2nd CERN IT ML workshop Report from CMS

Davide Valsecchi[1] for the CMS collaboration
[1]ETH Zurich (CH)

2nd CERN IT Machine Learning infrastructure workshop
10/10/2023

ETH zürich

Intro

This talk is based on a survey that we circulated among the **CMS community** to gather inputs from all the different teams working on ML applications.

It also reflects specific inputs from the **CMS CERN group**, as requested specifically by IT in its new “business engagement" scheme

Outline

1. Machine learning in CMS - R&D examples
   - Jet taggers
   - End-to-end reconstruction
   - ML particle flow
2. Status of ML software support in CMS
   - Supported platforms
   - Discussion about the need for GPUs in production
   - Indirect inference - SONIC
3. Training Resources and Infrastructure
4. Hardware discussion

1. ML Models in production - footprint

CMS uses many ML models running in production in the trigger and reconstruction sequence.
Until now all the models have been run in *direct inference* on CPU.

- **Time spent in evaluating ML models** is reaching 2% in the reconstruction sequence (RECO), and 10% in the ntuplization step (MiniAOD)

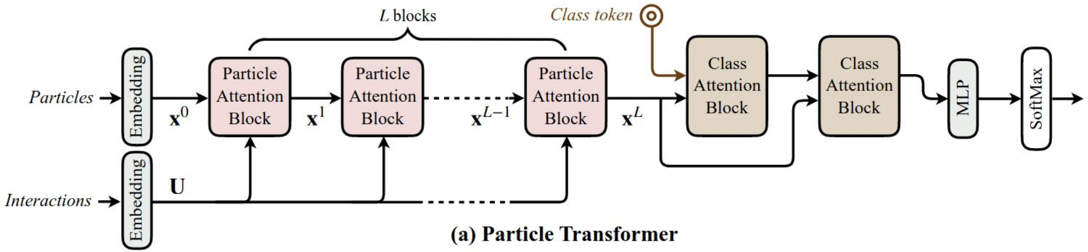
Few examples:

- **Tracking (**~1% of RECO time, TensorFlow): larger number of evaluations but with ~low complexity (DNNs)
- **Jets taggers** (~3% of MiniAOD time, ONNX):
  - Many variants of jet taggers and mass regression: AK4, AK8, AK15 jets
  - Medium size input: ~20 Jets, ~30 constituents, ~10 features x event
  - Complex operations (graphs and transformers)
- **DeepTau** (~3% of MiniAOD time, TensorFlow): complex CNN with ~large input: R&D ongoing to upgrade to graph networks
- **DeepMET** (~2% of MiniAOD time, TF): DNN with large inputs but single evaluation
- **Regression** and **classification** (XGboost, TF): very simple, BDTs or DNNs, usually needed in local reconstruction

Jet taggers

Jet tagging = playground for ML architectures

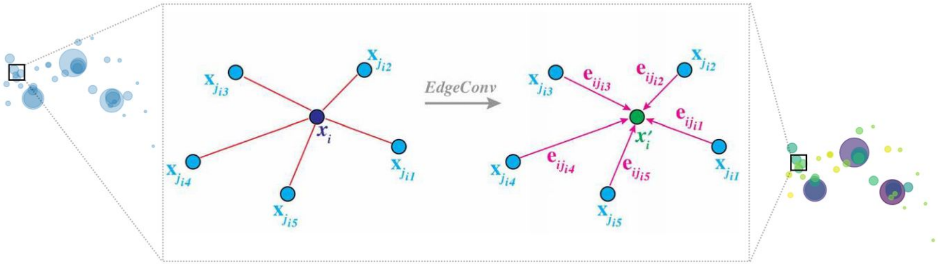
- 2 leading models emerged in CMS: *ParticleNet* (graph conv.), *ParticleTransformer* (ParT)



(a) Particle Transformer

- Probably we still haven't reached the *ultimate* performance, but CMS focus is also shifting more on consolidation:
    - energy / pT calibration for jet and taus
    - working on jet flavor, tau decay mode and lepton tagging
    - Exploring adv attack and data adaptation to improve stability and minimize efficiency corrections

ParticleNet

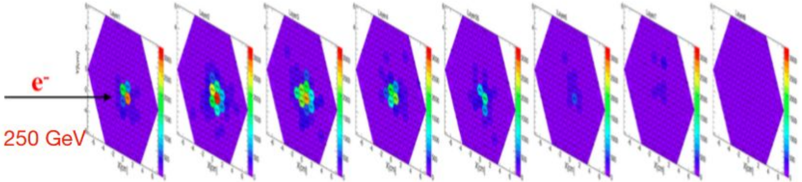


End-to-end reconstruction

Heavy R&D on end-to-end high granularity calorimeter (HGCAL) reconstruction arxiv2204.01681

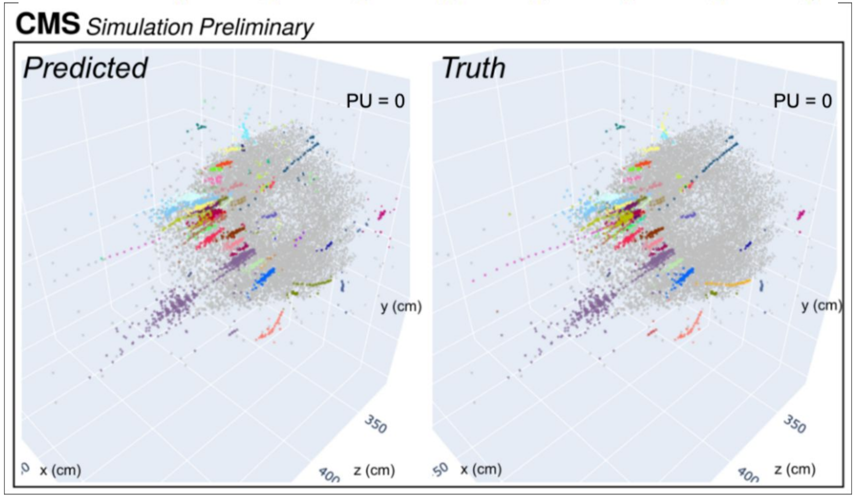
From **hits → clusters position and properties:** perfect application of GNNs and *object condensation.*

