ETH zürich

spcl.inf.ethz.ch
@spcl_eth

D INFK

T. Hoefler, T. Ben-Nun

Optimizing and Benchmarking Large-Scale Deep Learning

Machine Learning Day at ISC'19, Frankfurt, Germany

With contributions from Dan Alistarh, Yosuke Oyama, Cedric Renggli, and others at SPCL, IST Austria, and Tokyo Tech

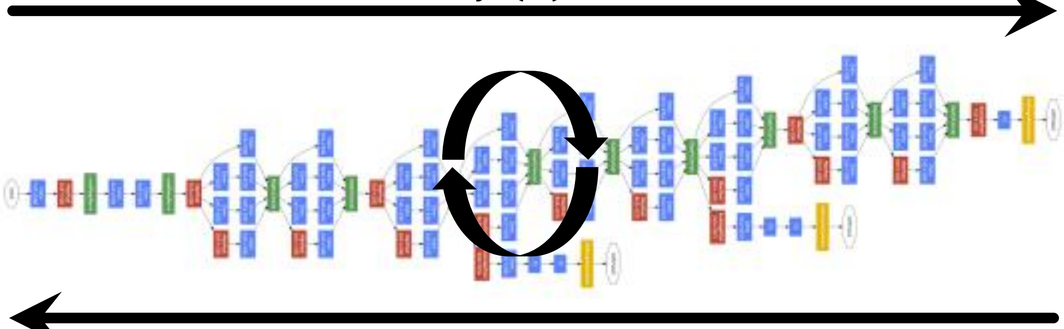
SPCL

EuroMPI '19
September 10-13, 2019
Zurich, Switzerland

A brief theory of supervised deep learning (minibatch SGD)

labeled samples $x \in X \subset \mathcal{D}$

$f(x)$



layer-wise parameter update

Cat	0.54	Cat	1.00
Dog	0.28	Dog	0.00
Airplane	0.07	Airplane	0.00
Horse	0.33	Horse	0.00
Banana	0.02	Banana	0.00
Truck	0.02	Truck	0.00

label domain Y true label $l(x)$

$f(x): X \to Y$

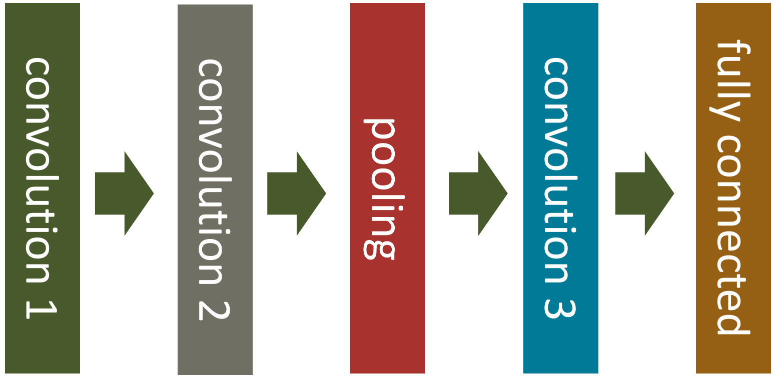
network structure (fixed)

weights w (learned)

$$w^* = \mathrm{argmin}_{w \in \mathbb{R}^d}\, \mathbb{E}_{x \sim \mathcal{D}}[\ell(w,x)]$$

$$f(x) = f_n\left(f_{n-1}\left(f_{n-2}\left(\dots f_1(x)\dots\right)\right)\right)$$

$f_1(x)$ $f_2(f_1(x))$ … $f(x)$

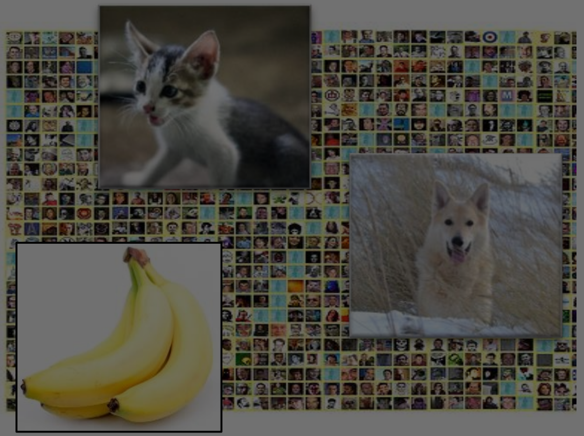


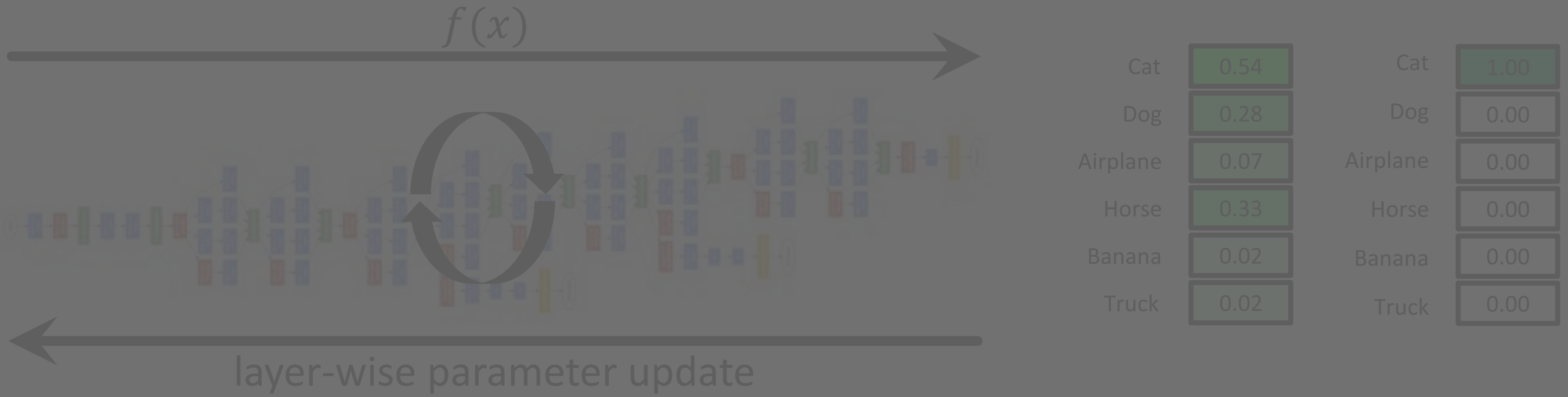
$$\ell_{sq}(w,x) = \left(f(x) - l(x)\right)^2$$

$$\ell_{0-1}(w,x) = \begin{cases} 0 & f(x) = l(x) \\ 1 & f(x) \neq l(x) \end{cases}$$

$$\ell_{ce}(w,x) = -\sum_i l(x)_i \cdot \log \frac{e^{f(x)_i}}{\sum_k e^{f(x)_k}}$$

A brief theory of supervised deep learning (minibatch SGD)





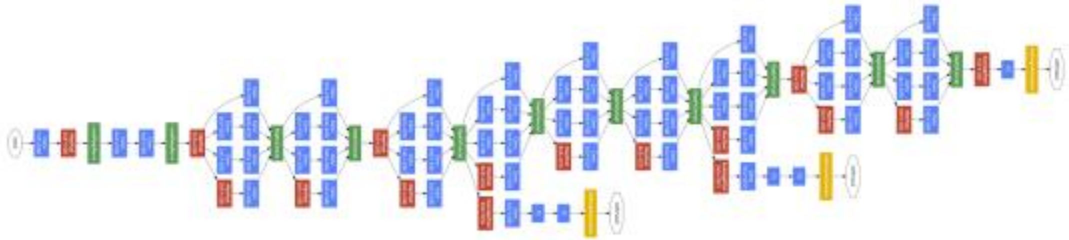
≥TBs of random access

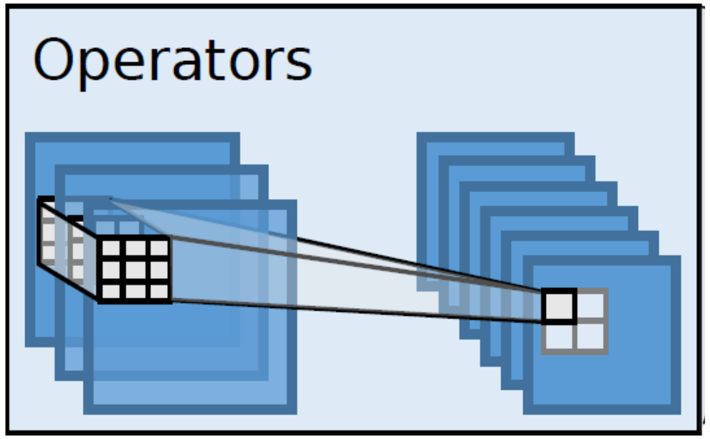
100MiB-26GiB and beyond

22k-millions

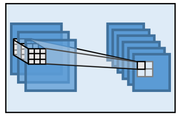
Deep Learning is Supercomputing!

