DISTRIBUTED DEEP LEARNING: CHALLENGES & OPPORTUNITIES

Peter Labus, Ph.D.

Competence Center for High Performance Computing, Fraunhofer ITWM, Kaiserslautern.

Fraunhofer Center Machine Learning.

ISC 2020

MLHPCS workshop

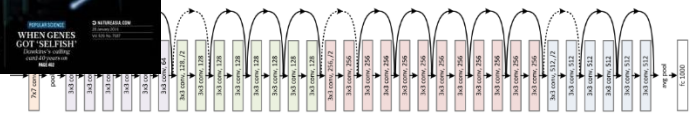
June 25, 2020

Achievements in AI need exponentially growing computing power

Achievements in AI…

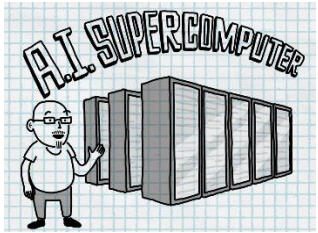
APRIL 15, 2019 • 7 MINUTE READ

OpenAI Five Defeats Dota 2 World Champions



OpenAI forms exclusive computing partnership with Microsoft to build new Azure AI supercomputing technologies

July 22, 2019 | Microsoft News Center

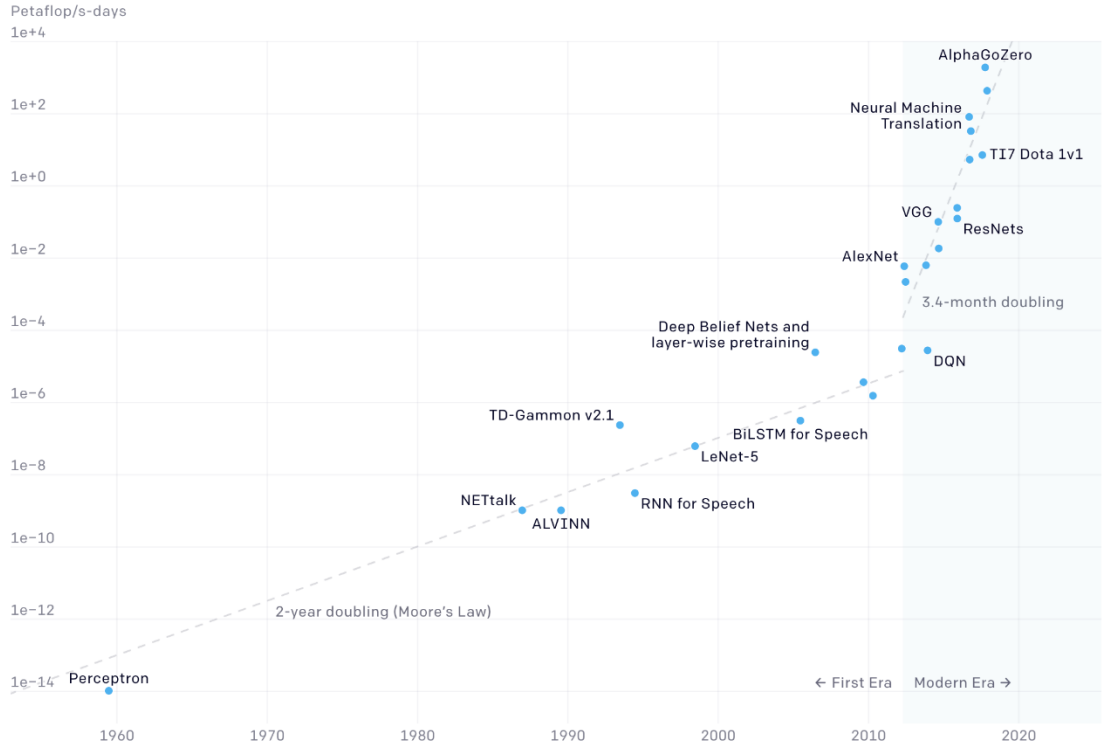


JULY 22, 2019 • 2 MINUTE READ

Microsoft Invests In and Partners with OpenAI to Support Us Building Beneficial AGI

Microsoft is investing $1 billion in OpenAI to support us building artificial general intelligence (AGI) with widely distributed economic benefits. We're