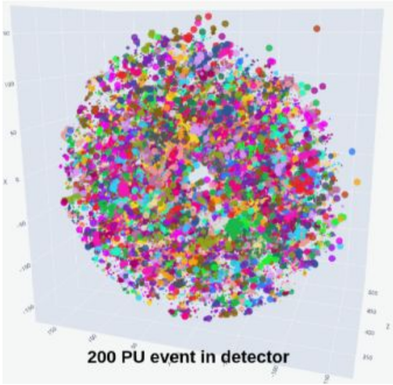
- GravNet architecture with dynamical graph building applied successfully arxiv 2106.01832



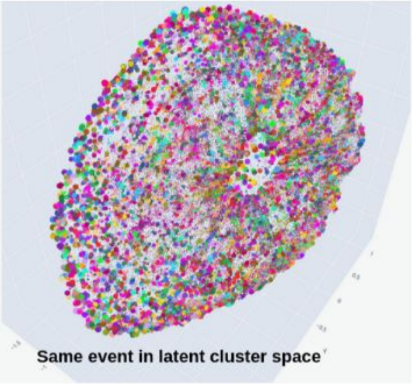
Challenges:

- large input dimensionality and non-regular data structure
- Implemented custom kNN kernel for in-memory operations (200x faster than pytorch geometric
- Multi-GPU training neede to speed up prototyping





200 PU event in detector



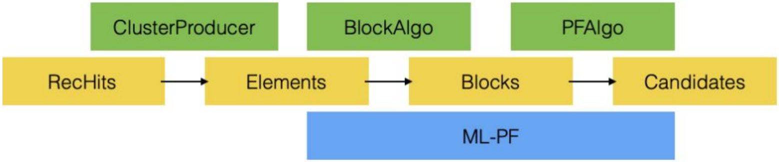
Same event in latent cluster space

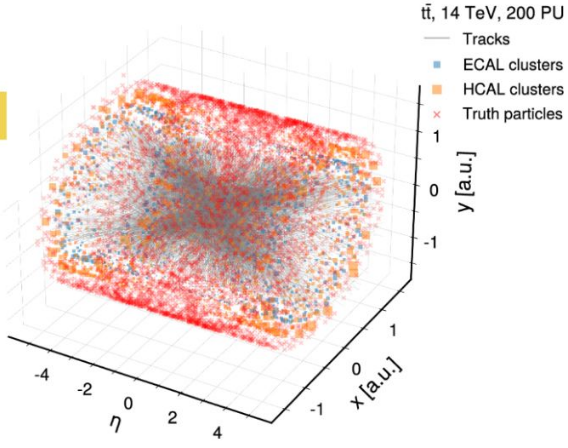
ML for Particle Flow

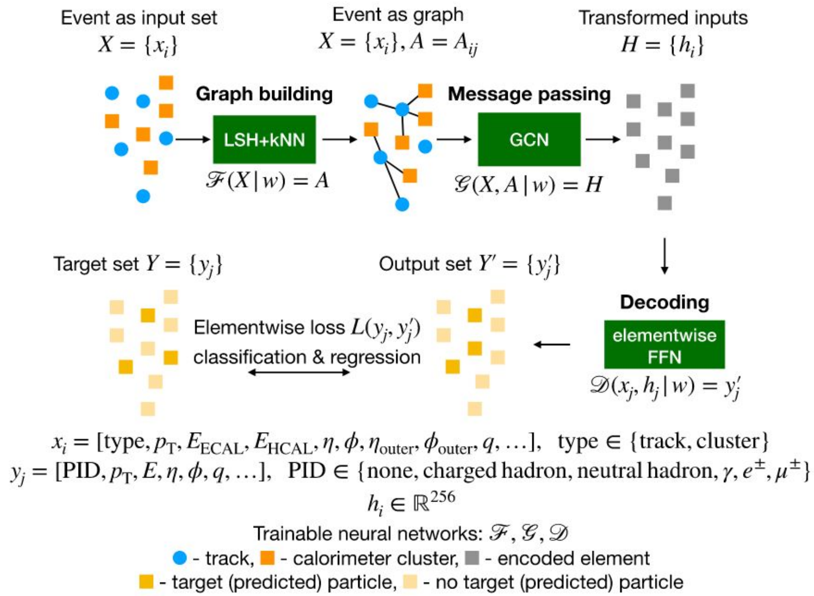
R&D effort in CMS, well advanced and implemented in CMSSW through ONXX inference.

