Improving AXOL1TL Trigger Anomaly Detection in CMS Level-1 Trigger

Sebastian Paucar[†*], Jennifer Ngadiuba[†‡], Cristina Botta.[†]
[†] *European Organization for Nuclear Research (CERN), Geneva, Switzerland*
[‡] *Fermi National Accelerator Laboratory (Fermilab), Illinois, United States*
[*] *Universidad Nacional de Ingenieria (UNI), Lima, Peru*

Abstract

In the search for the optimal triggers to avoid overlooking potential new physics information in proton collision events at the LHC, which are constrained by a low-level DAQ latency of 50 nanoseconds at CMS experiment, unsupervised machine learning (ML) has gained prominence as an efficient tool for encoding the Standard Model background and identifying anomalies using out-of-distribution metrics. AXOL1TL is a trigger located within the μGT system of the CMS Level-1 Trigger currently deployed. We demonstrate that the inclusion of tau leptons into the model, along with adjustments to the dense model architecture, improves sensitivity to rare SM signals or potential BSM escenarios, as well as anomaly classification compared to the baseline.

Keywords: Anomaly detection, trigger algorithm, autoencoder, signal, background.

1 Introduction

At the CERN Large Hadron Collider (LHC), 40 million proton collisions occur every second, resulting in massive volumes of data. The CMS experiment generates tens of terabytes of data per second during the current Run 3. To store and process this massive volume of data, a sophisticated two-level trigger system for DAQ is currently being implemented. The two stages are the Level-1 Trigger (L1T), where FPGAs are the preferred hardware along with high-speed optical links, and the High-Level Trigger (HLT), based on software. The L1T receives information from the calorimetry and muon systems, generating an initial real-time selection with a fixed latency of 3.8 μs and a maximum output rate of 100 kHz, rejecting 97.5% of the initial events. The accepted events are reconstructed in the HLT, which employs a farm of 30,000 cores, reducing the output rate to approximately 1 kHz on average, with a latency on the order of 100 ms, thereby achieving a further 99% reduction in the amount of data to be analyzed.

While this filtering system is crucial for managing the enormous amount of data generated, the supervised, theory-driven algorithms used in systems such us L1T have proven effective for identifying specific theoretical physics scenarios, as was the case with the discovery of the Higgs boson. However, these algorithms are exclusive of potential new physics signatures not anticipated by kinematic cuts on event-level observables. They lack robustness against non-saturated Beyond the Standard Model (BSM) *signals*, such as those associated with Supersymmetry (SUSY), due to their reliance on predefined theoretical scenarios.

In response, unsupervised and signal-model-agnostic, out-of-distribution-based anomaly detection (AD) algorithms are being deployed to extract a rich spectrum of rare events as candidates for new physics signatures. The primary challenge for these algorithms is to meet the extreme latency requirements at the LHC and computational resource constraints, while preserving high inference performance.

The *CMS collaboration* has developed an AD trigger at the final stage of the L1T based on a variational autoencoder to fulfill this task, and it is currently in use. The goal of this project is to extend the current baseline by incorporating new high-energy features, improving the spectrum of triggered rich events, as well as enhancing robustness against detector changes and pileup.

2 Background

2.1 The Large Hadron Collider

The Large Hadron Collider is the most powerful and complex particle collider in the world, located in a circular tunnel with a circumference of 26.7 km and a depth of 175 m, situated on the border between Switzerland and France. Constructed between 1998 and 2008 by the European Organization for Nuclear Research (CERN) in collaboration with approximately 10,000 scientists and over 100 countries, the LHC is designed to collide beams of protons traveling in a high-quality vacuum, maintained by dipole and quadrupole magnets. These protons are accelerated in opposite directions using superconducting radiofrequency cavities, achieving a center-of-mass energy of 14 TeV. Currently, during Run 3, the LHC operates at an energy of 13.6 TeV with a collision rate of around 40 MHz.