…need exponentially growing computing power

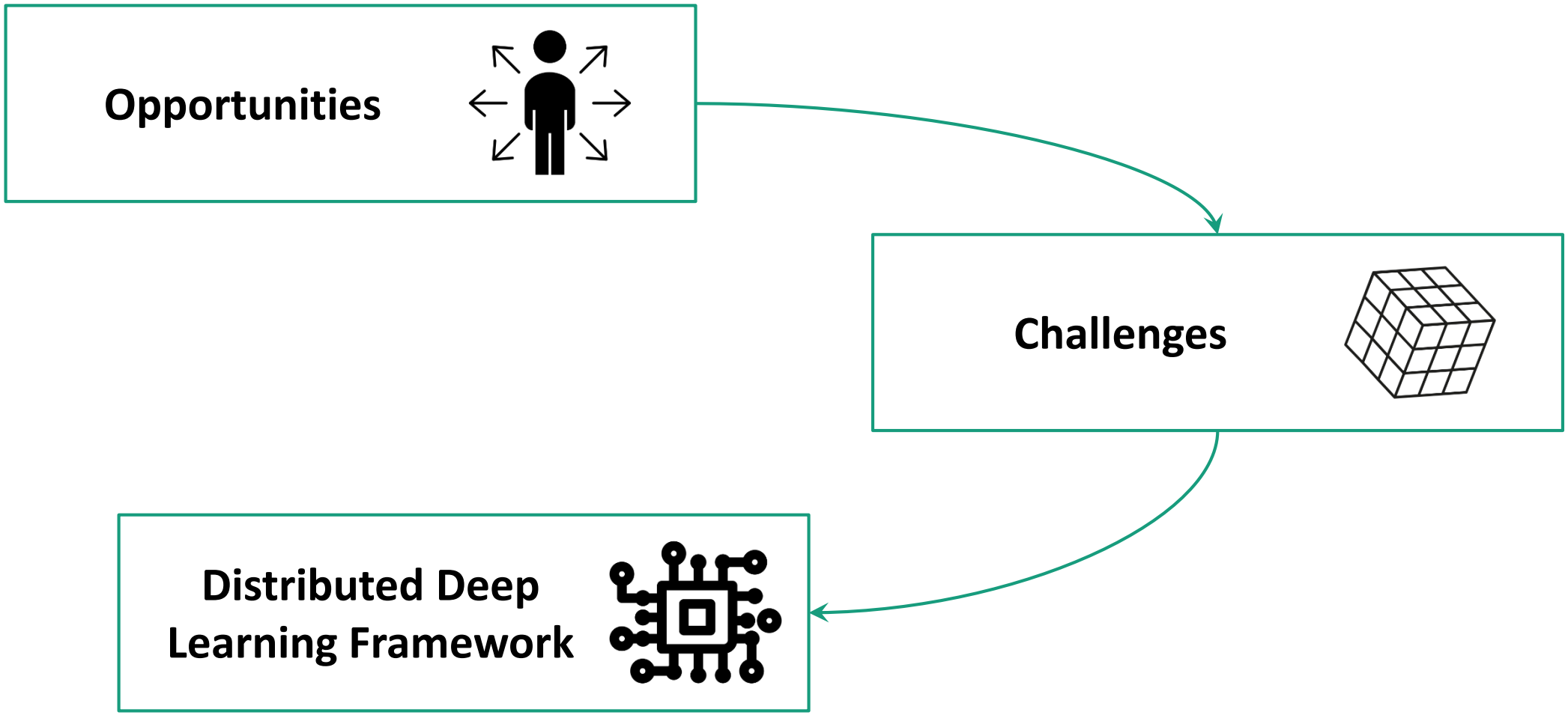


"Moore's Law of AI": FLOP/s-days double every 3.5 months

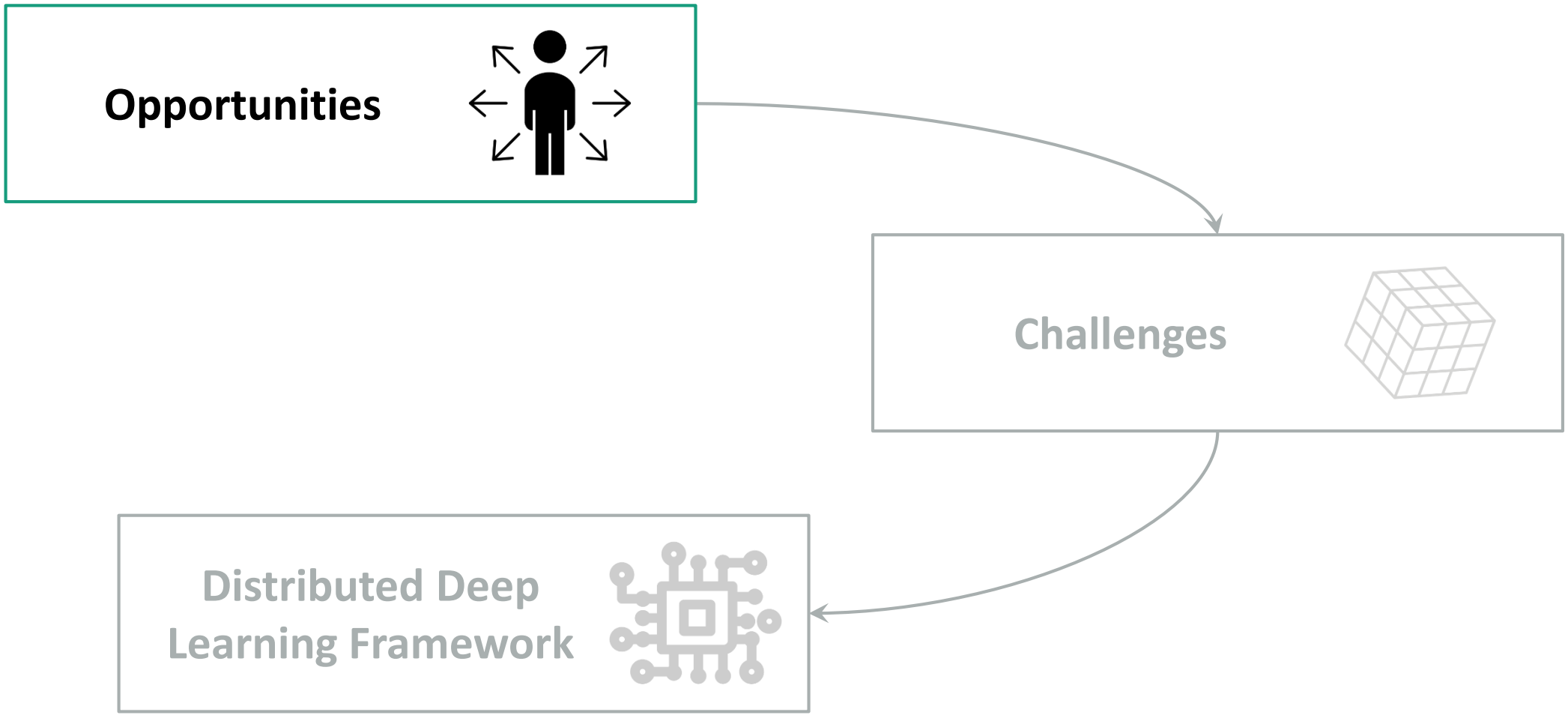
[https://openai.com/blog/ai-and-compute/]

Fraunhofer

Overview: Distributed Deep Learning



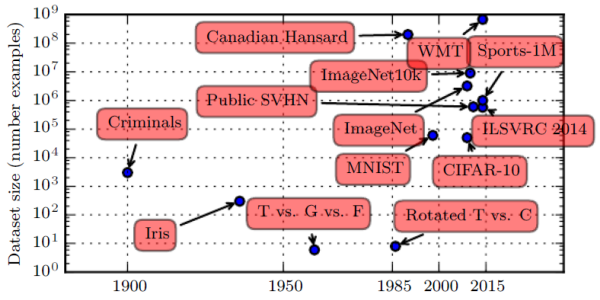
Overview: Distributed Deep Learning



HPC can enable…

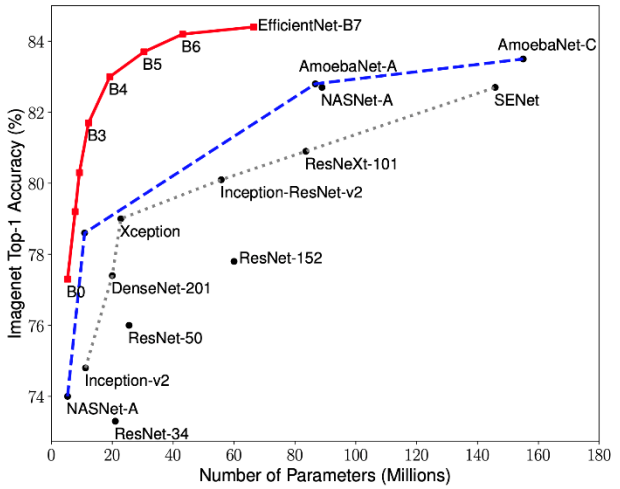
Larger Datasets

- labeled datasets grow in size
  - more samples
  - larger samples
- shift towards unlabeled/generated datasets
  - reinforcement learning
  - self-supervised learning



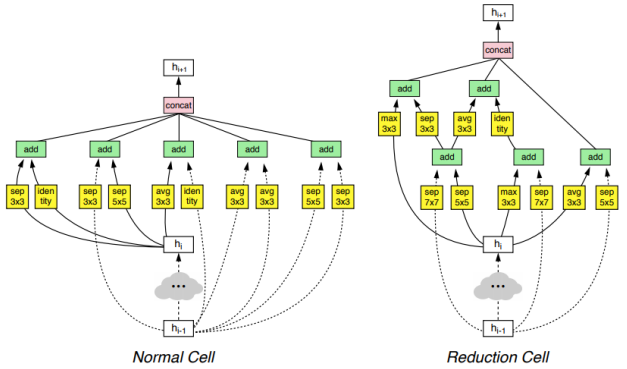
Larger Models

- learn complex tasks & improve accuracy
- may not fit on single device
- often take weeks to train



Meta-Learning

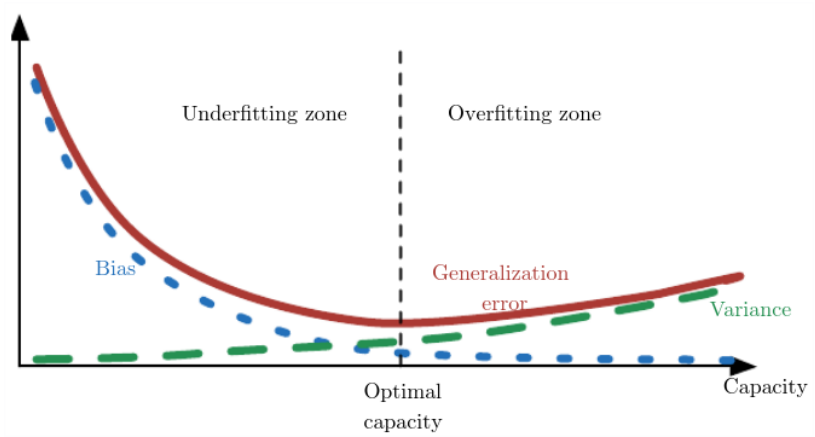
- train several DNNs in parallel
  - hyperparameter optimization
  - ensembles
  - Neural Architecture Search



