CoE RAISE

HDCRS Summer School

31.05.2022

Rocco Sedona, Marcel Aach

Jülich Supercomputing Centre – Forschungszentrum Jülich GmbH

University of Iceland

Speaker Introduction

- **Rocco Sedona:** PhD student at Juelich Supercomputing Centre and University of Iceland (Supervisors: Dr. Cavallaro, Prof. Riedel, Prof. Book)
- **Marcel Aach:** PhD student at Juelich Supercomputing Centre and University of Iceland (Supervisors: Prof. Riedel and Dr. Lintermann)

Juelich Supercomputing Centre

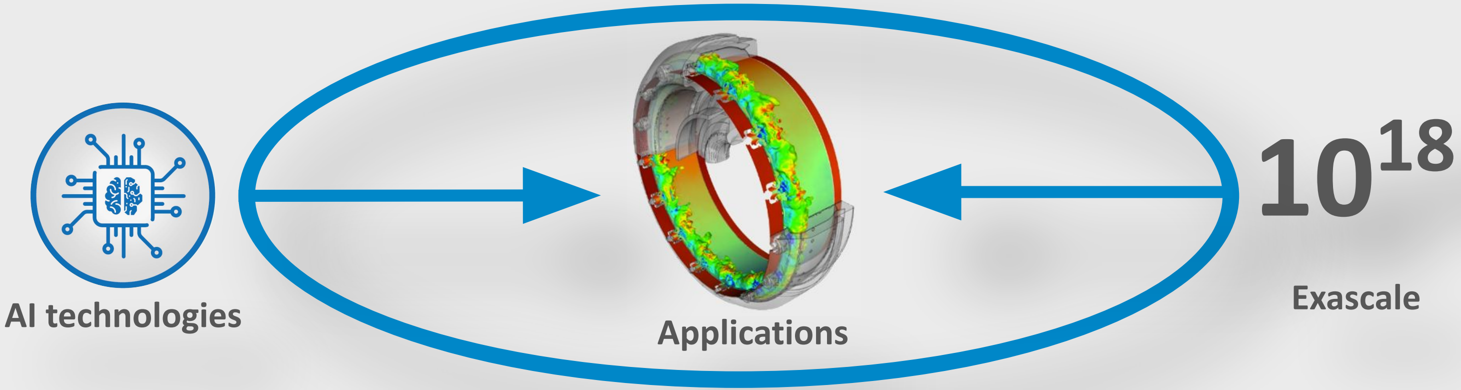
RAISE Center of Excellence

- Supercomputing center as as part of the Juelich Research Centre
- Hosts one of the **most powerful supercomputers** in Europe (JUWELS BOOSTER)
- First **D-Wave Quantum Annealer** in Europe

Motivation

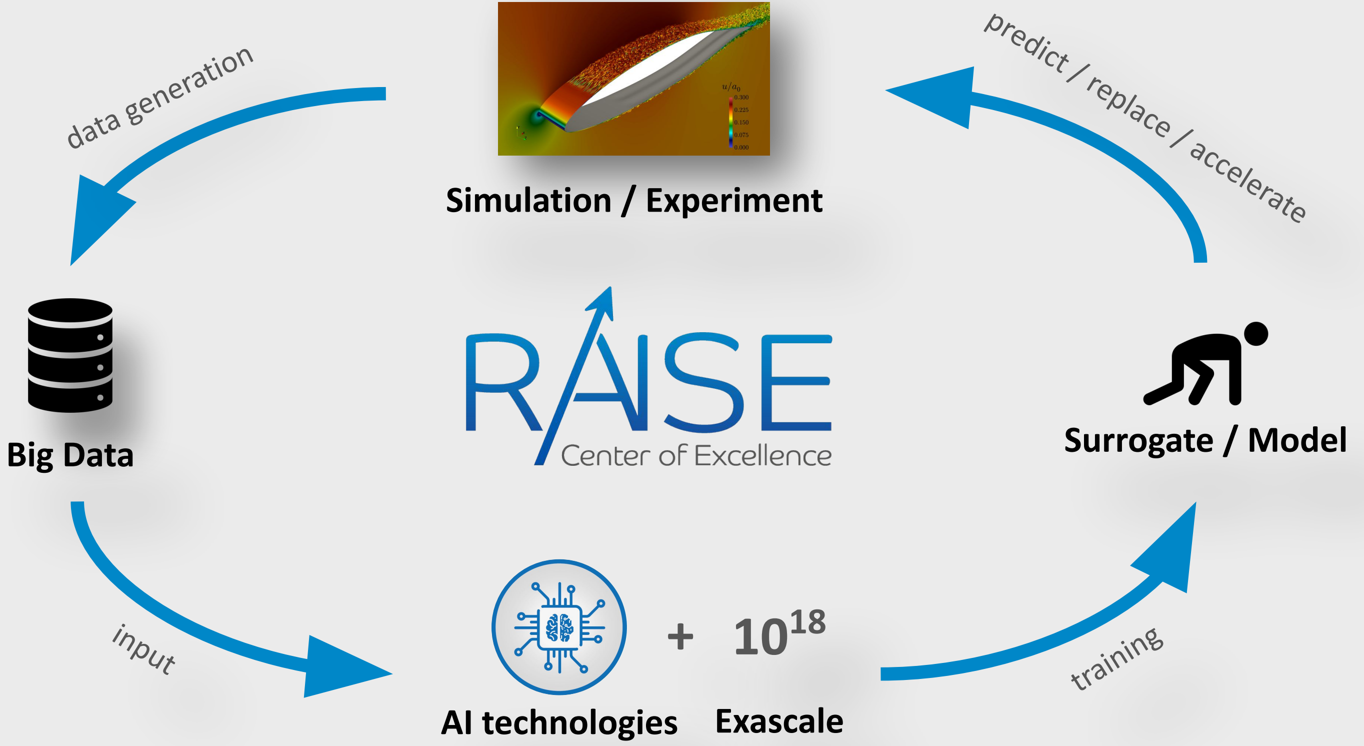
Motivation

- AI technologies are key to
  - extract knowledge from big data collections
  - reason from existing knowledge
  - find hidden features and detect unseen correlations in massively large data sets
- High-Performance Data Analytics (HPDA) requires
  - intelligent analytic tools
  - scalable systems



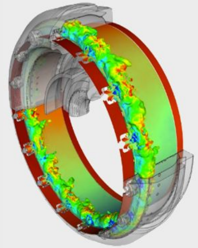
Introduction to CoE RAISE

CoE RAISE: Motivation



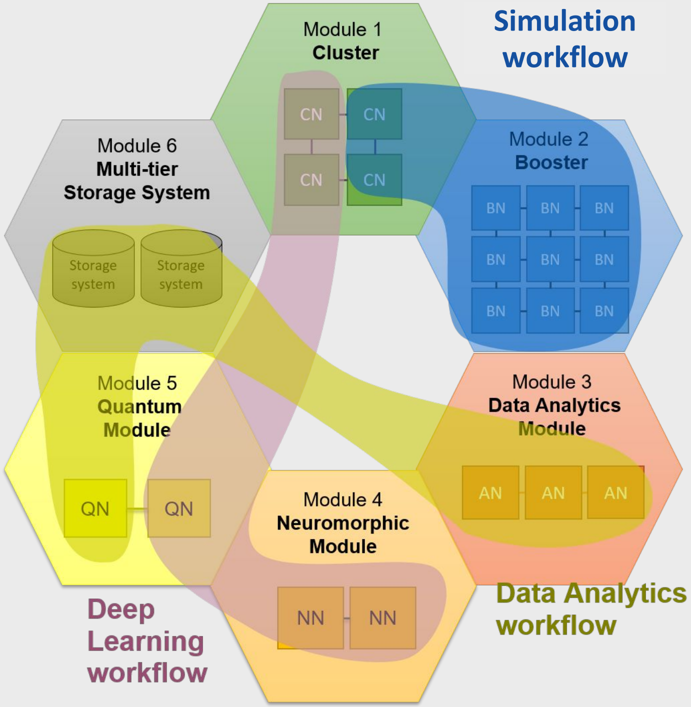
CoE RAISE: Modularity of Next-Generation HPC Systems

Complex hardware



Complex task

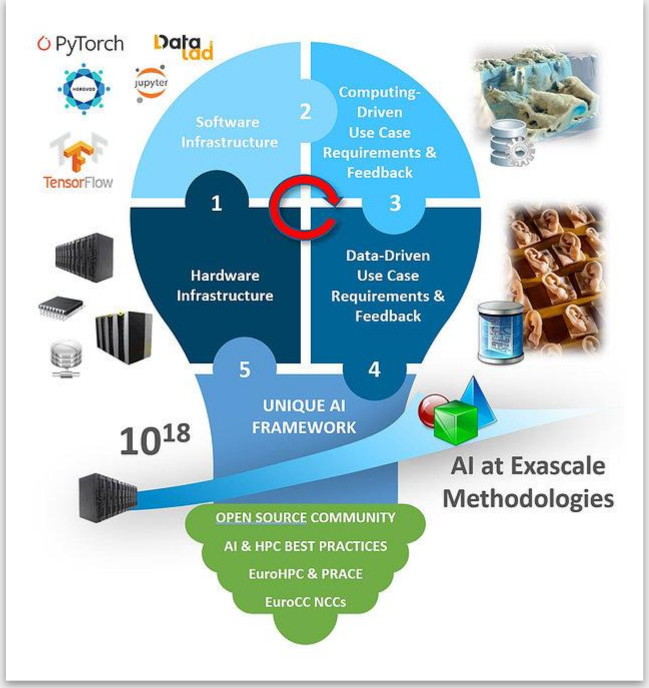
- Find the most suitable hardware for a specific task
- Enable intertwined AI- and HPC-workflows



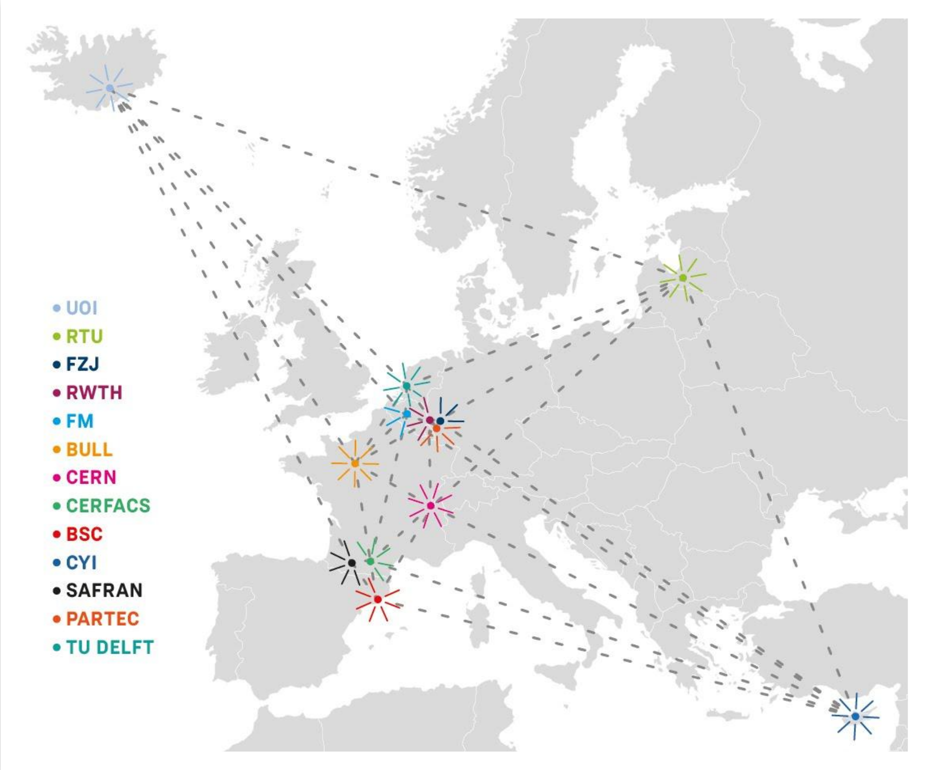
CoE RAISE's Major Objectives

- Development of AI methods towards Exascale
- Connect
  - hardware infrastructure,
  - software infrastructure,
  - compute-driven use cases,
  - and data-driven use cases

to create a <u>**Unique AI framework**</u> for academia and industry



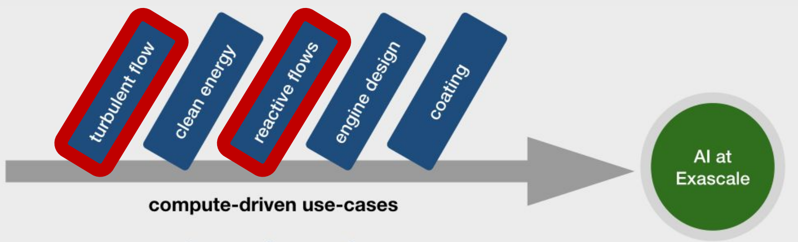
Partners in CoE RAISE

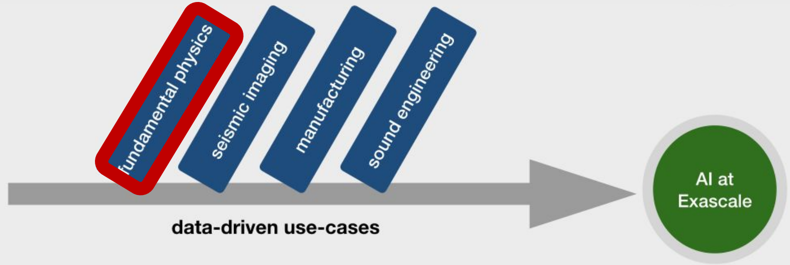


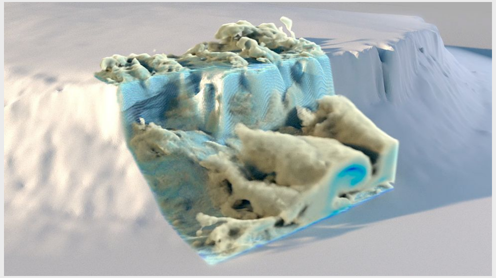
CoE RAISE
Use Cases

Use Cases in CoE RAISE

- Two kinds of use cases:



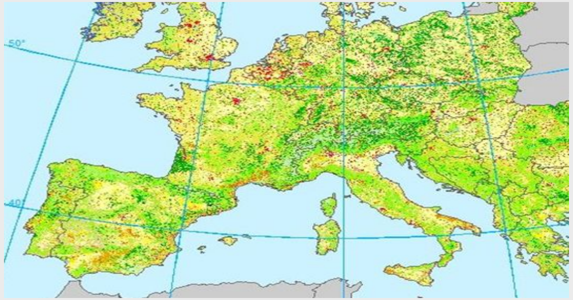




Example from use case "AI for wind farm layout": Turbulence generated by a cliff on Bolund Island, Denmark.

Example from use case " Seismic imaging with remote sensing for energy applications": Snapshot from a wavefield.