Proton beams for the LHC are generated in the Large Hadron Collider Proton Synchrotron (LHCPS) and the Proton Synchrotron Booster (PSB), accelerated through the Super Proton Synchrotron (SPS), and finally introduced into the LHC, where they are accelerated to 13.6 TeV. The beams collide at the LHC's four interaction points (ATLAS, CMS, LHCb, and ALICE). As the beam intensity decreases over time due to collisions, the beams are typically dumped approximately 12 hours after a collision starts to maintain experimental quality.

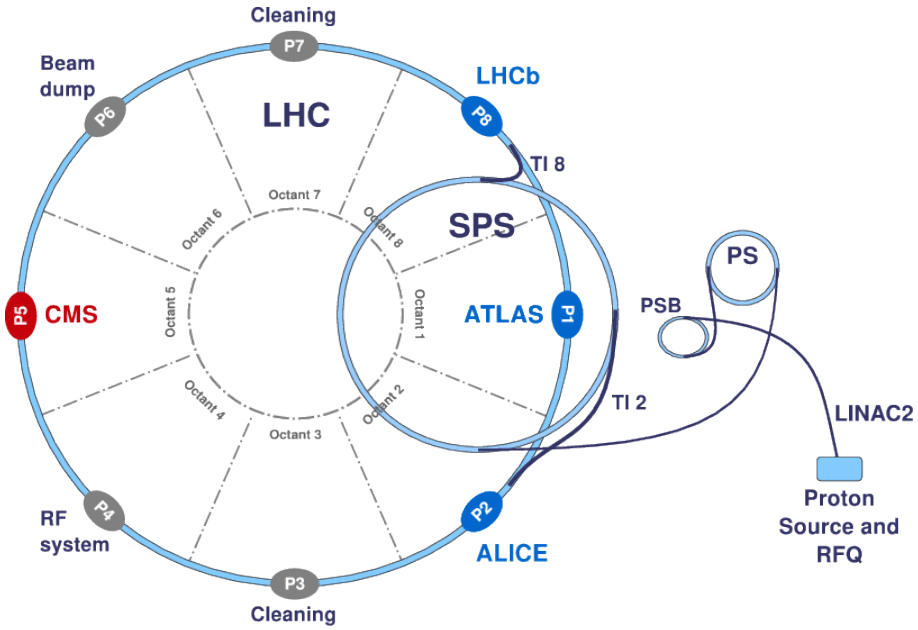


Figure 1: Schematic of the LHC, showing the collider's eight octants and highlighting the four main interaction points (LHCb, ALICE, ATLAS, and CMS). The key accelerator stages are also illustrated: the RFQ, which bunches and initially accelerates the protons; the LINAC2, which provides the first significant acceleration; and the PSB, PS, and SPS systems, which progressively increase the beam energy and prepare it for injection into the LHC, where it reaches the required collision energies.

The LHC has expanded its focus to conducting advanced research to explore BSM phenomena. The LHC experiments face the challenge of detecting signals of new particles and forces, such as the Higgs boson, high-mass quarks, or potential indications of new physics. The sensitivity of searches is limited to a specific subset of BSM models that resemble the theoretical targets. This is because many analysis techniques are focused on detecting signals matching specific theoretical models, restricting the ability to identify unexpected new physics. Advances in machine learning (ML) techniques have improved the sensitivity of new physics searches by enabling broader exploration of theoretical BSM scenarios. However, developing new search strategies is crucial to encompass a greater diversity of possible scenarios and unanticipated events.

In the coming years, the LHC will undergo a significant upgrade with the transition to the High Luminosity LHC (HL-LHC), scheduled to begin in 2029. This upgrade will enable a considerably higher

instantaneous luminosity and a modest increase in collision energy to 14 TeV. However, any further increase in collision energy will require advancements in magnet technology and a collider with a larger radius. The HL-LHC aims to overcome current limitations and provide more detailed data, facilitating a deeper and more precise exploration of BSM physics.

