Hyperparameter Optimization on High Performance Computing systems

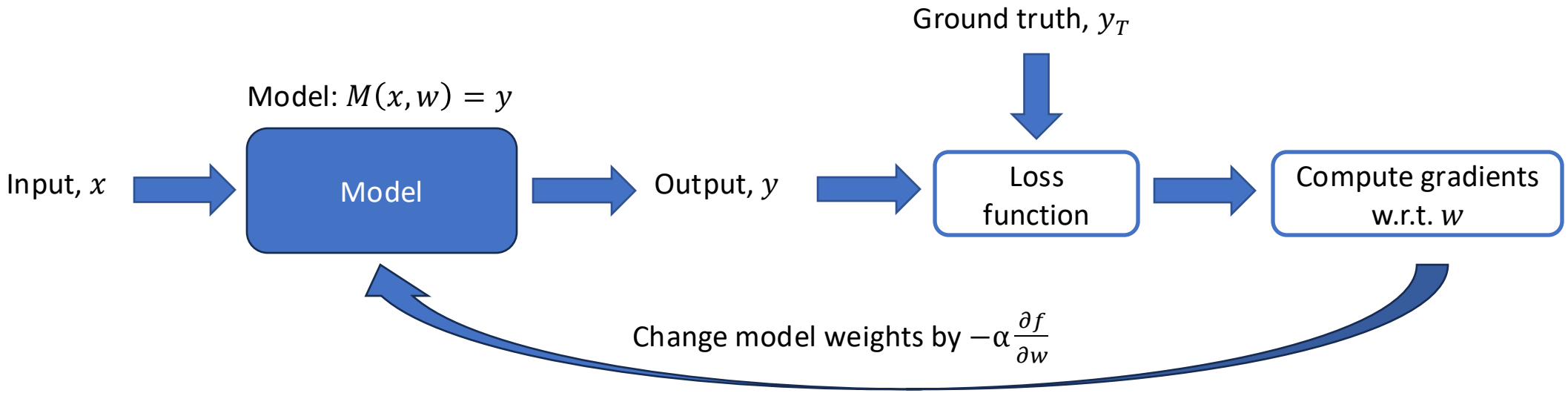
Eric Wulff

Computing Engineer, CERN

CERN openlab

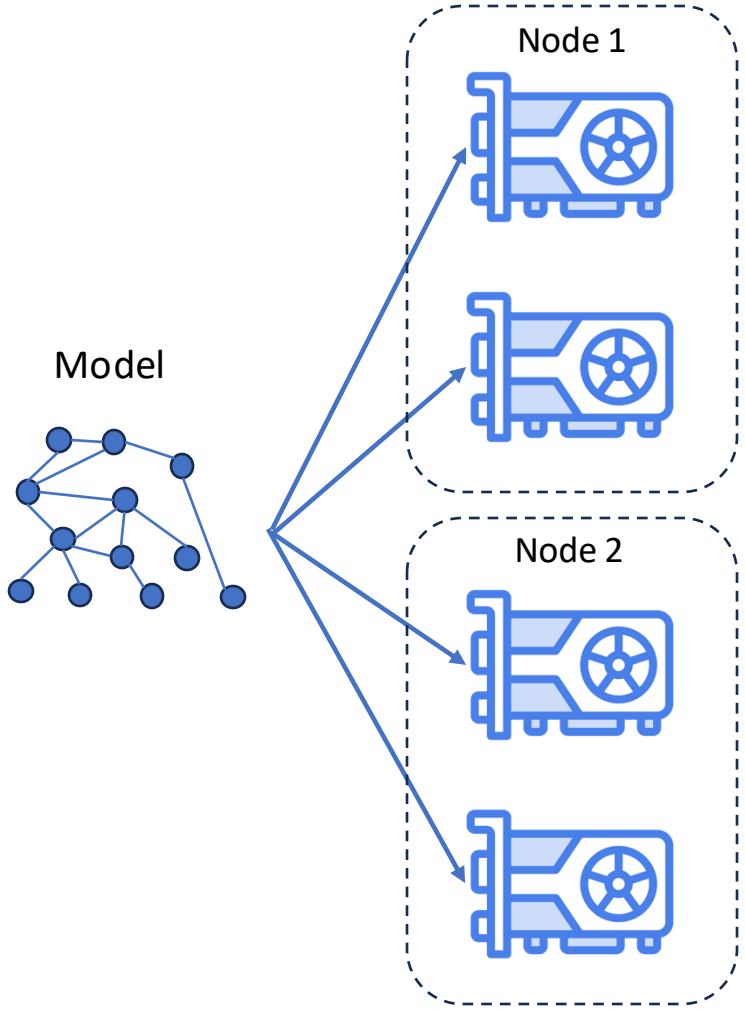
Training deep learning models

- In DL, model parameters w are learned using backpropagation and gradient descent to minimize some objective, or loss, $f(w, \theta)$
- Training:
  - For each x in training data, compute the gradients of the loss and change the model's weights by subtracting from them the gradients multiplied by some learning rate, α
  - Repeat until convergence or reaching some other stopping criterion



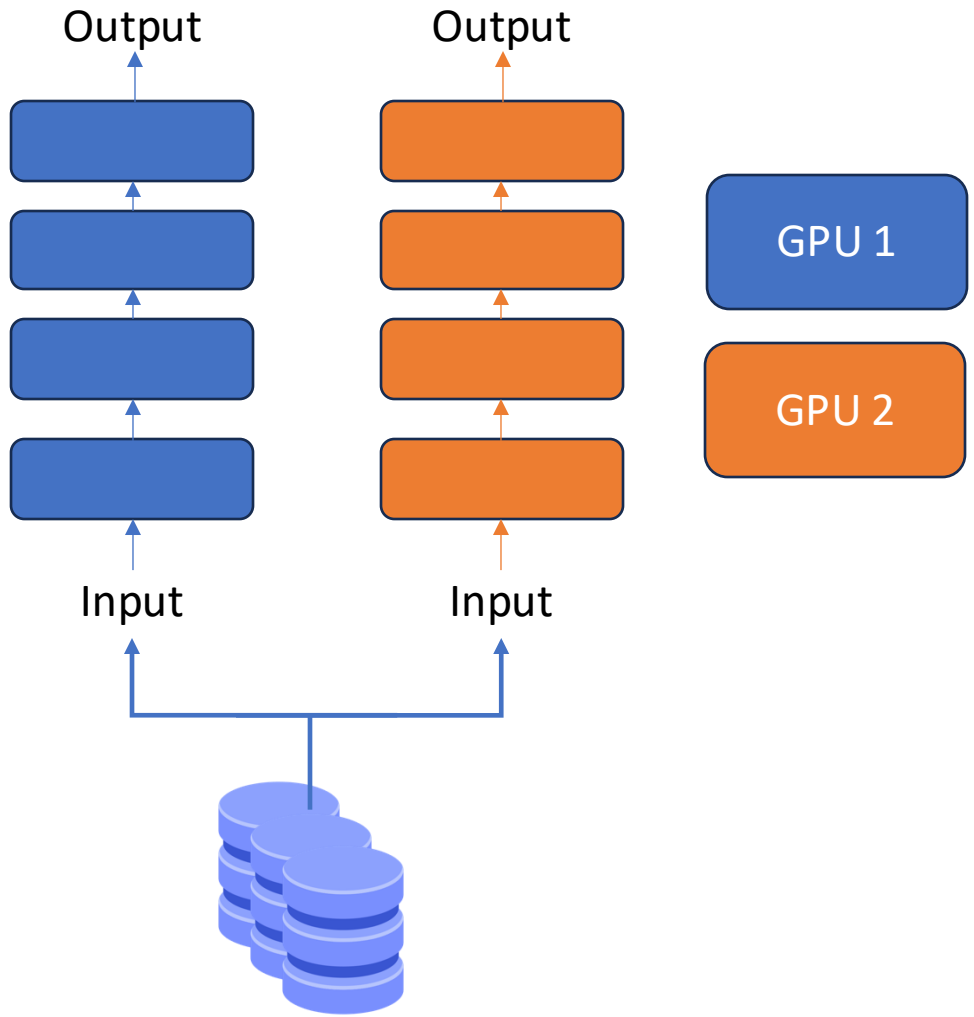
Distributed training

- What is Distributed Training?
  - Training models across multiple devices/nodes
  - Enables scaling to larger models and datasets
- Why It Matters:
  - Faster training times
  - Faster development cycles
  - Overcome memory constraints
- Examples of distributed strategies:
  - Data parallel
  - Fully sharded data parallel, ZeRO-3, ZeRO-2, ZeRO-1
  - Tensor parallel
  - Pipeline parallel
  - Etc.