Fraunhofer

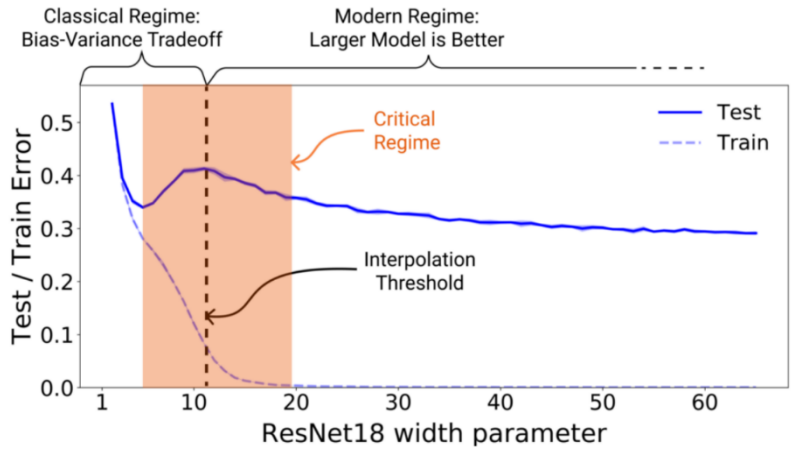
Larger models & datasets lead to better accuracy

Classical picture: bias-variance trade-off



Balance between over- and underfitting

Modern picture: deep double descent



Bigger models and more data increase accuracy without overfitting

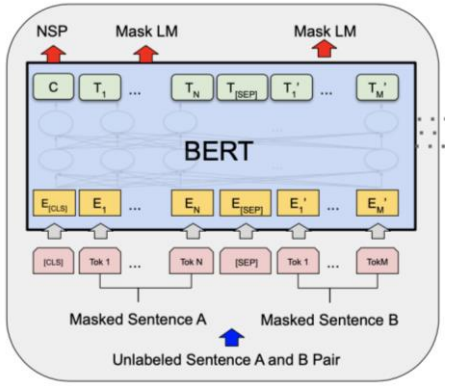
The growth of models & datasets is here to stay!

[Deep Learning, Goodfellow, Bengio, Courville]
[Deep Double Descent: Where Bigger Models and More Data Hurt, Nakkiran et al.]

Fraunhofer

HPC can help sharing models, datasets & tools to foster progress in Deep Learning

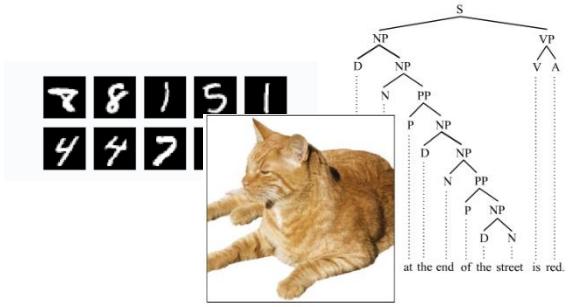
Models



- often SOTA models cannot be trained on workstations
- *solution:* train on HPC systems / share / fine-tune on downstream task
- ***Do not waste compute resources!***

Datasets

MNIST, Penn Treebank, ImageNet, … changed the DL landscape



We need more of them!

- How to generate?
- How to provide / host?

Tools

High quality Deep Learning software available: communication libraries, DNN operator kernel libraries, frameworks, …

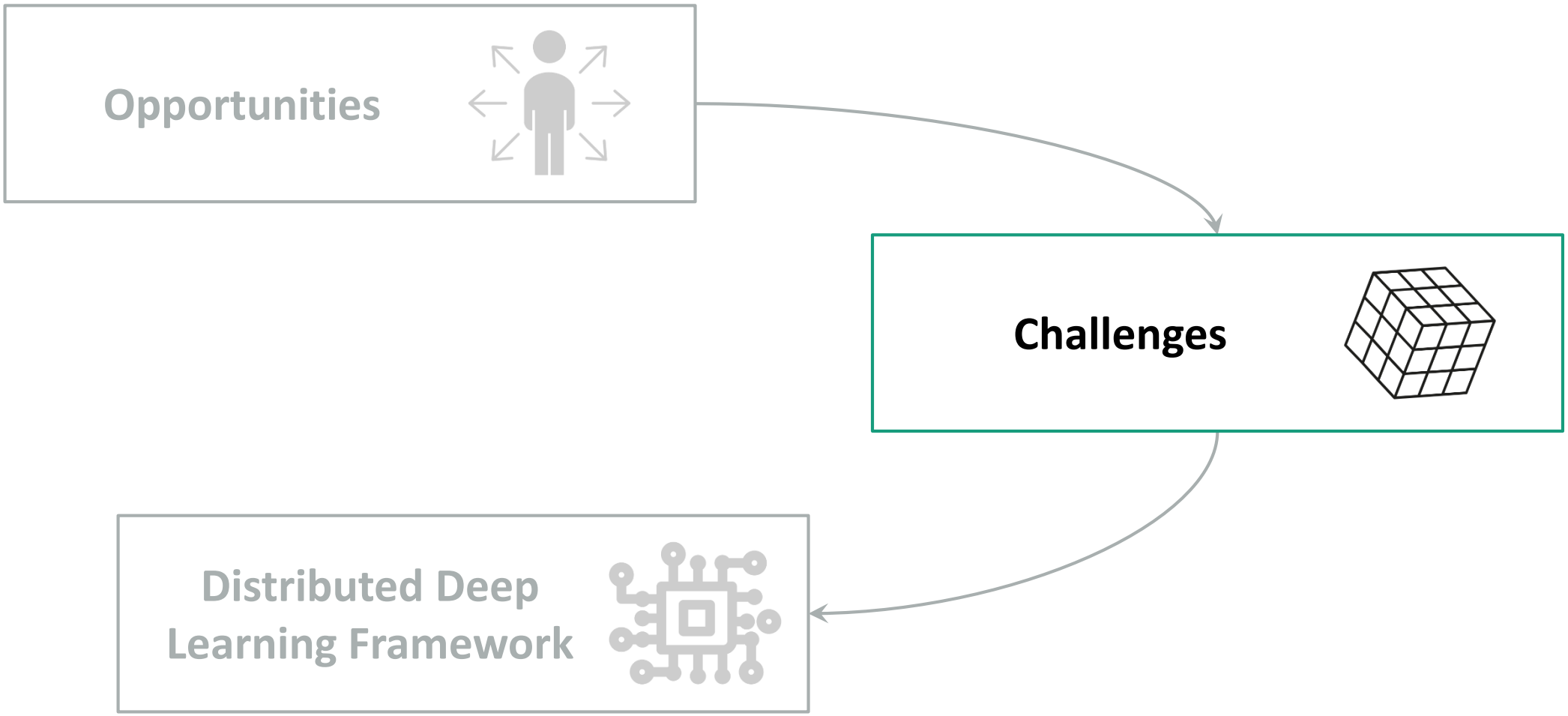
ONNX mxnet oneAPI cuDNN

- build on / support existing tools
- **provide hardware- / vendor-independent solutions**

Foster Deep Learning Research: towards reproducible, accessible & energy-efficient AI