Remote Sensing

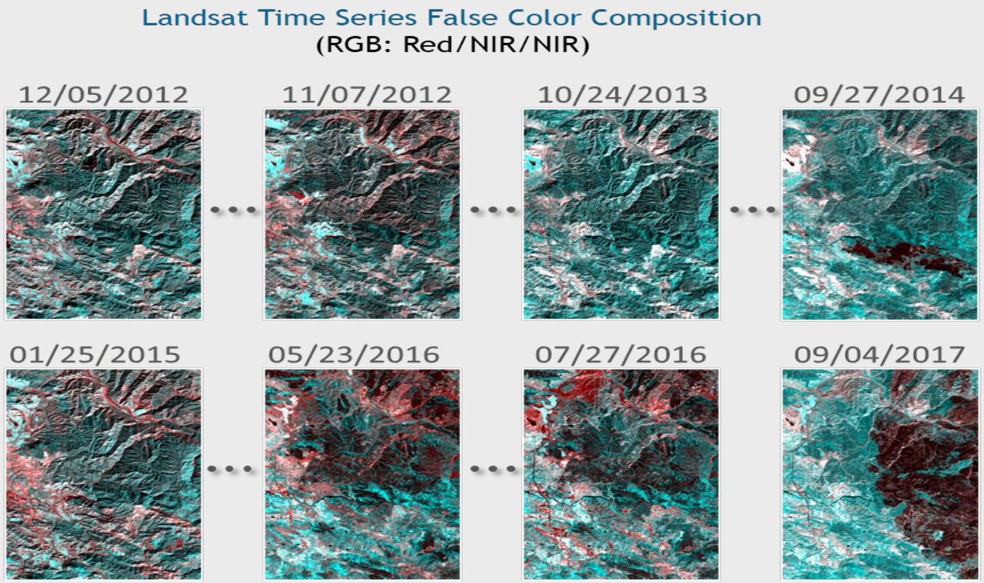


https://land.copernicus.eu/pan-european/corine-land-cover

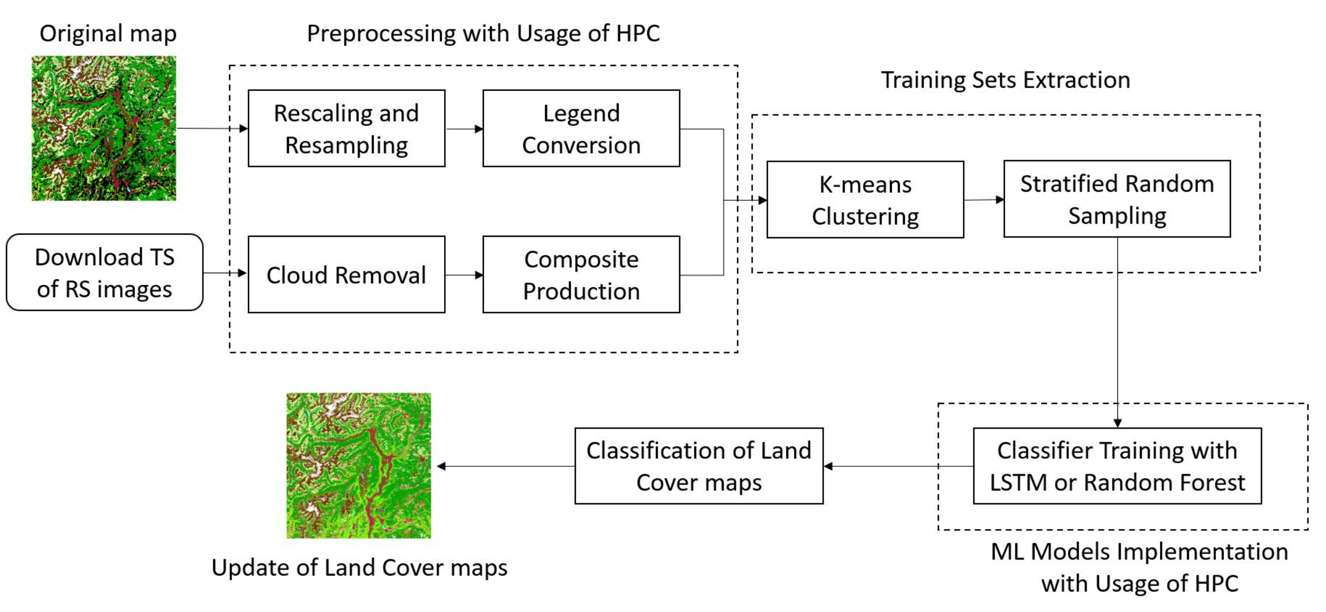
Problem: long time to create new thematic maps

Goal: more frequent update of maps

How: using satellite imagery



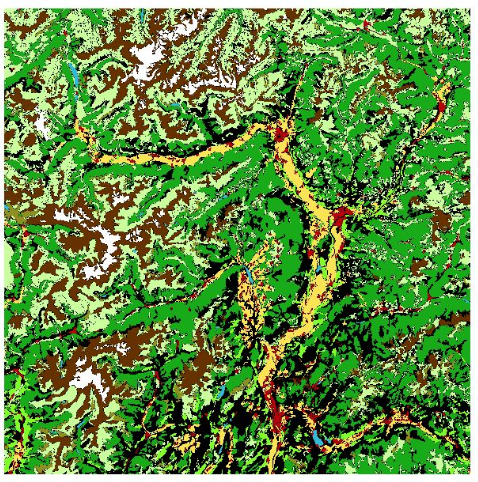
Remote Sensing: Framework



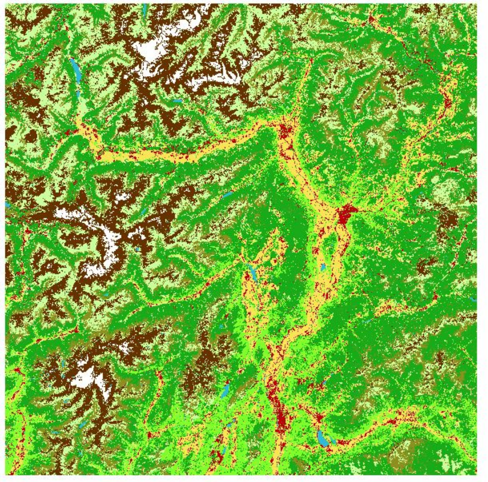
C. Paris, L. Bruzzone, D. Fernández-Prieto, "A Novel Approach to the Unsupervised Update of Land-Cover Maps by Classification of Time Series of Multispectral Images," IEEE Transactions on Geoscience and Remote Sensing, Vol. 57, No. 7, pp. 4259-4277, 2019,

Rocco Sedona, Claudia Paris, Liang Tian, Morris Riedel, Gabriele Cavallaro, "An automatic approach for the production of a time series of consistent land-cover maps based on long-short term memory", IEEE International Geoscience and Remote Sensing Symposium (IGARSS) 2022 (accepted)

Remote Sensing: Land Cover Classification



Original CORINE map
of TPS32 (2018)

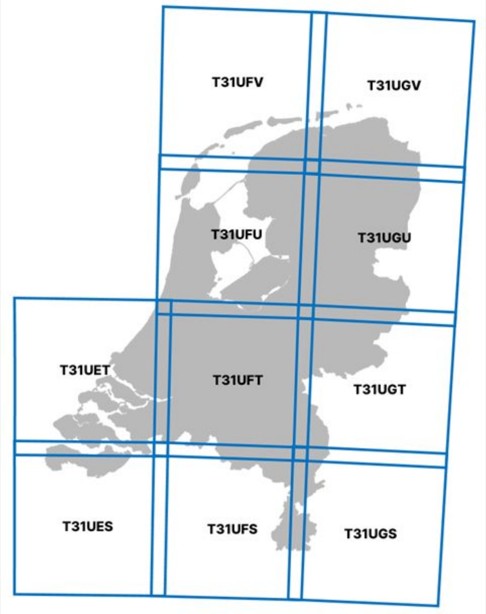


Predicted map of
TPS32 with RF (2018)

- Artificial
- Grass
- Crops
- Rice fields
- Mineral, Rocks, Sand
- Broadleaves
- Conifers
- Shrubland
- Lagoons
- Lake
- Snow

Remote Sensing: Study Area

Retrieval of Sentinel-2 time series for the Netherlands and related CORINE thematic maps



tile	num 2018	size 2018 [GB]	num 2019	size 2019 [GB]	num 2020	size 2020 [GB]
31UFT	17	16.16	14	15.33	19	20.91
31UGS	19	21.17	15	16.56	21	23.4
32ULC	20	22.08	18	19.55	20	21.16
31UGU	19	19.97	20	20.38	20	20.92
31UGV	20	18.68	17	15.5	27	21.7
31UFV	19	16.59	17	14.83	22	18.8
31UFU	18	16.65	15	15.24	21	21.31
31UFS	21	23.66	15	16.63	23	25.5
31UES	12	13.56	16	16.82	19	21.11
31UET	16	13.42	14	13.17	20	18.43
total	181	181.94	161	164.01	212	213.24

drive. enable. innovate.

RAISE Center of Excellence

The CoE RAISE project have received funding from the European Union's Horizon 2020 – Research and Innovation Framework Programme H2020-INFRAEDI-2019-1 under grant agreement no. 951733

Distributed Deep Learning

HDCRS Summer School

31.05.2022

Rocco Sedona

Jülich Supercomputing Centre – Forschungszentrum Jülich GmbH

University of Iceland

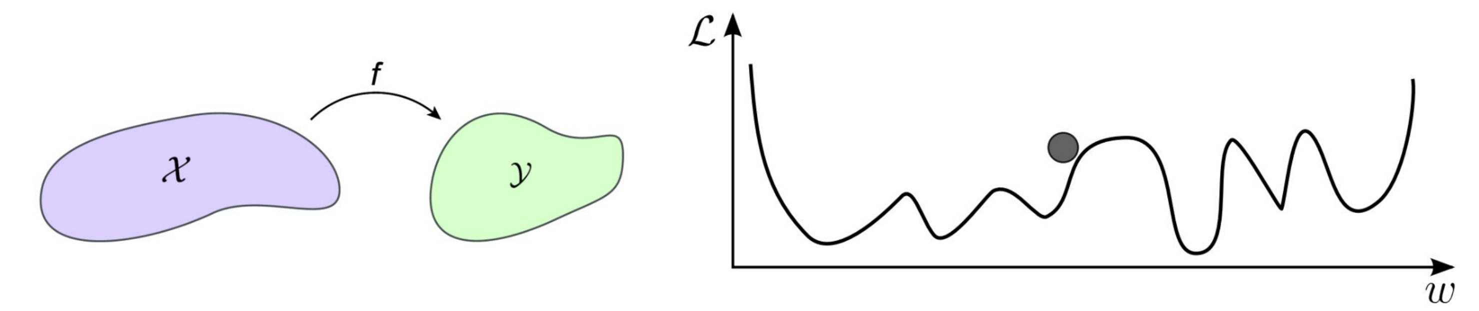
Outline

- Recap of basic concepts of Deep Learning
- Introduction to HPC
- MPI and other communication backends
- Introduction to Distributed Deep Learning
- Frameworks
- Final Remarks

Recap of basic DL concepts

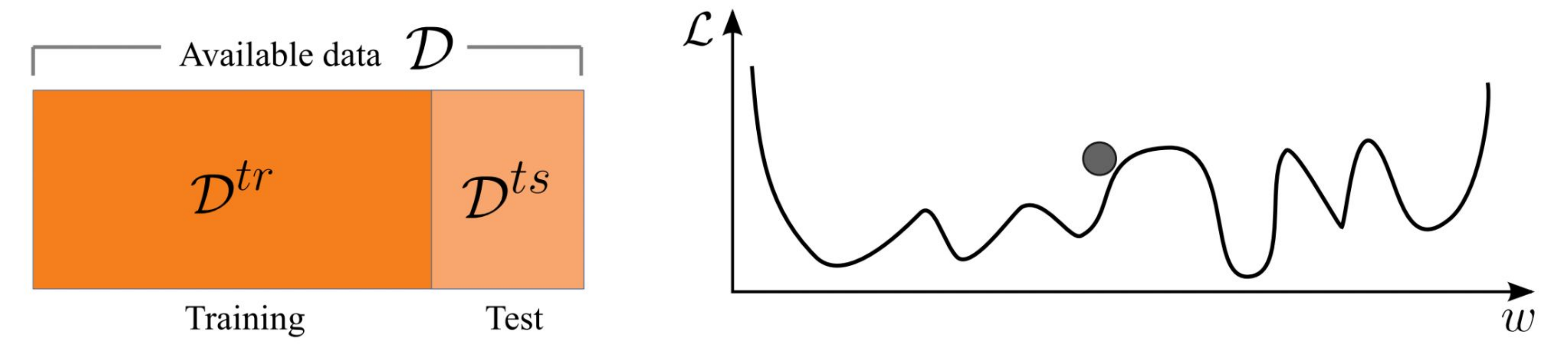
Optimization

- Optimizing loss (objective) of a (complex) model f on data D
- a (complex) model: function (or distribution) family $f(X;\theta)$ $(p(X;\theta))$
- parameters θ are to adapt ("fit") given the data $X \in D$
- optimization: defining a loss $L(f(X;\theta), D)$
- loss L: measure of quality ("fit") of the model f in terms of a task solution on D
- Objective: minimize $L(f(X;\theta), D)$



Generalization

- Estimate $L(f(X;\theta), D^{unseen})$: aiming for good generalization capability
- General approach: split D into disjoint D_{tr} and D_{ts}, $D_{tr} \cap D_{ts} = \varnothing$
- train on D_{tr}
- generalization error on D_{ts} after training

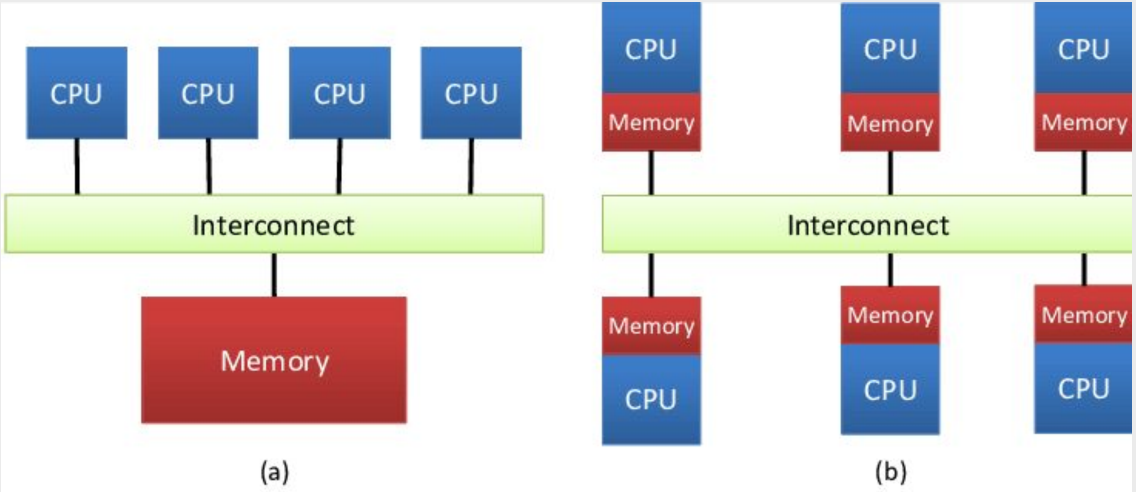


Scalable Learning & Multi-Purpose AI Lab, Helmholtz AI @ JSC

HPC

Hardware Levels of Parallelism

- (a) Single-machine (shared memory) (b) Multi-machine (distributed memory)



https://www.researchgate.net/publication/302245489_Adaptation_Strategies_in_Multiprocessors_System_on_Chip

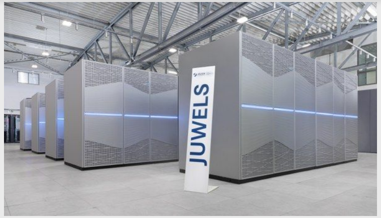
HPC

HPC at the frontline of computing power

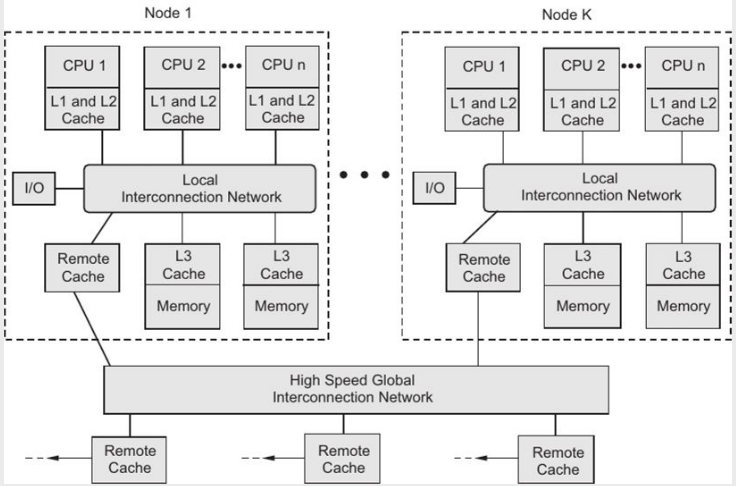
It includes work on 'four basic building blocks':

- Theory (numerical laws, physical models, speed-up performance, etc.)
- Technology (multi-core, supercomputers, networks, storages, etc.)
- Architecture (shared-memory, distributed-memory, interconnects, etc.)
- Software (libraries, schedulers, monitoring, applications, etc.)