Data parallelism

- Process :
  - Replicate the same model across multiple devices
  - Each device processes a different subset of data
  - Each device holds a full copy of the model but trains on different mini-batches
  - Gradients are averaged and synchronized across workers in each optimization step
- Benefits:
  - Straightforward implementation
  - Scales well with data size
- Drawbacks:
  - Model size limits due to memory constraints



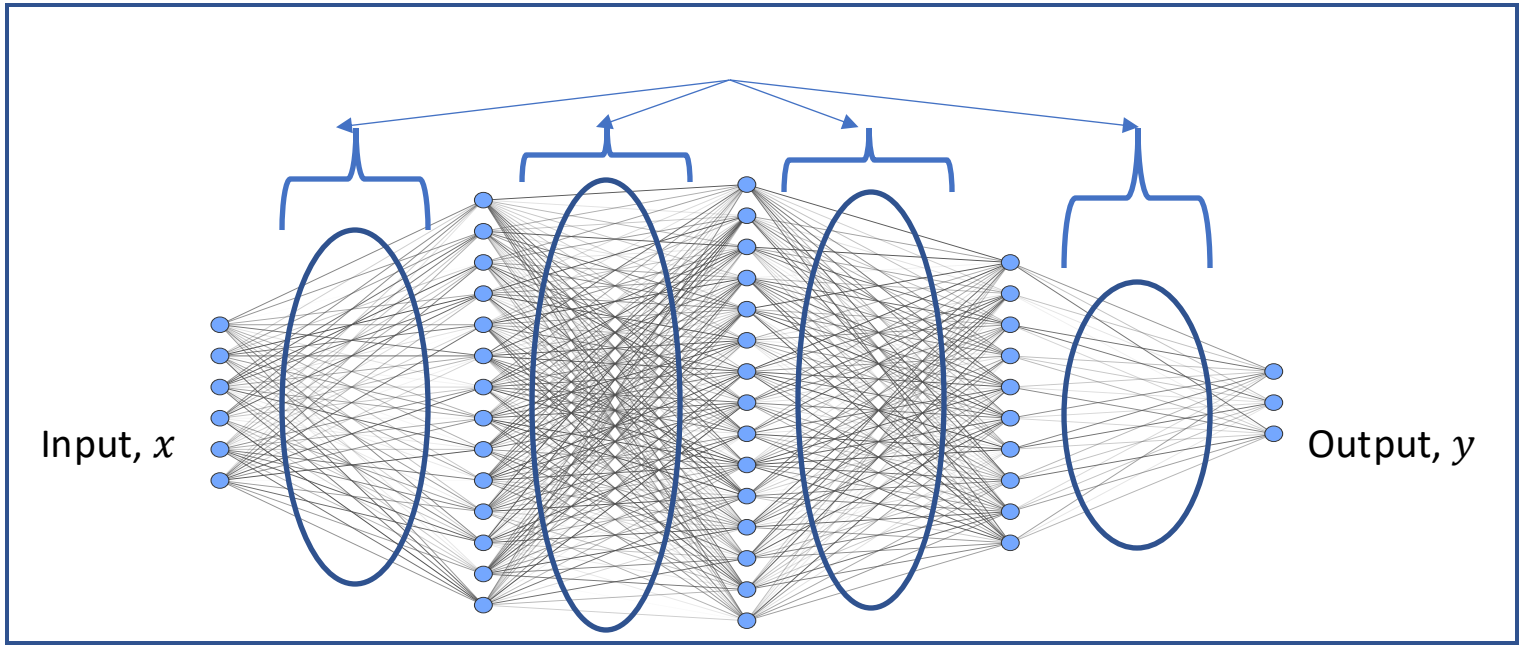
What is hyperparameter optimization? (1/3)

- $f(w, \theta)$, depends not only on w, but also on θ
  - w: Model parameters
  - θ: Hyperparameters
    - Number of layers
    - Number of nodes
    - Choice of optimizer, learning rate, batch size, etc.

- Hyperparameter optimization (HPO) is the process of tuning θ to improve performance

f is the final validation loss after completed training, f is not the model itself

Model: $M(x, w) = y$

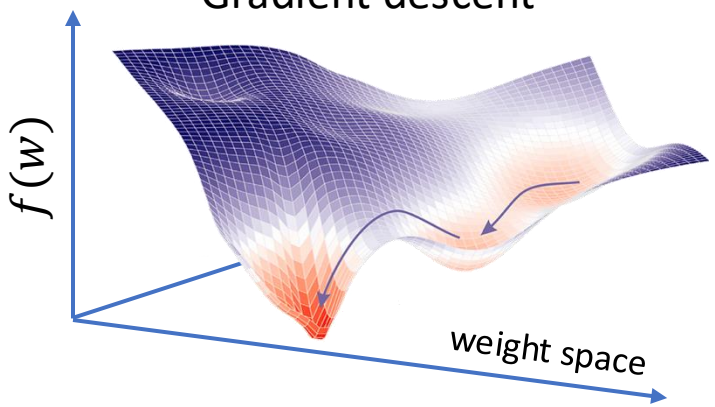


What is hyperparameter optimization? (2/3)

- Optimizing the objective $f(w, \theta)$ is done in two different ways
  1. Training: Optimize f w.r.t. w by gradient descent $\Longrightarrow$ search for $w^* = \arg\min_w f(w, \theta)$
  2. HPO: Optimize f w.r.t. $\theta \Longrightarrow$ search for $\theta^* = \arg\min_\theta f(w, \theta)$

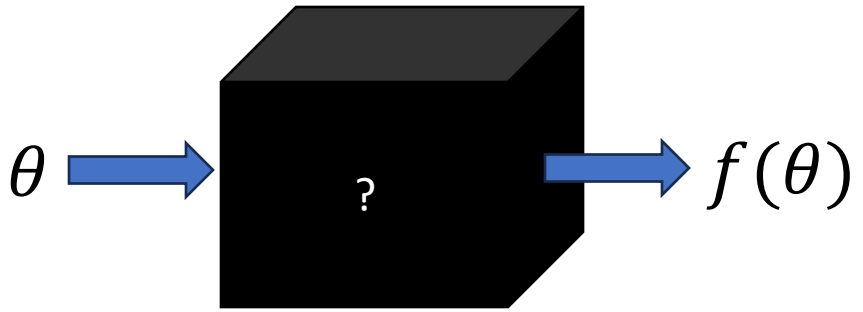
$f(w, \theta)$ is differentiable w.r.t. w

Gradient descent



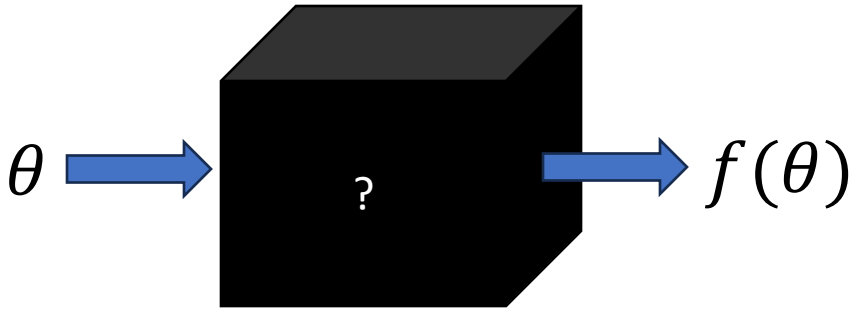
$f(w, \theta)$ non-differentiable w.r.t. θ