Fraunhofer

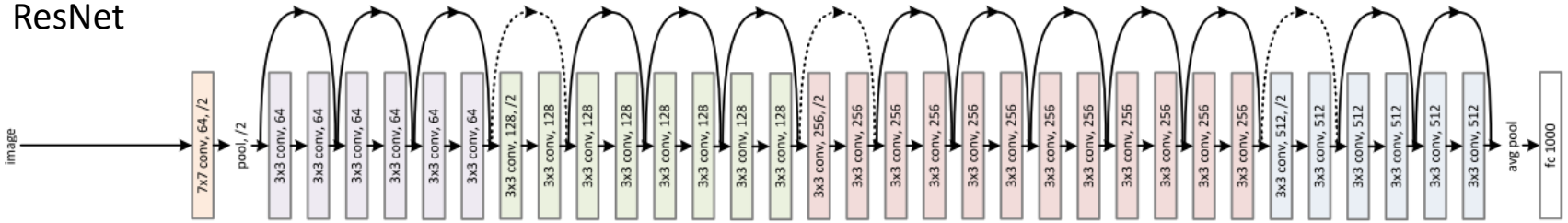
Overview: Distributed Deep Learning



DNN training is inherently sequential & iterative

DNN training

ResNet



- (mostly) linear chains of *dependent* operations
- backward pass needs to be executed after forward pass
- weight optimization via *iterative* algorithms

sequential

&

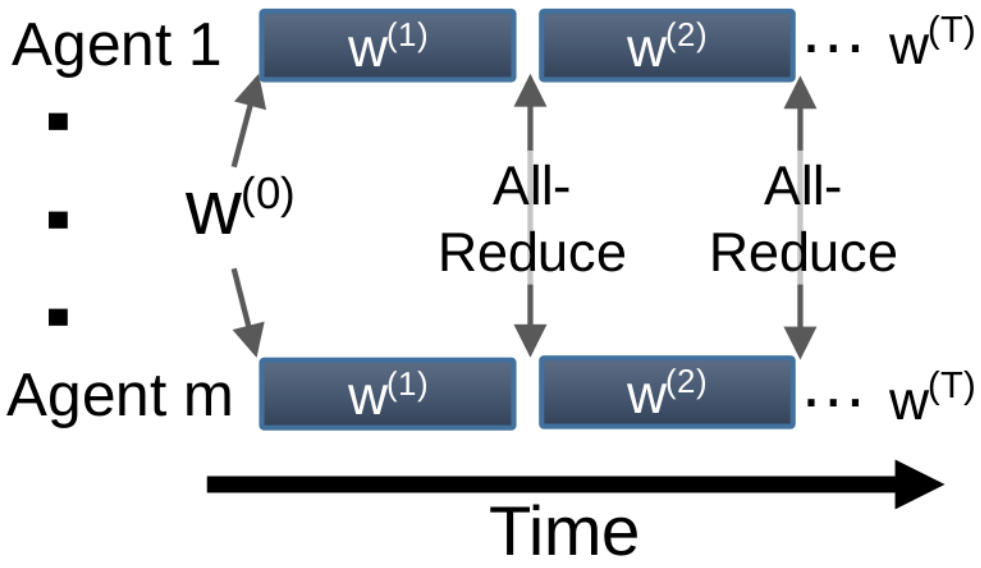
iterative

- repeated synchronization is unavoidable
- many levels of parallelism need to be exploited for good performance
- simplest parallelization strategy: **data parallelism**

Fraunhofer

Even data parallelism faces several challenges

Distributed (synchronous) data parallelism



available implementations:

Horovod, native support in *TensorFlow* & *pyTorch*

1. replicate model on all agents with same initial weights
2. process one micro-batch for each agent & iteration
3. average gradients & redistribute to all agents (*allreduce*)

Challenges:

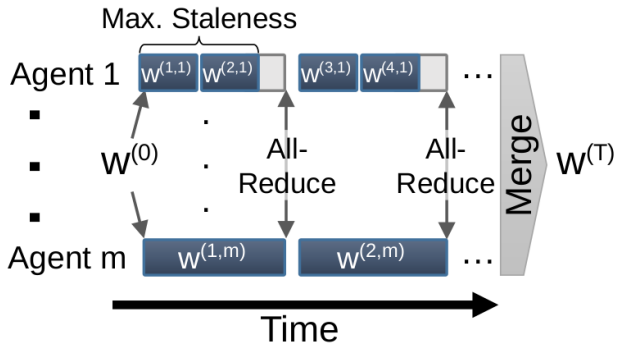
- introduces synchronization points
- needs (very) large batch sizes to scale well
- How to hide *allreduce* communication behind backpropagation calculations?

pictures taken from: [Demystifying parallel and distributed deep learning: An in-depth concurrency analysis, Ben-Nun et al.]
[Horovod: fast and easy distributed Deep Learning in TensorFlow, Sergeev et al.]

Going beyond the synchronous optimizers may reduce statistical efficiency

beyond synchronous optimizers

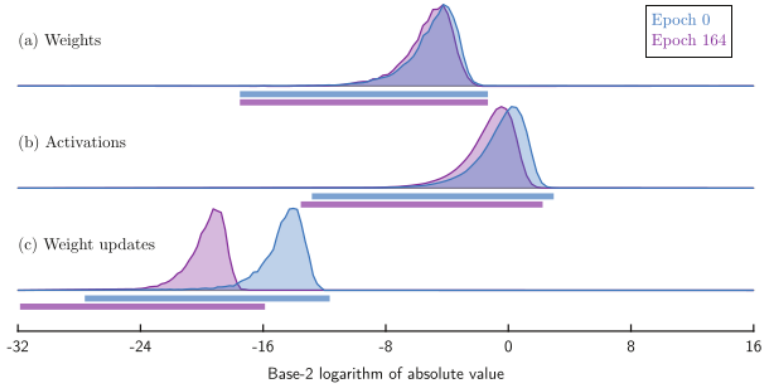
asynchronous optimizers



- **Hogwild!**
- **stale SGD**

reduce communication frequency

sparsity & compression



- **quantization**
- **top-*k* sparsification**

reduce communication volume

alters the model & may reduce statistical efficiency!

pictures taken from: [Demystifying parallel and distributed deep learning: An in-depth concurrency analysis, Ben-Nun et al.]
[Hogwild: A Lock-Free Approach to Parallelizing Stochastic Gradient Descent, Recht et al.]
[SparCML: High-performance sparse communication for machine learning, Renggli et al.]

Fraunhofer

Time to accuracy is influenced by the time per epoch & the statistical efficiency

Time to accuracy:

We want to optimize the training time to solution for a **target test accuracy**

Time per epoch

- How much time to process all samples of the input dataset once?
- ***focus today***

Statistical efficiency

- How many epochs do we need until convergence?
- *will change, when changing the optimization algorithm*
- **keep constant through the use of *synchronous optimizers***

MLPerf

Benchmark	Dataset	Quality Target	Reference Implementation Model
Image classification	ImageNet (224x224)	75.9% Top-1 Accuracy	Resnet-50 v1.5
Object detection (light weight)	COCO 2017	23% mAP	SSD-ResNet34
Object detection (heavy weight)	COCO 2017	0.377 Box min AP, 0.339 Mask min AP	Mask R-CNN
Translation (recurrent)	WMT English-German	24.0 BLEU	GNMT
Translation (non-recurrent)	WMT English-German	25.0 BLEU	Transformer
Recommendation	Undergoing modification		
Reinforcement learning	N/A	Pre-trained checkpoint	Mini Go

- many task domains, datasets & model architectures
- set quality targets
- reference implementations in *TensorFlow* & *pyTorch*
- **optimize *time to accuracy***

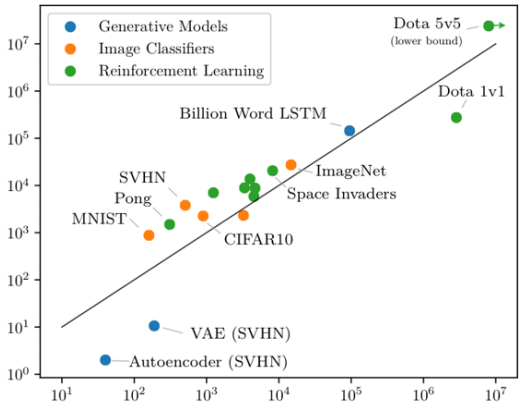
[MLPerf training benchmark, Mattson et al.]