Architecture: Shared-memory building blocks interconnected with a fast network (e.g., InfiniBand)



https://www.fz-juelich.de/de/ias/jsc

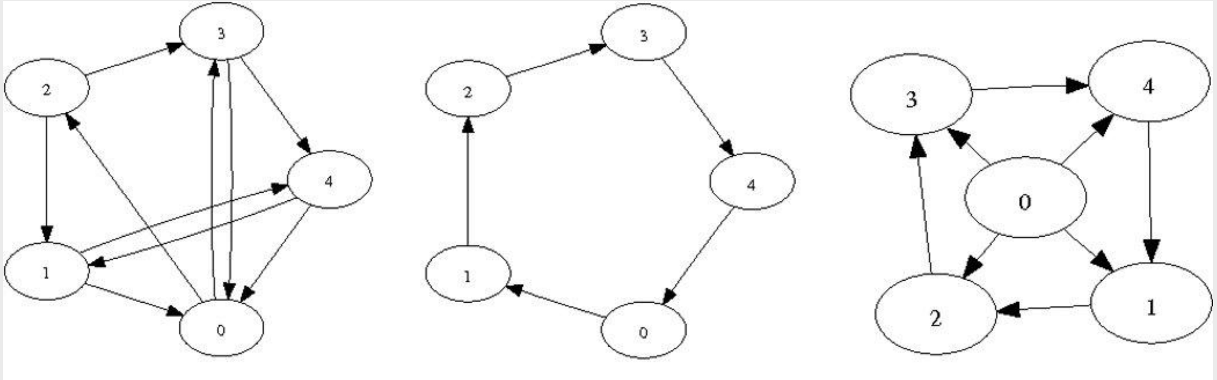


https://ebrary.net/206293/computer_science/distributed_shared_memory_multiprocessors_numa_model

Communication Backend

MPI

- MPI is a **standard** for **exchanging messages** between multiple computers running a parallel program across **distributed memory**
- Point-to-point and collective communication are supported
- **Different topologies** can be implemented
- Parallel I/O operations
- Blocking and non blocking statements

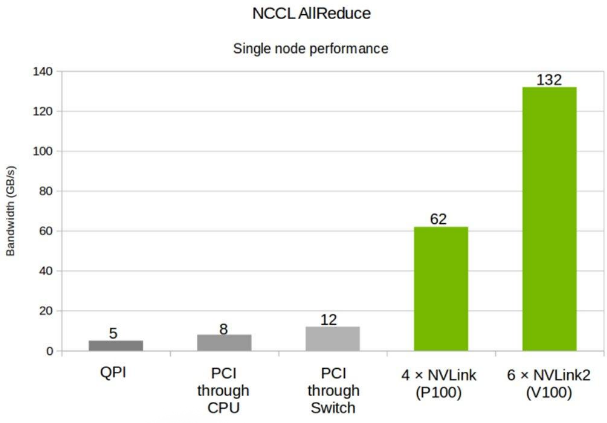


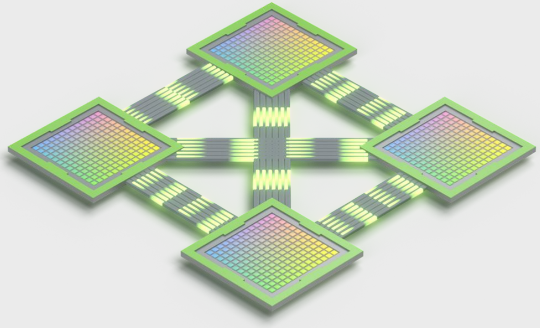
(a) Random (b) Ring (c) Wheel

Hoefler, T., Rabenseifner, R., Ritzdorf, H., de Supinski, B. R., Thakur, R., & Träff, J. L. (2010). The scalable process topology interface of MPI 2.2. Concurrency and Computation: Practice and Experience, 23(4), 293–310. https://doi.org/10.1002/cpe.1643

NCCL

- NVIDIA Collective Communications Library (NCCL) [19]
- Provides **optimized implementation of inter-GPU communication** operations, such as allreduce and variants
- Optimized for **high bandwidth and low latency** over PCI and NVLink/NVSwitch high speed interconnect for intra-node communication (up to 16 GPUs)
- Sockets and InfiniBand for inter-node communication
- For a comparison between communication backends look at: [https://mlbench.github.io/2020/09/08/communication-backend-comparison/]





https://developer.nvidia.com/blog/scaling-deep-learning-training-nccl/

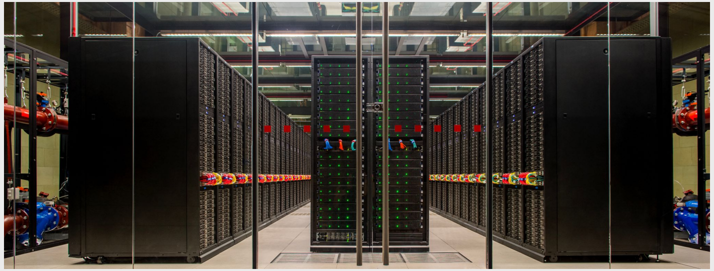
RCCL

- AMD's port of NCCL: ROCm Communication Collectives Library (RCCL) uses the same C API as NCCL
- NCCL APIs do not need to be converted

https://github.com/RadeonOpenCompute/ROCm

https://hwrig.com/amd-instinct-gpu-and-epyc-are-making-lumi-in-2021/

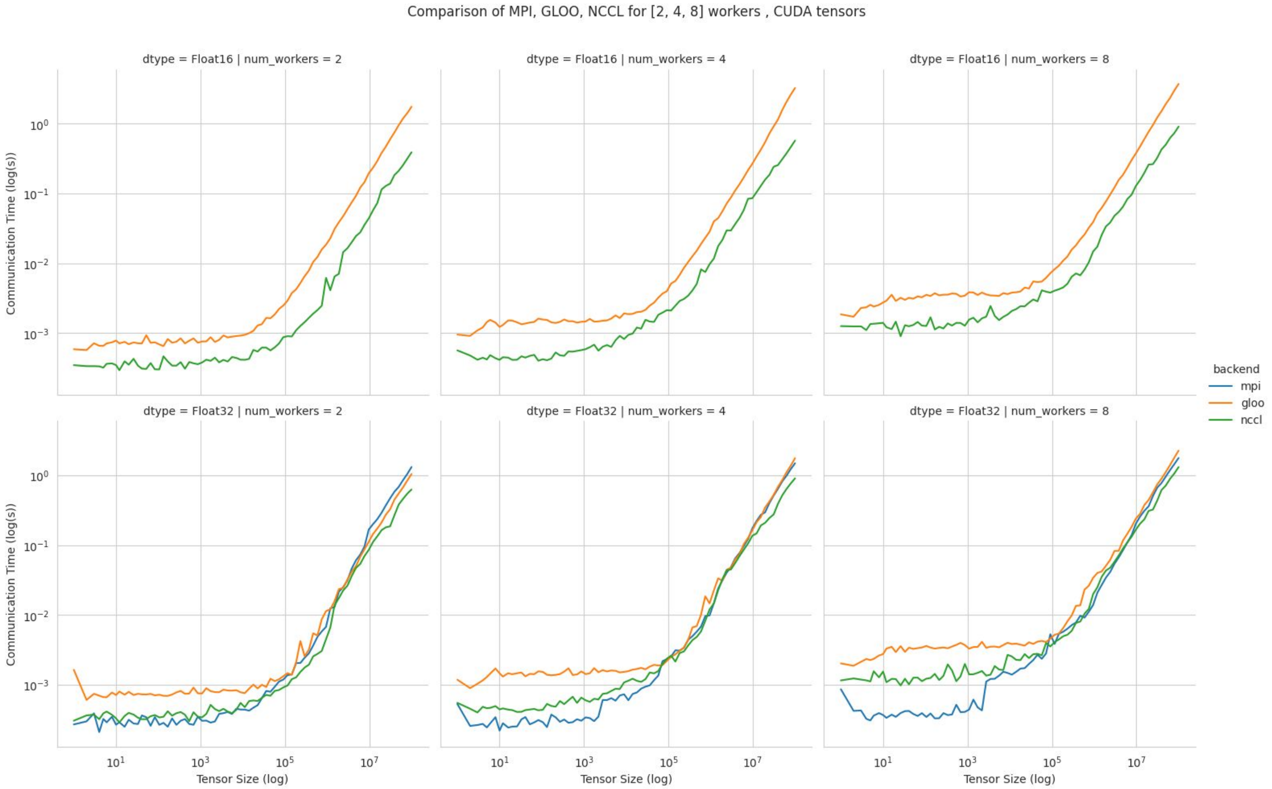
https://lumi-supercomputer.eu/easybuild-lumis-primary-software-installation-tool-introduced/



https://www.bsc.es/innovation-and-services/technical-information-cte-amd

Benchmark

- For a comparison between communication backends look at: [https://mlbench.github.io/2020/09/08/communication-backend-comparison/]
- MPI vs Gloo vs NCCL

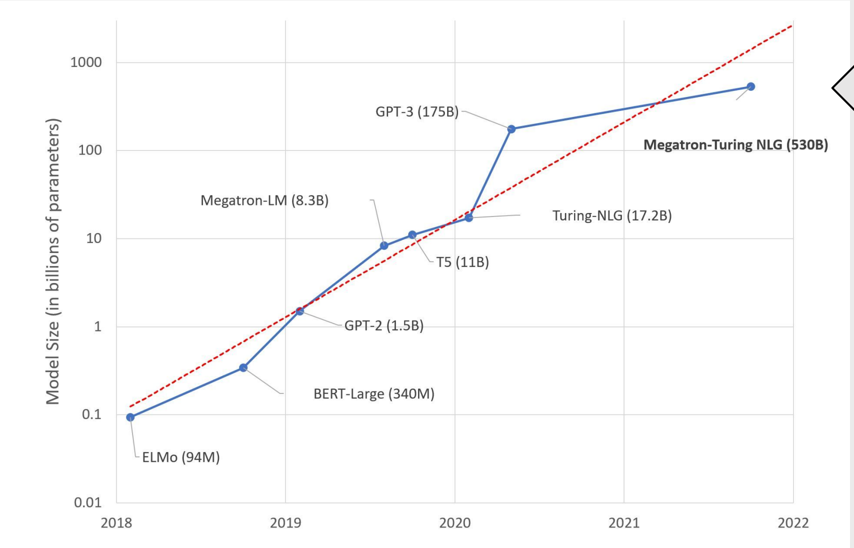


Motivation

Motivation

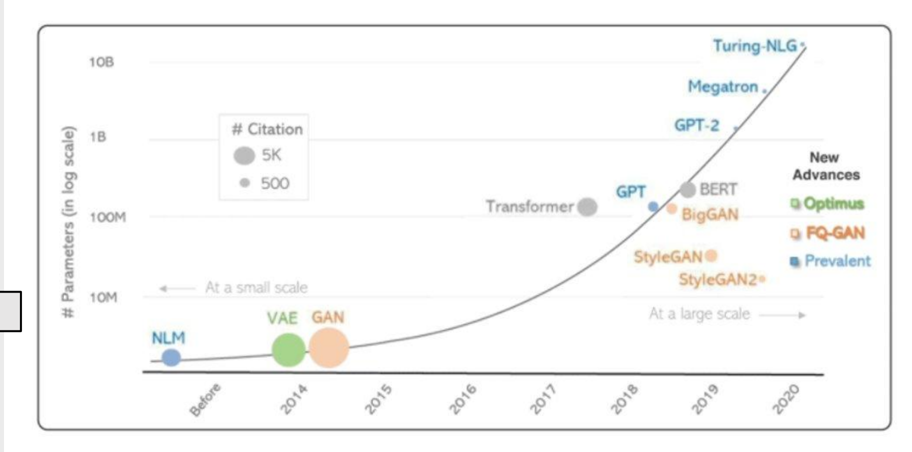
In recent years almost exponential increase of number of parameters of the models

2022



https://huggingface.co/blog/large-language-models

2020

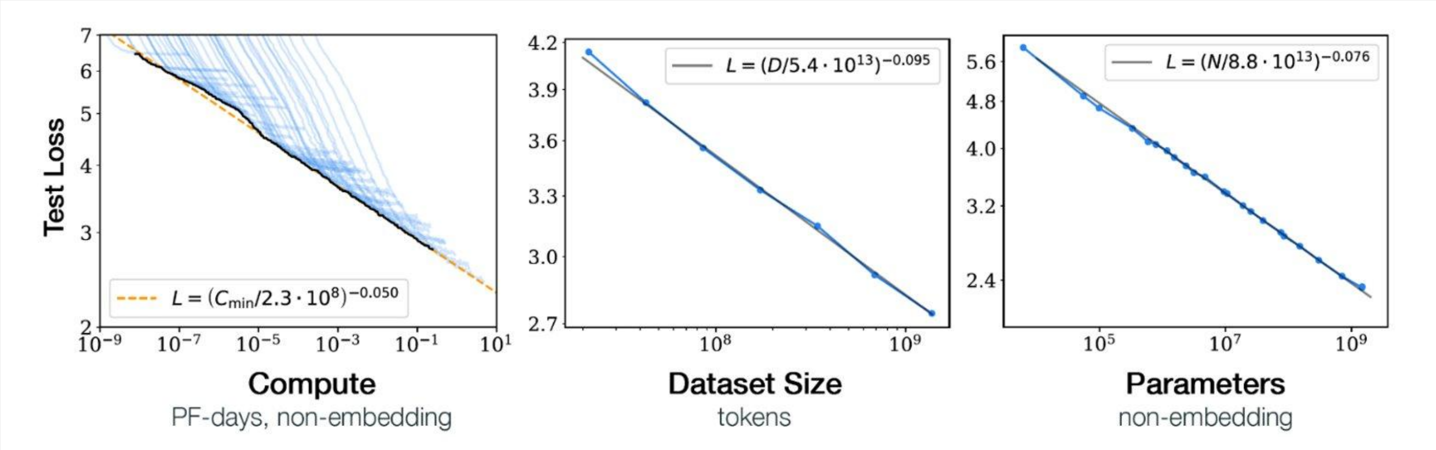


https://www.microsoft.com/en-us/research/blog/a-deep-generative-model-trifecta-three-advances-that-work-towards-harnessing-large-scale-power/

Motivation

- Bigger models require bigger datasets
- Consequence –> More resources are needed (both memory and computation power)

	Data Set	Type	Task	Size
small	MNIST	Image	Classification	55,000
small	Fashion MNIST	Image	Classification	55,000
small	CIFAR-10	Image	Classification	45,000
large	ImageNet	Image	Classification	1,281,167
large	Open Images	Image	Classification (multi-label)	4,526,492
large	LM1B	Text	Language modeling	30,301,028
large	Common Crawl	Text	Language modeling	~25.8 billion

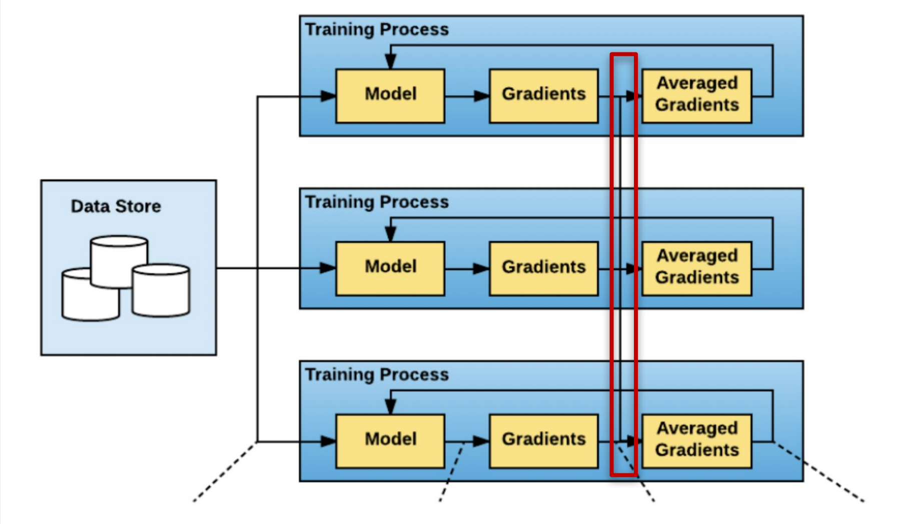


Kaplan et al., "Scaling Laws for Neural Language Models", 2020, https://arxiv.org/abs/2001.08361