- Black-box optimization
- No straightforward update rule for θ



What is hyperparameter optimization? (3/3)

- HPO is a black box optimization problem, we want to find $\theta^* = \arg\min_{\theta} f(w, \theta)$ but only get to query values of f, not compute its gradient w.r.t. θ
  - w: Model parameters (learned by gradient descent)
  - θ: Hyperparameters
  - $f(w, \theta)$: What we're trying to minimize, e.g., loss
  - f is non-differentiable w.r.t. θ
- f is often expensive to evaluate
- HPO is compute-resource intensive
  - Benefits greatly from HPC resources
  - In need of smart, efficient search algorithms



Some popular HPO algorithms

- Search algorithms
  - Model free:
    - Grid search
    - Random search
    - Evolutionary search
  - Model-based:
    - Bayesian optimization
- Scheduling algorithms
  - Successive Halving (SHA)
  - Hyperband
  - Asynchronous SHA (ASHA)

Static configuration ***selection***

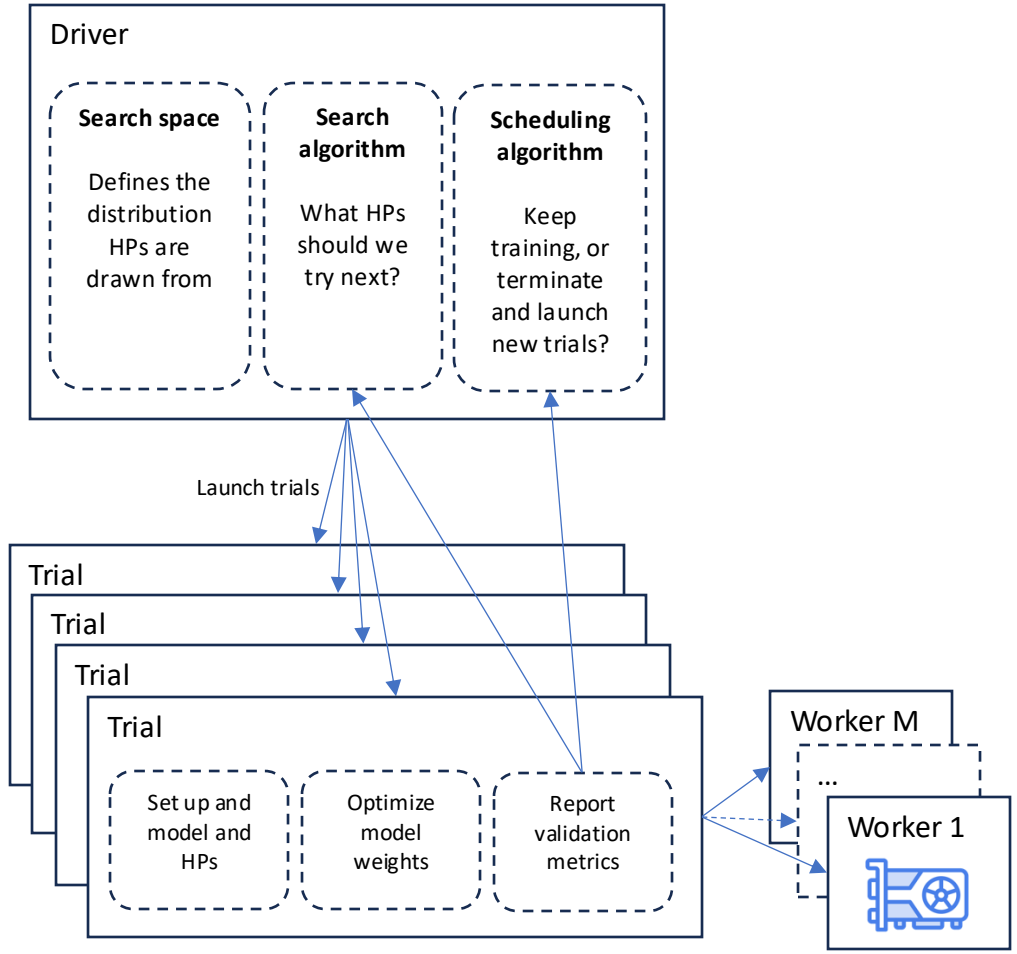
Adaptive configuration ***selection***

Adaptive configuration ***evaluation***

Selection- and evaluation-based algorithms can easily be combined

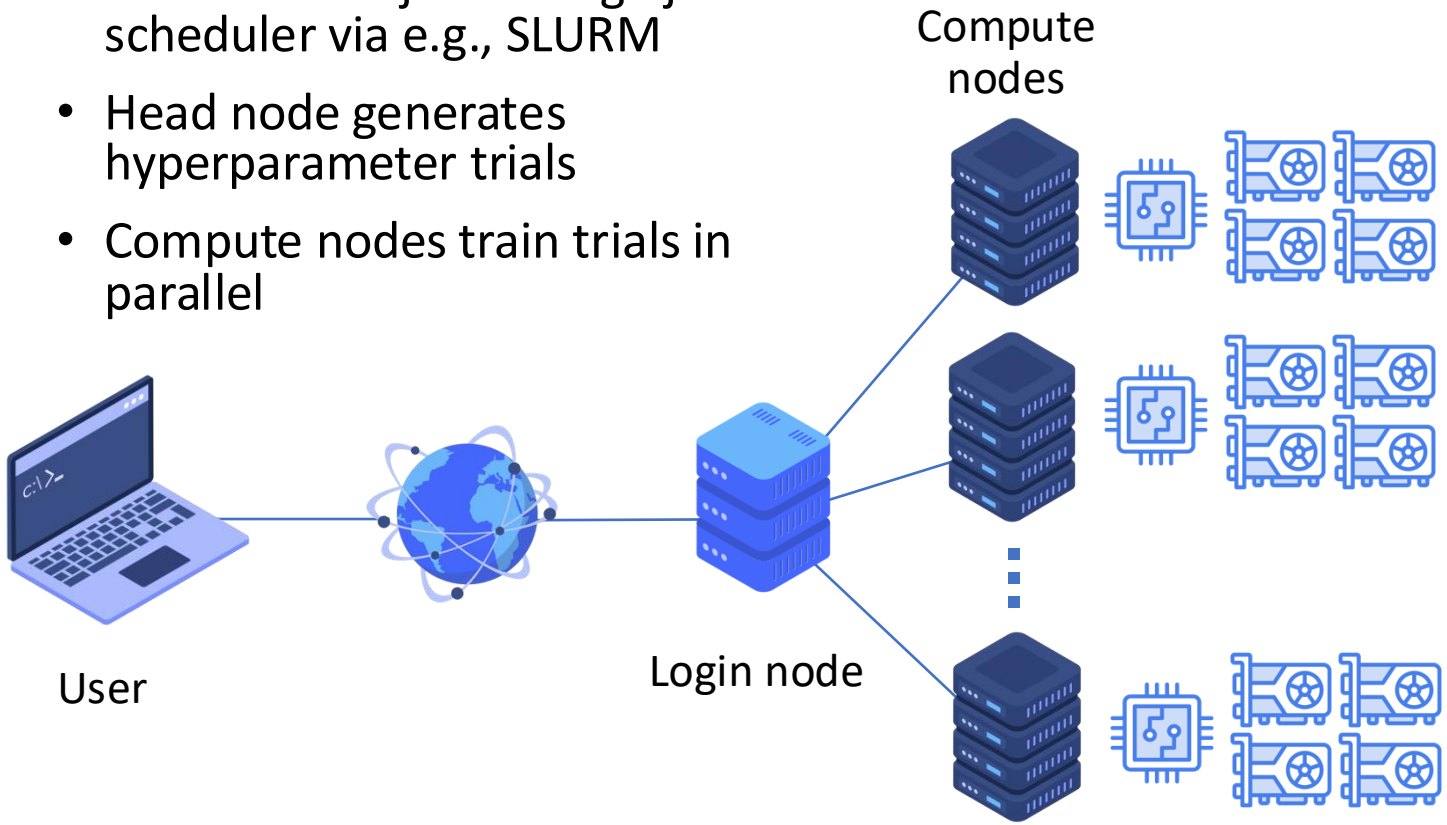
Distributed HPO workflow

- Driver
  - Defines search space
  - Generates hyperparameter trials
  - Launches, monitors, and terminates trials
- Trials
  - Sets up model and HPs
  - Distributes model training across M workers
  - Reports metrics back to driver