Fraunhofer

Data parallelism suffers from two major shortcomings

Large batch size problem

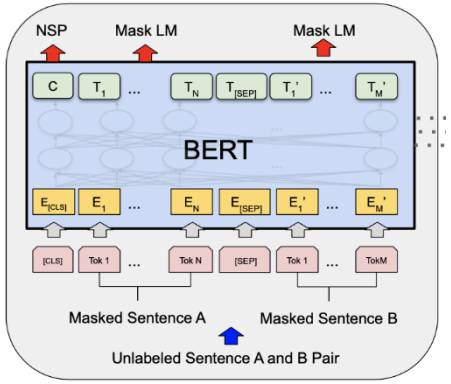
- **reduced test accuracy**
  can be avoided with hyperparameter tuning & appropriate learning rate schedules
- **critical batch size**
  batch size beyond which there is no linear scaling in terms of iterations anymore

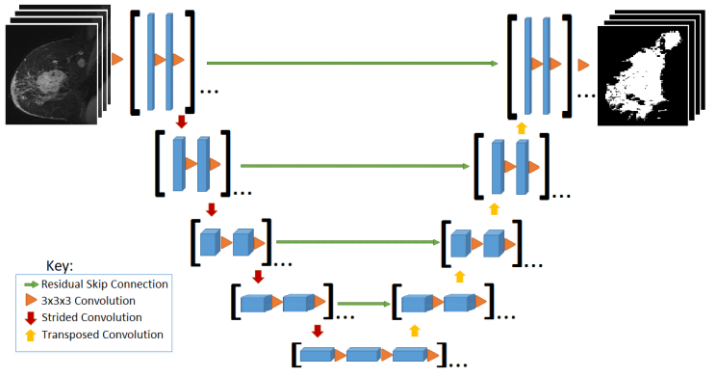


critical batch size grows with dataset complexity!

Model size

- **model must fit on single device**
- **limiting for models with large inputs or for complex tasks**





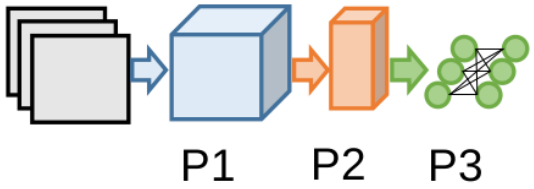
[Measuring the Effects of Data Parallelism on Neural Network Training, Shallue et al.]
[An Empirical Model of Large-Batch Training, McCandlish et al.]

Fraunhofer

Model parallelism can eliminate the shortcomings of data parallelism

model parallelism

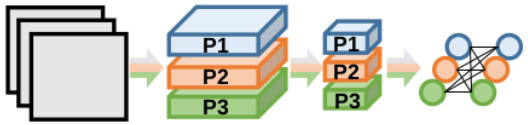
pipelining



- split DNN into partitions
- communicate activations on boundaries
- existing approaches:

  GPipe, Pipedream, HyPar-Flow

layer parallelism



- split individual layers
- might introduce more complex communication patters (convolutions, BatchNorm, …)
- existing approaches:

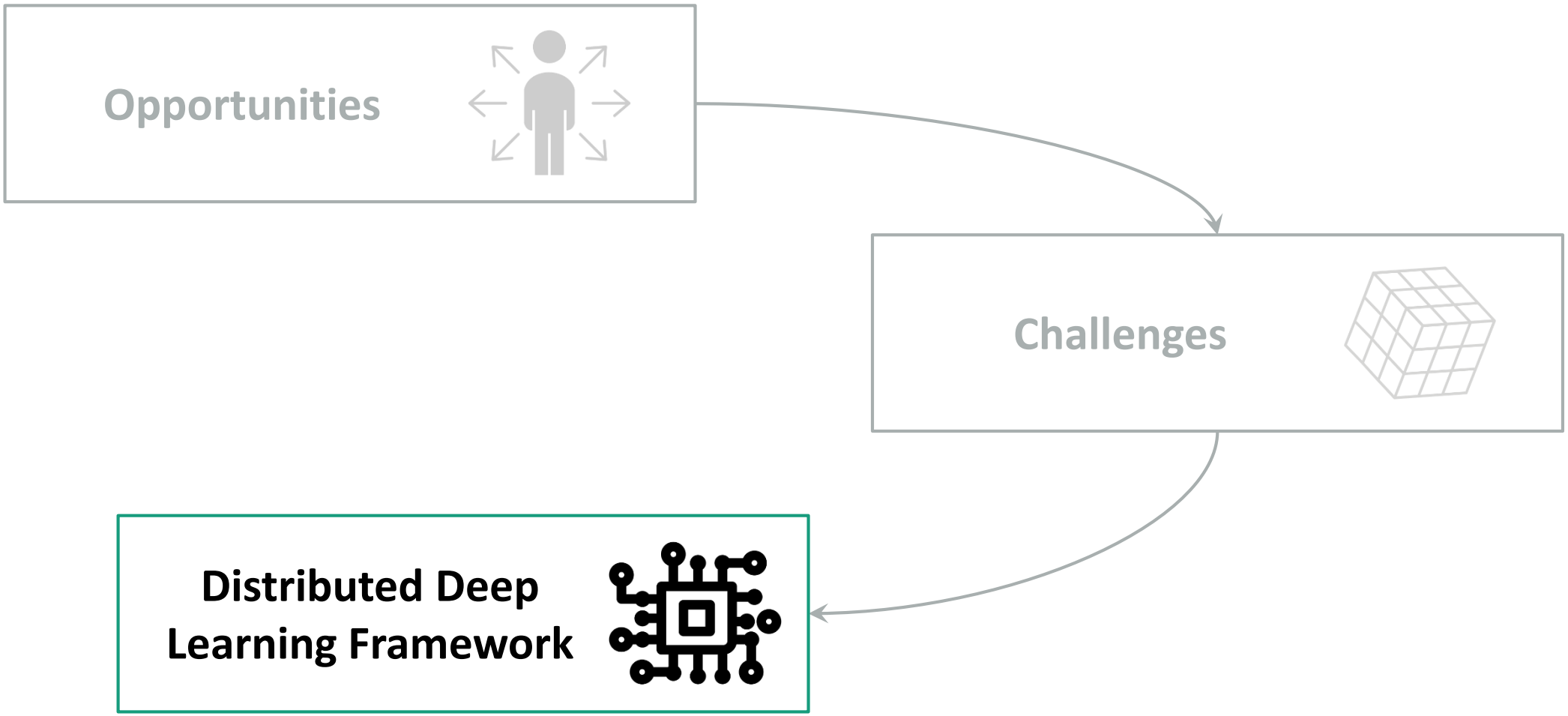
  Mesh-TensorFlow, TF-Replicator, LBANN

eliminate shortcomings of data parallelism

- train models of any size
- exploit additional levels of parallelism

pictures taken from: [Demystifying parallel and distributed deep learning: An in-depth concurrency analysis, Ben-Nun et al.]

Overview: Distributed Deep Learning



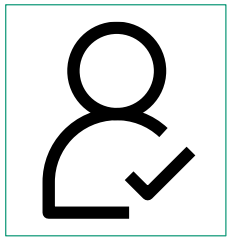
Our distributed Deep Learning framework *Tarantella* has three goals

distributed Deep Learning with *Tarantella*

three goals

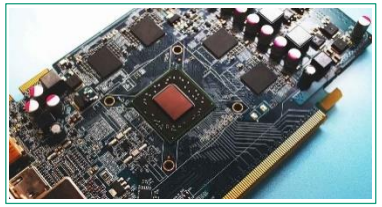
High usability without HPC expertise

- high-level user interface
- integrate well with existing tools (*TensorFlow 2*)



Deep Learning without memory limits

- automatic use of pipelining & layer-parallelism



Good scalability on many systems

- leverage highly optimized data parallel implementation based on GASPI
- vendor-independent solution

SPONSORED BY THE
Federal Ministry of Education and Research

ZIH
Center for Information Services & High Performance Computing

UNIVERSITÄT HEIDELBERG
ZUKUNFT SEIT 1386

Fraunhofer

Tarantella is based on GASPI

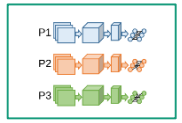
Global Address Space Programming Interface GASPI