[[Paper](#)]

Sketch of PF algorithm step in CMS







- Starting from **all sub-detector ingredients**
  - Output directly the list of candidates
  - Particle ID and properties regression in one go

- **Dynamic graph building** done in an efficient way:
  - Locality sensitive hashing (LSH) [arxiv](#)

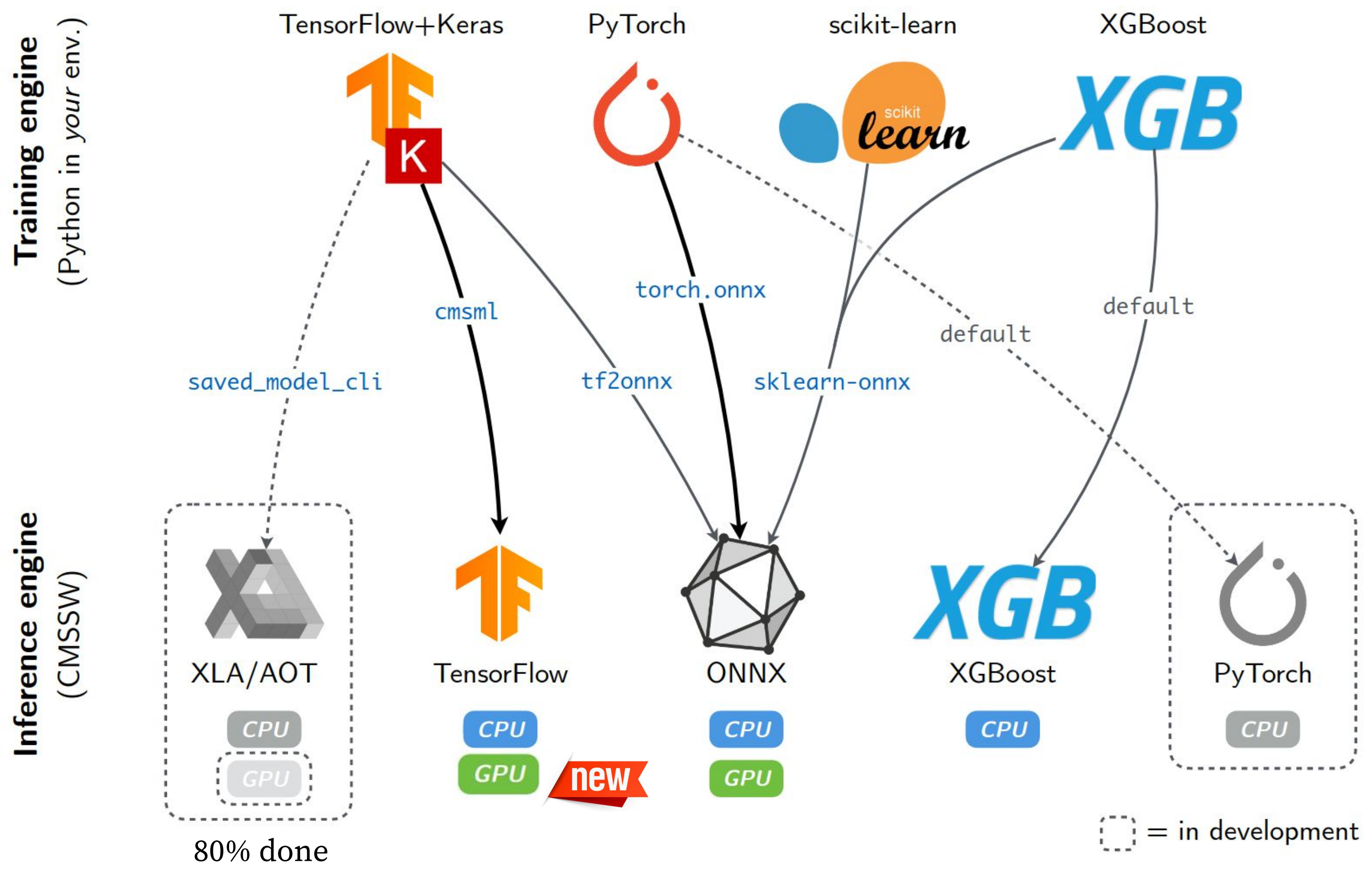
- Based on dense operations for portability:
  - demonstrating **good scaling on GPU**

2. Software platforms support

CMS supports inference of ML models within its software framework CMSSW in two ways:

- **Direct inference:**
    - ML libraries compiled and shipped along the CMSSW binaries
    - CPU and GPU support
    - Platforms:
        - Tensorflow: 2.12 CPU+GPU
        - ONNX 1.10.0 CPU+GPU
        - PyTorch: currently under development
- **Indirect inference (currently in R&D):**
    - CMS developed a service to externalize ML inference:
        - Services for Optimized Network Inference on Coprocessors (SONIC) [MLG-23-001 PAS]
    - Based on Triton (Nvidia). It can handle different accelerators and models.
    - **Factorizes backends out of CMSSW**, many jobs can access the same GPU resources
    - Dynamic batching of events calls → improve throughput

1. Direct Inference status



Discussion about Software dependencies for direct inference

Maintaining TensorFlow updated along the CMSSW software stack is becoming **increasingly difficult**

CMSSW is moving to **C++20, GCC12, CUDA12**

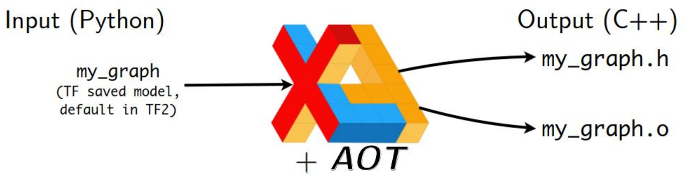
- **ONNX** compiles happily, both CPU and GPU version, no problems
- **TensorFlow** **has been a blocker**:
    - **Fixed by moving to TF2.12**
    - Tensorflow has many dependencies (eigen, abseil, etc) used in other libraries in CMSSW: keeping libraries version consistent and side-effects are **painful** to address
- **PyTorch** discussion: adding pytorch to the mix is possible, but it will increase even more the complexity of the dependency matrix.