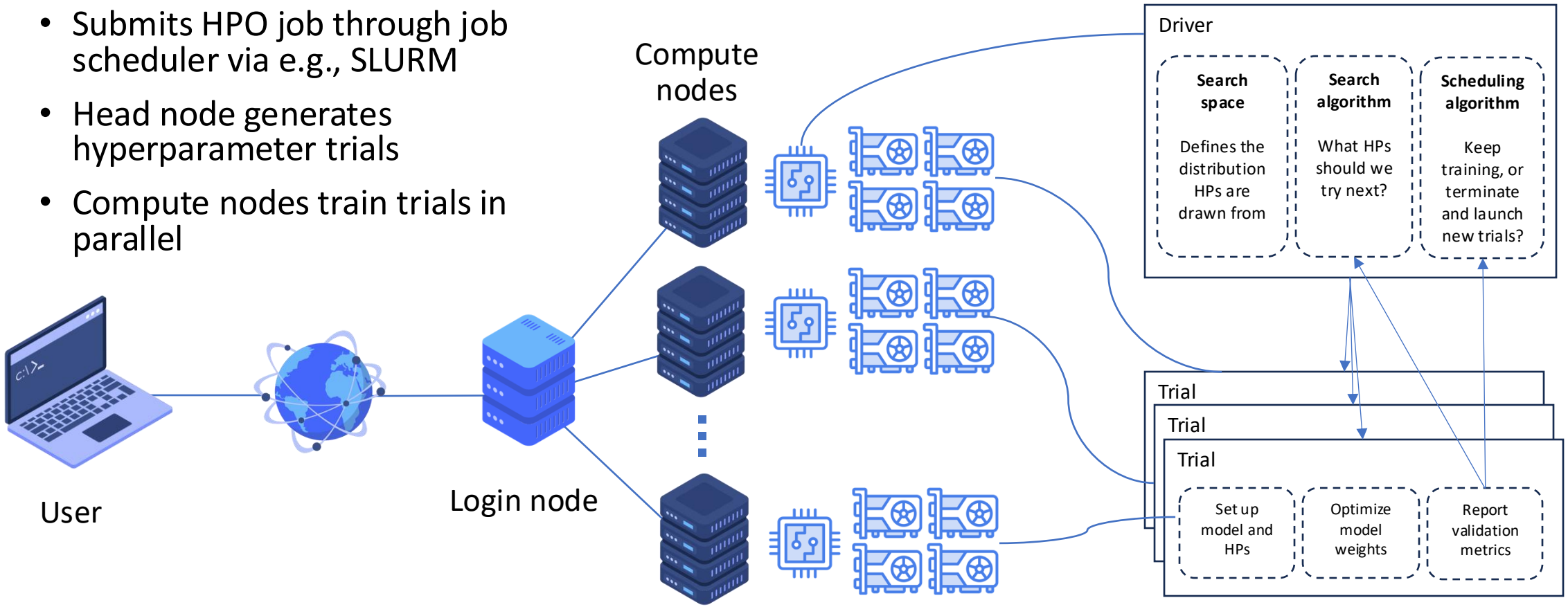
Distributed HPO on HPC systems

- User connects to HPC via ssh
- Submits HPO job through job scheduler via e.g., SLURM
- Head node generates hyperparameter trials
- Compute nodes train trials in parallel



Distributed HPO on HPC systems

- User connects to HPC via ssh
- Submits HPO job through job scheduler via e.g., SLURM
- Head node generates hyperparameter trials
- Compute nodes train trials in parallel



Distributed training and hyperparameter optimization for MLPF

Improved particle-flow event reconstruction with scalable neural networks for current and future particle detectors

Joosep Pata[1*], Eric Wulff[2], Farouk Mokhtar[3], David Southwick[2], Mengke Zhang[3], Maria Girone[2], Javier Duarte[3]

[1*]National Institute of Chemical Physics and Biophysics (NICPB), Rävala pst 10, 10143 Tallinn, Estonia.
[2]European Center for Nuclear Research (CERN), CH 1211, Geneva 23, Switzerland.
[3]University of California San Diego, La Jolla, CA 92093, USA.

*Corresponding author(s). E-mail(s): joosep.pata@cern.ch;
Contributing authors: eric.wulff@cern.ch; fmokhtar@ucsd.edu; david.southwick@cern.ch; mezhang@ucsd.edu; maria.girone@cern.ch; jduarte@ucsd.edu;

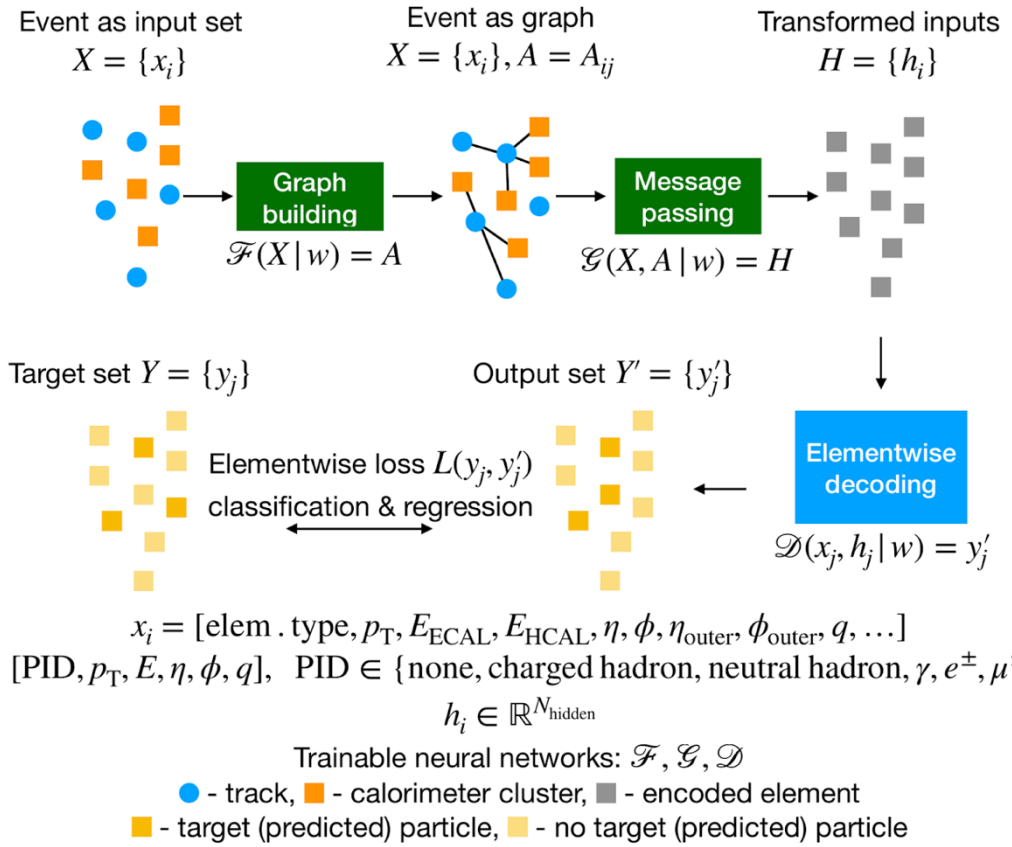
Pata, J., Wulff, E., Mokhtar, F. et al. Improved particle-flow event reconstruction with scalable neural networks for current and future particle detectors. Commun Phys 7, 124 (2024). https://doi.org/10.1038/s42005-024-01599-5