Communication Library Zoo

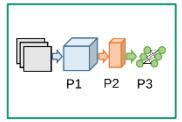
MPI NCCL BlueConnect

Blink oneCCL Gloo Aluminum

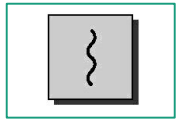
Tarantella



- overlap several *allreduces* with back-propagating gradients

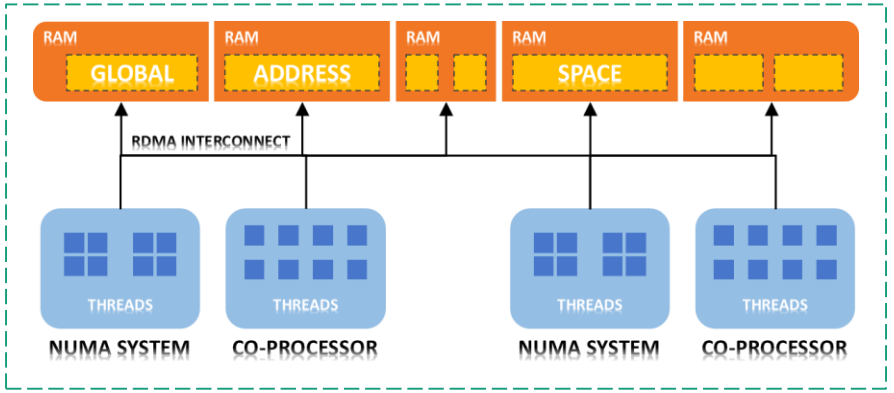


- non-blocking point-to-point communication



- dedicated thread to trigger progress in background

GASPI/GPI

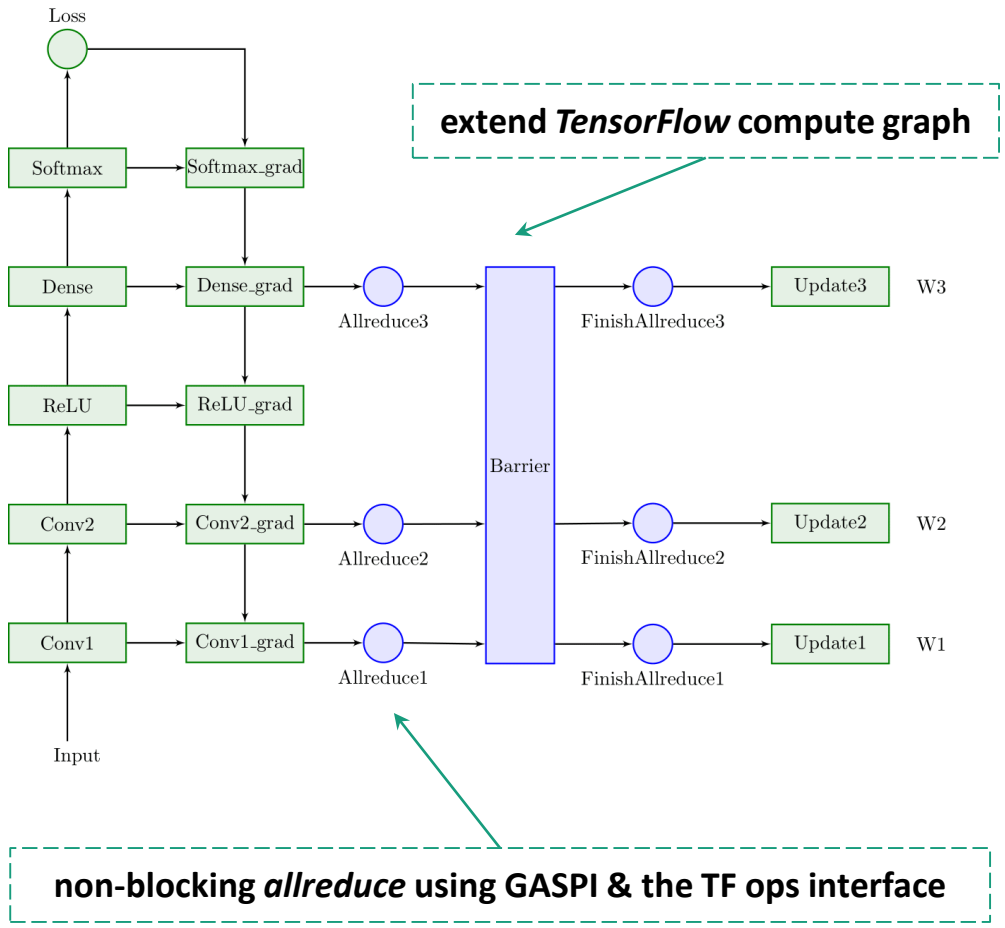


- matured standard, developed since 2005 at Fraunhofer ITWM
- natively one-sided, non-blocking & asynchronous communication using RDMA
- ideal API to overlap computation & communication in training DNNs

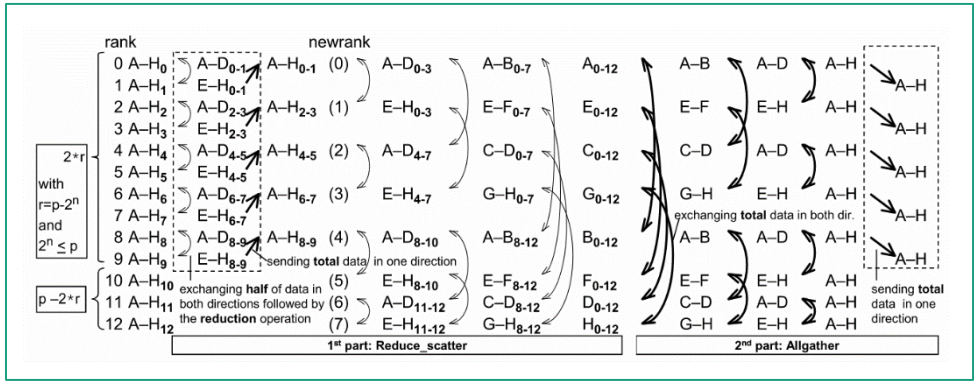
Fraunhofer

Tarantella's data parallelism overlaps *allreduces* with backpropagation

Tarantella's data parallelism



allreduce



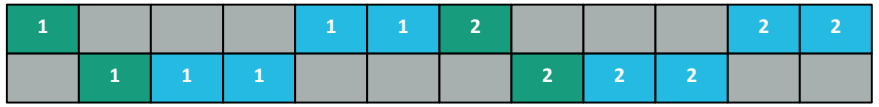
- *reduce scatter* & *allgather*
- bandwidth efficient algorithm for large message sizes
- interleave iterations of multiple *allreduces*
- dedicated communication thread triggers progress in the background

[Optimization of Collective Communication Operations in MPICH, R. Thakur et al.]

Fraunhofer

Tarantella's pipelining improves existing approaches

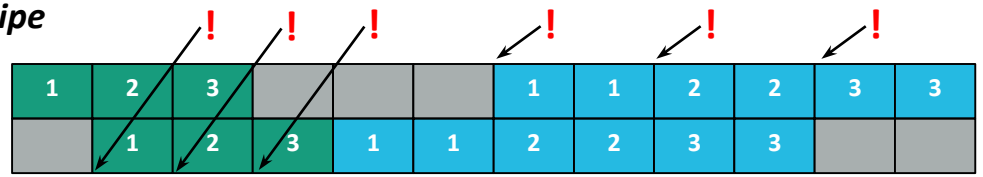
naïve model parallelism



- 2 partitions
- process 1 mini-batch at a time
- not performant ✗
- synchronous scheme ✓