Development on the direct inference

XLA+Ahead-Of-Time (AOT): compilation and optimization of ML models graphs to C++ code.

Enables several types of graph optimizations

- On graph level:
  - kernel fusion
  - Buffer analysis for allocating runtime memory (eliminates intermediate caches)
  - Common subexpression elimination
  - Pruning of unused kernel
- On hardware level:
  - TPU, GPU or CPU (different backends)



AOT

- Converts graphs into self-contained library
  - Graph becomes a series of C++ compute kernels
  - **No dependence** on main `libtensorflow.so`
- Pros:
  1. Reduced memory footprint
  2. Trivial multi-threading behavior
  3. Runtime potentially faster (depends on degree of optimization)
- Cons:
  1. No dynamic batching (fixed memory layout) but can be emulated (**padding/stitching**, **shown later**)
  2. Graph needs to be XLA compatible

Cutting edge technique to **speed up models evaluation** and reduce the central software maintenance.

Currently under integration in the CMS software.

5-10x reduction **in memory**.

Speed-up depends on the models but many optimizations possible when exporting the graph.

The needs for GPUs in production

GPU support is there for ONNX and **TensorFlow,** as direct inference:

- still not used in production
- It will become crucial to be able to integrate architectures

The performance scaling of the different models **is not the same in all our applications.**
Depends in first approximation on:

- Number of calls (batch dimension) to the inference engine → ML for object vs for global event reconstruction.
- Inputs dimension → jet constituents vs all reconstructed particles
- model complexity → DNNs vs large transformers

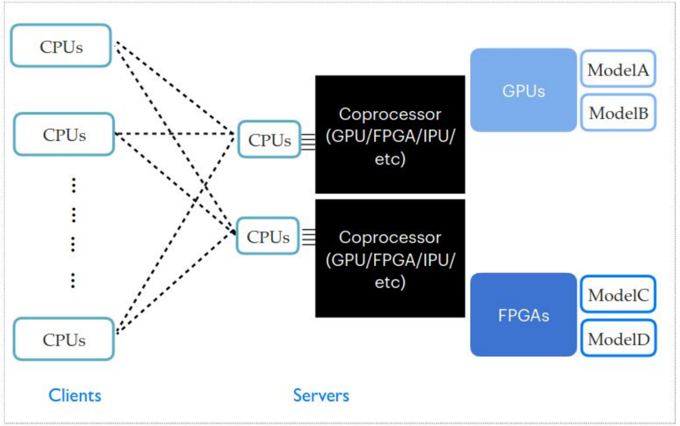
GPUs can help a lot with large batch dimension + complex operations.

- With direct inference we are limited to **single event inference calls**
- At the HLT trigger we have GPU but busy with tracking → memory is very scarce
- Indirect inference would help by batching many requests and offloading the handling of GPU dedicated machines

Indirect inference - SONIC

- Public Analysis Summery: MLG-23-001
  - Document R&D about offloading the ML models evaluation for MiniAOD step to inference-as-a-service Triton based software

- Some **benefits** of SONIC:
  - **Factorizes backends out of CMSSW**
  - Allows for use of **remote** coprocessors
  - Allows for adjustable CPU-to-coprocessor ratios
  - Can be used with multiple types of coprocessors, GPU, IPU, FPGA
  - multi-event inputs batching

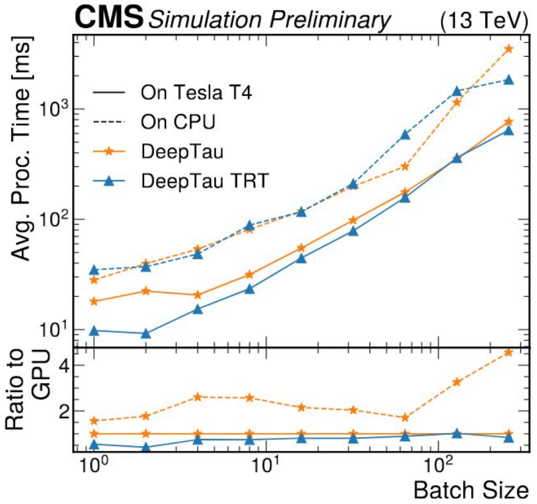


- Some **difficulties**:
  - Dealing with server failures
  - Developing a scheme for server discovery
  - Using load balancing for different workflows with different coprocessor demands

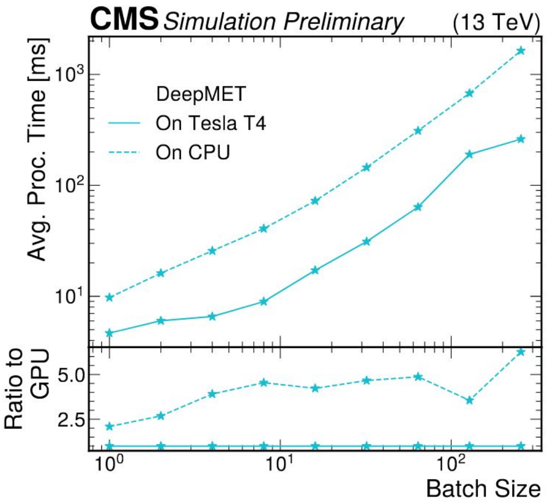
Established clear advantages of the approach, now dealing with stabilization for the actual use in production.