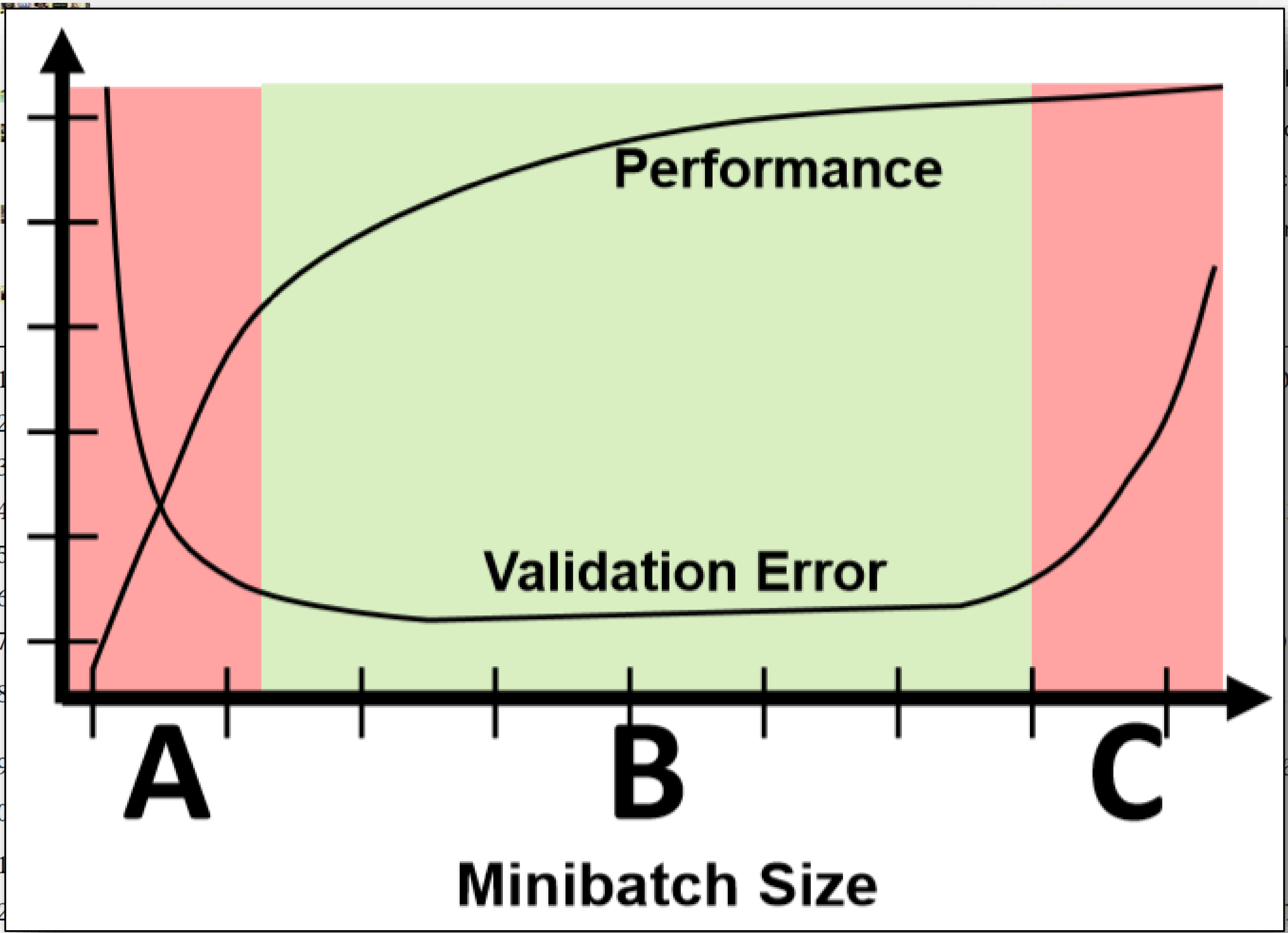
Computational Principles





Minibatch Stochastic Gradient Descent (SGD)





Cat	0.54
Dog	0.28
lane	0.07
orse	0.04
ycle	0.03
ruck	0.02
	0.02

Cat	1.00
Dog	0.00
Airplane	0.00
Horse	0.00
Bicycle	0.00
Truck	0.00
	0.00

ondition
dataset
ion of ℓ
agation
r.t. data
ameters
ction u

13: **end for**

T. Ben-Nun, T. Hoefler: Demystifying Parallel and Distributed Deep Learning: An In-Depth Concurrency Analysis, CSUR 2019

Microbatching (μ-cuDNN) – how to implement layers best in practice?

- In cuDNN there are ~16 convolution implementations
- Performance depends on temporary memory (workspace) size
- Key idea: segment minibatch into microbatches, reuse workspace, use different algorithms

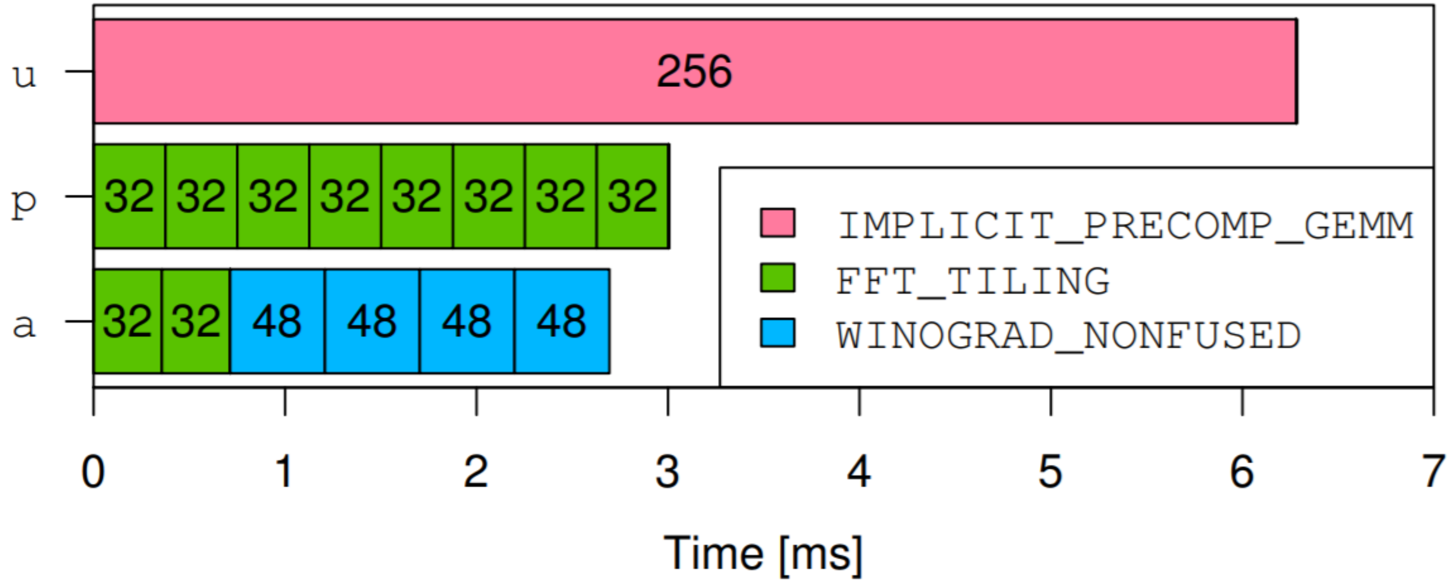
Microbatching Strategy

none (undivided)

powers-of-two only

any (unrestricted)

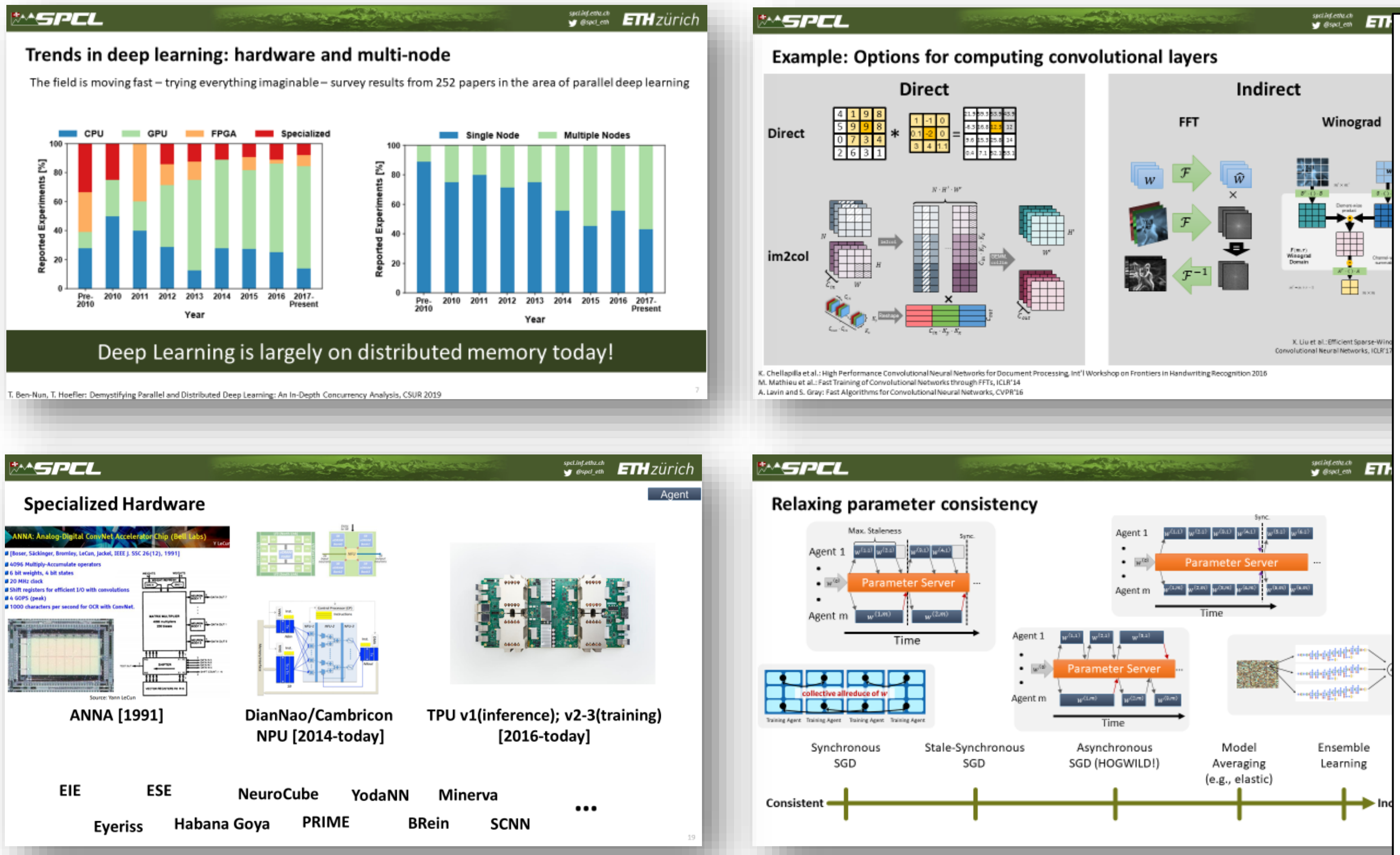
Fast (up to 4.54x faster on DeepBench)