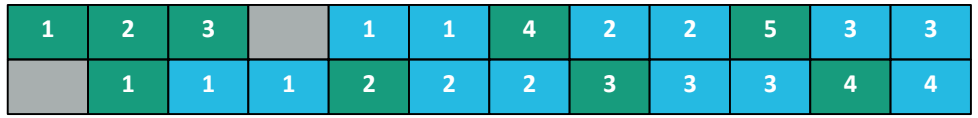
existing pipelining approaches

GPipe



- sub-optimal overlap ✗
- synchronous ✓

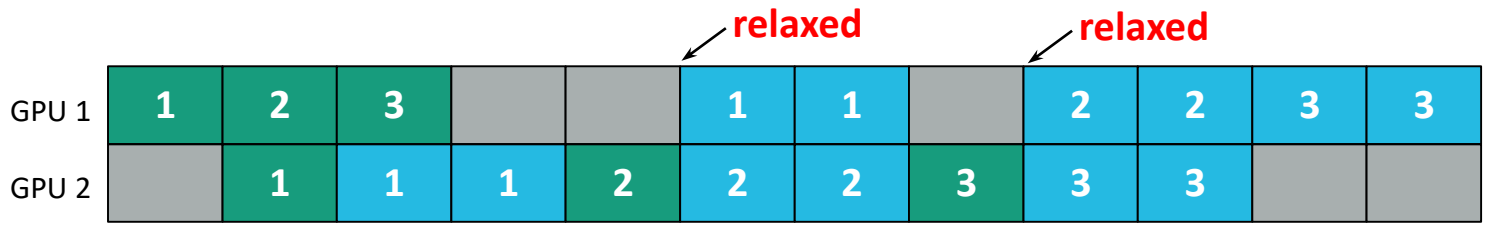
PipeDream



- full pipeline ✓
- stale weights ✗

improve

Tarantella's pipelining



- good overlap ✓
- synchronous scheme ✓
- support full *Keras* interface ✓

[GPipe: Efficient training of giant neural networks using pipeline parallelism, Huang et al.]
[PipeDream: generalized pipeline parallelism for DNN training, Narayanan et al.]
[HyPar-Flow: Exploiting MPI and Keras for Scalable Hybrid-Parallel DNN Training using TensorFlow, Awan et al.]

Tarantella integrates well into existing *TensorFlow 2 / Keras* models

Keras model

```
import tensorflow as tf

# Create Keras model
model = tf.keras.Model(resnet56.get_model())

# Define optimizer with learning rate
sgd = tf.keras.optimizers.SGD(learning_rate=base_learning_rate)

# Build Keras model
model.compile(optimizer = sgd,
              loss = 'categorical_crossentropy',
              metrics = (['categorical_accuracy']))

# Load input data in mini-batches
train_dataset = tf.data.FixedLengthRecordDataset(filenames_train)
train_dataset = train_dataset.shuffle().repeat().batch(batch_size)
val_dataset = tf.data.FixedLengthRecordDataset(filenames_validation)

# Perform synchronous training
model.fit(train_dataset, nepochs, val_dataset)
```

execute *Tarantella* with

```
tarantella_run -np 8 -npernode 4 -m machinefile --no-gpus
./models/resnet50.py --batch-size=1024 -e 100
```

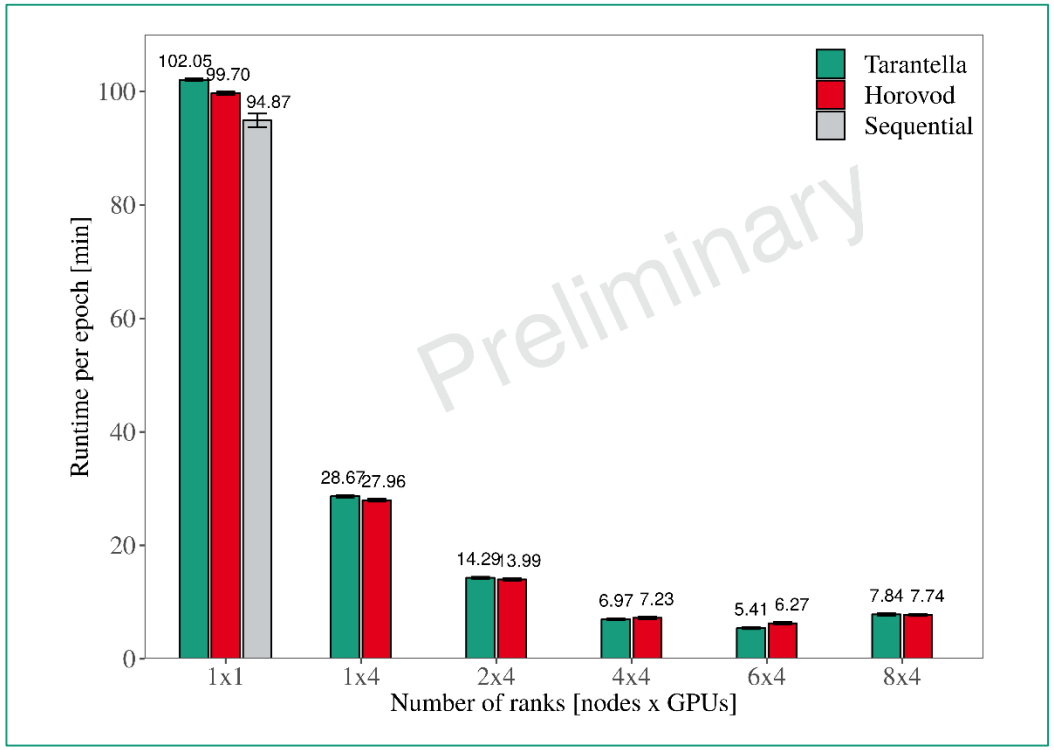
Tarantella model

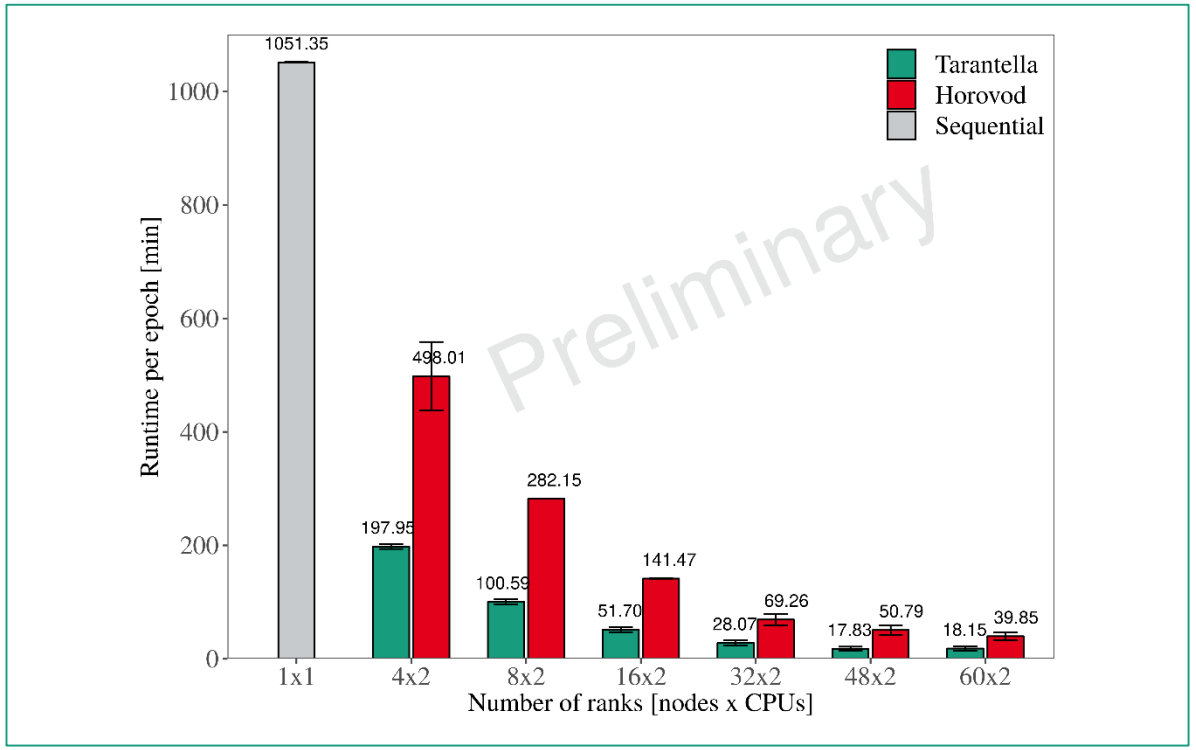
```
import tensorflow as tf

# Step (1)
# Initialize the framework
import tarantella as tnt

# Create Keras model
model = tf.keras.Model(resnet56.get_model())

# Step (2)
# Wrap model
model = tnt.TarantellaModel(model)

# Define optimizer with appropriate learning rate for large batch sizes
sgd = tf.keras.optimizers.SGD(learning_rate=base_learning_rate)

# Build Keras model
model.compile(optimizer = sgd,
              loss  ='categorical_crossentropy',
              metrics = (['categorical_accuracy']))

# Load input data distributed in micro-batches
train_dataset = tf.data.FixedLengthRecordDataset(filenames_train)
train_dataset = train_dataset.shuffle().repeat().batch(batch_size)
val_dataset = tf.data.FixedLengthRecordDataset(filenames_validation)

# Perform distributed synchronous training
model.fit(train_dataset, nepochs, val_dataset)
```