Distributed Deep Learning

Data Parallelism

- Concept: split the data
- The gradients for different batches of data are calculated separately on each node
- But averaged across nodes to apply consistent updates to the model copy in each node

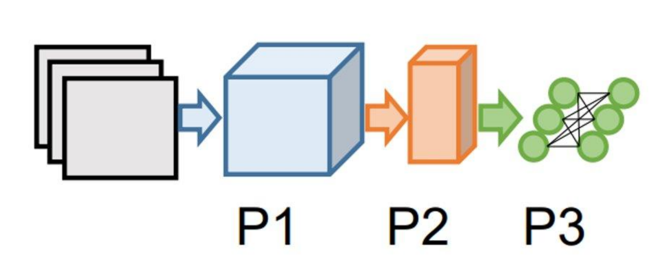


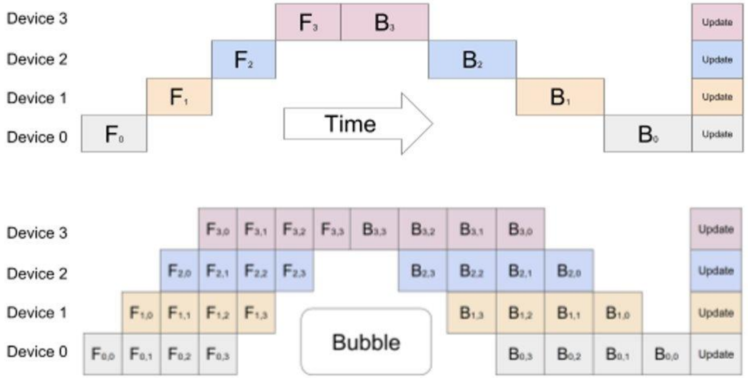
A. Sergeev and M. D. Balso, “Horovod: Fast and Easy Distributed Deep Learning in TensorFlow”, arXiv:1802.05799, 2018

Model Parallelism

Concept: split the model
Pipelining:

- partitioning the DNN according to depth, assigning layers to specific processors
- overlapping computations, i.e., between one layer and the next (as data becomes ready)





Top: The naive model parallelism strategy leads to severe underutilization due to the sequential nature of the network. Only one accelerator is active at a time. ***Bottom:*** *GPipe divides the input mini-batch into smaller micro-batches, enabling different accelerators to work on separate micro-batches at the same time.*

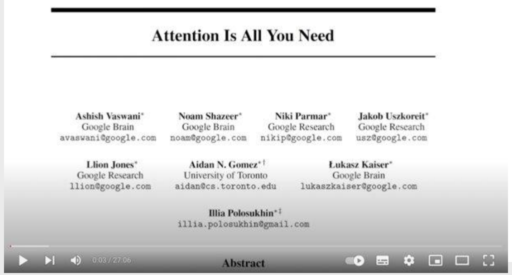
[https://huggingface.co/docs/transformers/parallelism]

Model Parallelism

RAISE Center of Excellence

Tensor parallelism:

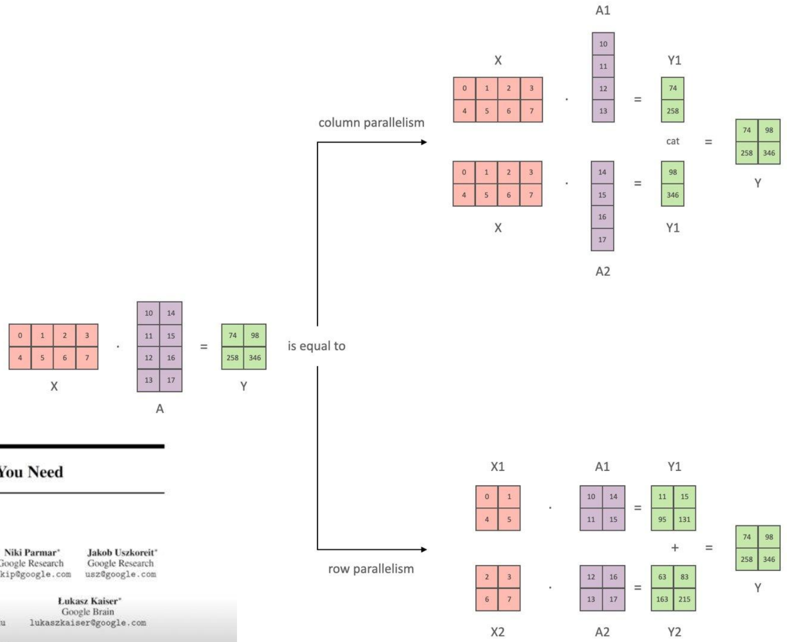
- matrix operations (f.e. matrix multiplication) can be split between multiple GPUs
- Scaling large transformers with multihead self-attention is based on this concept



Attention Is All You Need

Ashish Vaswani* Google Brain avaswani@google.com
Noam Shazeer* Google Brain noam@google.com
Niki Parmar* Google Research nikip@google.com
Jakob Uszkoreit* Google Research usz@google.com
Llion Jones* Google Research llion@google.com
Aidan N. Gomez*† University of Toronto aidan@cs.toronto.edu
Łukasz Kaiser* Google Brain lukaszkaiser@google.com
Illia Polosukhin*‡ illia.polosukhin@gmail.com

Abstract



[https://www.youtube.com/watch?v=iDulhoQ2pro&ab_channel=YannicKilcher]

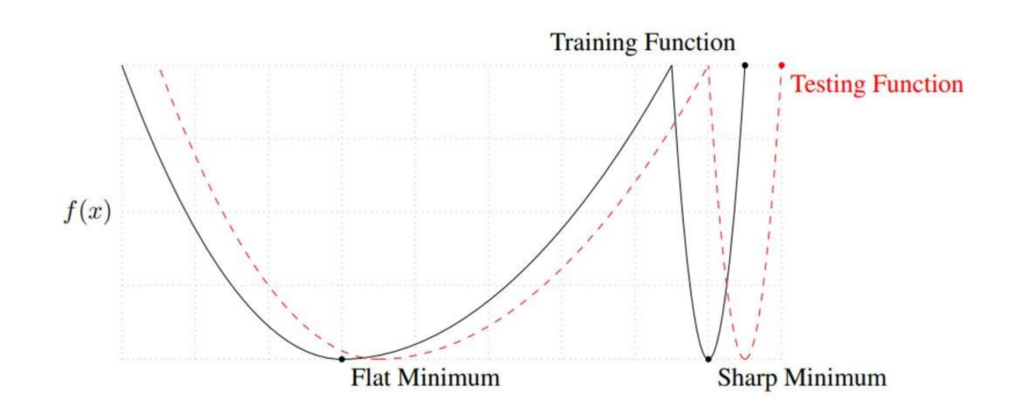
[https://huggingface.co/docs/transformers/parallelism]

Challenges

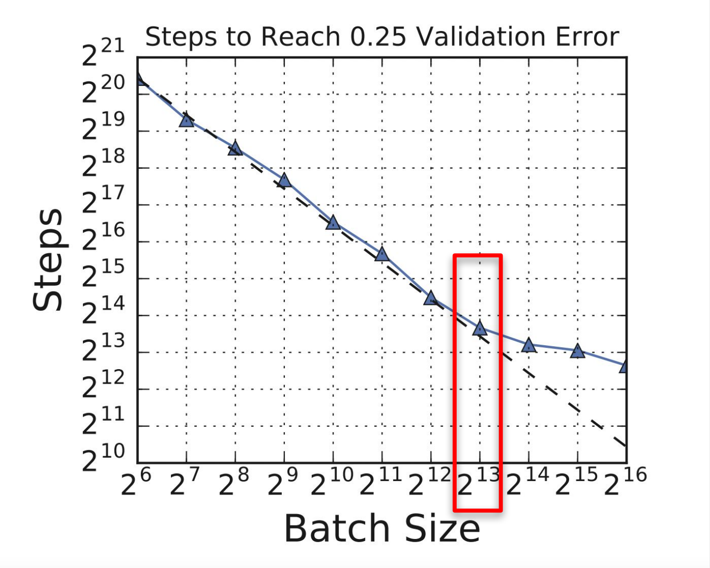
- Poor generalization due to sharp minima

[Hochreiter, Sepp and Schmidhuber, Jürgen. Flat minima. Neural Computation, 9(1):1–42, 1997]

- Time to accuracy does not decrease



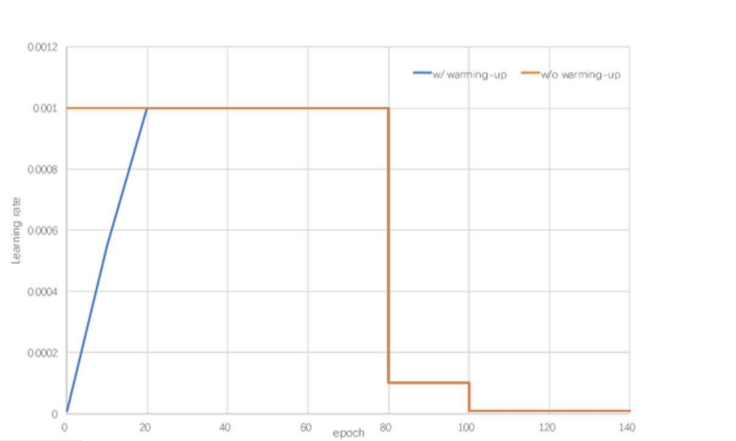
N. S. Keskar and D. Mudigere and J. Nocedal and M. Smelyanskiy and P.T.P. Tang, On Large-Batch Training for Deep Learning: Generalization Gap and Sharp Minima, 2016



Shallue et al., 2019, https://arxiv.org/pdf/1811.03600.pdf

Solution

- For batch size < 8000
  - Scale learning rate
  - Warm-up
- For batch size > 8000
  - Choice of the optimizer:
    - LARS
    - LAMB
    - post-local SGD



Warm-up

	Hardware	Software	Batch size	Optimizer	# Steps	Time/step	Time	Accuracy
Goyal *et al.* [3]	Tesla P100 × 256	Caffe2	8,192	SGD	14,076	0.255 s	1 hr	**76.3** %
You *et al.* [11]	KNL × 2048	Intel Caffe	32,768	SGD	3,519	0.341 s	20 min	75.4 %
Akiba *et al.* [10]	Tesla P100 × 1024	Chainer	32,768	RMSprop/SGD	3,519	0.255 s	15 min	74.9 %
You *et al.* [11]	KNL × 2048	Intel Caffe	32,768	SGD	2,503	0.335 s	14 min	74.9 %
Jia *et al.* [12]	Tesla P40 × 2048	TensorFlow	65,536	SGD	1,800	0.220 s	6.6 min	75.8 %
Ying *et al.* [16]	TPU v3 × 1024	TensorFlow	32,768	SGD	3,519	**0.037 s**	2.2 min	**76.3** %
Mikami *et al.* [13]	Tesla V100 × 3456	NNL	55,296	SGD	2,086	0.057 s	2.0 min	75.3 %
Yamazaki *et al.* [14]	Tesla V100 × 2048	MXNet	81,920	SGD	1,440	0.050 s	**1.2 min**	75.1 %

Osawa et al., 2020

Is that all?

- Still ongoing research
- Well-establish optimizers can match new ones with enough *hyperparameter tuning*

Batch size	Step budget	LAMB	Adam
32k	15,625	91.48	**91.58**
65k/32k	8,599	90.58	**91.04**
65k	7,818	–	**90.46**

https://openreview.net/pdf?id=Kloou2uk_Rz

Frameworks

Horovod

Horovod

- *Data parallel*, each GPU has a copy of the model and a chunk of the data
- Efficient *decentralized framework*,

based on MPI and NCCL libraries, where actors exchange parameters without the need of a parameter server

- Works on top of Keras, TensorFlow, PyTorch and Apache MXNet

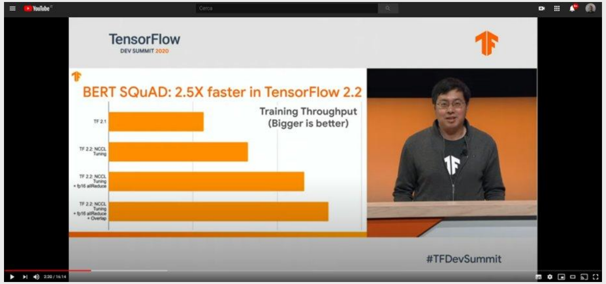
Tensorflow

- Parameter server for asynchronous training
- Mirrored strategy for synchronous training

Pytorch

- Distributed Data-Parallel Training (DDP)

A. Sergeev and M. D. Balso, "Horovod: Fast and Easy Distributed Deep Learning in TensorFlow", arXiv:1802.05799, 2018.

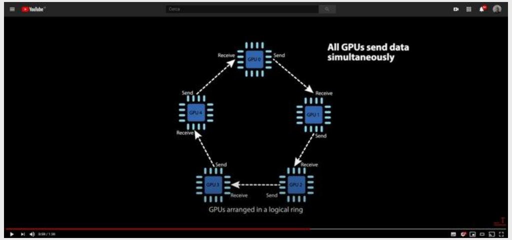


https://www.youtube.com/watch?v=6ovfZW8pepo&ab_channel=TensorFlow

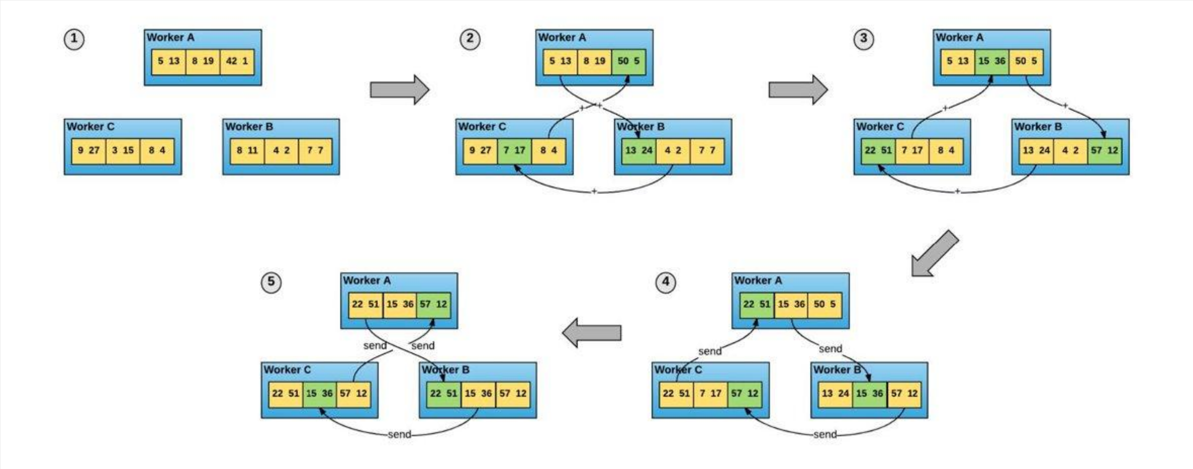
Ring allreduce

Two step process:

1. share-reduce step
2. share-only step



https://www.youtube.com/watch?v=4y0TDK3KoCA&t=585s&ab_channel=Uber Engineering

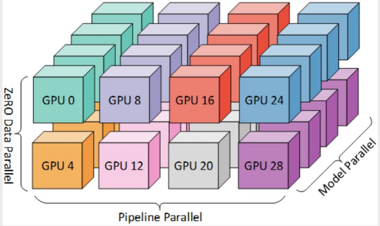


A. Sergeev and M. D. Balso, “Horovod: Fast and Easy Distributed Deep Learning in TensorFlow”, arXiv:1802.05799, 2018

Other Frameworks

[https://github.com/cc-hpc-itwm/tarantella]