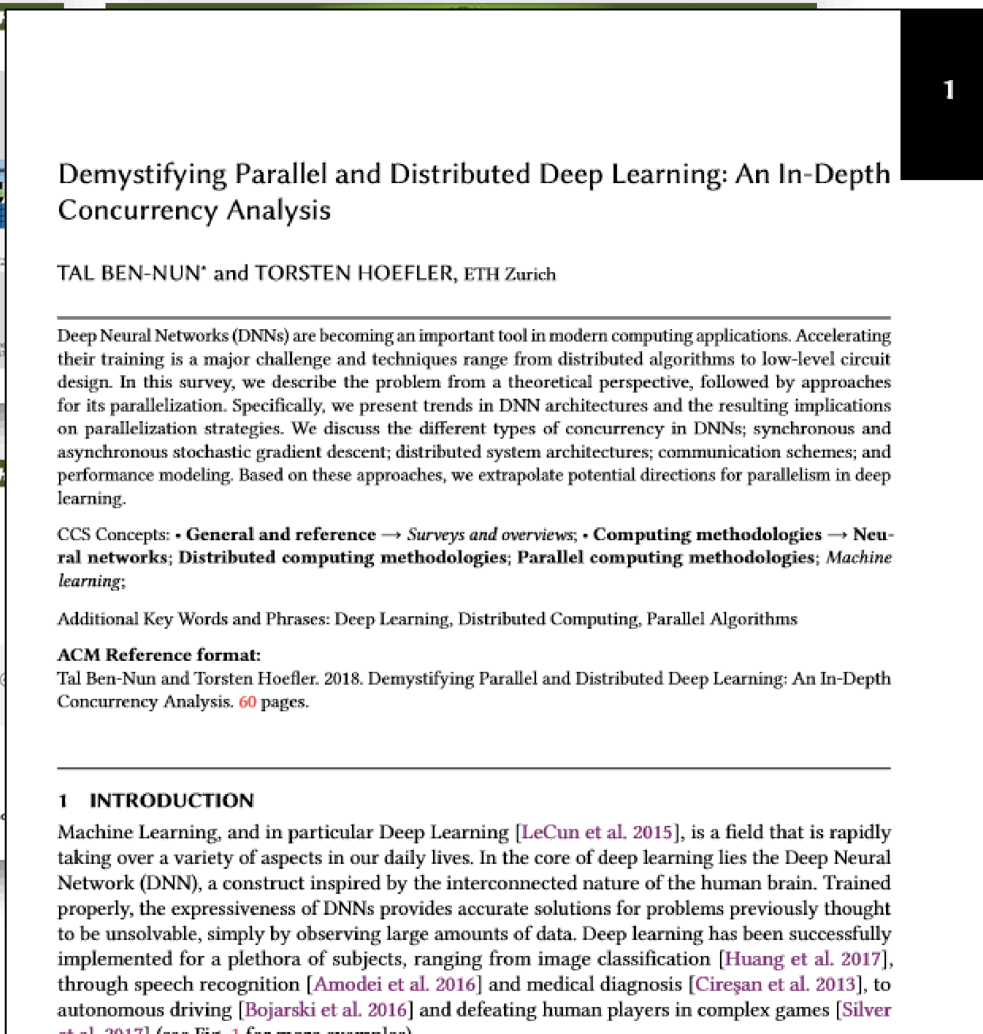


Y. Oyama, T. Ben-Nun, T. Hoefler and S. Matsuoka: μ-cuDNN: Accelerating Deep Learning Frameworks with Micro-Batching, Cluster 2018

Demystifying Parallel and Distributed Deep Learning: An In-Depth Concurrency Analysis

- https://www.arxiv.org/abs/1802.09941




1

Demystifying Parallel and Distributed Deep Learning: An In-Depth Concurrency Analysis

TAL BEN-NUN* and TORSTEN HOEFLER, ETH Zurich

Deep Neural Networks (DNNs) are becoming an important tool in modern computing applications. Accelerating their training is a major challenge and techniques range from distributed algorithms to low-level circuit design. In this survey, we describe the problem from a theoretical perspective, followed by approaches for its parallelization. Specifically, we present trends in DNN architectures and the resulting implications on parallelization strategies. We discuss the different types of concurrency in DNNs; synchronous and asynchronous stochastic gradient descent; distributed system architectures; communication schemes; and performance modeling. Based on these approaches, we extrapolate potential directions for parallelism in deep learning.

CCS Concepts: • **General and reference** → *Surveys and overviews*; • **Computing methodologies** → **Neural networks**; **Distributed computing methodologies**; **Parallel computing methodologies**; *Machine learning*;

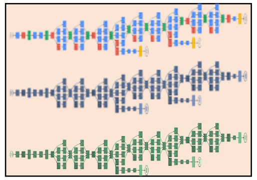
Additional Key Words and Phrases: Deep Learning, Distributed Computing, Parallel Algorithms

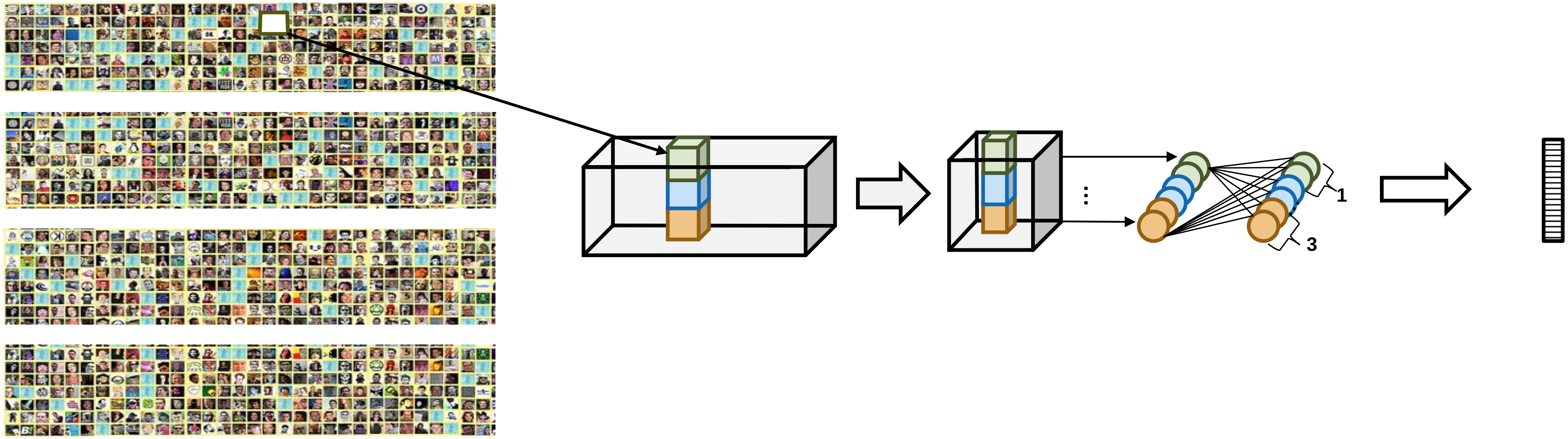
ACM Reference format:
Tal Ben-Nun and Torsten Hoefler. 2018. Demystifying Parallel and Distributed Deep Learning: An In-Depth Concurrency Analysis. 60 pages.

1 INTRODUCTION

Machine Learning, and in particular Deep Learning [LeCun et al. 2015], is a field that is rapidly taking over a variety of aspects in our daily lives. In the core of deep learning lies the Deep Neural Network (DNN), a construct inspired by the interconnected nature of the human brain. Trained properly, the expressiveness of DNNs provides accurate solutions for problems previously thought to be unsolvable, simply by observing large amounts of data. Deep learning has been successfully implemented for a plethora of subjects, ranging from image classification [Huang et al. 2017], through speech recognition [Amodei et al. 2016] and medical diagnosis [Cireşan et al. 2013], to autonomous driving [Bojarski et al. 2016] and defeating human players in complex games [Silver

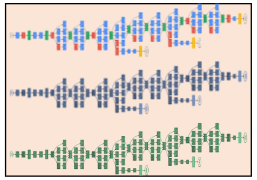
Model parallelism – limited by network size



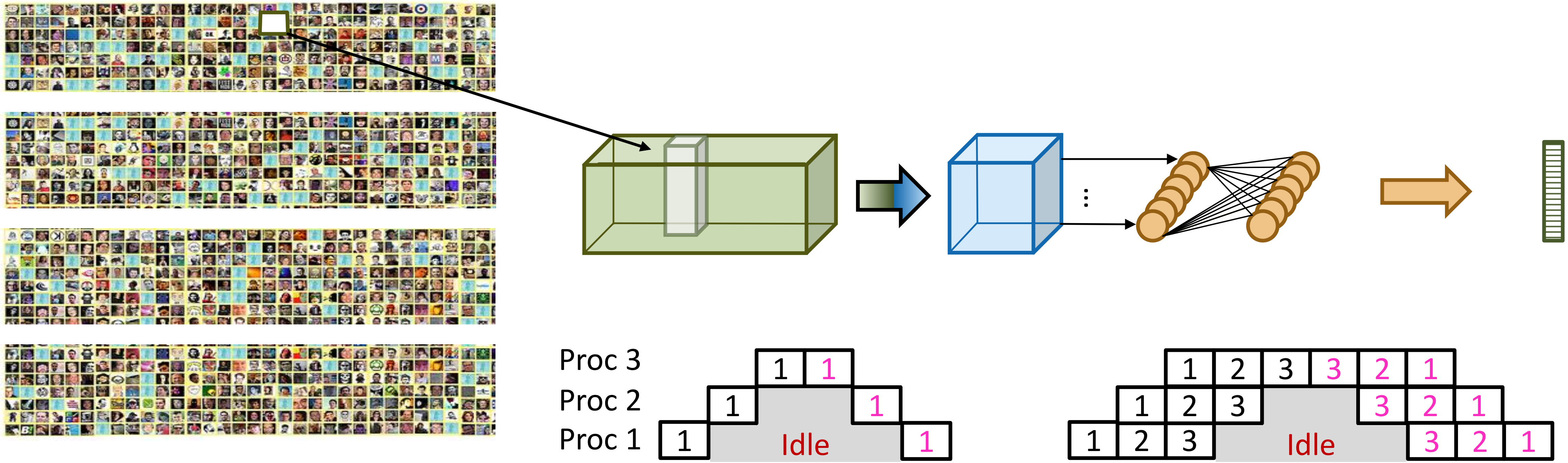


- Parameters can be distributed across processors
- Mini-batch has to be copied to all processors
- Backpropagation requires complex communication every layer

M. Forrest, D. Roweth, N. Stroud, D. J. Wallace, and G. V. Wilson. 1987. Implementing Neural Network Models on Parallel Computers. Comput. J.