Performance of models on GPU with SONIC

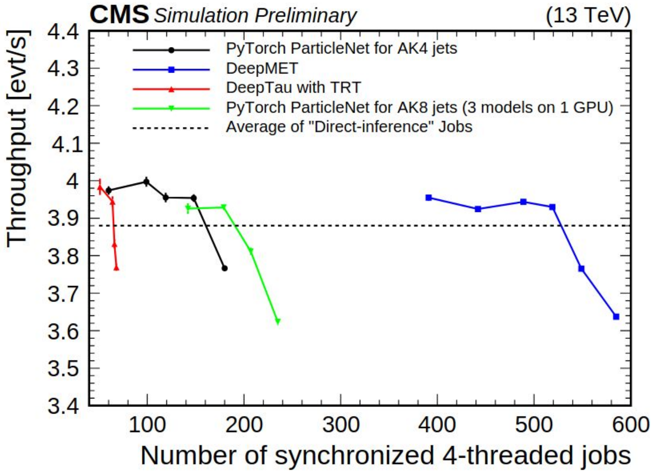
Speed-up both for **single event evaluation** and for **batched evaluation**



Model performance on SONIC comparing CPU vs GPU events/s



Model performance on SONIC comparing CPU vs GPU events/s



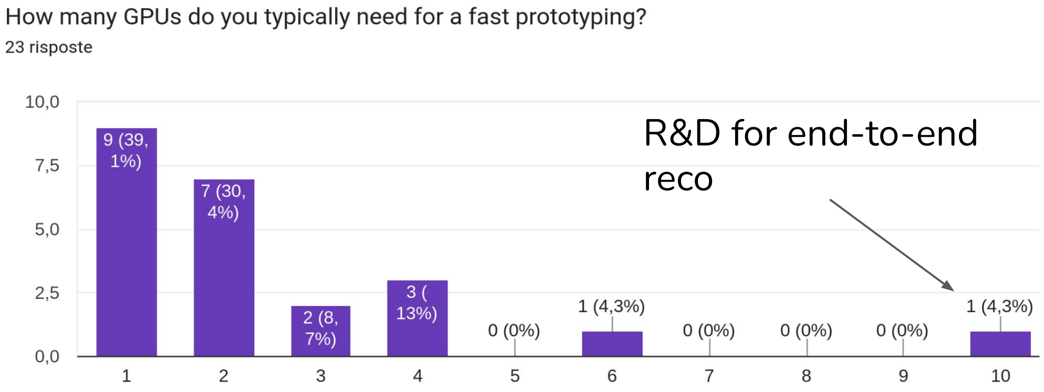
Batching multiple evaluations on single GPU. Throughput decreases on these plots when GPUs become saturated

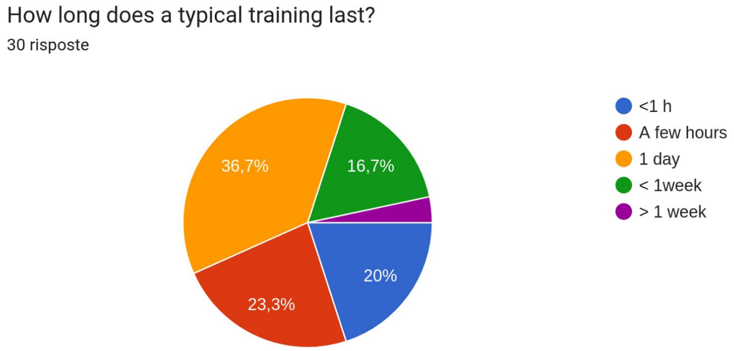
MLG-23-001-PAS

Training resources - survey

ML development in CMS is carried out independently by many groups: no central training infrastructure in place

- Analysis, reconstruction, trigger, DQM, anomaly detection, simulation → **many different requirements**
- Organized a **survey** to collect feedback about **training resources:**
  - ~small amount of answers (30 for now) but **main R&D efforts included**
  - the bulk of the distribution (model training for analysis) is still not covered
- Investigated GPU resources needed, resources provider, frequency of retraining, etc