2.2 The Compact Muon Solenoid Experiment

The CMS experiment is one of the two multipurpose detectors located at one of the interaction points of the LHC, in Cessy, France. It is a cylindrical detector measuring 28.7 meters in length and 17 meters in height, designed to precisely reconstruct p-p collisions in high-pileup environments, at a center-of-mass energy of 14 TeV (5.5 TeV per nucleon) and at luminosities of up to $10^{34}\,\mathrm{cm}^{-2}\mathrm{s}^{-1}$ ($10^{27}\,\mathrm{cm}^{-2}\mathrm{s}^{-1}$). Around 2 600 people from 180 different scientific institutions have contributed to its construction.

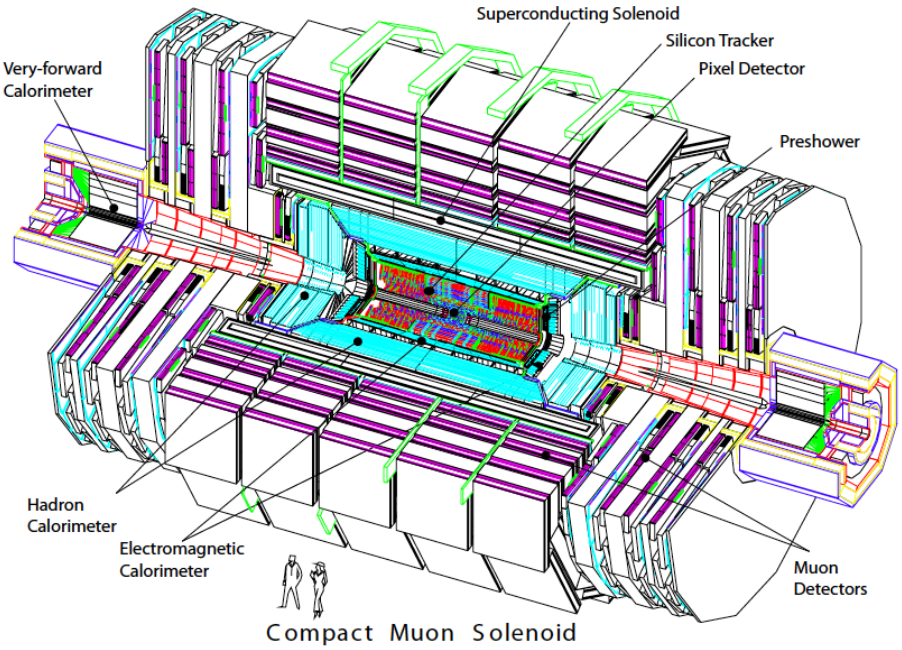


Figure 2: Technical perspective of the CMS Experiment. It consists of an inner silicon tracker (including pixel and strip detectors) that ensures optimal granularity and precision in measuring the charged particles trajectories ($|\eta| < 2.5$, where η denotes pseudorapidity). The electromagnetic (ECAL) and hadronic (HCAL) calorimeters measure the energy deposited by electrons, photons ($|\eta| < 3.0$), and strongly interacting particles, including hadrons. The Very-Forward Calorimeter extends the HCAL coverage up to $|\eta| \approx 5$. The preshower system, located in the endcap region, enhances photon identification by discriminating against background events, while the superconducting solenoid generates the magnetic field for the charged particle trajectories curvature and separates the muon system from the rest of the detector.

The main distinguishing features of the CMS design include a high-field, large-diameter superconducting solenoid, a fully silicon-based internal tracking system using pixels and strips, and a sampling electromagnetic hadronic calorimeter based on homogeneous scintillating crystals, along with the external muon systems.

One of the primary achievements of the CMS experiment was the discovery of the Higgs boson, a key component in understanding the nature of electroweak symmetry breaking and the mathematical SM consistency at TeV scales. BSM theories propose new symmetries, forces, and constituents, with the expectation of a unified theory in the form of supersymmetry or additional dimensions, which often require modified gravity at the TeV scale. Therefore, investigating the TeV energy scale as done in CMS allows for the potential study of new physics.

In the context of the HL-LHC, the CMS Phase-2 Upgrade aims to search for new physics BSM as well as to conduct high-precision measurements within the SM, significantly characterizing the Higgs boson.