Check out the previous talk by Joosep Pata for more details on the MLPF model and datasets https://indico.cern.ch/event/1440389/#6-machine-learned-particle-flo

Machine-Learned Particle Flow (MLPF)

- The Particle Flow (PF) Algorithm [1]
  - Tries to identify and reconstruct all stable individual particles from collision events by combining information from different subdetectors (tracks, calorimeter clusters)
- Machine-Learned Particle-Flow (MLPF) [2]
  - GPU accelerated, NN-based algorithm for PF
  - Code available on GitHub
  - See ACAT2021 talk by J. Pata (and proceedings) for more MLPF model details and ACAT 2021 talk by E. Wulff (and proceedings) for more details on the HPO of MLPF
  - See Nature Commun Phys 2024 paper for latest published results

The MLPF model

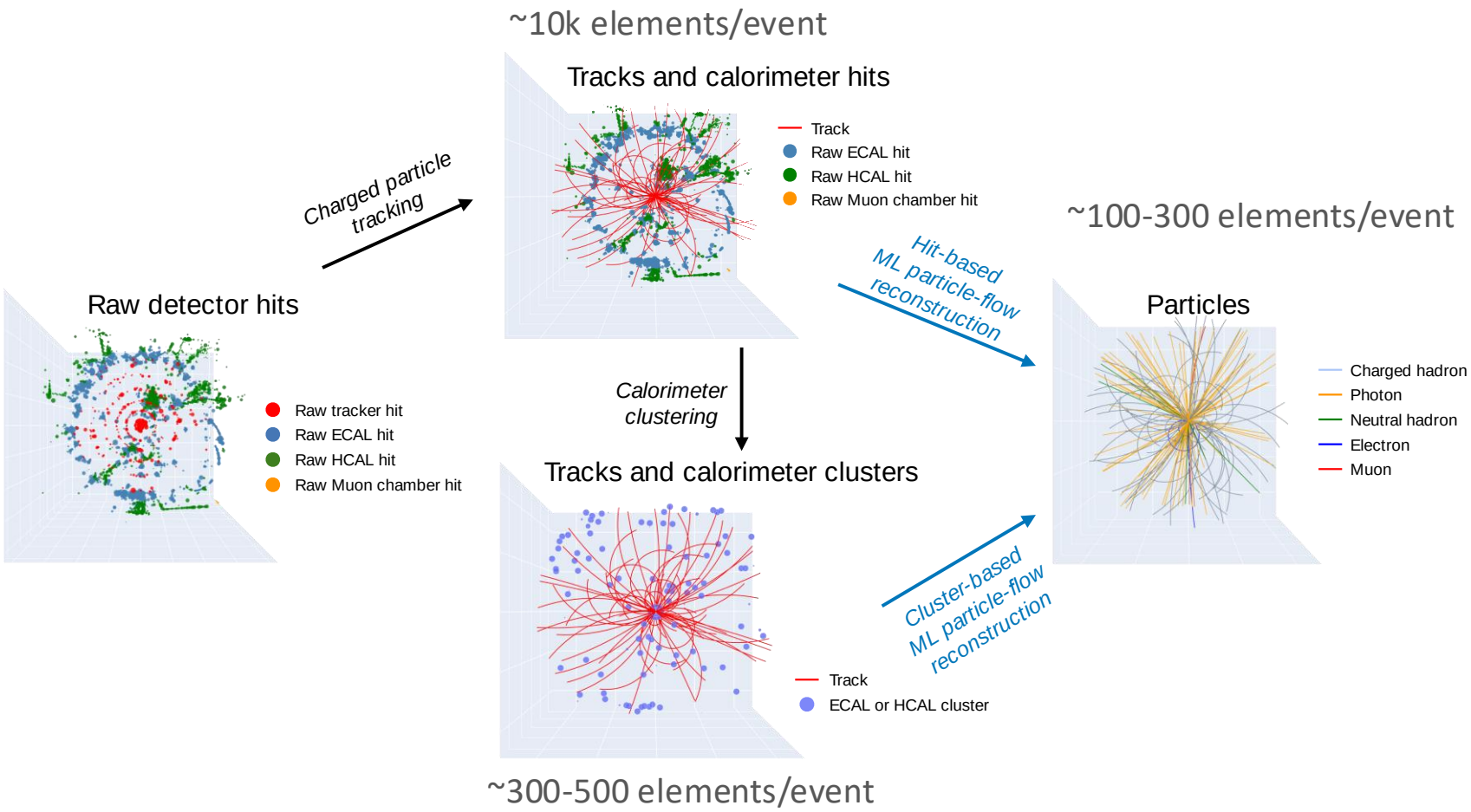


[1] CMS Collaboration https://cds.cern.ch/record/1194487?ln=en

[2] Pata, J., Duarte, J., Vlimant, JR. *et al.* MLPF: efficient machine-learned particle-flow reconstruction using graph neural networks. *Eur. Phys. J. C* **81,** 381 (2021). https://doi.org/10.1140/epjc/s10052-021-09158-w

Open dataset for supervised learning

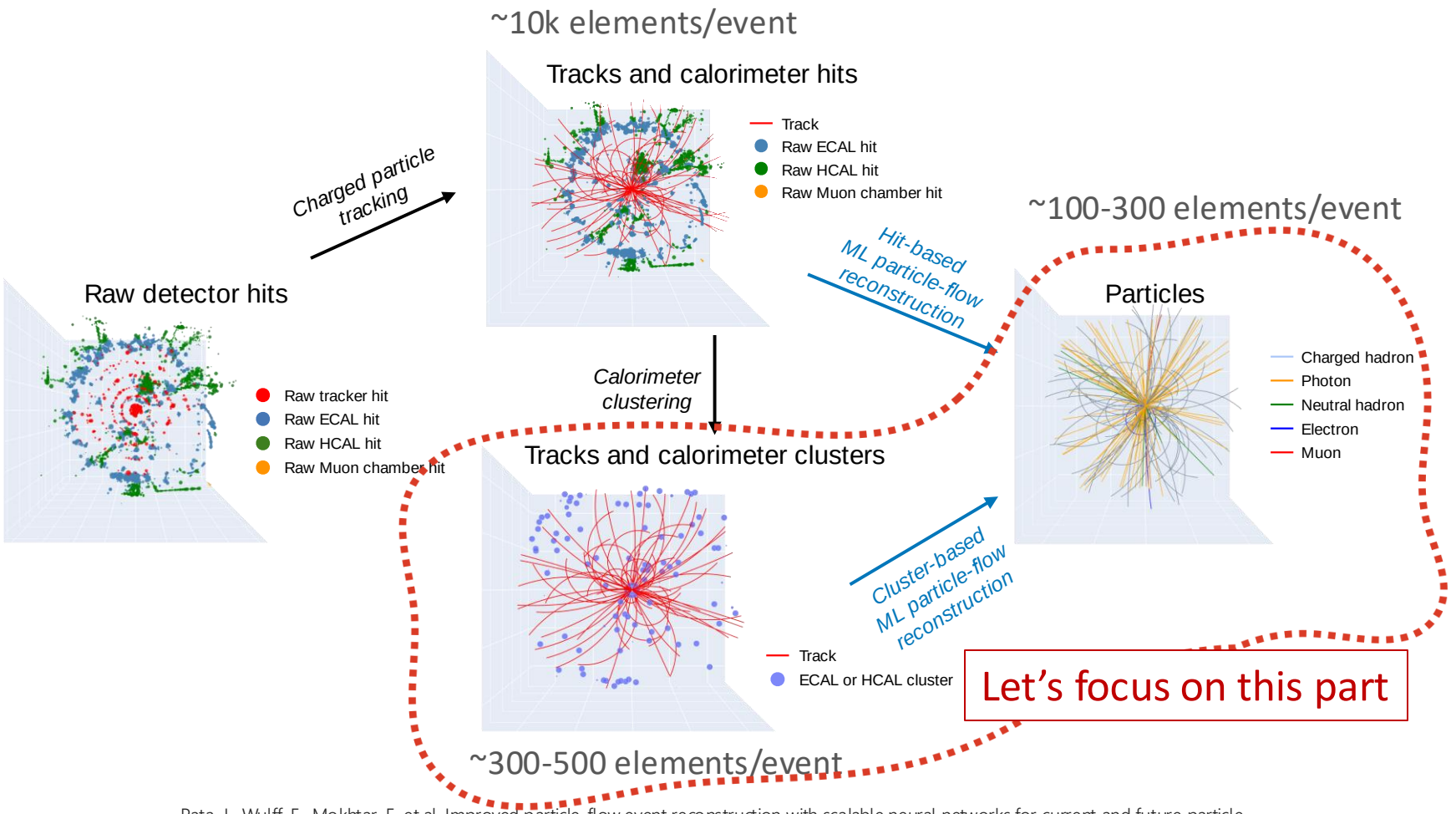
- Full detector simulation using GEANT4
- Electron-positron collision in CLIC detector geometry
- Dataset contains
  - Calorimeter and tracker hits
  - Tracks and calorimeter clusters
  - Generator-level particles (ground truth for supervised learning)
  - Baseline reconstructed particles (from a non-ML PF algo)