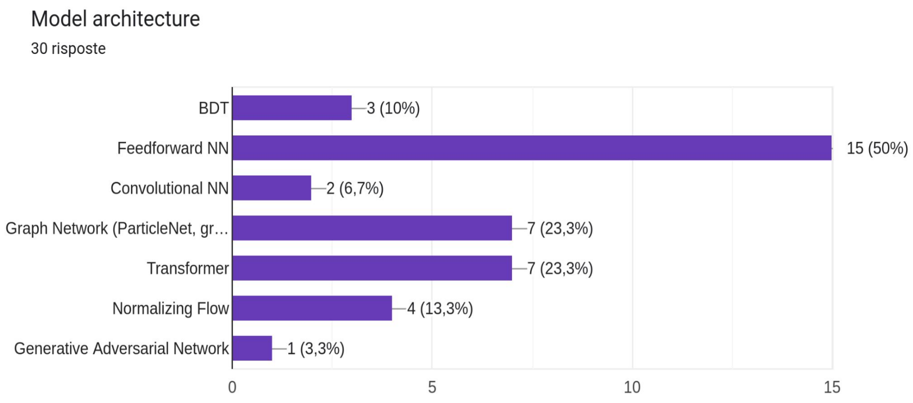


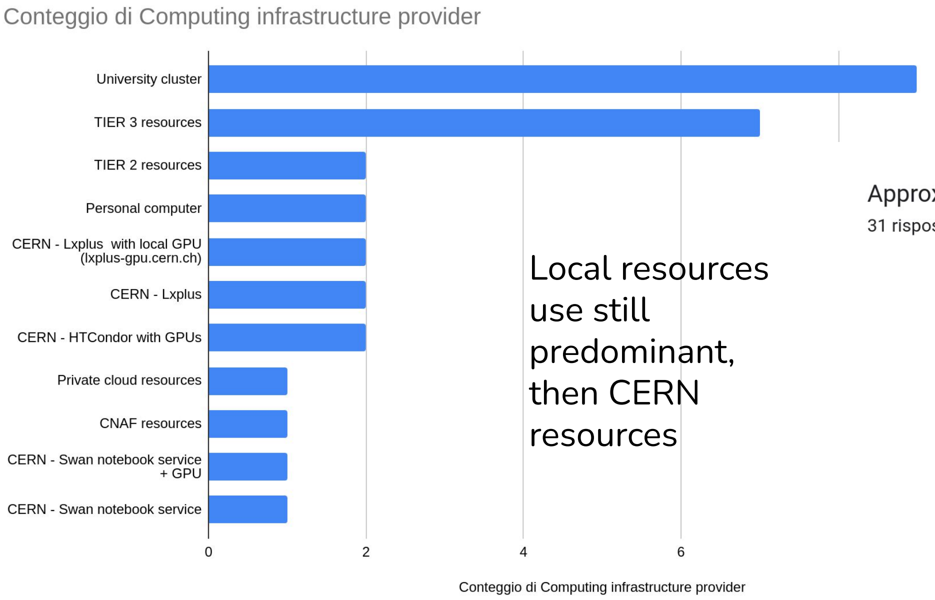


Training resources - survey

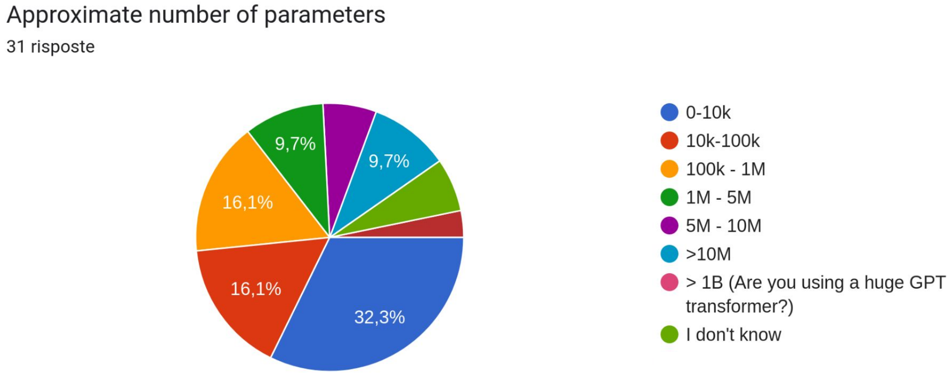
Most of the ML efforts haven't still performed an extensive hyper-parameters optimization due to **resources limitation**.

Modern architectures (graph, transformer, NF), which are heavier to train, are now common





Local resources use still predominant, then CERN resources



Support for training worflows

General feedback for the needs of tools that can improve the ML training workflows

- **Tracking the full lifecycle of ML models:**
  - model repository store, with versioning, metadata and other artefacts
  - keep track **centrally** of the models used in the **data taking of the experiment** for **its full lifecycle**
  - A good candidate would be mlflow
  - a good integration with training facilities would be needed:
    - integration with CERN Kubeflow instance? Still not used in CMS

- **Continuous training infrastructure**
  - Some teams, like L1 trigger, are looking at CD/CI for continuous training
  - Having **GPU capable runners in the Gitlab** infrastructure would be useful

Infrastructure needs for L1 Trigger

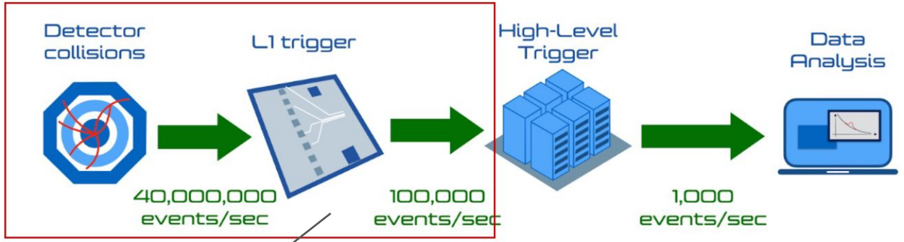
ML is being developed and deployed at the CMS L1 trigger for anomaly detection:

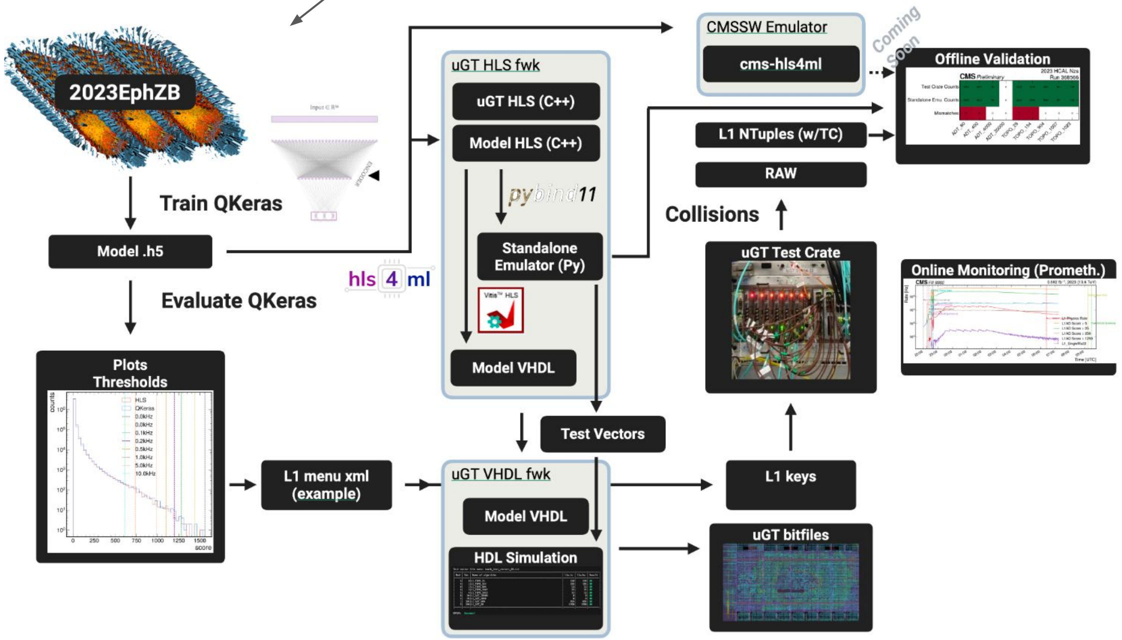
- Auto-encoders on FPGAs!

Very complex chain of software and hardware tools, **many components developed by CMS** and integrated in a complex ecosystem.

Frequent retraining, depending on detector conditioning

R&D well advanced. Models deployed at L1T (on test hardware connect to the real trigger) and commissioning on going. Expected to be in production in Phase 2





[Detector Performance Note]

Machine Learning beyond training

In terms of computing resources, we need more than **just a training facility.** There is a big push to use Machine Learning in **custom environments,** e.g., hardware L1 triggers running on electronic boards.

We need IT support for that workflow

- Most of the adopted solutions rely on a **High-Level synthesis library** (hls4ml for neural networks, Conifer for BDTs).
  - HLS synthesis is a memory-demanding task that can take a lot of time
  - So far, we used an old machine with plenty of memory, provided by OpenLab
  - A machine dedicated to HLS tasks, with all hardware on it (Vivado, Catapult, Quartus, etc.) would be **extremely beneficial,** event beyond the ML use case

- We need to **test inference on FPGA**
- We need to do **R&D on emerging AI-friendly technologies** (graphcore, cerebras AI, etc.), for future trigger design, as well as for offline AI inference-as-a-service tasks (see recent CMS note)

Establish a large(er) scale TechLab-like facility with an easier-access policy to the machine (shared queues as opposed to 100% reserved running time) could address these needs

Our experience with distributed training

- In the past, we experienced with several training solutions (e.g, CERN, Google cloud, CSCS, Flatiron, Marconi) and **collected experience** of which configuration makes our work easier
- We reached consensus on the need of
  - An **HPC-like** setup (the Flatiron one, with /cvmfs mount was particularly user-friendly for us)
  - **GPU interconnection** is a must
  - Support for software **containers**, possibly powered by the usual LCG software stack behind Swan, is more than sufficient
  - Graphic interfaces (e.g., Swan as a jupyter notebook service) are appreciated at design stage, but not needed when running heavy jobs
  - On the long term, cyclical training tasks will happen as part of CMSSW software release for production (data taking, MC production, etc) in a **semi-automatic workflow** -> a GUI based environment would be disruptive

Conclusions

ML inference in CMS is **crucial** and it will become even more important when larger model will reach the production phase

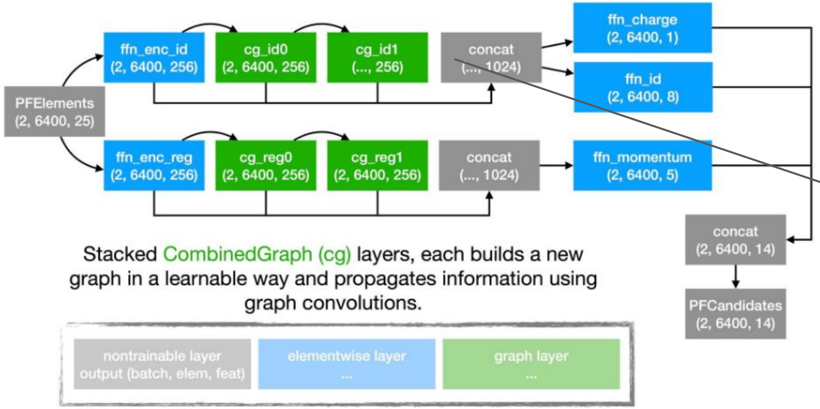
- Software dependencies for **direct inference** is becoming a long term dependency problem
  - we have a strategy to ease the problem thanks to **XLA/AOT**
- **Indirect inference strategies** has a lot of potential to reduce the dependencies problems and developments are ongoing towards production
- The **training infrastructure** is still very frammented and not centralized
  - **Central tools for models tracking** and **continuous training** are necessary
- CMS **L1 Trigger** developments with ML have **special hardware requirements**:
  - CERN should aim to provide an R&D environment suitable for cutting edge applications
- A training infrastructure with a **large number of interconnected GPUs** is becoming crucial for the development of heavy applications like end-to-end reconstruction or fast simulation.