[https://github.com/hpcaitech/ColossalAI]



https://www.deepspeed.ai/

https://github.com/helmholtz-analytics/heat

Model size	Hidden size	Number of layers	Number of parameters (billion)	Model-parallel size	Number of GPUs	Batch size	Achieved teraFlOPs per GPU	Percentage of theoretical peak FLOPs	Achieved aggregate petaFLOPs
1.7B	2304	24	1.7	1	32	512	137	44%	4.4
3.6B	3072	30	3.6	2	64	512	138	44%	8.8
7.5B	4096	36	7.5	4	128	512	142	46%	18.2
18B	6144	40	18.4	8	256	1024	135	43%	34.6
39B	8192	48	39.1	16	512	1536	138	44%	70.8
76B	10240	60	76.1	32	1024	1792	140	45%	143.8
145B	12288	80	145.6	64	1536	2304	148	47%	227.1
310B	16384	96	310.1	128	1920	2160	155	50%	297.4
530B	20480	105	529.6	280	2520	2520	163	52%	410.2
1T	25600	128	1008.0	512	3072	3072	163	52%	502.0

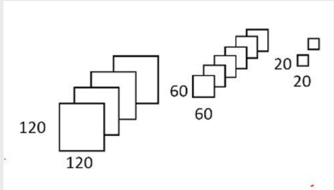
[https://github.com/NVIDIA/Megatron-LM]

A Remote Sensing Use Case

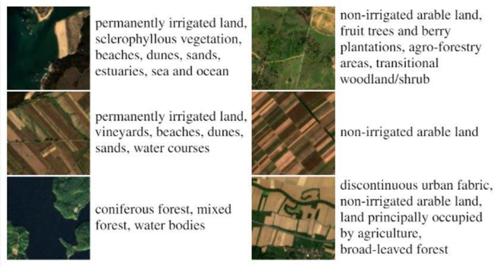
Dataset: BigEarthNet, Sentinel-2 Data Patches and Annotated with CORINE Land Covers

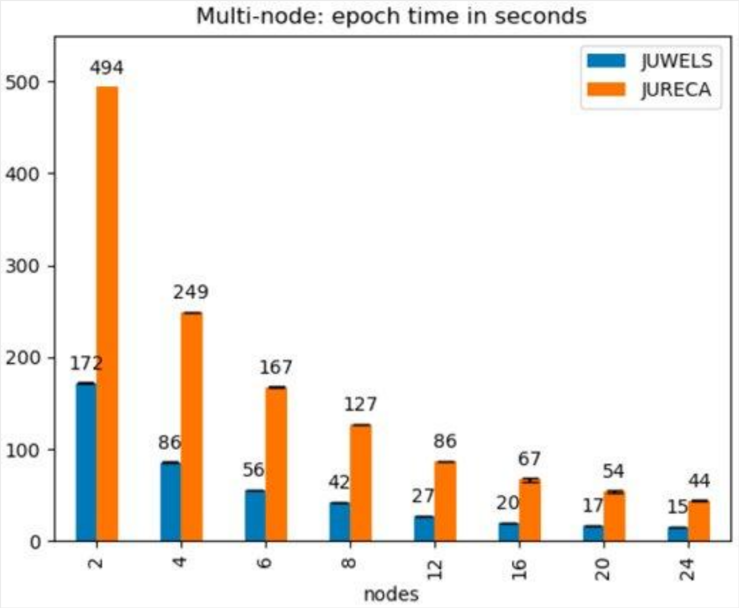
Model: ResNet50

0.74 F1-score up to 24 nodes – 96 GPUs with a global batch size of 8K samples



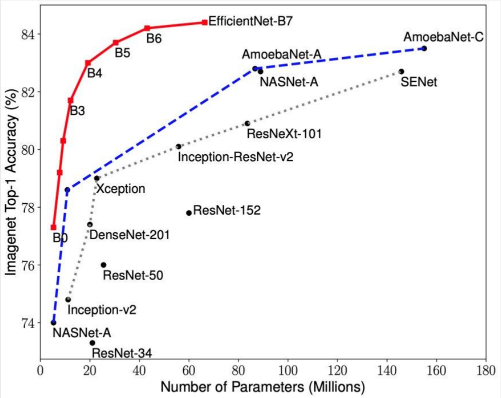
Patch and its dimension (px)





R. Sedona et al., Remote Sensing Big Data Classification with High Performance Distributed Deep Learning, 2019

- Adopted TensorFlow Dataset API to build a pipeline with integrated data augmentation, caching and prefetching of the data
- Deploying on 64 nodes / 256 GPUs of the Juwels Booster (Nvidia A100)
- New CNNs as EfficientNet, less parameters than ResNet, faster to train and higher accuracy
- Testing newer optimizers: LARS, LAMB, NovoGrad
- As the number of hyperparameters grows, there is the need to automatize the search for the optimal values (NAS)
- Hyper parameter tuning with **Ray Tune (embedded in Horovod)**: 'IGARSS2022 ACCELERATING HYPERPARAMETER TUNING OF A DEEP LEARNING MODEL FOR REMOTE SENSING IMAGE CLASSIFICATION', M. Aach, R. Sedona, A. Lintermann, G. Cavallaro, H. Neukirchen, M. Riedel, IGARSS2022 (accepted)



https://github.com/qubvel/efficientnet

Final Remarks

Final Remarks

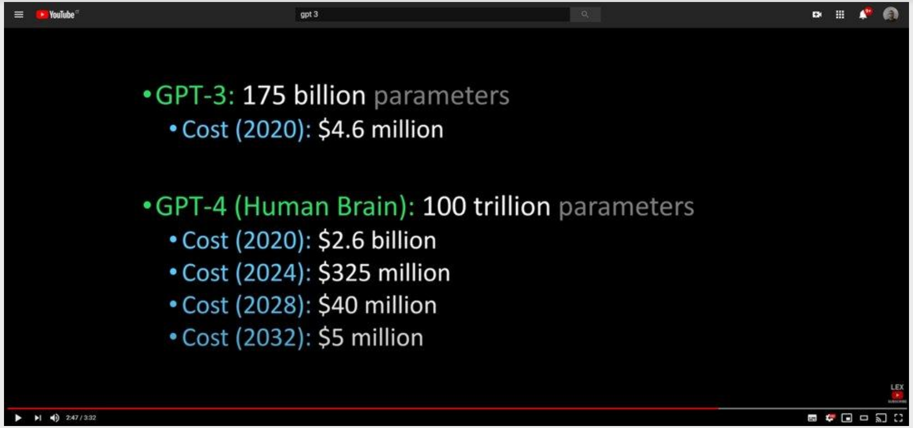
- The trend is to make distributed deep learning easier
- Not only frameworks, but integrated products
- Example: Dataflow-as-a-Service by SambaNova
- Intel's OpenAPI for heterogeneous computing
  [https://www.intel.com/content/www/us/en/developer/tools/oneapi/overview.html#gs.u1eb1g]
- AMD's GPUs using ROCm (similar to Nvidia's NCCL)

[f.e. https://www.bsc.es/innovation-and-services/technical-information-cte-amd]

[https://www.hpcwire.com/2020/12/09/ai-newcomer-sambanova-gas-product-lineup-and-offers-new-service/]

DL and Cloud Computing

- Trend towards cloud-based HPC
- What about costs?
- Let's have a look at **NCsv3-series** [25]
- **355 years** to train GPT-3 on a Tesla V100
- Training cost = 355Y×365D/Y×24H/D×0.9792$/H = **3.045.116$**



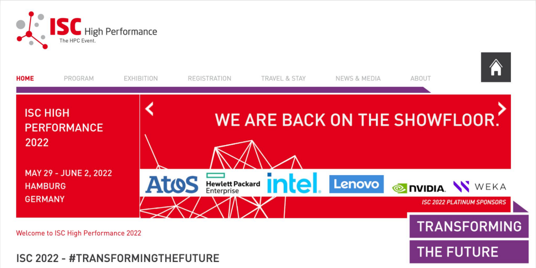
https://www.youtube.com/watch?v=kpiY_LemaTc&ab_channel=LexFridman

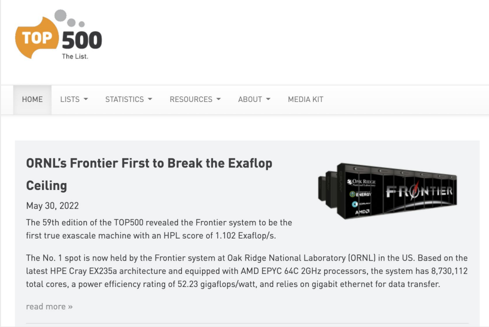
Add to estimate	Instance	Core	RAM	Temporary storage	GPU	Pay as you go	1 year reserved (% Savings)	3 year reserved (% Savings)	Spot (% Savings)
⊕	NC6s v3	6	112 GiB	736 GiB	1X V100	$3.06/hour	$1.9492/hour (~36%)	$0.9792/hour (~68%)	$0.306/hour (~90%)
⊕	NC12s v3	12	224 GiB	1,474 GiB	2X V100	$6.12/hour	$3.8984/hour (~36%)	$1.9585/hour (~68%)	$0.612/hour (~90%)
⊕	NC24rs v3	24	448 GiB	2,948 GiB	4X V100	$13.464/hour	$8.5766/hour (~36%)	$5.1002/hour (~62%)	$1.3464/hour (~90%)
⊕	NC24s v3	24	448 GiB	2,948 GiB	4X V100	$12.24/hour	$7.7970/hour (~36%)	$3.9169/hour (~68%)	$1.224/hour (~90%)

https://azure.microsoft.com/en-us/pricing/details/virtual-machines/linux/

Towards Exascale

Frontier (First supercomputer to Break the Exaflop Ceiling at Oak Ridge National Laboratory (ORNL) in the US





TOP 500 The List.

HOME LISTS STATISTICS RESOURCES ABOUT MEDIA KIT

ORNL's Frontier First to Break the Exaflop Ceiling

May 30, 2022

The 59th edition of the TOP500 revealed the Frontier system to be the first true exascale machine with an HPL score of 1.102 Exaflop/s.

The No. 1 spot is now held by the Frontier system at Oak Ridge National Laboratory (ORNL) in the US. Based on the latest HPE Cray EX235a architecture and equipped with AMD EPYC 64C 2GHz processors, the system has 8,730,112 total cores, a power efficiency rating of 52.23 gigaflops/watt, and relies on gigabit ethernet for data transfer.

read more »

The New York Times

U.S. Retakes Top Spot in Supercomputer Race

A massive machine in Tennessee has been deemed the world's speediest. Experts say two supercomputers in China may be faster, but the country didn't participate in the rankings.

Give this article

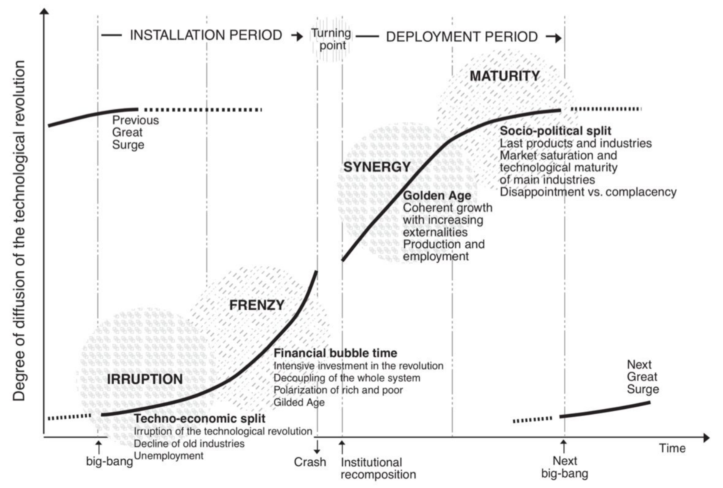
Exascale Application Readiness
[https://www.olcf.ornl.gov/caar/frontier-caar/?fbclid=IwAR0JvTHz9rc_um_OGQbN28J8MDw5sv5yMF2OBWy2u5RKdMVyxODseWlnP7E]

Final Remarks

- Takeaways:
    - Frontier technology is fast paced
    - But successful solutions tend to become stable
    - Great opportunities for Distributed Deep Learning with the increased availability of computing resources

- Aknowledgement: Helmholtz AI Consultants
  [https://www.helmholtz.ai/themenmenue/our-research/consultant-teams/helmholtz-ai-consultants-fzj/index.html]
  [PRACE course "Introduction to Scalable Deep Learning" https://events.prace-ri.eu/event/1310/]

Figure 5.1 Recurring phases of each great surge in the core countries



Carlota Perez, 2002. "Technological Revolutions and Financial Capital," Books, Edward Elgar Publishing, number 2640.

drive. enable. innovate.

RAISE Center of Excellence

The CoE RAISE project have received funding from the European Union's Horizon 2020 – Research and Innovation Framework Programme H2020-INFRAEDI-2019-1 under grant agreement no. 951733

Follow us:

RAISE Center of Excellence

Distributed Hyperparameter Tuning with HPC

HDCRS Summer School

31.05.2022

Marcel Aach

Jülich Supercomputing Centre – Forschungszentrum Jülich GmbH

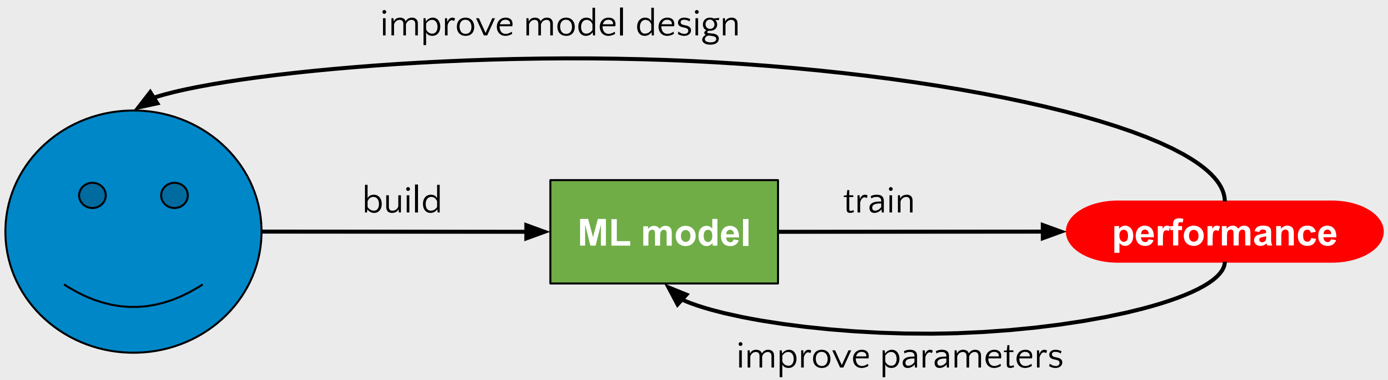
University of Iceland

Outline

- What are hyperparameters and why are they so important?
- Hyperparameter optimization (HPO) methods
- Running hyperparameter optimization on HPC with Ray Tune
- Remote sensing use-case

What Are Hyperparameters and Why Are They So Important?

