Machine-Learning Pile-Up-Suppression Trigger Algorithms for the HL-LHC

Applicant: Peilin Ye

Mentors: Ariel Schwartzman, Rainer Bartoldus

Duration: May 12, 2025 – August 3, 2025 (12 weeks)

1. Motivation

At the HL-LHC, the number of simultaneous proton-proton interactions per bunch crossing (pileup) will increase dramatically from approximately 60 at Run 3, to an average of 200 pp collisions. This large pileup will pose significant challenges to the trigger systems, especially for the selection of multi-jet final states. Traditional pile-up mitigation, tuned for stochastic soft-QCD jets, will be inadequate; without additional rejection of hard-QCD pairs of dijets, trigger accept rates will saturate and dilute the discovery potential of Run 4 physics (See Figure below).

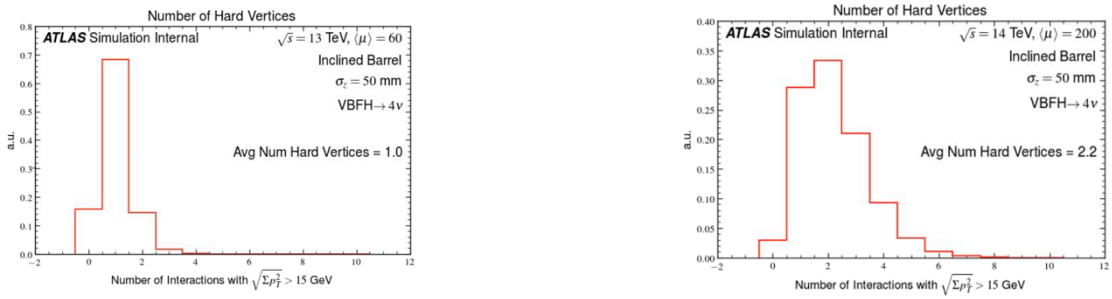

Figure 1: Left side: number of hard vertices per bunch crossing in LHC; Right side: number of hard vertices per bunch crossing expected in HL-LHC. Notice that the majority (>65%) has 2 or more hard vertices.

This project proposes to develop machine learning based trigger algorithms for the suppression of hard-QCD pileup at the HL-LHC, specifically optimized for the low-latency environment at the hardware trigger level. The primary goal is to distinguish multi-jet events originating from a single vertex (signal) from those stemming from multiple vertices (background), thus enhancing the sensitivity of analyses like Higgs boson pair production (HH → 4b). If time permits, such algorithms will be prepared for implementation in the ATLAS L1 Topological Trigger (L1Topo) so they could be put into use for Run 3 as a capability demonstration.

2. Objectives

Develop machine learning models (Boosted Decision Trees and simple neural networks) using simulated calorimeter trigger objects or processed beam (“bytestream”) data:

1. Emphasize fast, resource-constrained implementations suitable for FPGA hardware, avoiding complex operations (e.g., division).

2. Demonstrate performance gains in multi-vertex discrimination over traditional trigger strategies.
3. Investigate translation of the ML models into C++ and synthesis using High-Level Synthesis tools for FPGA deployment.

3. Procedure

1. Data Collection and Preparation: Gather either real beam data or simulated calorimeter trigger objects. Preprocess the data to match the trigger system's format.
2. Feature Engineering: Utilize jet kinematics (pT, η, φ), and angular variables (e.g., Δφ between jets) as inputs.
3. Model Development: Implement and train BDTs or lightweight NNs. Exploit topological differences between single-vertex and multi-vertex multi-jet events using calorimeter information. Performance metrics will include background rejection vs. signal efficiency for any signals (i.e. signal model independent).
4. Hardware Feasibility: Ensure models are compatible with FPGA limitations and FPGA synthesis tools (HLS4ML to C, Vitis Synthesis, etc).
5. Optional FPGA Translation: If time permits, translate the ML model to C++ and synthesize it for FPGAs, for a possible implementation in L1Topo as a capability demonstration.

4. Timeline

Week 1-2: Familiarize with calorimeter trigger data structures, hardware constraints, review BDT/NN literature for low-latency designs.

Week 3-4: Data gathering: collect or process simulation and/or bytestream data to resemble calorimeter trigger object outputs.

Week 5: Develop initial BDT model in Python, using jet kinematic/topological variables.

Week 6: Test discriminating power: differentiate between single-vertex and multi-vertex multi-jet events; check signal model independence.

Week 7-8: Explore simple neural networks; compare to BDTs in terms of performance and hardware-friendliness; conduct hyperparameter tuning if necessary. Improve on FPGA compatibility if necessary.

Week 9: Final validation: trigger efficiency and rejection rates on unseen data; assess feasibility for real-time use. Start preparing for the final presentation.

Week 10-12: Start translation of best-performing model to C++; investigate FPGA synthesis; (if time permits) try to implement the algorithm in L1Topo for demo purpose.