Backup

MLPF architecture

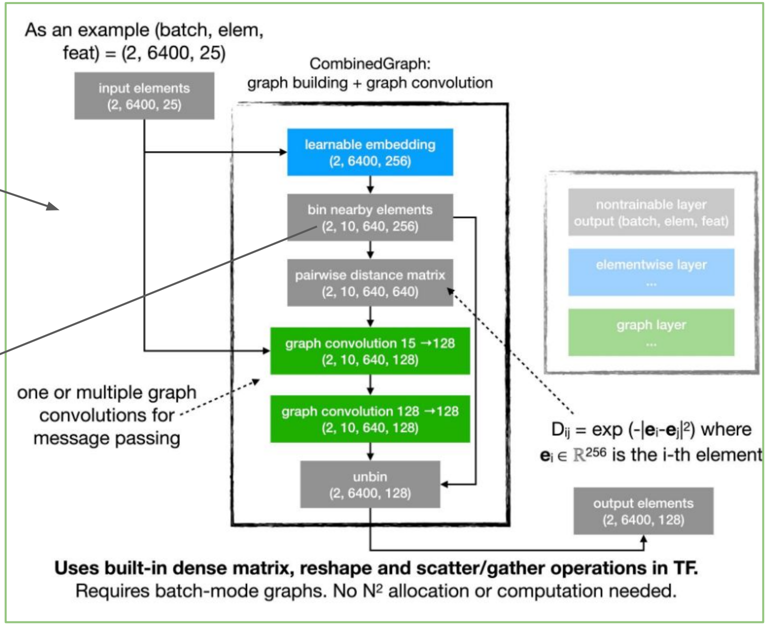
Particle ID an properties are stacked together in the decoder

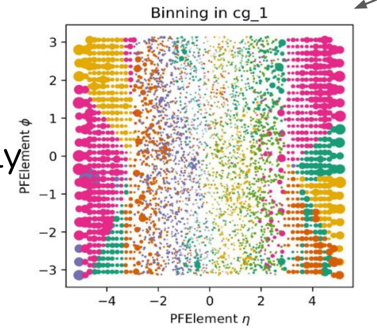
As an example (batch, elem, feat) = (2, 6400, 25)



CombinedGraph layer

- Learnable embedding to form sub-graph
- Multiple graph-conv to propagate info.



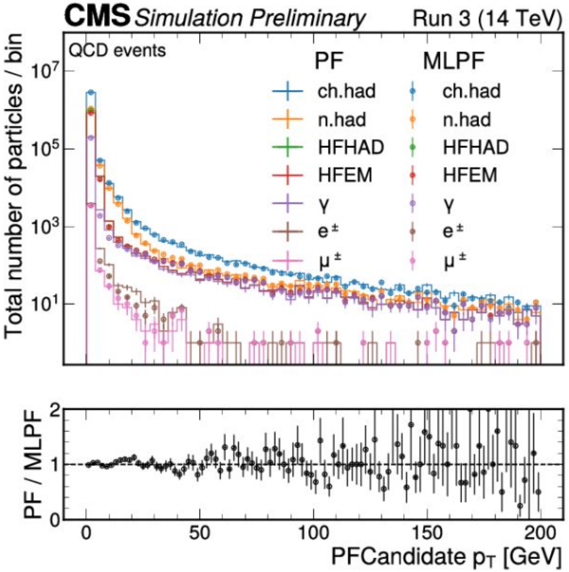


- subgraph binning is fully learnable
- no ground-truth

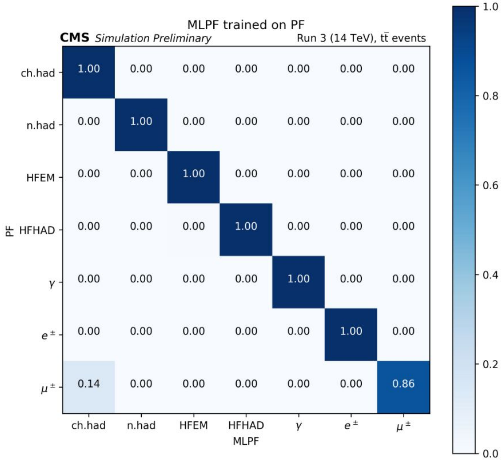
[Slides at CMS ML forum]

MLPF performance

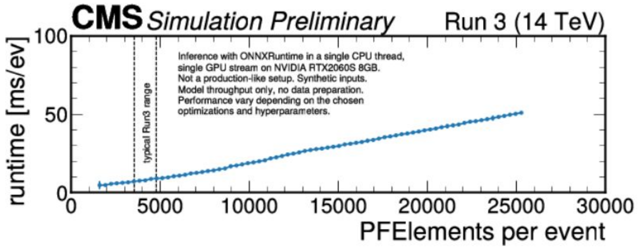
- Hyperparameters optimization is going, but the performance on a realistic environment is very promising.
- Until now trained on PF candidated → work ongoing to define the best possible GEN-level truth

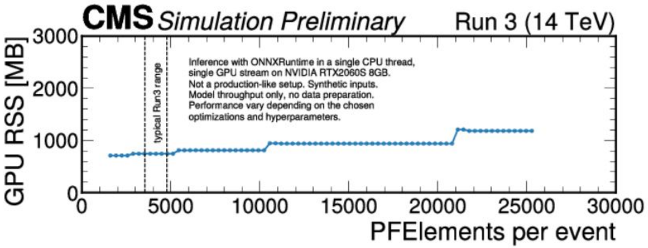


PID confusion matrix



Inference performance is under control





Pata, J. et al. Machine Learning for Particle Flow Reconstruction at CMS. ACAT 2021. https://doi.org/10.48550/arXiv.2203.00330