Design of ML Models



- ML models are designed by humans
- Usually start from experience, then fine tune
- Trial and error

Hyperparameters in ML Models

- Architectural parameters:
  - Type of ML model (neural network, SVM?)
  - Number and kind of layers (convolutional, dense, dropout?)
  - Number of neurons per layer
  - Activation functions (relu, sigmoid?)
  - Weight initialization and regularization
- Optimizer parameters:
  - Optimizer type (SGD, Adam?)
  - Batch size
  - Learning rate
  - Learning rate schedule
  - Momentum

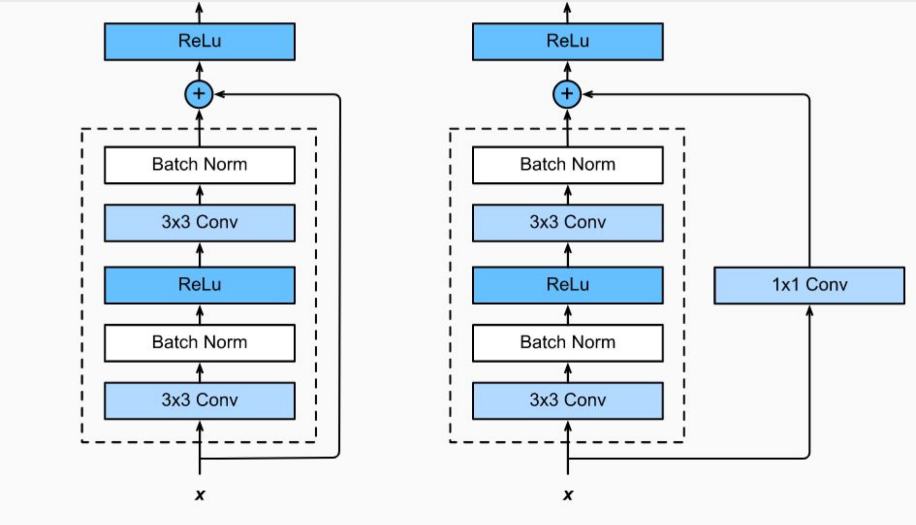


Image from opengenus.org

Hyperparameters in ML Pipeline

RAISE Center of Excellence

- Parameters of pre-processing
  - Image size
  - Image normalization
  - Image crop
  - Image rotation
  - Number of spectral bands

- **Mix of lots of discrete and continuous variables that influence each other**
- **Requires lots and lots of trials and error**
- **We optimize the “inner loop”, why not the “outer loop” as well?**

Importance of Hyperparameters

HPO models beat human design (CV: [Real 2018] , NLP: [Melis 2017])

HPO Methods

HPO Terminology

- **search space:** hyperparameters and their sampling interval
- **configuration:** a set of hyperparameters sample from the search space
- **trial:** one training run of a configuration
- **“inner loop” optimization:** adjusting the parameters of a model (e.g. via SGD)
- **“outer loop” optimization:** adjusting the ***hyper***parameters of a model

The Easy Way

- **Random Search:**
  - Sample a random configuration from the search space grid and look at the performance
- **Grid Search:**
  - Sample every grid point

- Good starting point
- Embarrassingly parallel
- Burns lots and lots of resources

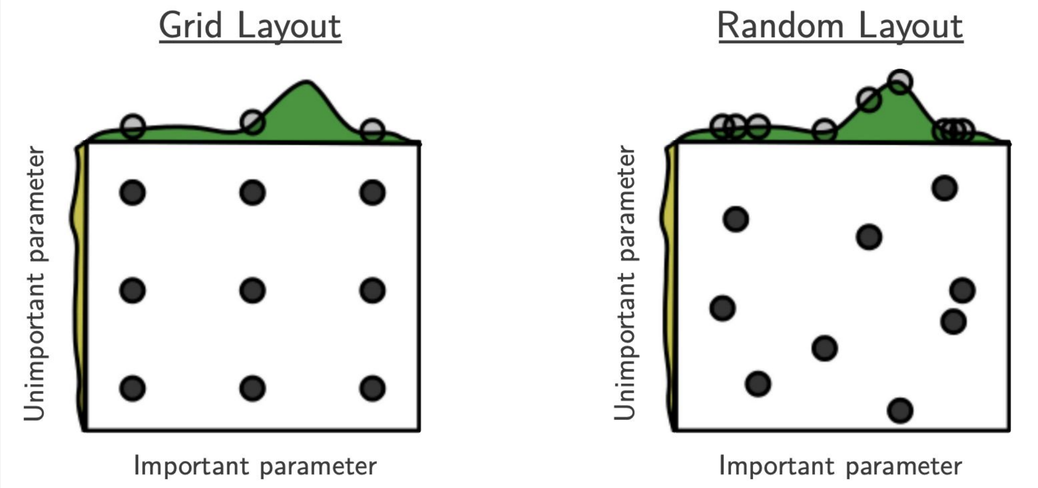


Image from [Bergstra 2012]

Accelerating HPO

- **“Smart” choice of configurations:**
  - Bayesian Optimization (black box optimizer)
  - High dimensional search space
  - Parallelism problem: some trials finish before others
- **Acceleration of the trials:**
  - Distributed deep learning
  - Early stopping “bad” trials
  - “Smart” scheduling of the trials

Smart Scheduling: SHA

- **Successive Halving (or Thirding) Algorithm (SHA, Li 2018):**
  - Idea: Stop bad trials early and allocate resources to more promising trials
  - Sample N trials randomly, keep only best N/2, then N/4 ...
  - Problem: What about late learners?/Run trials for longer or explore more trials?

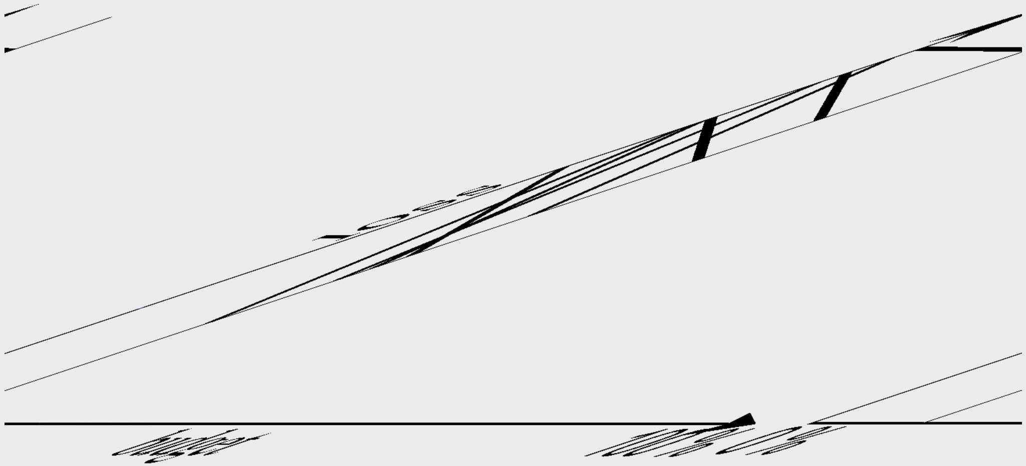


Figure from AutoML.org

Smart Scheduling: HyperBand

- Solution to exploration vs. exploitation trade-off: HyperBand [Li 2018]
- Perform multiple SHA runs with different budget allocations

time ↓

	$s=4$		$s=3$		$s=2$		$s=1$		$s=0$	
i	n_i	r_i	n_i	r_i	n_i	r_i	n_i	r_i	n_i	r_i
0	81	1	27	3	9	9	6	27	5	81
1	27	3	9	9	3	27	2	81		
2	9	9	3	27	1	81				
3	3	27	1	81						
4	1	81								

level of "pruning aggression" →

n = number of configurations
r = resources allocated per configuration

Improvements to HyperBand

- **Bayesian Optimization + HyperBand (BOHB, [Falkner 2018]):**
  - Use HyperBand for scheduling but choose new trial parameters with Bayesian methods
- **Asynchronous Successive Halving (ASHA, [Li 2020]):**
  - Like HyperBand but do not wait for all trials to finish before halving
  - Allocate resources faster massively parallel, **very well suited for HPC applications**

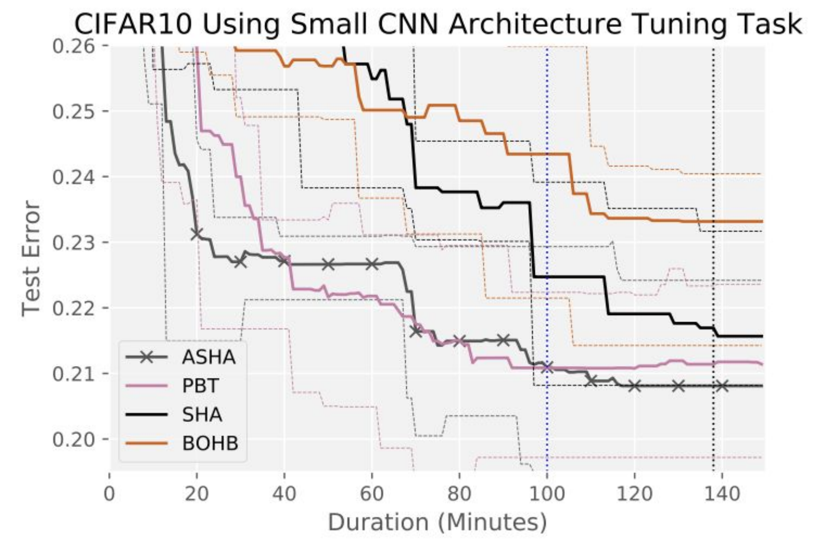
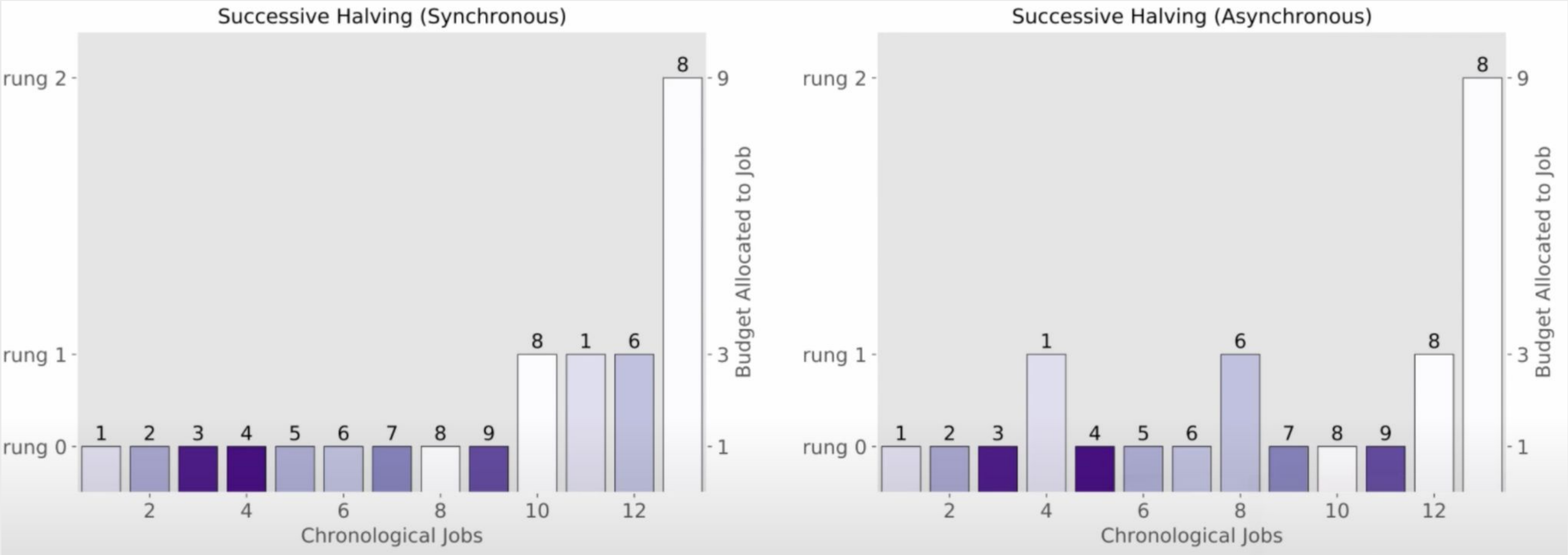


Image from [Li 2020]

SHA vs. ASHA



Slide from Ameet Talwalkar: Massively Parallel Hyperparameter Tuning

Other Options

- **Population Based Training (PBT):**
  - Genetic algorithm – mutate best performing trials randomly [Jaderberg 2017]
- **Reinforcement Learning:**
  - Agent selects the parameters – use trial performance as reward [Zoph 2016]
- **Differentiable Architecture Search (DARTS):**
  - Continuous representation of the architecture search space, use gradient descent for optimization [Liu 2018]

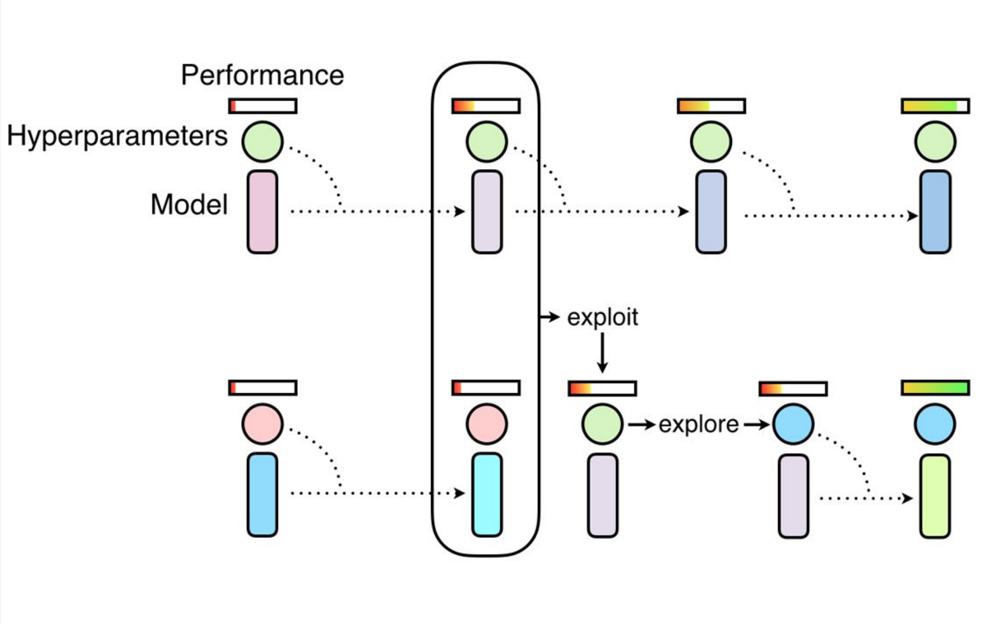


Image from [Jaderberg 2017]

Hyperparameter Optimization on HPC

Library: Ray

- Universal python API for distributed computing
  - Simple primitives to run and build distributed applications
  - Parallelize single machine code with little code changes
  - Works with lots of different libraries
- Open source, maintained by Anyscale

Ray Framework

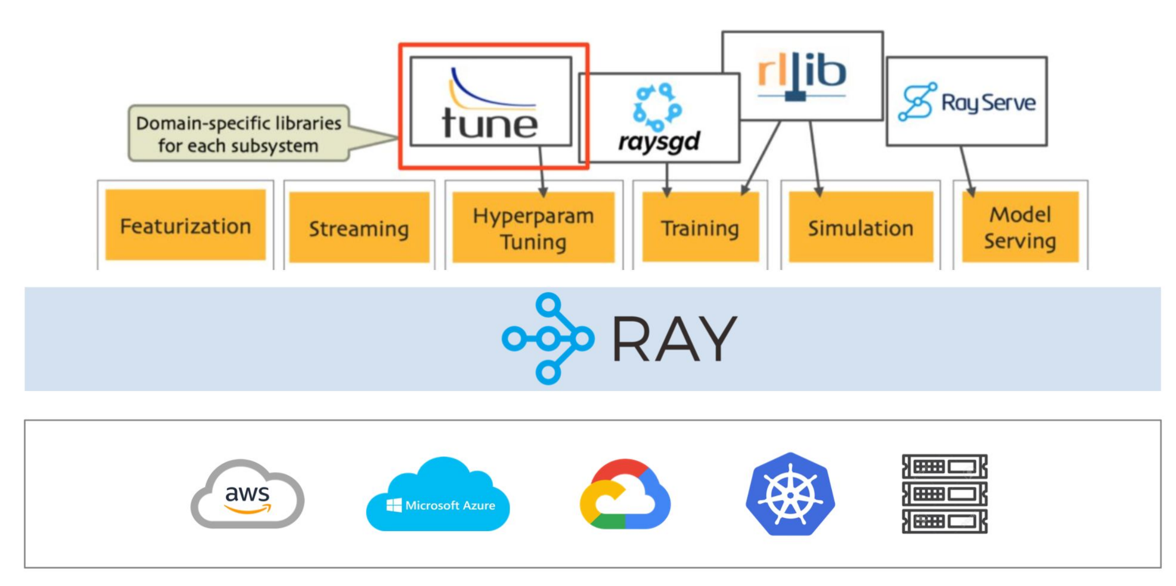


Figure taken from Anyscale website

Ray Tune

- Focus on distributed hyperparameter tuning
- Support for multiple machine learning frameworks (PyTorch, Tensorflow, sklearn, MXNet, Horovod etc.) –> **Trials in parallel on the outer loop, trials in parallel on the inner loop**
- Logging via Tensorboard (or other frameworks)
- Debugging and monitoring via Ray Dashboard
- Compatible with lots of optimization algorithms

Ray Tune Workflow

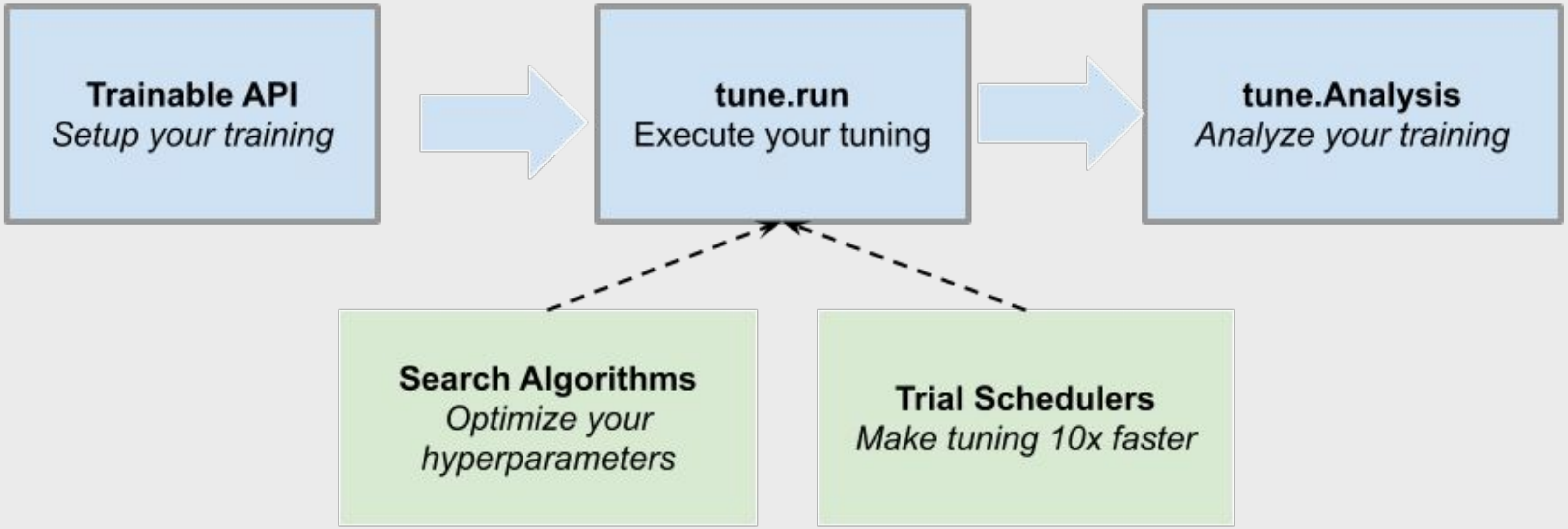


Figure taken from Anyscale website

How Does Ray Tune Distribute Work?