With the unprecedented data samples provided by the HL-LHC and the new capabilities offered by the detector upgrade, a wide spectrum of physics analyses will become possible, requiring more advanced selection algorithms and sophisticated triggers to effectively select electroweak-scale processes with 200 pileup events.

3 Related work

Anomaly detection in HEP is a cutting-edge research area focused on the search for new physics, event selection, and hardware monitoring, and is continually advancing. The particle physics community, in its effort to explore beyond the Standard Model, has developed state-of-the-art ML implementations for AD.

Govorkova, Puljak, Aarrestad, et al. (2021) demonstrated that a DNN architecture based on VAEs, quantization-aware trained and implemented in the L1T of the LHC, enables real-time anomaly detection within 130 ns, using only 5% of the resources of the Xilinx VU9P FPGA. This makes it feasible to accelerate the search for new physical signatures during the LHC's Run 3 . This aligns with the findings of An and Cho (2015) regarding the effectiveness of VAEs for AD and with Gemici et al. (2017) on generative models in general. In the unsupervised approach, Halilovic (2017) developed an AD algorithm based on Isolation Forests for injection magnets at CERN. On the supervised side, Roche et al. (2023)Roche et al. (2024) explored decision trees for nanosecond-order anomaly inference using a modified AE, with applications in exotic Higgs decays.

Mikuni and Canelli (2020) introduced an unsupervised statistical method to detect non-resonant anomalies, based on training multiple mutually independent AEs using the DisCo method; in this context, an event is considered anomalous if its reconstruction quality is poor across all autoencoders, and an event classified as anomalous by only one autoencoder contextualizes the SM background.

Anomaly detection strategies for HEP at the forefront of computing use quantum computing (QC), particularly quantum machine learning (QML). Bermot et al. (2023) employed a quantum GAN (qGAN) to extract an anomaly score for each data input. Using the Higgs boson and Graviton production as anomalies, they demonstrated that the qGAN architecture can detect anomalous events using only a tenth of the data points required by classical schemes.

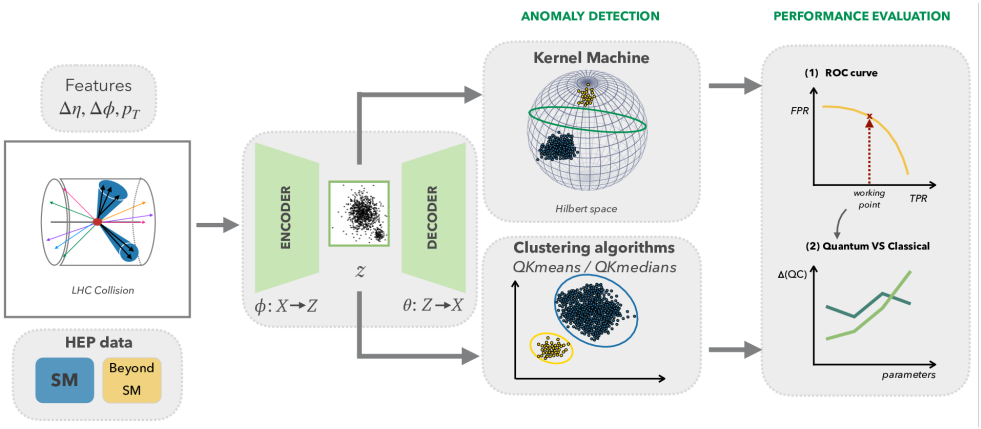


Figure 3: Typical classical-quantum pipeline AE based for AD. Simulated LHC collision data are projected into a latent space where unsupervised QML algorithms are trained on SM data, learning to recognize anomalies in unseen data. $\Delta\eta$, $\Delta\phi$, and p_T are the per-particle features relative to the jet axis. Receiver Operating Characteristic (ROC), Area Under the Curve (AUC), True Positive Rate (TPR), and False Positive Rate (FPR) evaluate the model's performance.

Blance and Spannowsky (2021) used Gaussian Boson Sampling (GBS) to create a lower-dimensional representation of BSM events, simulated by photonic quantum computers, combined with a quantum

version of K-means, called Q-means, for AD. This method has proven to scale more efficiently to large feature vectors compared to the classical version, reducing the complexity from $O(N)$ in the classical version to $O(\log(N))$ in the quantum version (where N is the length of the feature vector).