Pata, J., Wulff, E., Mokhtar, F. et al. Improved particle-flow event reconstruction with scalable neural networks for current and future particle detectors. Commun Phys 7, 124 (2024). https://doi.org/10.1038/s42005-024-01599-5

Open dataset for supervised learning

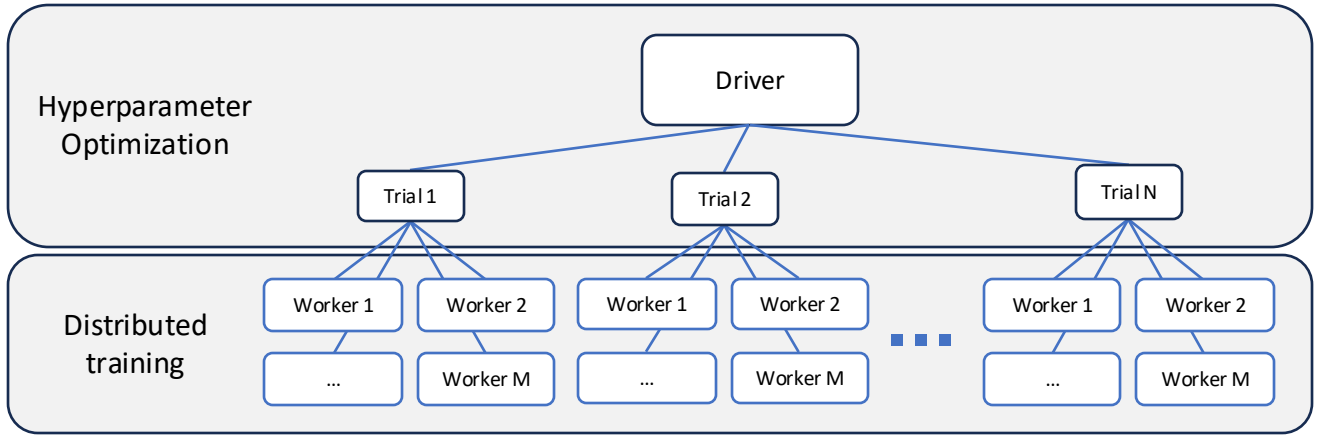
- Full detector simulation using GEANT4
- Electron-positron collision in CLIC detector geometry
- Dataset contains
  - Calorimeter and tracker hits
  - Tracks and calorimeter clusters
  - Generator-level particles (ground truth for supervised learning)
  - Baseline reconstructed particles (from a non-ML PF algo)

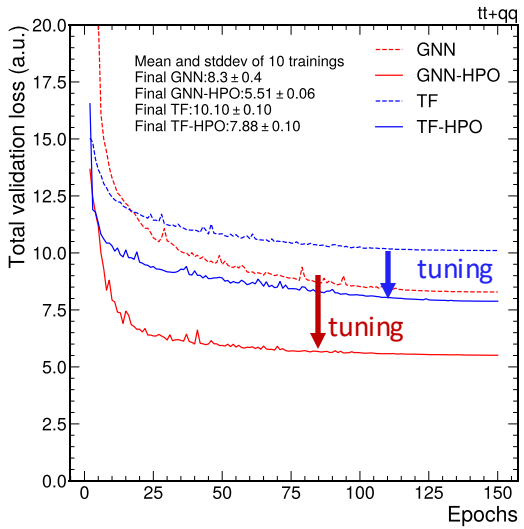


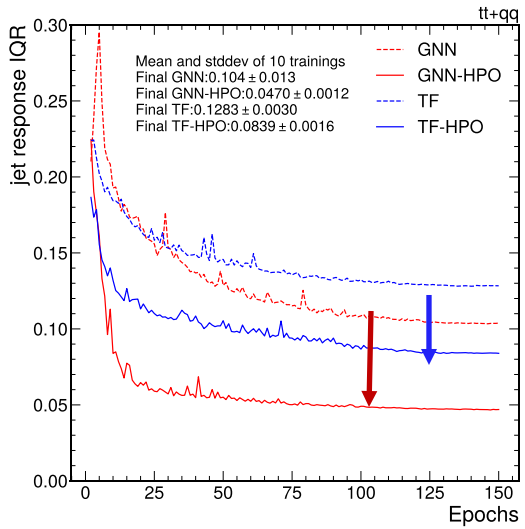
Pata, J., Wulff, E., Mokhtar, F. et al. Improved particle-flow event reconstruction with scalable neural networks for current and future particle detectors. Commun Phys 7, 124 (2024). https://doi.org/10.1038/s42005-024-01599-5

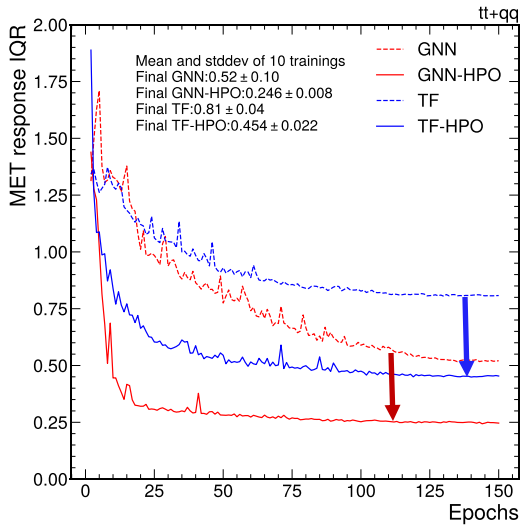
Distributed hyperparameter optimization

- Two levels of parallelization
- Using ASHA + Bayesian Optimization for HPO
- Final validation loss decreased by **~34%** giving a significant performance improvement from HPO
- Run on 96 A100 GPUs spread across 24 nodes at the Jülich Supercomputer Center
- Around 5000 GPU-hours









Pata, J., Wulff, E., Mokhtar, F. et al. Improved particle-flow event reconstruction with scalable neural networks for current and future particle detectors. Commun Phys 7, 124 (2024). https://doi.org/10.1038/s42005-024-01599-5

Summary

- HPC systems provide large numbers of powerful GPUs suitable for AI training
- Distributed training speeds up training and model development cycles
- HPO is well suited for distribution across many compute nodes
- Large-scale distributed HPO significantly increased model performance in the example of MLPF

Thank you!