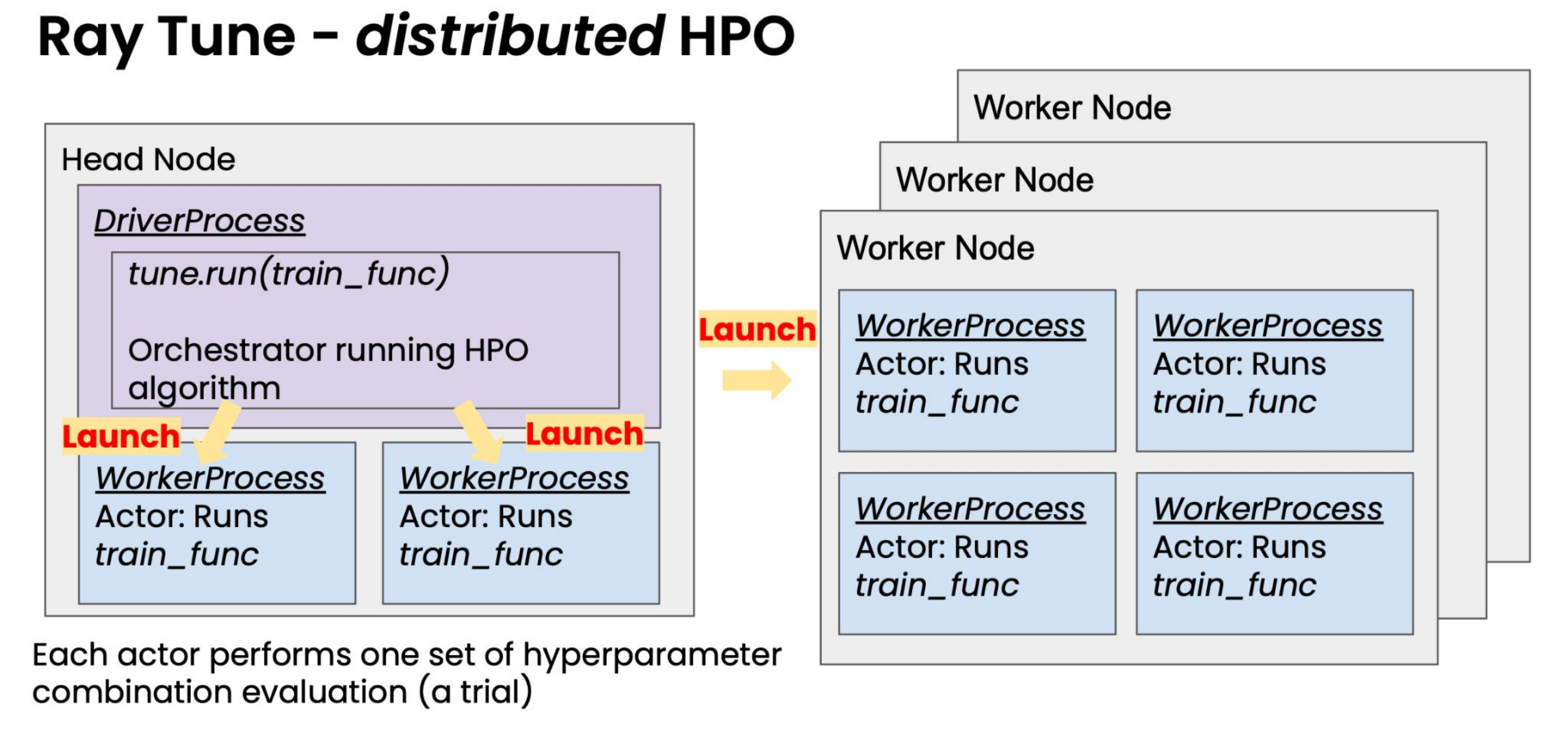


Figure taken from Anyscale website

How Does Ray Tune Distribute Work?

RAISE Center of Excellence

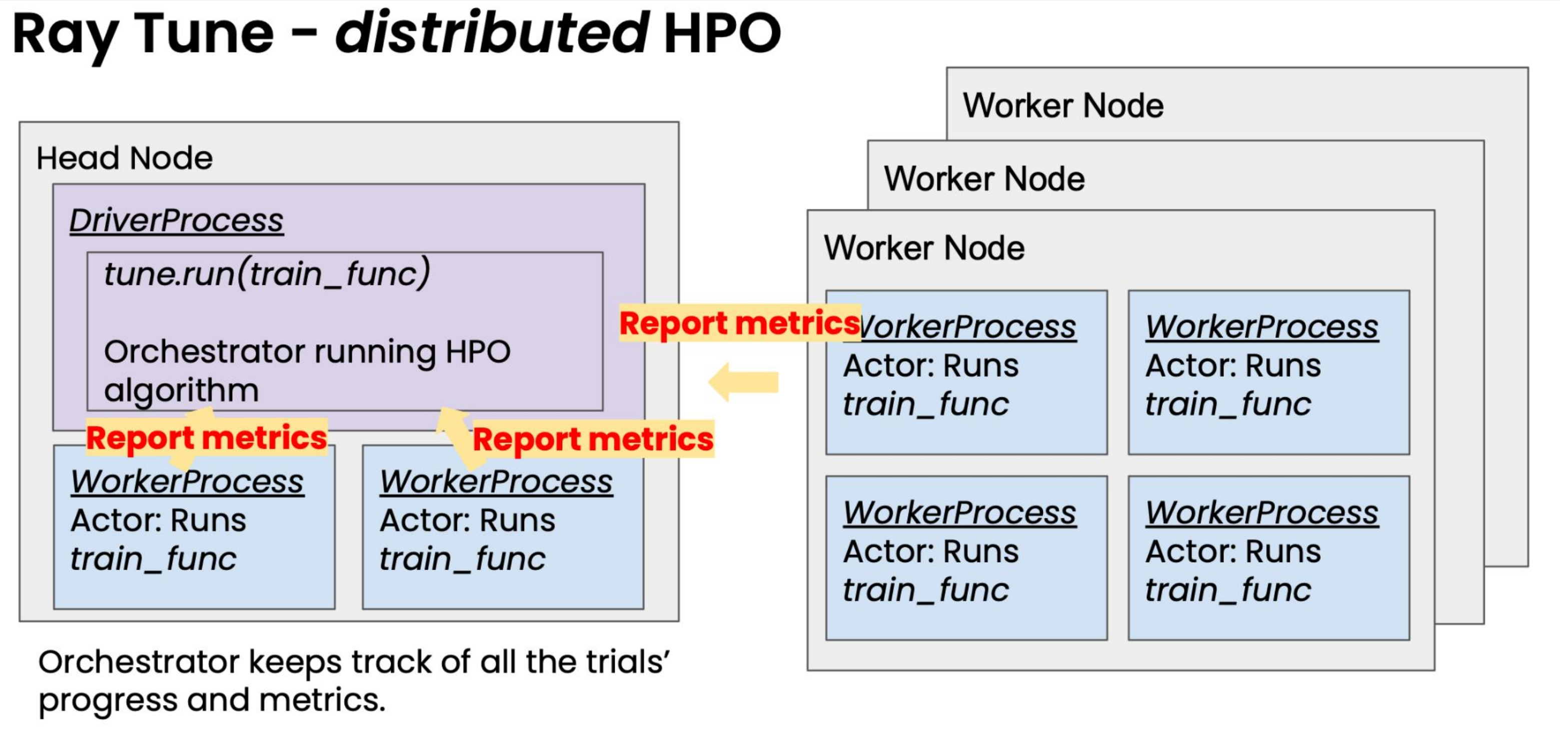


Figure taken from Anyscale website

How Does Ray Tune Distribute Work?

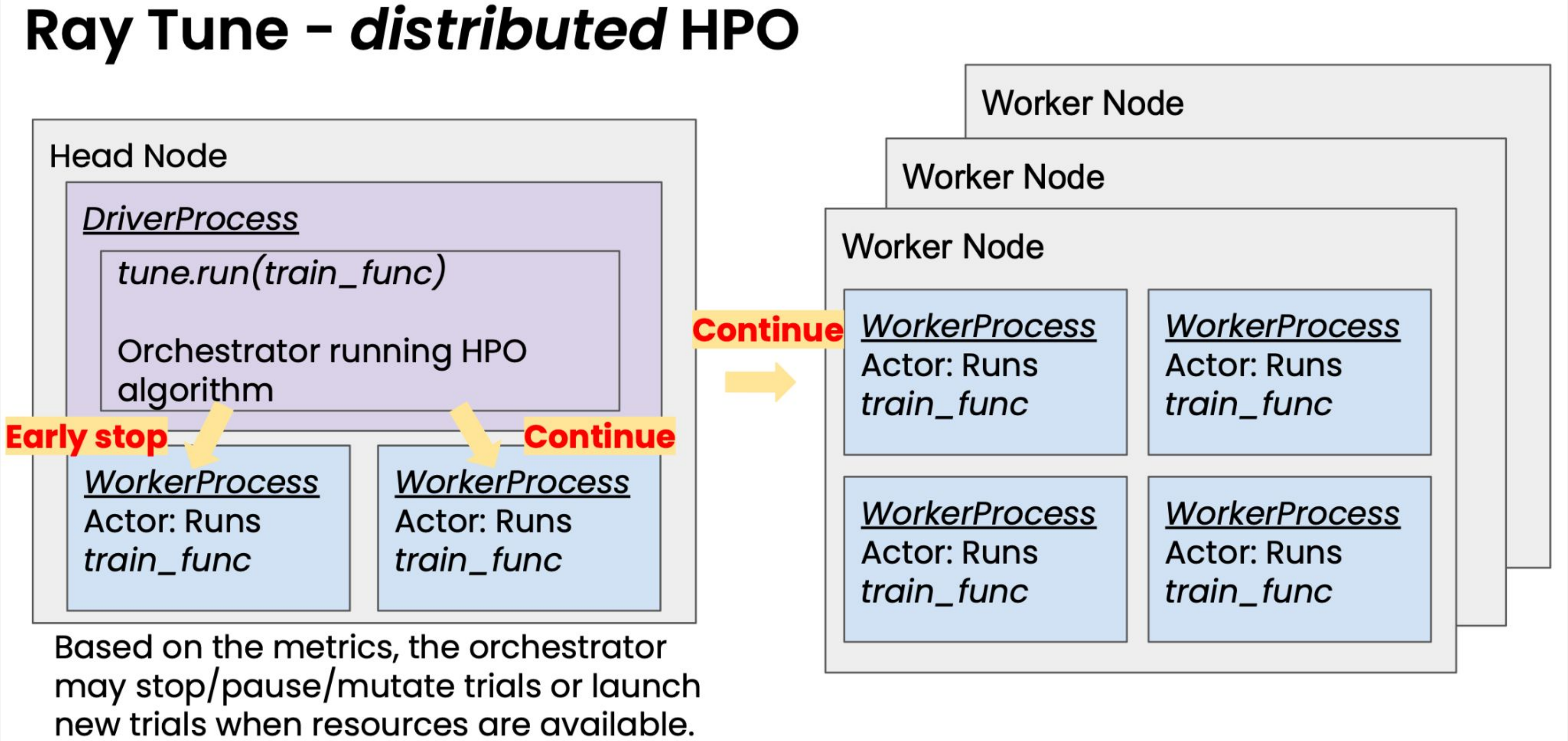


Figure taken from Anyscale website

Integrating Ray Tune

```
ray.init()

config = {
    "num_layers_conv": tune.choice([2,3,4]),
    "num_layers_linear": tune.choice([1,2,3]),
    "num_filters": tune.choice([16,32,48,64]),
    "weight_init_conv": tune.loguniform(10e-4,10e-1),
    "weight_init_linear": tune.loguniform(10e-3,1),
    "weight_decay": tune.loguniform(10e-4,1),
    "batch_size": tune.choice([64, 128, 256, 512]),
    "lr": tune.loguniform(10e-5, 1)}
```

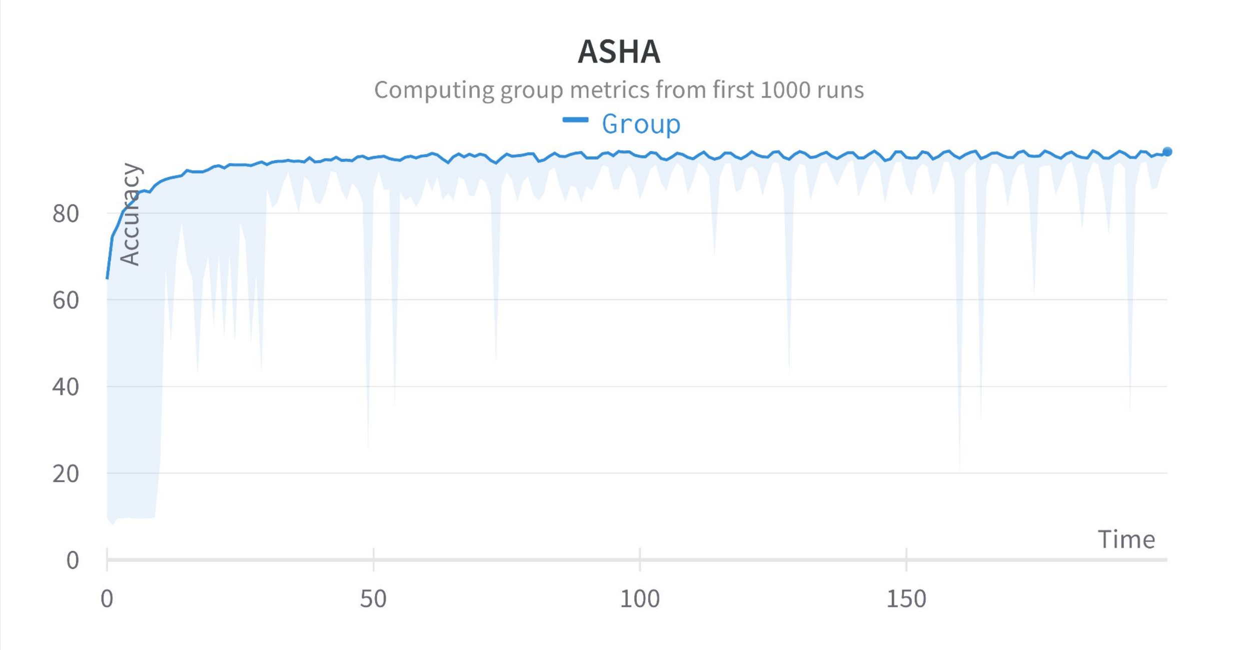
```
scheduler = ASHAScheduler(
    metric="accuracy",
    mode="max")
result = tune.run(
    function_to_train,
    resources_per_trial={"cpu": 9, "gpu": 1},
    config=config,
    num_samples=100,
    scheduler=scheduler)

ray.shutdown()
```