Pipeline parallelism – limited by network size

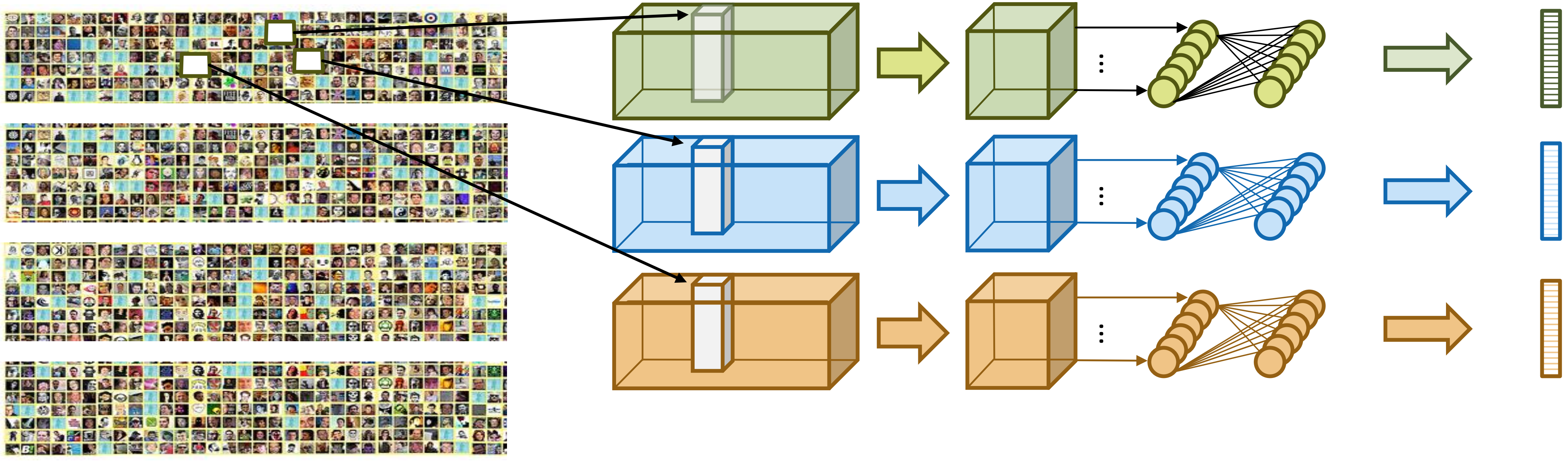


Microbatching

- Layers/parameters can be distributed across processors
- Sparse communication pattern (only pipeline stages)
- Mini-batch has to be copied through all processors
- Consistent model introduces idle-time “Bubble”

G. Blelloch and C.R. Rosenberg: Network Learning on the Connection Machine, IJCAI’87

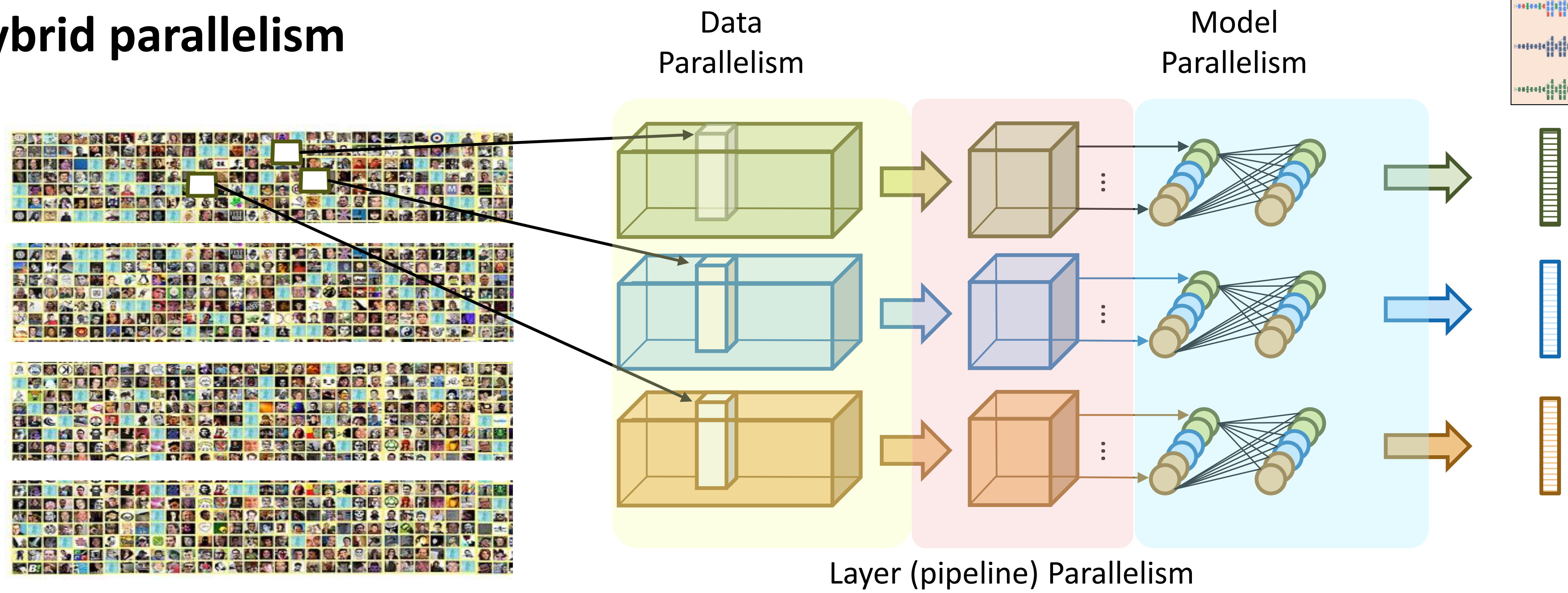
Data parallelism – limited by batch-size



- Simple and efficient solution, easy to implement
- Duplicate parameters at all processors
- Affects generalization

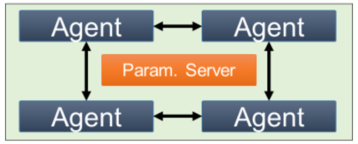
X. Zhang et al.: An Efficient Implementation of the Back-propagation Algorithm on the Connection Machine CM-2, NIPS'89

Hybrid parallelism

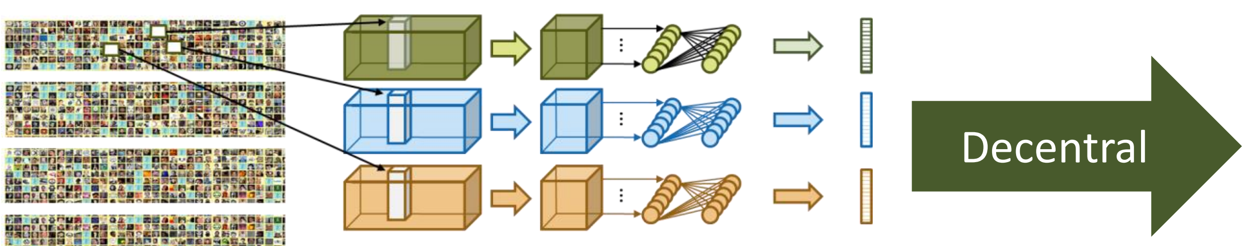


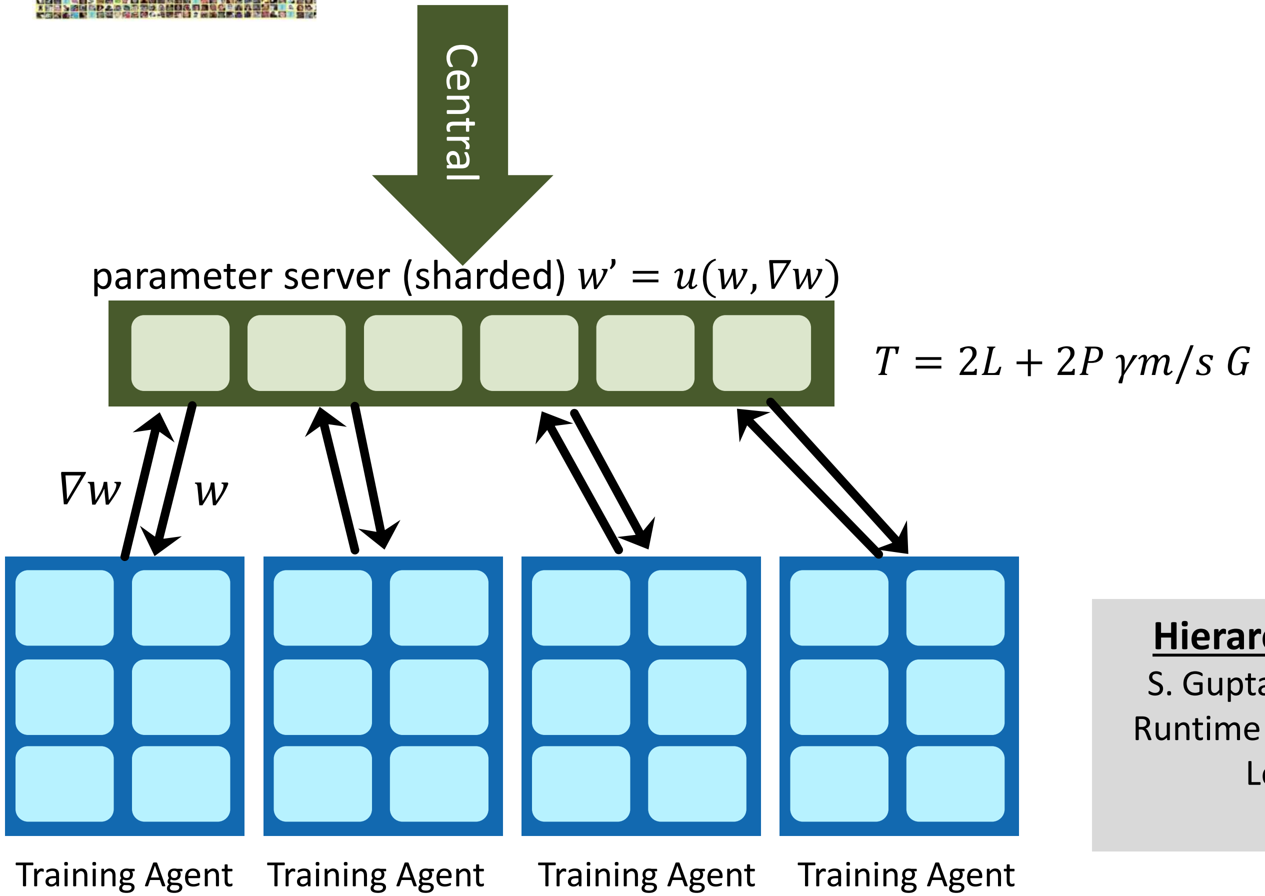
- **Layers/parameters can be distributed across processors**
- **Can distribute minibatch**
- **Often specific to layer-types (e.g., distribute fc layers but handle conv layers data-parallel)**
  - Enables arbitrary combinations of data, model, and pipeline parallelism – very powerful!

A. Krizhevsky: One weird trick for parallelizing convolutional neural networks, arXiv 2014
J. Dean et al.: Large scale distributed deep networks, *NIPS'*12.
T. Ben-Nun, T. Hoefler: Demystifying Parallel and Distributed Deep Learning: An In-Depth Concurrency Analysis, CSUR 2019



Updating parameters in distributed data parallelism

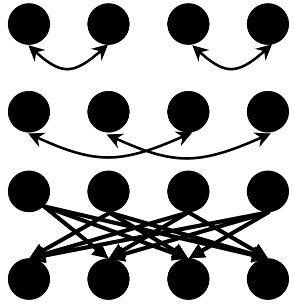




$T = 2L + 2P\ \gamma m/s\ G$

MPI

- Collective operations
- Topologies
- Neighborhood collectives
- RMA?



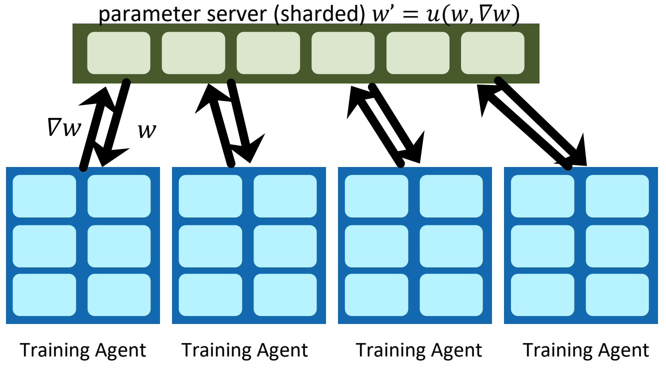
$T = 2L \log_2 P + 2\gamma m G (P-1)/P$

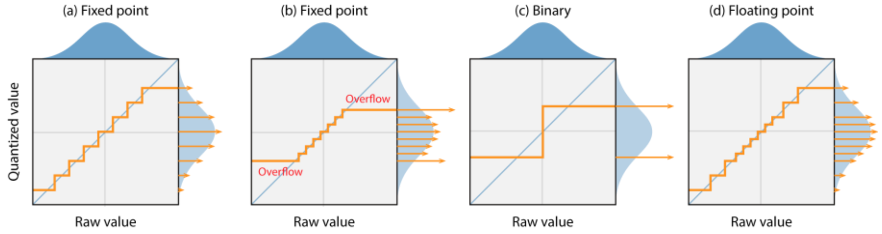
Hierarchical Parameter Server
S. Gupta et al.: Model Accuracy and Runtime Tradeoff in Distributed Deep Learning: A Systematic Study. ICDM'16

Adaptive Minibatch Size
S. L. Smith et al.: Don't Decay the Learning Rate, Increase the Batch Size, arXiv 2017

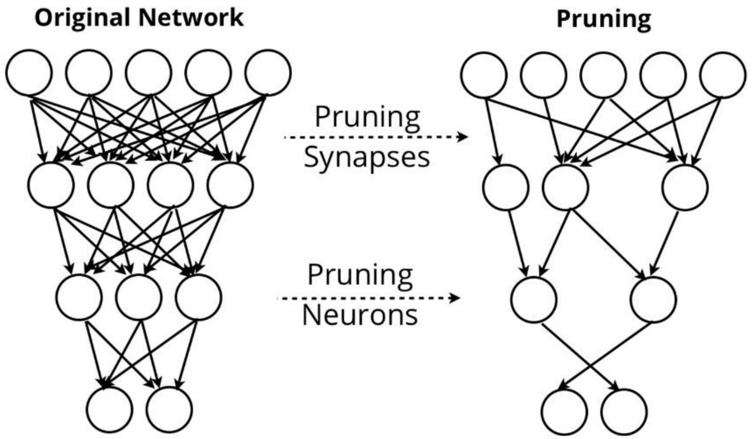
Communication optimizations

- **Different options how to optimize updates**
  - Send ∇w, receive w
  - Send FC factors (o_{l-1}, o_l), compute ∇w on parameter server
    Broadcast factors to not receive full w
  - Use lossy compression when sending, accumulate error locally!
- **Quantization**
  - Quantize weight updates and potentially weights
  - Main trick is stochastic rounding [1] – expectation is more accurate
    Enables low precision (half, quarter) to become standard
  - TernGrad - ternary weights [2], 1-bit SGD [3], …
- **Sparsification**
  - Do not send small weight updates **or** only send top-k [4]
    Accumulate omitted gradients locally

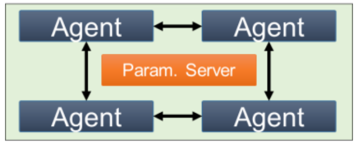




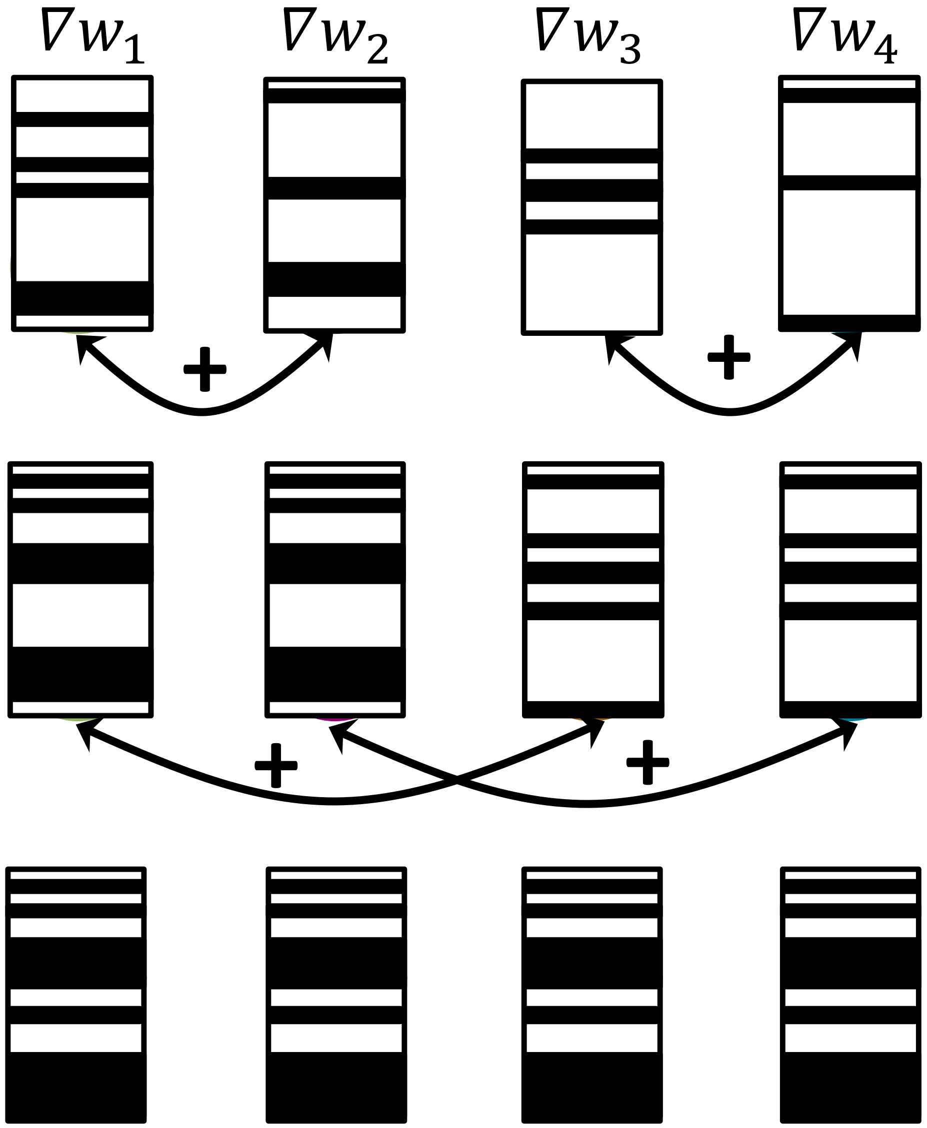
source: ai.intel.com



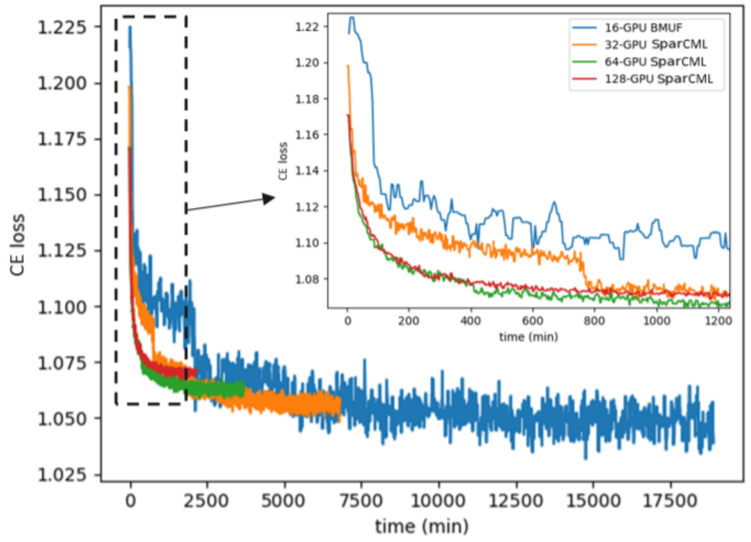
[1] S. Gupta et al. Deep Learning with Limited Numerical Precision, ICML'15
[2] F. Li and B. Liu. Ternary Weight Networks, arXiv 2016
[3] F. Seide et al. 1-Bit Stochastic Gradient Descent and Application to Data-Parallel Distributed Training of Speech DNNs, In Interspeech 2014
[4] C. Renggli et al. SparCML: High-Performance Sparse Communication for Machine Learning, arXiv 2018

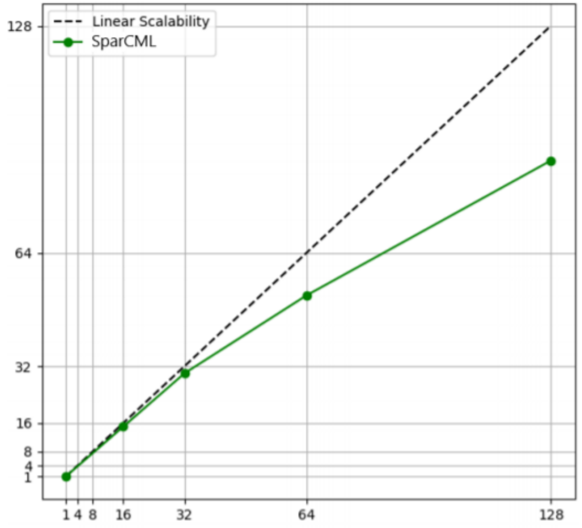


SparCML – Quantized sparse allreduce for decentral updates



Six epochs, 60 million params





Microsoft Speech Production Workload Results – **2 weeks → 2 days**!

