ML in LL workflows

John Veitch and Steven Schramm
21 May 2025

Low-latency computing requirements

- In late February, had to provide LL computing requirement estimates for ET-PP D8.1
  - Of course the situation is going to change considerably between now and the start of ET
  - However, we need to provide rough estimates of how to extrapolate
- Current (O4) LVK LL computing is > 1/20th of a future HL-LHC trigger farm
  - Naive scaling by signal multiplicity (10^3 increase) unfeasible, thus must rely on speedups
  - ML for fast parameter estimation has shown promise, with studies in the blue book quoting up to 10^5 speedups
  - However, requirement is sum of parts, not only CBC parameter estimation
  - Therefore assume 100x speedup, thus estimate is half of a future HL-LHC trigger farm
- This is for post-merger detection, thus neglects pre-merger regime
  - Assume pre-merger uses a similar amount of resources
- Total LL computing estimate is therefore at the level of a future HL-LHC trigger farm
  - A significant amount of resources, but does not have to be located at a single site - can be distributed (like now)

ML workflows: the basics

- Trade in-advance training time for significant execution (inference) speedup
  - May spend days of time training; a huge up-front cost
  - However, can take seconds or less for inference; huge speed-up compared to traditional approaches
  - Not useful if inference is rare, but can be extremely powerful when inference is repeated many times
- Full ML workflows are possible, but could also be used to narrow the range for matched filtering/etc
  - ML models are likelihood approximants, so can be a strong seed for further processing
  - ML models are a trade-off between performance and execution time/comp. requirements
  - Mixed approaches may be necessary for full performance and resiliency
- Hardware significantly impacts the efficiency of ML
  - Training will (for the foreseeable future) be run on GPUs or dedicated hardware (AI engines, etc)
  - Cannot really escape this, as rely on huge matrix operations to “learn” from data
  - In contrast, inference can be run on different hardware - still usually GPUs, but can be exported to CPU etc
  - Not a real change to the computing model, but does have the potential to impact computing requirements

ML workflows: the implications

- ML hardware dependence does impose some assumptions in comparisons
  - Current LVK is predominantly CPUs with some GPUs (likely for ML)
  - Hardware may thus become more heterogeneous, as each pipeline may have different requirements
  - If heterogeneous hardware needs to be supported, could impact the computing model requirements
  - In particular if large ML models are used, specific hardware is likely needed
- ML models are primarily designed to interpolate, not extrapolate
  - Changing detector conditions etc may necessitate rapid re-training
  - Breaks the idea of ML training done once in advance, rather can be an evolving process
  - Most likely implication is on detector calibration: may need to re-train on a regular basis
- ML is evolving at a rapid pace, and we cannot predict where it will be by the time ET begins
  - We have come an enormous way in the past few years, never mind the past decade
  - Designing a computing model must take this rapid rate of change into account
  - Already have a “technology tracking” group in the EIB, maybe we need an “architecture tracking” group too?
- ML is not a simple algorithm to get started with, and remains a very empirical approach
  - It will be important to envision proper training for students for them to be able to properly contribute
  - Lots of this will be at universities, but train-the-teachers or central advanced programs will be helpful (HSF-like)

ML and data quality

- ML can be used to identify good data, but also to identify bad data
  - Can be trained for specific issues (such as classification of glitch modes)
  - Can also be used for unknown issues (anomaly detection)
  - Need to be careful in the latter case to not remove real signals
  - In the era of ET 50-year infrastructure lifetime, ML could really help reduce shift load via first-pass automated DQ
- ML may be the key to understanding all of the auxiliary channels
  - Leverage the full power of the data streams being collected
  - Could better help to understand when we have glitches, what is causing them, etc
- Don’t see a direct impact on the low-latency computing model
  - Training would likely be offline, and then refined (transfer learning, etc) online
  - Would likely be an (online) input to low-latency, not part of low-latency

Publishing ML results

- ML models are limited in their interpretability
  - Not easy to condense the ML output into a simple equation that can be put in a paper
  - Others trying to reproduce your results will struggle - even the same architecture will differ with each training
  - May need to publish full ML models for the community to be able to interpret results
- As mentioned, the online and low-latency system is likely to be evolving
  - Do alerts have to include the ML likelihoods or calibrations/DQ/etc to go with them?
  - Do we need to store the ML models used for an alert so we can internally cross-check alerts after the fact?
  - More generally, does ML usage change desired alert contents?