- advanced interface for power-users available
- automatic distribution of datasets

Preliminary benchmarks…

- **image classification: ResNet50 on ImageNet**
- TensorFlow 2.1, Horovod 19.4 (with Infiniband)
- 5 epochs, 3 repetitions with different random seeds, micro-batch size 64 (GPU) / 256 (CPU)



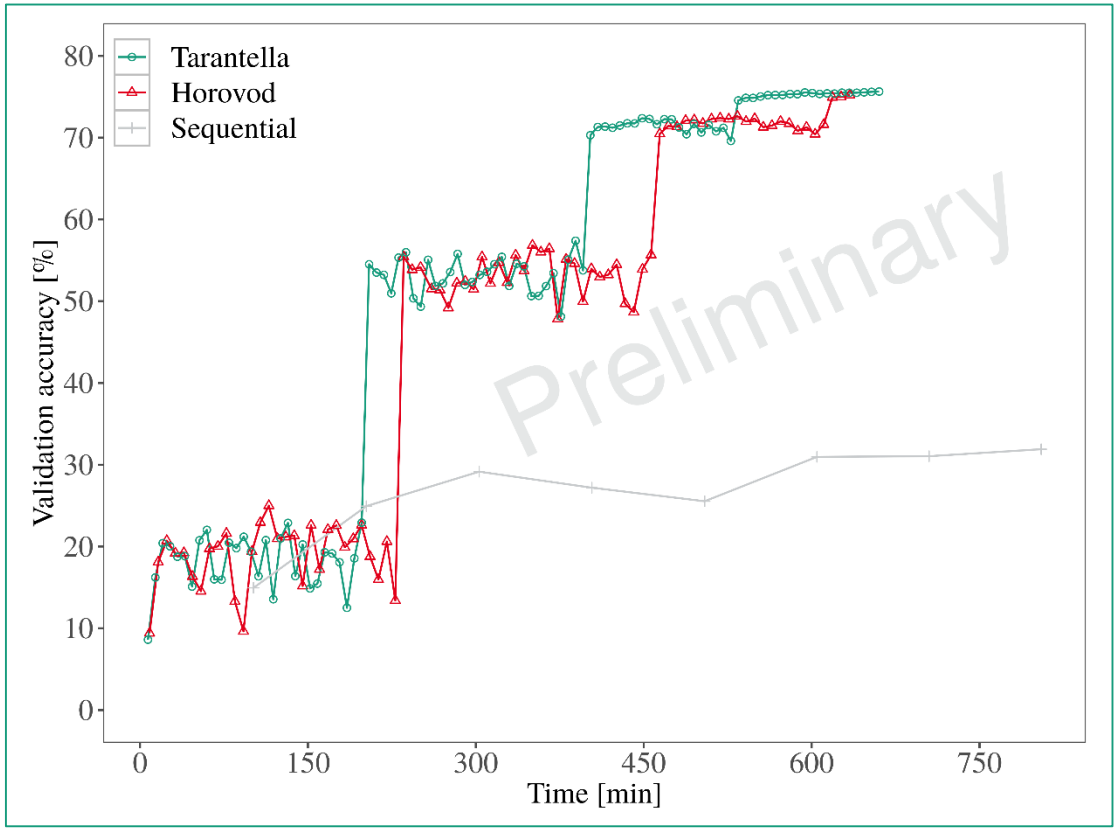


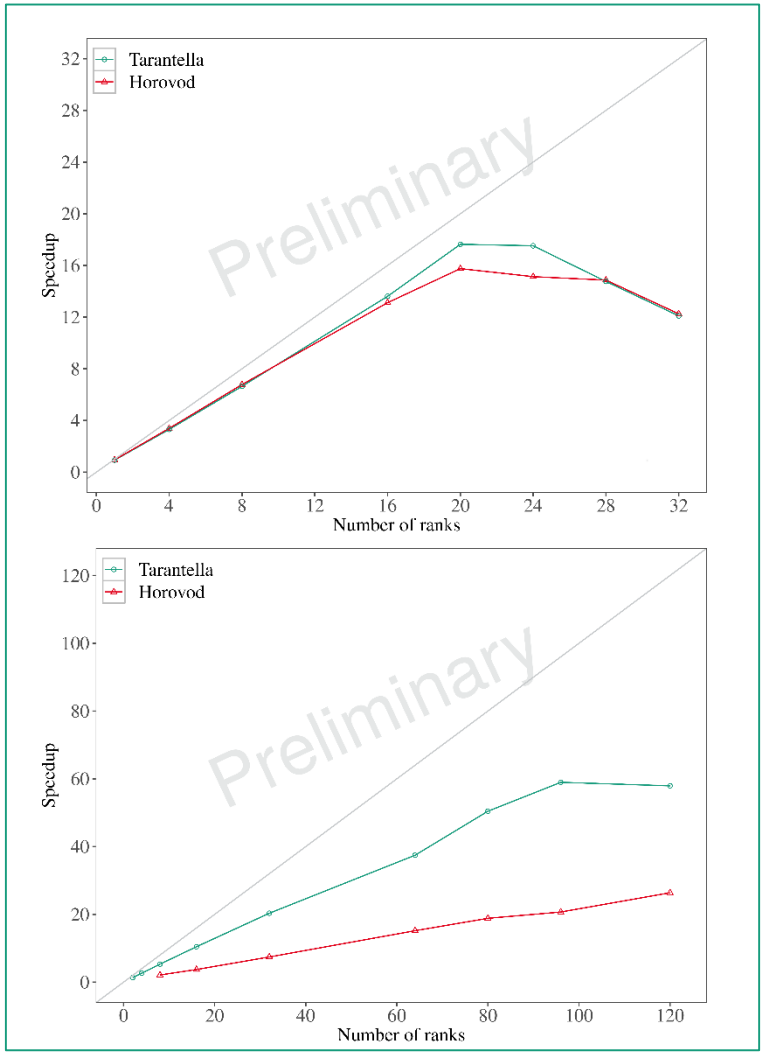
- NVidia Titan V (12 GB memory)
- 2 x Intel® Xeon® Silver 4108 CPU @ 1.80GHz (16 cores)
- 2 x Mellanox ConnectX-5 100Gbps Infiniband

- 2 x Intel® Xeon® Gold 6148 CPU @ 2.40GHz (20 cores/40 hyperthreads)
- Mellanox ConnectX-5 100Gbps Infiniband

Fraunhofer

…show very promising results





- on par / better than state-of-the-art (*Horovod*)
- good *strong scalability*
- further optimizations to be exploited

Next steps:

- ***MLPerf*** benchmark suite evaluation
- full model parallelism **in active development**

Fraunhofer

The *Tarantella* team

"None of us is as smart as all of us."

--- Ken Blanchard

research scientists

Peter Labus, Ph.D.

Alexandra Carpen-Amarie, Ph.D.

Martin Kuehn, Ph.D.

Master's students

Pengqiu Li

Kavyashree Renukachari

Fraunhofer

Distributed Deep Learning: summary & outlook

- HPC can super-charge the Deep Learning revolution!
- leverage data & model parallelism for large scale deep learning without memory limits
- build on existing solutions for fair & green AI

***Tarantella* open source release date:**

15.11.2020

SC20 Atlanta, GA | more than hpc.

Get in touch!

peter.labus@itwm.fhg.de

linkedin.com/in/PeterLabus/