Ray Tune Output

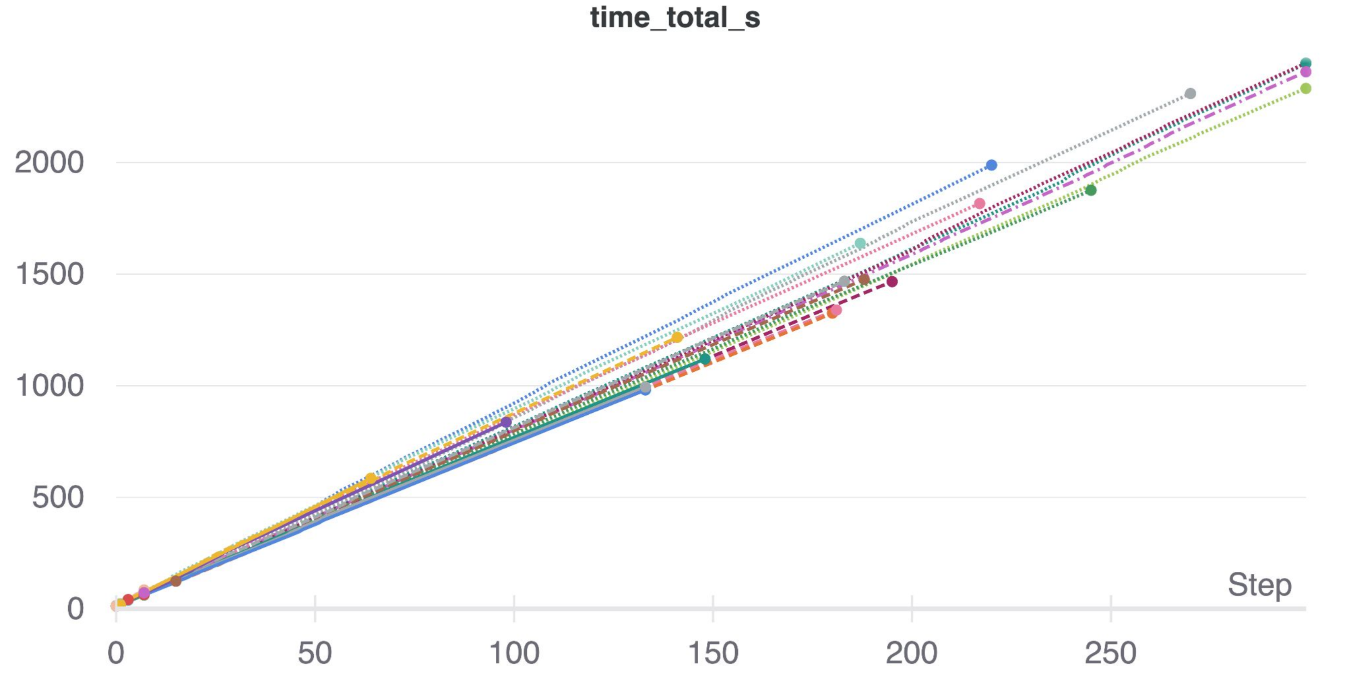
Trial name	status	loc	band_num	batch_size	lr	val_oa	val_mpca	val_miou	test_miou	training_iteration	time_total_s
train_aero_ac3b2_00032	RUNNING	10.2.17.202:20582	5	64	0.0001	0.916393	0.69643	0.600064	0.667609	35	969.47
train_aero_ac3b2_00033	RUNNING	10.2.17.219:25489	23	64	0.0001	0.890083	0.689603	0.570841	0.65651	29	890.717
train_aero_ac3b2_00034	RUNNING	10.2.17.203:22729	47	64	0.0001	0.889298	0.68893	0.575624	0.681428	26	873.42
train_aero_ac3b2_00035	RUNNING	dp-esb32i:23955	9	64	0.0001	0.899095	0.700847	0.595824	0.691052	31	888.104
train_aero_ac3b2_00036	RUNNING	10.2.17.221:375	11	64	0.0001	0.891821	0.678203	0.582706	0.619671	31	865.687
train_aero_ac3b2_00037	RUNNING	10.2.17.201:24116	46	64	0.0001	0.898306	0.680905	0.592579	0.713814	25	831.981
train_aero_ac3b2_00038	RUNNING	10.2.17.212:20998	4	64	0.0001	0.915348	0.702259	0.598816	0.714567	30	833.896
train_aero_ac3b2_00064	PENDING		4	64	0.0001						
train_aero_ac3b2_00065	PENDING		46	64	0.0001						
train_aero_ac3b2_00066	PENDING		7	64	0.0001						
train_aero_ac3b2_00067	PENDING		37	64	0.0001						
train_aero_ac3b2_00068	PENDING		32	64	0.0001						
train_aero_ac3b2_00069	PENDING		4	64	0.0001						
train_aero_ac3b2_00070	PENDING		22	64	0.0001						
train_aero_ac3b2_00000	TERMINATED	dp-esb32i:23956	11	64	0.0001	0.896212	0.671769	0.581059	0.705622	60	1789.01
train_aero_ac3b2_00001	TERMINATED	10.2.17.217:22795	28	64	0.0001	0.899371	0.688892	0.591628	0.740726	60	1924.39
train_aero_ac3b2_00002	TERMINATED	10.2.17.221:6054	14	64	0.0001	0.899756	0.683954	0.590084	0.732688	60	1800.57
train_aero_ac3b2_00003	TERMINATED	10.2.17.222:18550	25	64	0.0001	0.895409	0.674537	0.579146	0.746813	60	1932.59
train_aero_ac3b2_00004	TERMINATED	10.2.17.218:26357	9	64	0.0001	0.904167	0.677931	0.604722	0.74932	60	1857.76
train_aero_ac3b2_00005	TERMINATED	10.2.17.216:4064	26	64	0.0001	0.895551	0.674311	0.59516	0.713881	60	1896.93
train_aero_ac3b2_00006	TERMINATED	10.2.17.220:25578	18	64	0.0001	0.897533	0.677626	0.598319	0.730244	60	1857.71

... 76 more trials not shown (25 RUNNING, 25 PENDING, 25 TERMINATED)

ASHA with Ray Tune on a Supercomputer



ASHA with Ray Tune on a Supercomputer



Comparison with other Frameworks

	Open source	No cloud lock-in	Distributed	SOTA algorithms	Bring your own framework	Also runs
Ray Tune	✅	✅	✅	⭐⭐⭐	✅	HyperOpt, Optuna, SigOpt
HyperOpt	✅	✅	✅	⭐⭐	✅	
Optuna	✅	✅	❌	⭐⭐	✅	
SigOpt	❌	❌	✅	⭐	✅	
Vertex AI (Vizier)	❌	❌	✅	⭐	❌	
Sagemaker	❌	❌	✅	⭐	❌	
Azure ML	❌	❌	✅	⭐	❌	
Katib	✅	✅	✅	⭐⭐	❌	HyperOpt, Optuna
Spark ML	✅	✅	✅	❌	❌	HyperOpt

Figure taken from Anyscale website

Application in Remote Sensing

Remote Sensing Datasets

- Lots of “raw” satellite data available
- BigEarthNet-19 dataset
    - 600.000 image patches
    - Classification problem, measure the F1 micro and macro score of ML models
- **Idea: Use HPC to train classification models (and tune the hyperparameters of these models)!**

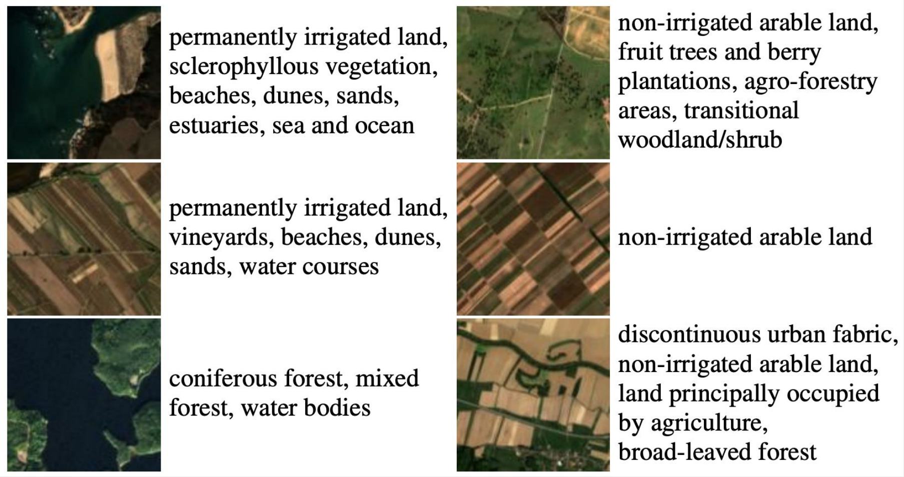


Image from [Sumbul 2019]

Scaling up Convolutional Neural Networks

- Network architecture: EfficientNet [Tan 2020]
- Distributed deep learning on HPC -> large batch size
- Investigation by [Sedona 2020]:
  - Drop in F1 score with larger batch sizes
  - Batch size: 512 -> F1 score: 0.78
  - Batch size: 8,192 -> F1 score: 0.74
  - Batch size 16,384 and larger -> divergence

Adapting the Batch Size

idea: Adapt the batch size during training!

-> Use a small batch size in the beginning to stabilize training process, use a bigger batch size afterwards for efficient resource utilization

-> Paper: "ACCELERATING HYPERPARAMETER TUNING OF A DEEP LEARNING MODEL FOR REMOTE SENSING IMAGE CLASSIFICATION" accepted at IGARSS 2022

Implementation in TensorFlow

RAISE Center of Excellence

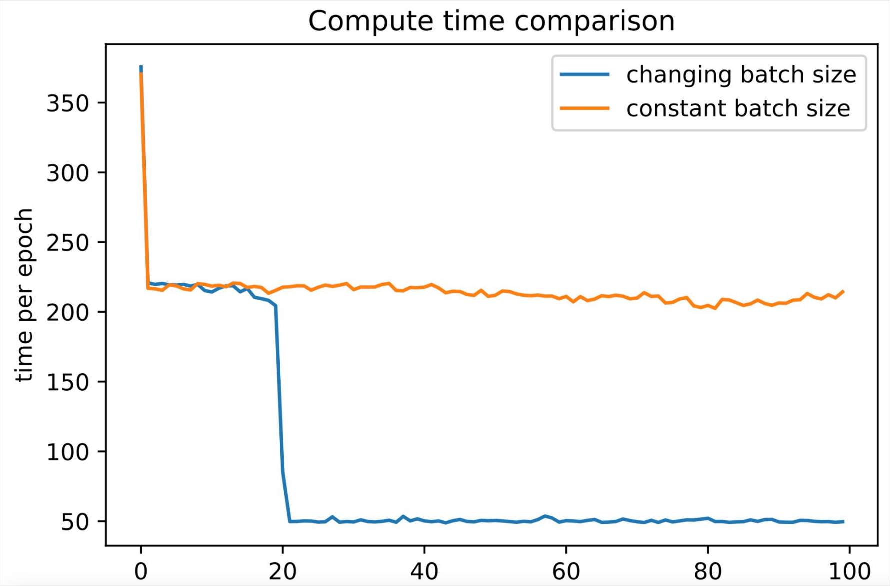
```
# training iteration loop
for batch, (images, labels) in enumerate(dataset):
    split = 32 # factor of bigger to smaller batch
    # small batch case
    if (epoch < 20):
        # split up the original big batch into
            smaller batches
        images_split = np.array_split(images,
            split)
        labels_split = np.array_split(labels,
            split)
        # call the training step on each of the
            small batches
        for i in range(split):
            loss_value = training_step(
                images_split[i], labels_split[i])
    # big batch case
    else:
        loss_value = training_step(images, labels)
```

Setup on the Supercomputer (JURECA-DC-GPU)

- 16 GPUs per trial (**data-parallel with Horovod**)
- **Batch size per GPU: 32 -> 1,024**
- **Total batch size: 512 -> 16,384**
- 6 trials in parallel (96 GPUs in total in with **Ray Tune**)
- Hyperparameter search space:
  - Learning rate in *[0.001, 1]*
  - Weight decay in *[0.0005, 0.1]*
  - Momentum in *[0, 0.9]*
  - Nesterov momentum in *[false, true]*
- No search or scheduling algorithm

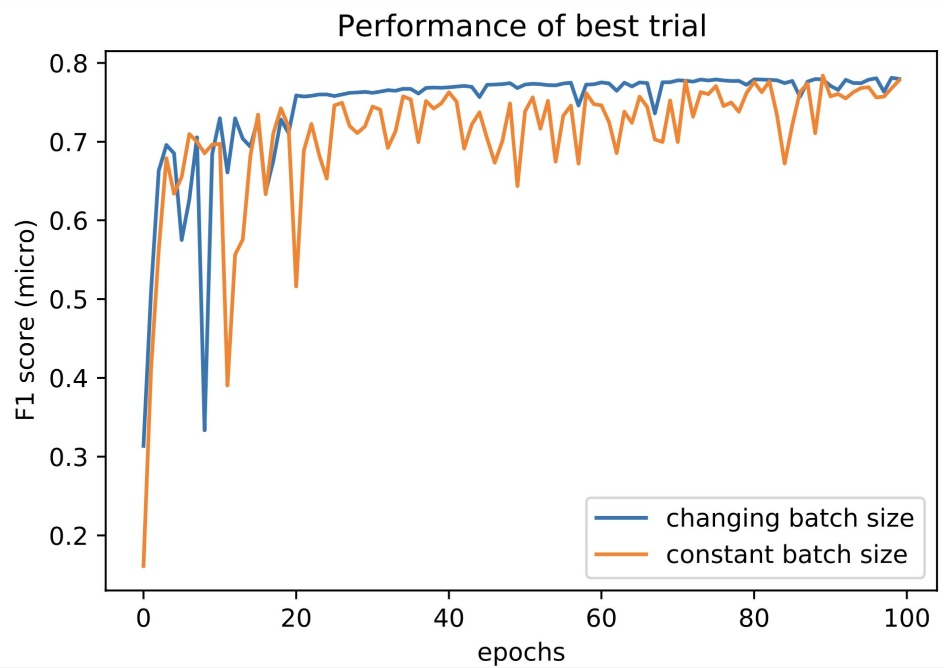
Results: Compute Time

- **4x speed-up**
- Change the batch size after 20 epochs



Results: Accuracy

BS (global)	total runtime	F1 scores
512	27 hrs	0.78 (0.72)
512->16,384	10 hrs	0.78 (0.70)



Pros and Cons of Adaptive Batch Size

- **Strengths:**
  - 3x speed-up
  - No drop in final validation accuracy
  - Efficient usage of HPC resources
  - Metrics grounded in theory (gradient noise scale by [McCandlish,2018]) exist for better batch size adaptation
- **Limitations:**
  - So far just tested on one RS dataset
  - Does not address issue of late learners

Summary Distributed HPO

- Optimize the **“outer loop”**
- Exploit full level of parallelism
  - Distributed deep learning on the “inner loop”
  - Distributed HPO on the “outer loop”
- Easy handling with Ray Tune library (also on HPC systems)
- Methods can be adapted to remote sensing use-cases

drive. enable. innovate.

RAISE Center of Excellence

The CoE RAISE project have received funding from the European Union’s Horizon 2020 – Research and Innovation Framework Programme H2020-INFRAEDI-2019-1 under grant agreement no. 951733

Follow us: