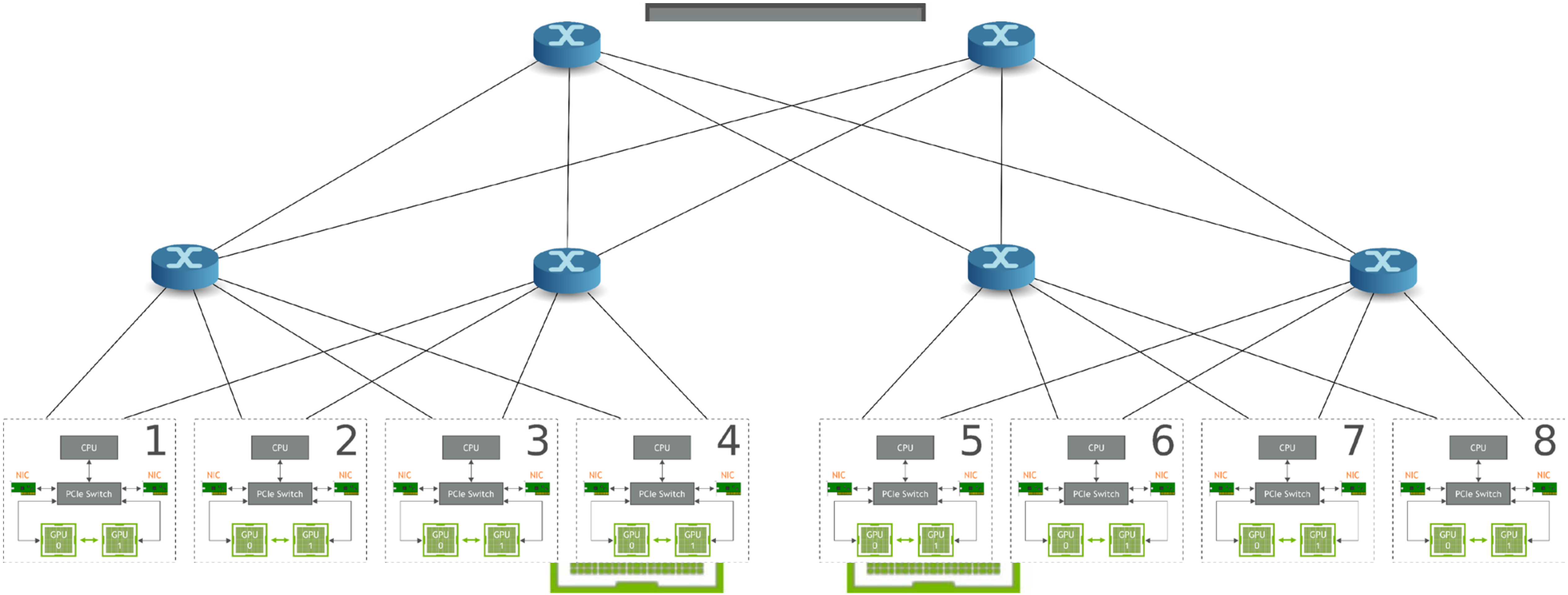
Communication Algorithm and Libraries

- Topology-aware, Adaptive to the configuration

In large-scale systems, a hierarchy of interconnect technologies are used giving different bandwidths, latencies, and connectivity



In large-scale systems, a hierarchy of interconnect technologies are used yielding different bandwidths, latencies, and connectivity



- In such distributed scenarios, topology and routing also plays an important role
  – Communication algorithm must adapt

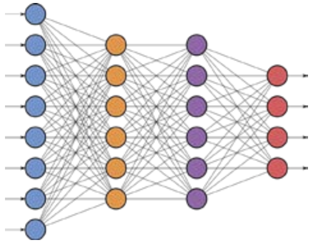
Communication algorithm can optimize utilization of the available link capacity by avoiding contention in data transfers

- Collective operations
  - All-Reduce operations in synchronous data parallelism
  - All-Gather and broadcast in asynchronous data parallelism / model parallelism

- Communication Libraries
  - MPI is very mature, standardized, and provide excellent scale-out performance
    - Scale-Up?
  - NICCL
    - Scale Up, Across multiple nodes is an issue even though supported now

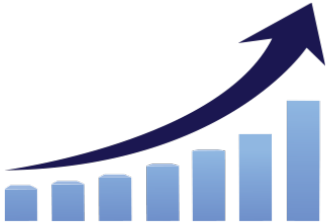
Design Scalable Models

Distributed deep learning can be accelerated through the use of computationally lighter & scalable models and define-by-run methodologies