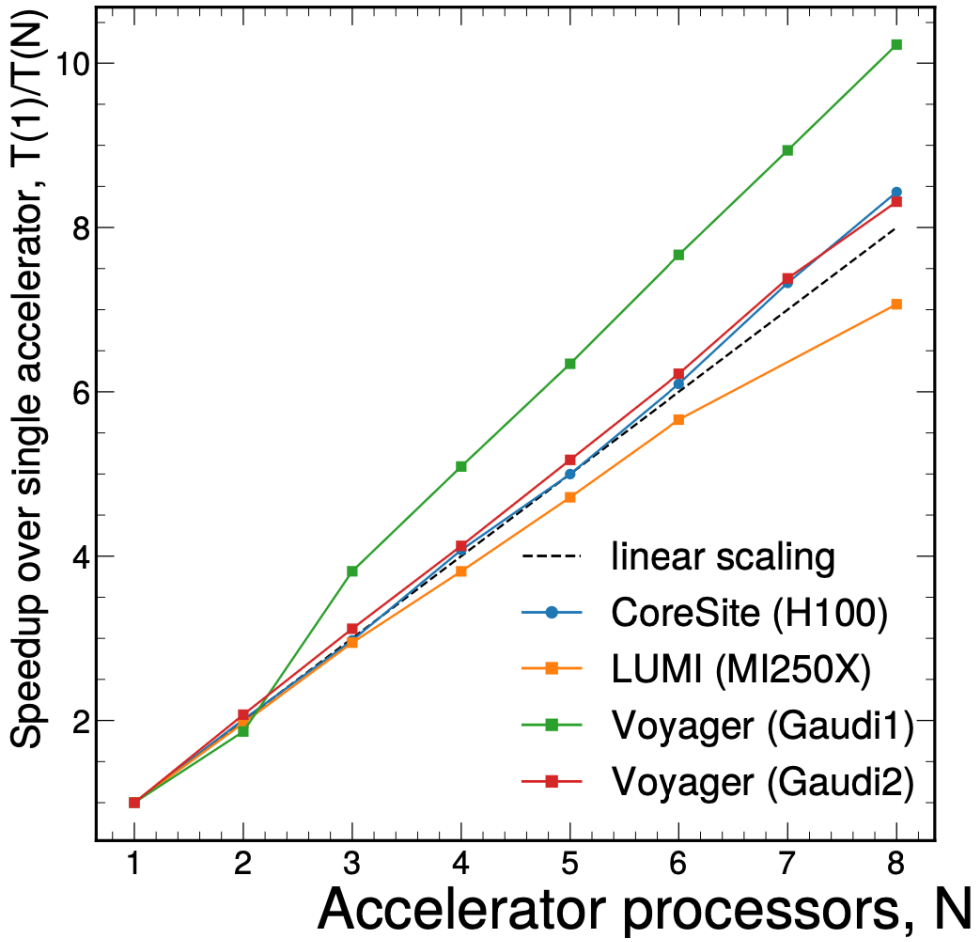
CERN openlab

Backup slides

CERN openlab

Distributed training

- Distributed multi-GPU training allows faster development cycle
- The HPC AI chip landscape is diversifying, and we need flexible and portable codes to make use of these resources
- MLPF distributed training is portable and runs on NVIDIA and AMD GPUs as well as Intel Habana Gaudi cards

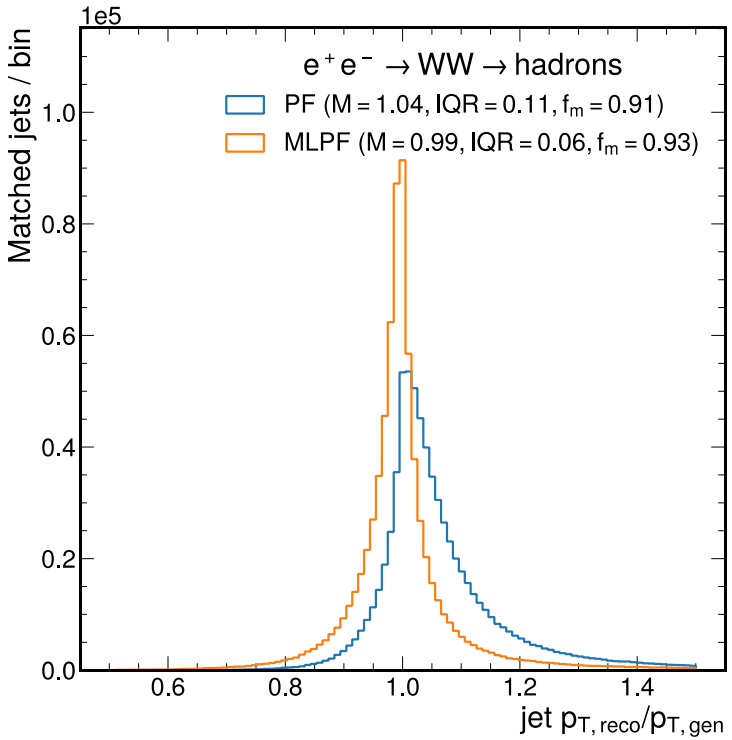


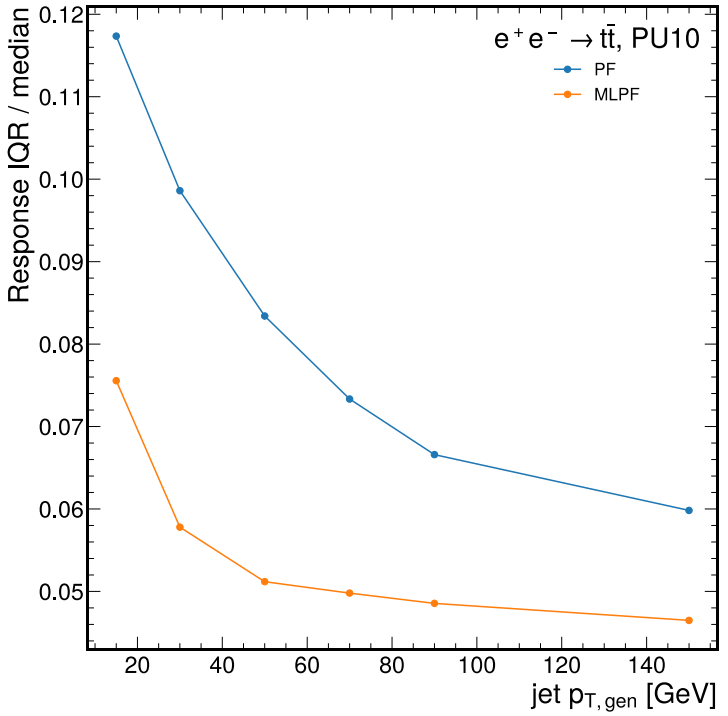
Pata, J., Wulff, E., Mokhtar, F. et al. Improved particle-flow event reconstruction with scalable neural networks for current and future particle detectors. Commun Phys 7, 124 (2024). https://doi.org/10.1038/s42005-024-01599-5

Jet resolution improvements over the baseline in test set

Even though we optimize a **per-particle**-loss, better than STOTA **event** reconstruction emerges naturally!