System	Dataset	Model	# of nodes	Algorithm	Speedup
Piz Daint	ImageNet	VGG19	8	Q4	1.55 (3.31)
Piz Daint	ImageNet	AlexNet	16	Q4	1.30 (1.36)
Piz Daint EC2	MNIST	MLP	8	Top16_Q4 Top16_Q4	3.65 (4.53) 19.12 (22.97)

C. Renggli et al. SparCML: High-Performance Sparse Communication for Machine Learning, arXiv 2018

Reproducing and Benchmarking Deep Learning

- **End result – generalization**

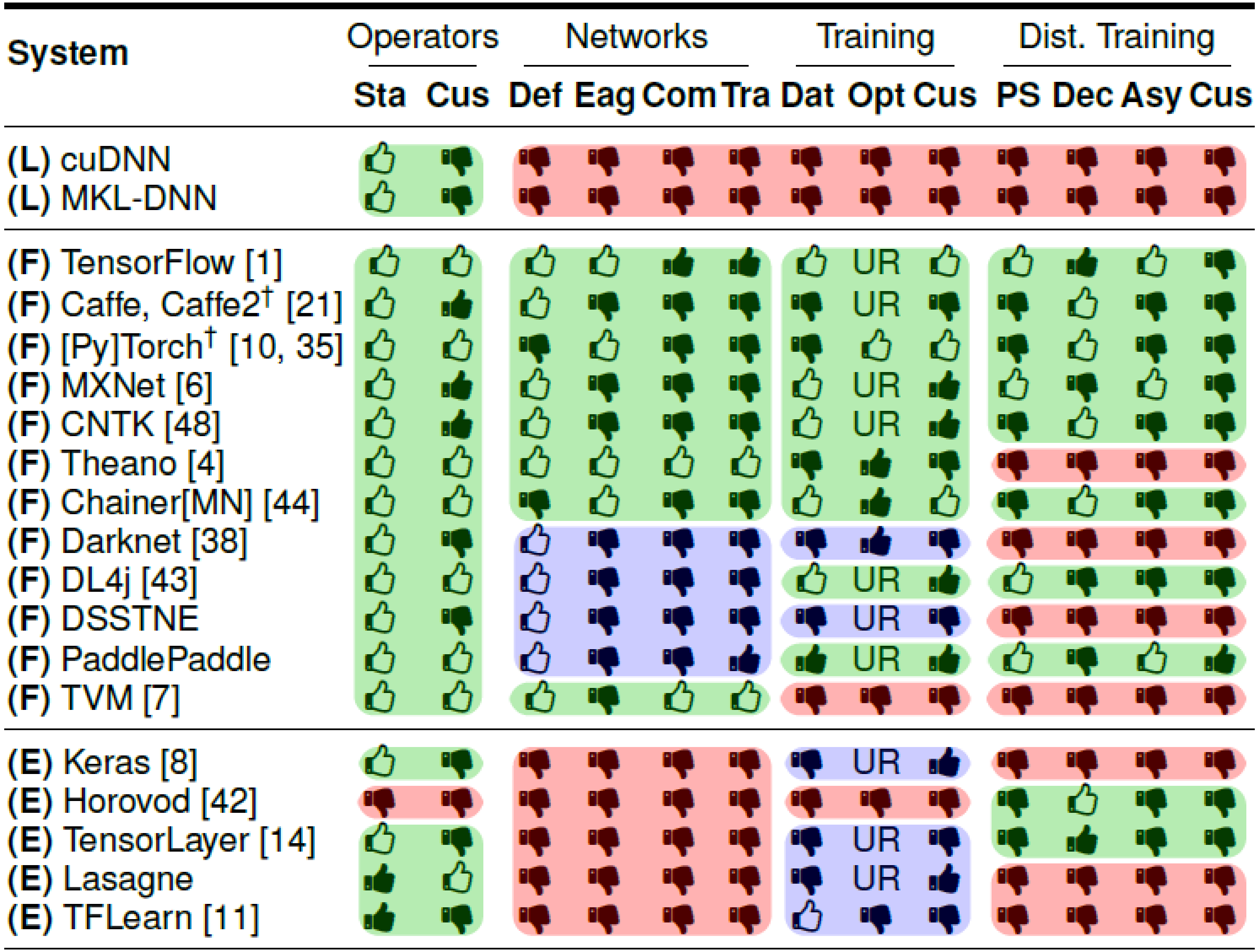
Benchmark	Focus			Metrics									Criteria			Customizability			DL Workloads						Remarks
	Perf	Con	Acc	Tim	Cos	Ene	Util	Mem	Tput	Brk	Sca	Com	TTA	FTA	Lat	Clo	Ope	Inf	Ops	Img	Obj	Spe	Txt	RL	
DeepBench [39]	👍	👎	👎	👍	👎	👎	👎	👎	👎	👎	👎	👎	👎	👎	👍	👍	👎	👍	👍	👎	👎	👎	👎	👎	Ops: Conv., GEMM, RNN, Allreduce
TBD [47]	👍	👎	👎	👎	👎	👎	👍	👍	👍	👎	👎	👎	👎	👎	👍	👍	👎	👎	👎	👍	👍	👍	👍	👍	+GANs
Fathom [2]	👍	👎	👎	👍	👎	👎	👎	👎	👍	👍	👍	👎	👎	👎	👍	👍	👎	👎	👎	👍	👎	👍	👍	👍	+Auto-encoders
DAWNBench [9]	👍	👍	👎	👍	👍	👎	👎	👎	👎	👎	👎	👎	👍	👎	👍	👎	👍	👎	👎	👍	👎	👎	👍	👎	
Kaggle [21]	👎	👎	👍	👎	👎	👎	👎	👎	👎	👎	👎	👎	👎	👍	👎	👎	👍	👎	👎	👍	👍	👍	👍	👍	Varying workloads
ImageNet [13]	👎	👎	👍	👎	👎	👎	👎	👎	👎	👎	👎	👎	👎	👍	👎	👎	👍	👎	👎	👍	👍	👎	👎	👎	
MLPerf [30]	👍	👍	👍	👍	👍	👍	👎	👎	👎	👎	👎	👎	👍	👍	👍	👍	👍	👎	👎	👍	👍	👍	👍	👍	
Deep500	👍	👍	👍	👍	👍	🖒	👍	🖒	👍	👍	👍	👍	👍	👍	👍	👍	👍	👍	👍	👍	🖒	🖒	🖒	🖒	

TABLE II: **An overview of available DL benchmarks**, focusing on the offered functionalities. **Perf**: Performance, **Con**: Convergence, **Acc**: Accuracy, **Tim**: Time, **Cos**: Cost, **Ene**: Energy, **Util**: Utilization, **Mem**: Memory Footprint, **Tput**: Throughput (Samples per Second), **Brk**: Timing Breakdown, **Sca**: Strong Scaling, **Com**: Communication and Load Balancing, **TTA**: Time to Accuracy, **FTA**: Final Test Accuracy, **Lat**: Latency (Inference), **Clo**: Closed (Fixed) Model Contests, **Ope**: Open Model Contests, **Inf**: Fixed Infrastructure for Benchmarking, **Ops**: Operator Benchmarks, **Img**: Image Processing, **Obj**: Object Detection and Localization, **Spe**: Speech Recognition, **Txt**: Text Processing and Machine Translation, **RL**: Reinforcement Learning Problems, 👍: A given benchmark does offer the feature. 🖒: Planned benchmark feature. 👎: A given benchmark does not offer the feature.

- **Sample throughput**

T. Ben-Nun et al., A Modular Benchmarking Infrastructure for High-Performance and Reproducible Deep Learning, IPDPS 2019.

Existing Deep Learning Frameworks



- Customizing operators relies on framework
- Network representation
- Dataset representation
- Training algorithm
- Distributed training (e.g., asynchronous SGD)

T. Ben-Nun et al., A Modular Benchmarking Infrastructure for High-Performance and Reproducible Deep Learning, IPDPS 2019.

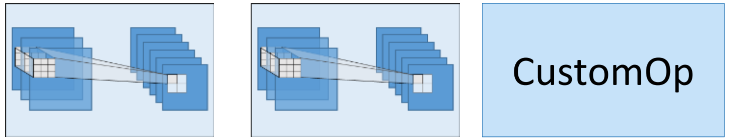
Deep500

DEEP 500

- **Deep learning meta-framework: a framework for frameworks to reside in**

Level 0

```
forward()
gradient()
```



Operators

T. Ben-Nun et al., A Modular Benchmarking Infrastructure for High-Performance and Reproducible Deep Learning, IPDPS 2019.

Deep500

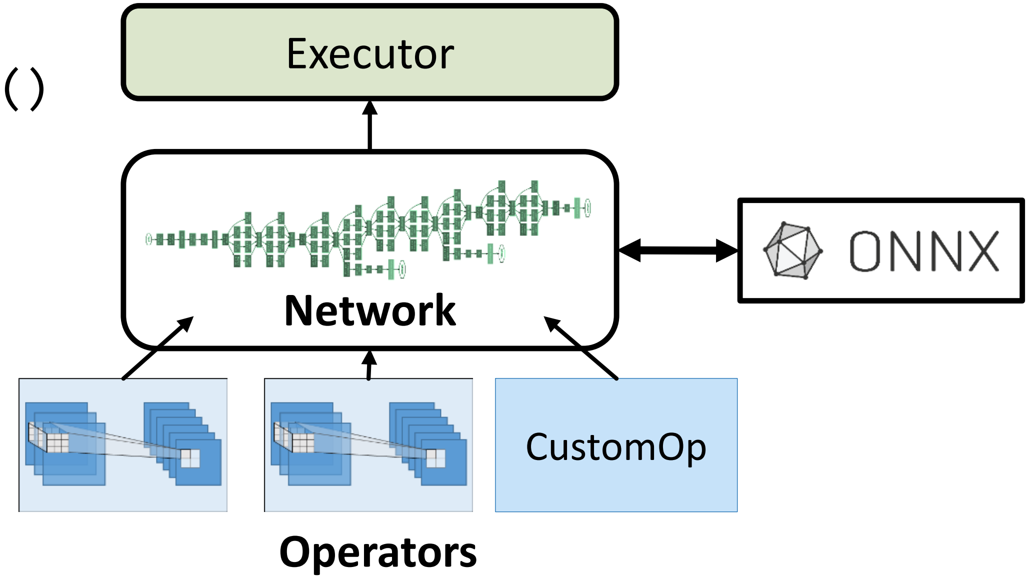
DEEP 500

- Deep learning **meta-framework**: a framework for frameworks to reside in

Level 0

Level 1

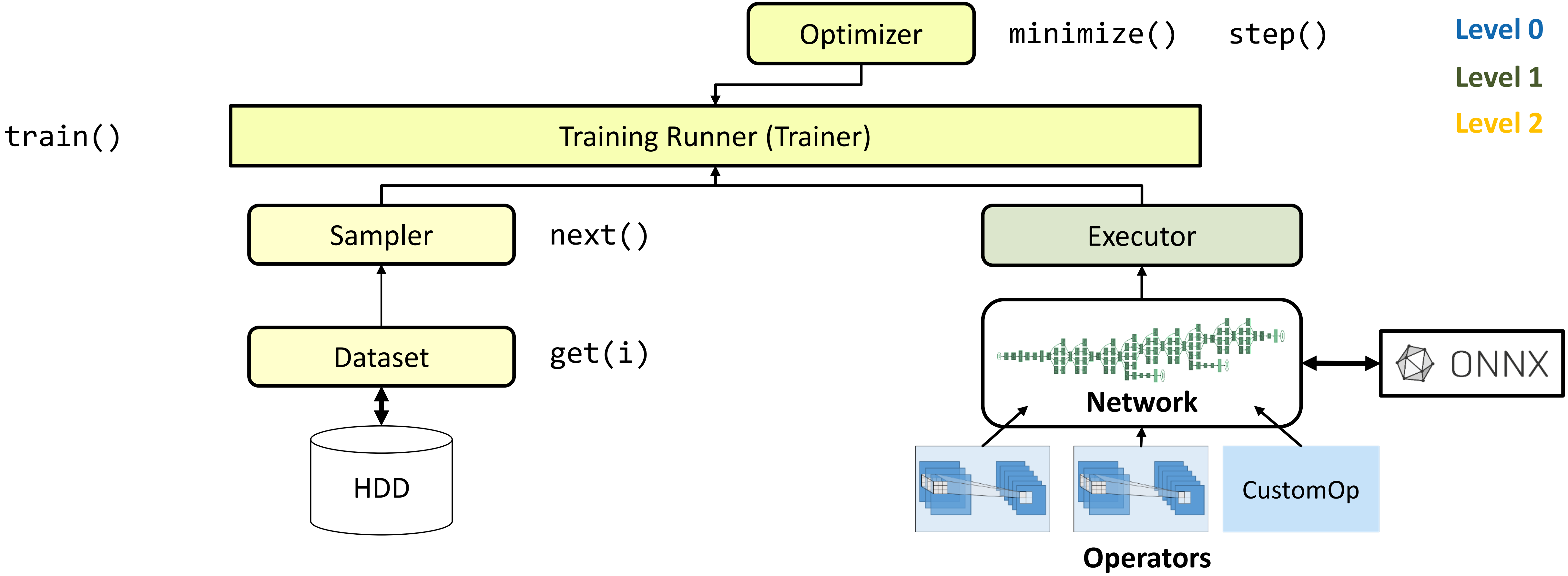
```
inference()
inference_and_backprop()

      add_node()
      add_edge()
      remove_...
```



T. Ben-Nun et al., A Modular Benchmarking Infrastructure for High-Performance and Reproducible Deep Learning, IPDPS 2019.

Deep500

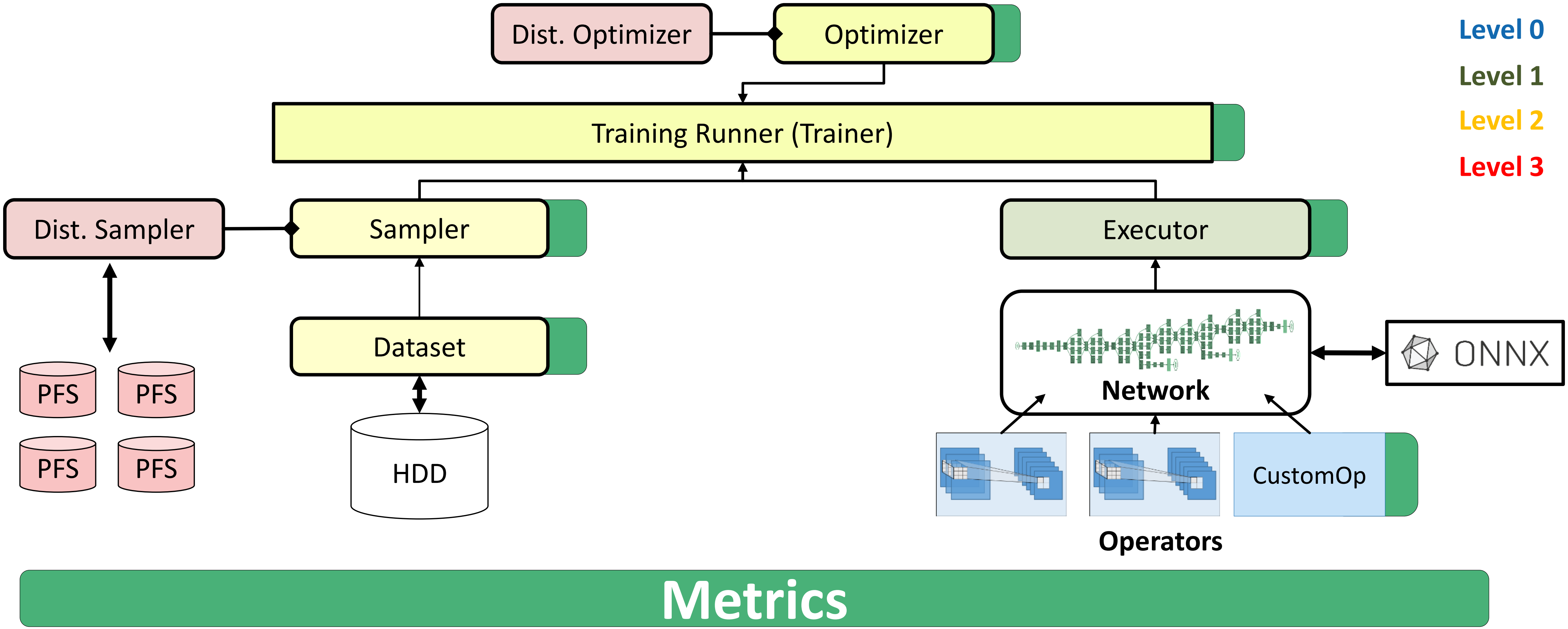
- **Deep learning meta-framework: a framework for frameworks to reside in**



T. Ben-Nun et al., A Modular Benchmarking Infrastructure for High-Performance and Reproducible Deep Learning, IPDPS 2019.

Deep500

- **Deep learning meta-framework: a framework for frameworks to reside in**



T. Ben-Nun et al., A Modular Benchmarking Infrastructure for High-Performance and Reproducible Deep Learning, IPDPS 2019.

For Benchmarking: Recipes

Fixed definitions + mutable definitions + acceptable metric set = Recipe

T. Ben-Nun et al., A Modular Benchmarking Infrastructure for High-Performance and Reproducible Deep Learning, IPDPS 2019.

For Benchmarking: Recipes

Fixed definitions + mutable definitions + acceptable metric set = Recipe

```python
""" A recipe for running the CIFAR-10 dataset with ResNet-44 and a momentum
    optimizer, with metrics for final test accuracy. """

import deep500 as d5
from recipes.recipe import run_recipe

# Using PyTorch as the framework
import deep500.frameworks.pytorch as d5fw


# Fixed Components
FIXED = {
    'model': 'resnet',
    'model_kwargs': dict(depth=44),
    'dataset': 'cifar10',
    'train_sampler': d5.ShuffleSampler,
    'epochs': 1
}

# Mutable Components
MUTABLE = {
    'batch_size': 64,
    'executor': d5fw.from_model,
    'executor_kwargs': dict(device=d5.GPUDevice()),
    'optimizer': d5fw.MomentumOptimizer,
    'optimizer_args': (0.1, 0.9),
}

# Acceptable Metrics
METRICS = [
    (d5.TestAccuracy(), 93.0)
]


if __name__ == '__main__':
    run_recipe(FIXED, MUTABLE, METRICS) or exit(1)
```

https://github.com/deep500/deep500/blob/master/recipes/cifar10_resnet44.py

T. Ben-Nun et al., A Modular Benchmarking Infrastructure for High-Performance and Reproducible Deep Learning, IPDPS 2019.

For Customizing: New Operator

```python
class IPowOp(CustomPythonOp):
    def __init__(self, power):
        super(IPowOp, self).__init__()
        self.power = power
        assert int(power) == power # integral

    def forward(self, inputs):
        return inputs[0] ** self.power

    def backward(self, grads, fwd_inputs, fwd_outputs):
        return (grads[0] * self.power *
            (fwd_inputs[0] ** (self.power - 1)))
```

Python

```cpp
template<typename T>
class ipowop : public deep500::CustomOperator {
protected:
    int m_len;
public:
    ipowop(int len) : m_len(len) {}
    virtual ~ipowop() {}

    void forward(const T *input, T *output) {
        #pragma omp parallel for
        for (int i = 0; i < m_len; ++i)
            output[i] = std::pow(input[i], DPOWER);
    }

    void backward(const T *nextop_grad,
                  const T *fwd_input_tensor,
                  const T *fwd_output_tensor,
                  T *input_tensor_grad) {
        #pragma omp parallel for
        for (int i = 0; i < m_len; ++i) {
            input_tensor_grad[i] = nextop_grad[i] * DPOWER *
                std::pow(fwd_input_tensor[i], DPOWER - 1);
        }
    }
};
```

C++

T. Ben-Nun et al., A Modular Benchmarking Infrastructure for High-Performance and Reproducible Deep Learning, IPDPS 2019.