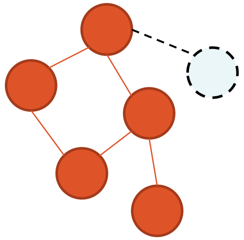
Computationally Lighter Models

- Some models are inherently more computationally expensive than others, VGG vs Inception



Scalable Models

- Models that are more scalable, tolerant to delayed parameter updates



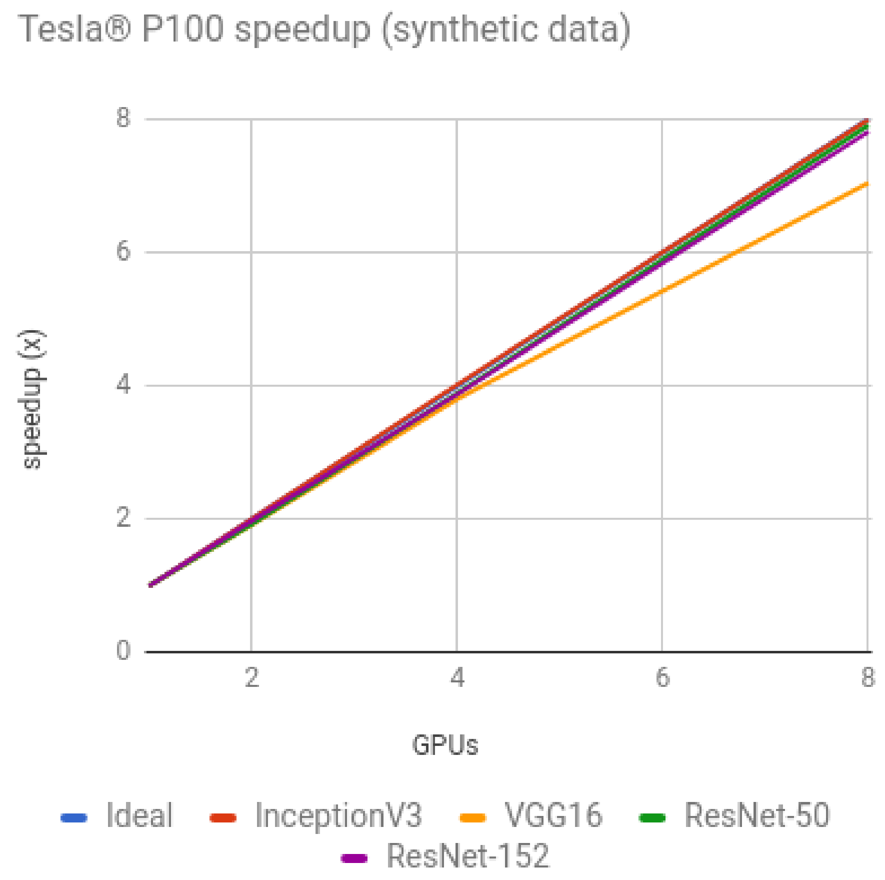
Define-by-Run Paradigm

- Dynamic graph updates, Potential for adaptation to the changing data

Design Better Models B

Models can also be designed in a way that they are more scalable and suited for the distributed deep learning

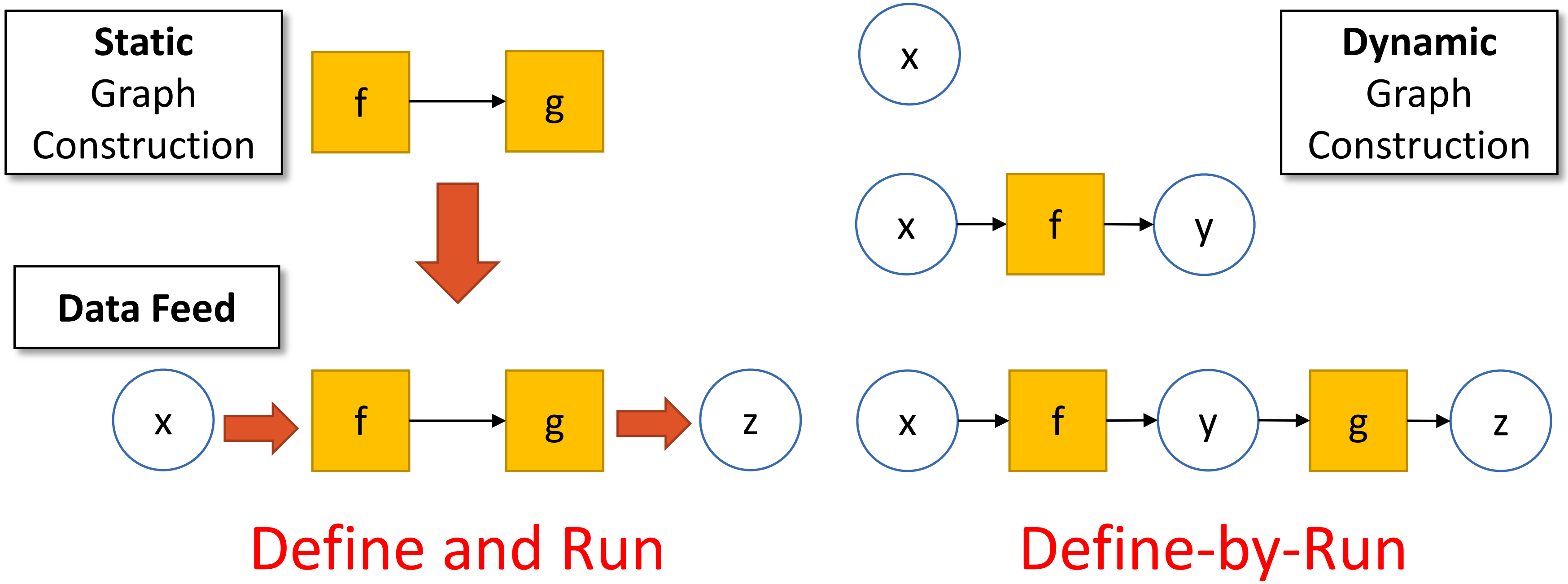
Example: VGG16 vs ResNet-152 – Single node, ImageNet, TensorFlow



Source: https://www.tensorflow.org/performance/benchmarks

In Define-by-Run frameworks, graphs constructions *on the fly* giving more flexibility and potential performance improvements

- Chainer
  - The first define-by-run deep learning framework
- TensorFlow
  - Eager execution



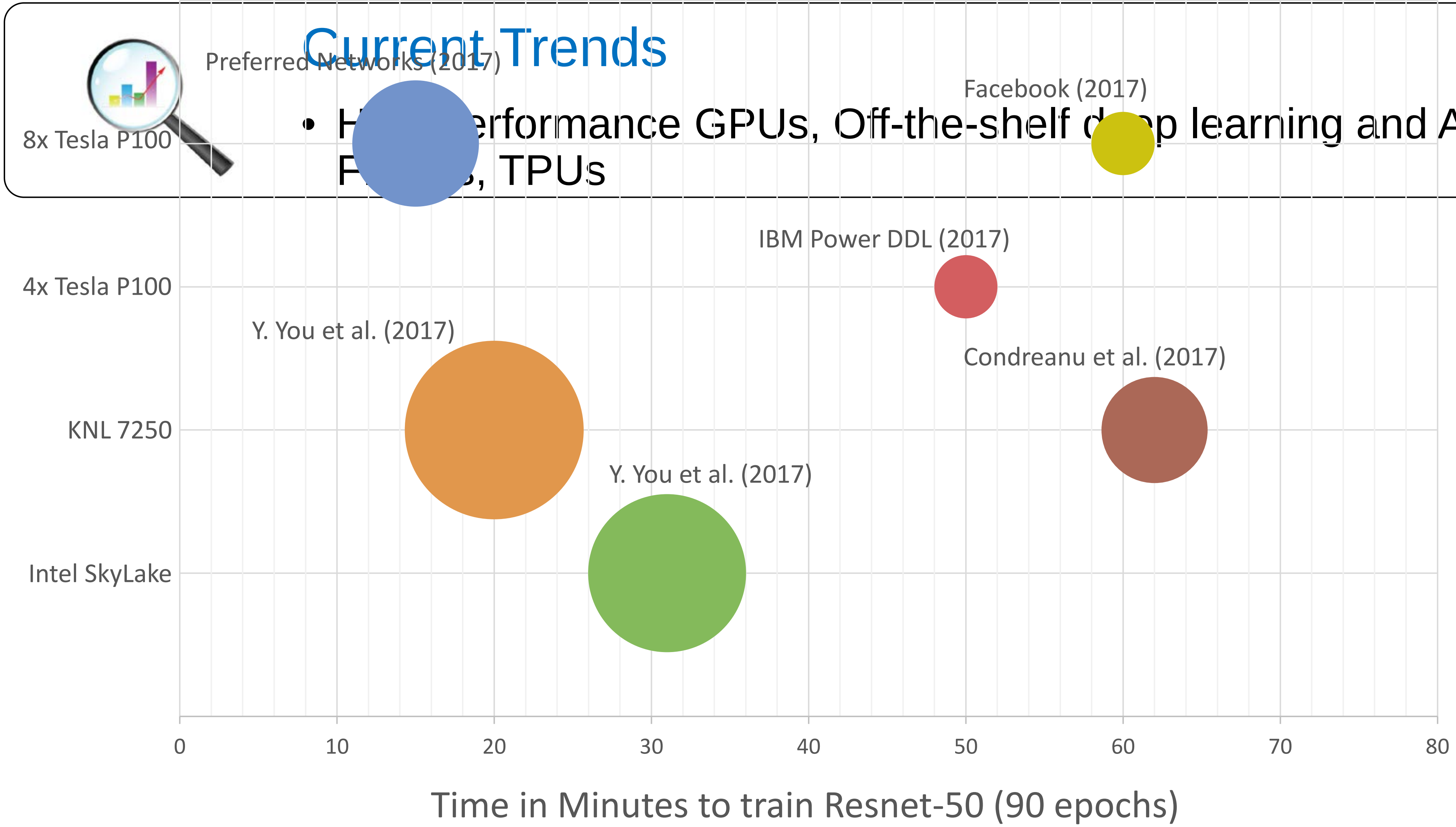
Source: Kenta Oono, Preferred Networks Inc.

Improve Hardware

Hardware advances will still play an integral role in improving efficiency of deep learning algorithms

Current Trends

- H[illegible]erformance GPUs, Off-the-shelf d[illegible]ep learning and AI systems, F[illegible]s, TPUs



Currently big players with a lot of (advanced) hardware are winning!

Improve Hardware C

This presentation will introduce distributed deep learning, walk through prominent techniques, and identify existing challenges and future directions

Introduction and Motivation

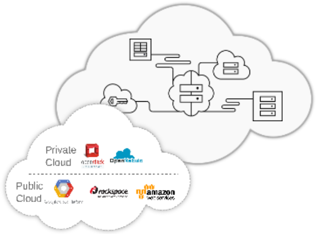
Existing Techniques and Toolsets

Future Directions

Future research directions include a holistic approach for variety of workloads & heterogeneous environments, and NN-aware communication

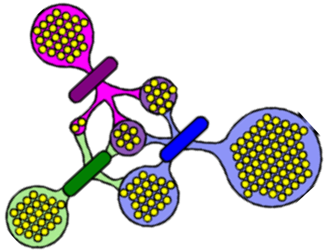
Holistic Approach

- New workloads are emerging, different toolsets, in modern apps workload of different types will co-exist



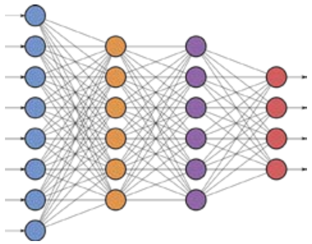
Heterogeneous Environments

- Accelerators, Adaptive to the configuration, Dynamic resource allocations, Clouds (challenges related to multi-tenancy)



Neural Network Aware Communication

- Pipelining communication with computation, Topology-aware, Offloading



Improved Algorithms and Models

- Less communication requirements, Good accuracy with large batch sizes, adaptive learning rates, Automatic neural network

References – Incomplete list

Dean, Jeffrey, et al. "Large scale distributed deep networks." Advances in neural information processing systems. 2012.

Cho, Minsik, et al. "PowerAI DDL." arXiv preprint arXiv:1708.02188 (2017).

Goyal, Priya, et al. "Accurate, Large Minibatch SGD: Training ImageNet in 1 Hour." arXiv preprint arXiv:1706.02677 (2017).

https://www.tensorflow.org/

Akiba, Takuya, Shuji Suzuki, and Keisuke Fukuda. "Extremely Large Minibatch SGD: Training ResNet-50 on ImageNet in 15 Minutes." arXiv preprint arXiv:1711.04325 (2017).

Shazeer, Noam, et al. "Outrageously large neural networks: The sparsely-gated mixture-of-experts layer." arXiv preprint arXiv:1701.06538 (2017).

Chilimbi, Trishul M., et al. "Project Adam: Building an Efficient and Scalable Deep Learning Training System." OSDI. Vol. 14. 2014.

Abadi, Martín, et al. "Tensorflow: Large-scale machine learning on heterogeneous distributed systems." arXiv preprint arXiv:1603.04467 (2016).

Tokui, Seiya, et al. "Chainer: a next-generation open source framework for deep learning." Proceedings of workshop on machine learning systems (LearningSys) in the twenty-ninth annual conference on neural information processing systems (NIPS). Vol. 5. 2015.

Jozefowicz, Rafal, et al. "Exploring the limits of language modeling." arXiv preprint arXiv:1602.02410 (2016).

https://www.nvidia.com/en-us/deep-learning-ai/

https://keras.io/

Bottou, Léon. "Large-scale machine learning with stochastic gradient descent." Proceedings of COMPSTAT'2010. Physica-Verlag HD, 2010. 177-186.

Zoph, Barret, et al. "Learning transferable architectures for scalable image recognition." arXiv preprint arXiv:1707.07012 (2017).