- Using data never seen in training
- Almost **50% improvement** in jet response width over the baseline
- Consistent improvement over the entire p_T spectrum



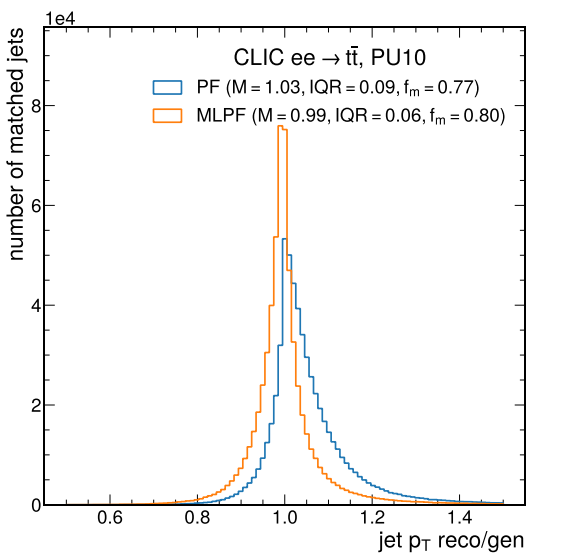


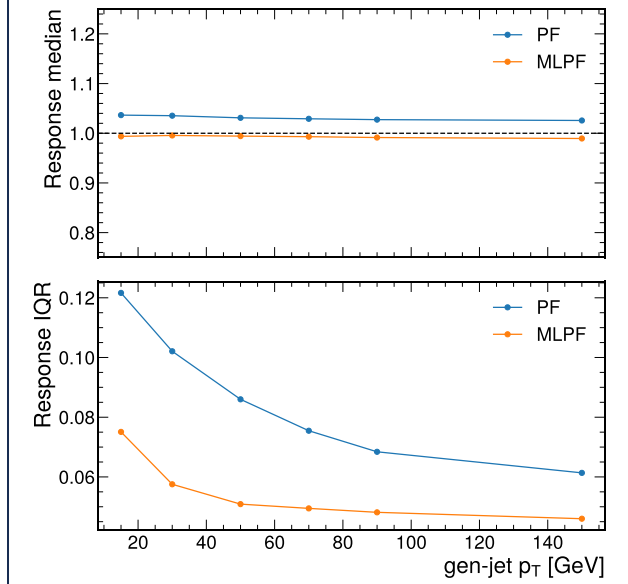
Pata, J., Wulff, E., Mokhtar, F. et al. Improved particle-flow event reconstruction with scalable neural networks for current and future particle detectors. Commun Phys 7, 124 (2024). https://doi.org/10.1038/s42005-024-01599-5

Jet and MET in ttbar + PU10 test data

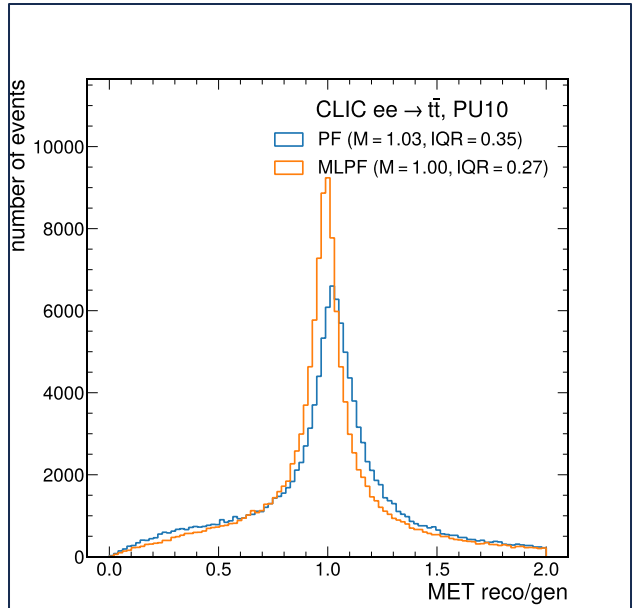
- For all test samples MLPF outperforms PF in Jet and MET reconstruction in terms of response width (quantified by median and interquartile range (IQR))
- MLPF also outperforms PF in terms of fraction of reconstructed jets ($^{n_{reco\ jets}}/_{n_{ground-truth\ jets}}$)
- Very similar results are seen in ZH and WW events

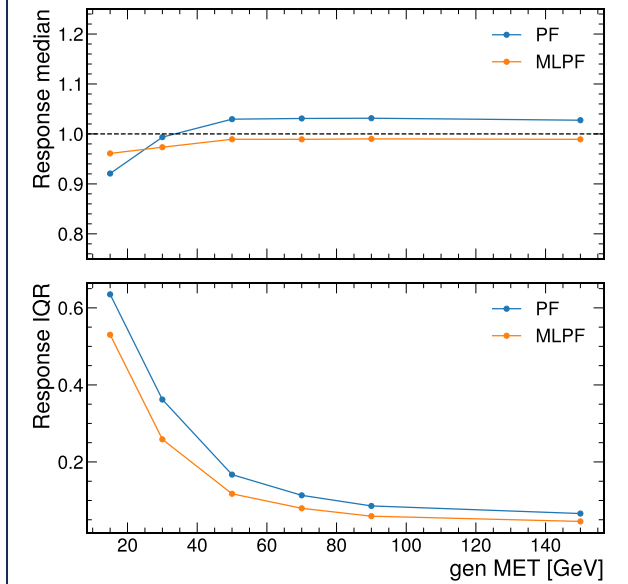
Jets resolution





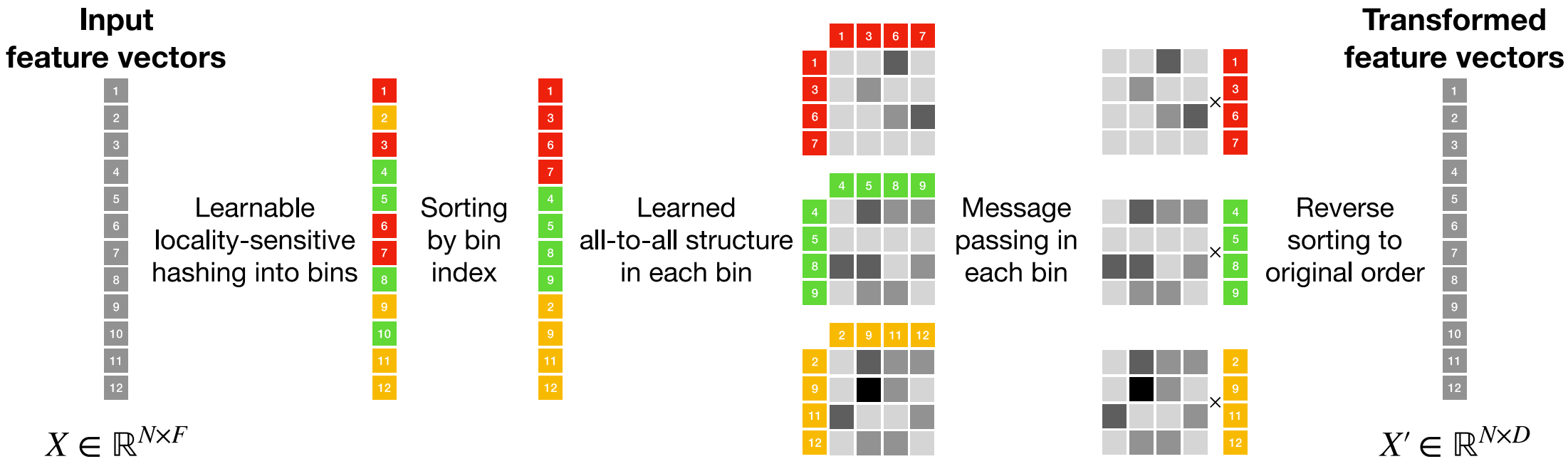
MET resolution





Graph Neural Network (GNN) with Locality Sensitive Hashing (LSH)

One layer of learnable graph building with locality sensitive hashing and message passing



Kernel-based self-attention Transformer

One layer of kernel-based self attention with the FAVOR mechanism.