For Customizing: Distributed Optimization

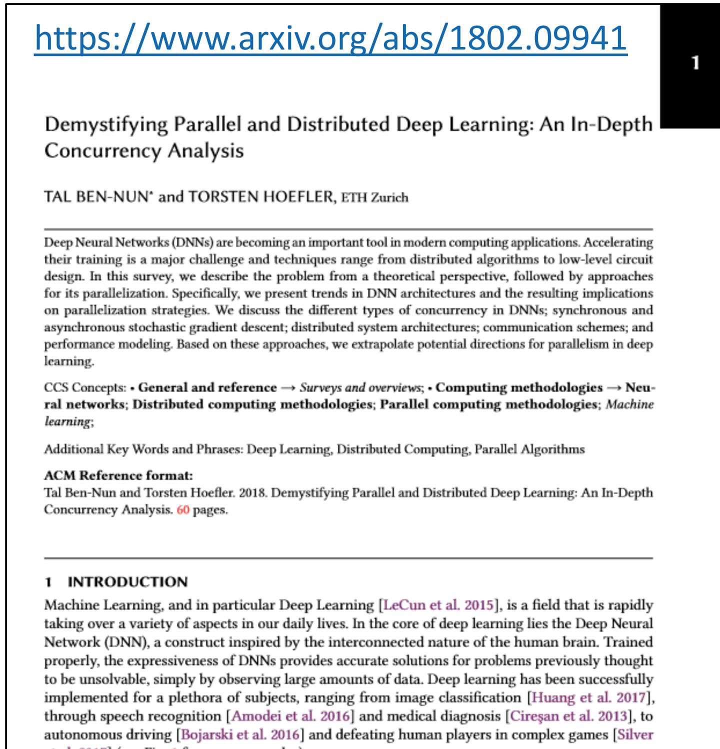
```python
class ConsistentNeighbors(DistributedOptimizer):
    # Follows communication scheme from https://arxiv.org/pdf/1705.09056.pdf

    def step(self, inputs):
        self.base_optimizer.new_input()
        for param in self.network.get_params():
            self.base_optimizer.prepare_param(param)
        output = self.executor.inference_and_backprop(inputs, self.base_optimizer.loss)
        gradients = self.network.gradient(self.base_optimizer.loss)
        for param_name, grad_name in gradients:
            param, grad = self.network.fetch_tensors([param_name, grad_name])
            grad = self.communication.reduce_from_neighbors(grad) / 3
            param = self.base_optimizer.update_rule(grad, param, param_name)
            self.network.feed_tensor(param_name, param)
        return output
```

https://github.com/deep500/deep500/blob/master/deep500/frameworks/reference/distributed_optimizers.py#L68

T. Ben-Nun et al., A Modular Benchmarking Infrastructure for High-Performance and Reproducible Deep Learning, IPDPS 2019.

HPC for Deep Learning – Summary

- **A supercomputing problem - amenable to established tools and tricks from HPC**
- **Concurrency is easy to attain, hard to program beyond data-parallelism**
- **Main bottleneck in distributed is communication – reduction by using the robustness of SGD**
- **Co-design is prevalent**
- **Very different environment from traditional HPC**
  - Trade-off accuracy for performance!
- **Main objective is generalization**
  - Performance-centric view in HPC can be harmful for accuracy


https://www.arxiv.org/abs/1802.09941

1

Demystifying Parallel and Distributed Deep Learning: An In-Depth Concurrency Analysis

TAL BEN-NUN* and TORSTEN HOEFLER, ETH Zurich

Deep Neural Networks (DNNs) are becoming an important tool in modern computing applications. Accelerating their training is a major challenge and techniques range from distributed algorithms to low-level circuit design. In this survey, we describe the problem from a theoretical perspective, followed by approaches for its parallelization. Specifically, we present trends in DNN architectures and the resulting implications on parallelization strategies. We discuss the different types of concurrency in DNNs; synchronous and asynchronous stochastic gradient descent; distributed system architectures; communication schemes; and performance modeling. Based on these approaches, we extrapolate potential directions for parallelism in deep learning.

CCS Concepts: • **General and reference** → *Surveys and overviews*; • **Computing methodologies** → **Neural networks; Distributed computing methodologies; Parallel computing methodologies**; *Machine learning*;

Additional Key Words and Phrases: Deep Learning, Distributed Computing, Parallel Algorithms

ACM Reference format:
Tal Ben-Nun and Torsten Hoefler. 2018. Demystifying Parallel and Distributed Deep Learning: An In-Depth Concurrency Analysis. 60 pages.

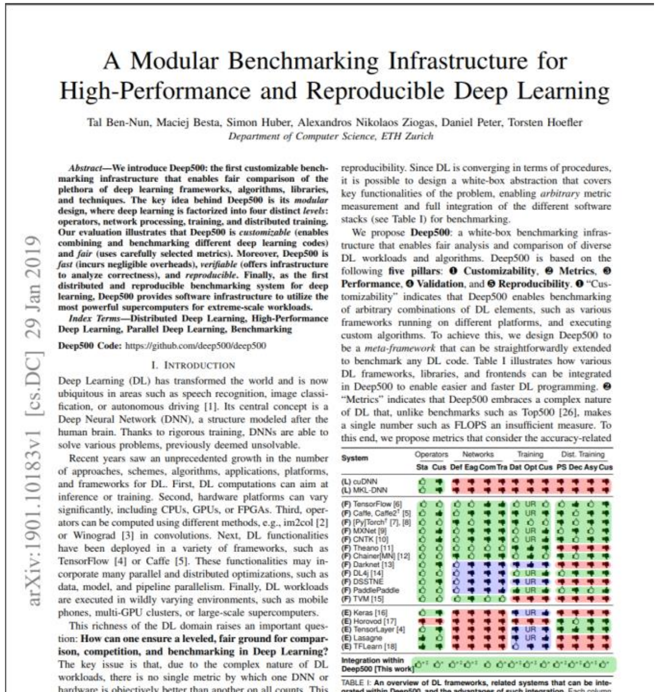
1 INTRODUCTION

Machine Learning, and in particular Deep Learning [LeCun et al. 2015], is a field that is rapidly taking over a variety of aspects in our daily lives. In the core of deep learning lies the Deep Neural Network (DNN), a construct inspired by the interconnected nature of the human brain. Trained properly, the expressiveness of DNNs provides accurate solutions for problems previously thought to be unsolvable, simply by observing large amounts of data. Deep learning has been successfully implemented for a plethora of subjects, ranging from image classification [Huang et al. 2017], through speech recognition [Amodei et al. 2016] and medical diagnosis [Cireşan et al. 2013], to autonomous driving [Bojarski et al. 2016] and defeating human players in complex games [Silver

Next steps – Community!

https://www.deep500.org/
https://www.github.com/deep500/deep500

- **More recipes**
- **More datasets (scientific computing)**
- **Use concepts from HPC to improve ML**
  - Better formats
  - Communication schemes
- **Implement reproducible methods**
- **Metrics and aggregate scores**



arXiv:1901.10183v1 [cs.DC] 29 Jan 2019

A Modular Benchmarking Infrastructure for High-Performance and Reproducible Deep Learning

Tal Ben-Nun, Maciej Besta, Simon Huber, Alexandros Nikolaos Ziogas, Daniel Peter, Torsten Hoefler
Department of Computer Science, ETH Zurich

***Abstract*—We introduce Deep500: the first customizable benchmarking infrastructure that enables fair comparison of the plethora of deep learning frameworks, algorithms, libraries, and techniques. The key idea behind Deep500 is its *modular* design, where deep learning is factorized into four distinct *levels*: operators, network processing, training, and distributed training. Our evaluation illustrates that Deep500 is *customizable* (enables combining and benchmarking different deep learning codes) and *fair* (uses carefully selected metrics). Moreover, Deep500 is *fast* (incurs negligible overheads), *verifiable* (offers infrastructure to analyze correctness), and *reproducible*. Finally, as the first distributed and reproducible benchmarking system for deep learning, Deep500 provides software infrastructure to utilize the most powerful supercomputers for extreme-scale workloads.**

***Index Terms*—Distributed Deep Learning, High-Performance Deep Learning, Parallel Deep Learning, Benchmarking**

Deep500 Code: https://github.com/deep500/deep500

I. Introduction

Deep Learning (DL) has transformed the world and is now ubiquitous in areas such as speech recognition, image classification, or autonomous driving [1]. Its central concept is a Deep Neural Network (DNN), a structure modeled after the human brain. Thanks to rigorous training, DNNs are able to solve various problems, previously deemed unsolvable.

Recent years saw an unprecedented growth in the number of approaches, schemes, algorithms, applications, platforms, and frameworks for DL. First, DL computations can aim at inference or training. Second, hardware platforms can vary significantly, including CPUs, GPUs, or FPGAs. Third, operators can be computed using different methods, e.g., im2col [2] or Winograd [3] in convolutions. Next, DL functionalities have been deployed in a variety of frameworks, such as TensorFlow [4] or Caffe [5]. These functionalities may incorporate many parallel and distributed optimizations, such as data, model, and pipeline parallelism. Finally, DL workloads are executed in wildly varying environments, such as mobile phones, multi-GPU clusters, or large-scale supercomputers.

This richness of the DL domain raises an important question: **How can one ensure a leveled, fair ground for comparison, competition, and benchmarking in Deep Learning?** The key issue is that, due to the complex nature of DL workloads, there is no single metric by which one DNN or hardware is objectively better than another on all counts. This

reproducibility. Since DL is converging in terms of procedures, it is possible to design a white-box abstraction that covers key functionalities of the problem, enabling *arbitrary* metric measurement and full integration of the different software stacks (see Table I) for benchmarking.

We propose **Deep500**: a white-box benchmarking infrastructure that enables fair analysis and comparison of diverse DL workloads and algorithms. Deep500 is based on the following **five pillars**: ❶ **Customizability**, ❷ **Metrics**, ❸ **Performance**, ❹ **Validation**, and ❺ **Reproducibility**. ❶ "Customizability" indicates that Deep500 enables benchmarking of arbitrary combinations of DL elements, such as various frameworks running on different platforms, and executing custom algorithms. To achieve this, we design Deep500 to be a *meta-framework* that can be straightforwardly extended to benchmark any DL code. Table I illustrates how various DL frameworks, libraries, and frontends can be integrated in Deep500 to enable easier and faster DL programming. ❷ "Metrics" indicates that Deep500 embraces a complex nature of DL that, unlike benchmarks such as Top500 [26], makes a single number such as FLOPS an insufficient measure. To this end, we propose metrics that consider the accuracy-related

TABLE I: An overview of DL frameworks, related systems that can be integrated within Deep500, and the advantages of such integration.

https://arxiv.org/abs/1901.10183