Ngairangbam et al. (2021) presented a quantum autoencoder (QAE) as a promising approach for BSM Higgs decay scenarios, executable on real quantum hardware. Meanwhile, Schuhmacher et al. (2023) explore new physics searches assisted by simulation using both quantum and classical supervised SVMs. Woźniak et al. (2023) introduced a typical classical-quantum workflow inspired by AE compression, now coupled with an unsupervised quantum kernel machine and clustering algorithms.

While there have been significant algorithmic, computational, and conceptual advancements in the field of anomaly detection, several challenges remain such as the development of an independent and rigorous anomaly metric, the validation of algorithm performance and its robustness against external factors, as well as the solid and consistent independence of models, which require that this field continue to develop over the coming years.

3.1 Variational Autoencoder

A VAE is a generative model that learns to map observed data from a space x, whose empirical probability distribution $q_D(x)$ is typically complex, to a latent space z of lower dimensionality, where the distribution is simpler. This process involves learning a joint distribution $p_\theta(x, z)$ (θ is a variational parameter) that describes the probabilistic relationship between the data x and the latent variables z. Typically, this distribution is factorized as $p_\theta(x, z) = p_\theta(z)p_\theta(x|z)$, where $p_\theta(z)$ is the prior distribution over z, often modeled as a multivariate spherical normal distribution, and $p_\theta(x|z)$, the stochastic decoder, describes how the observed data x are generated from a specific value of z. See Figure 4

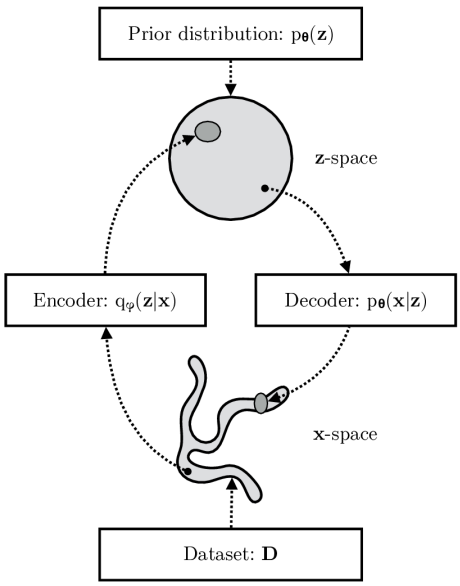


Figure 4: Variational Autoencoders mappings.

During training, the VAE optimizes an objective composed of two key terms. First, the fidelity of data reconstruction, measured by the probability that the decoder $p_\theta(x|z)$ reproduces x from a sample of z that comes from the encoder $q_\varphi(z|x)$. This refers to the **reconstruction loss** ($\mathcal{L}_{\text{recon}}$):

$$\mathcal{L}_{\text{recon}} = E_{q_\varphi(z|x)} \left[-\log p_\theta(x|z) \right] \tag{1}$$

The second term measures the similarity between the approximate distribution $q_\varphi(z|x)$ and the prior distribution $p_\theta(z)$, which is evaluated using the **Kullback-Leibler divergence**, D_{KL}:

$$D_{\text{KL}} \left(q_\varphi(z|x) \| p_\theta(z) \right) = \int q_\varphi(z|x) \log \left[\frac{q_\varphi(z|x)}{p_\theta(z)} \right] dz \tag{2}$$

4 The AXOL1TL System

The **A**nomaly e**X**traction **O**nline **L**evel **1** **T**rigger **L**ightweight (**AXOL1TL**) is an aggressively quantized VAE-based trigger algorithm (self-supervised), integrated into the CMS Global Trigger at the L1T. It is designed for real-time detection of anomalous events associated with new physics in a generic, model-agnostic, and data-driven way, being sensitive to a wide range of signals. **AXOL1TL** is sensitive to a wide range of signals and can capture potential BSM signals, overcoming the acceptance limitations of the traditional L1 system (biased by energy and momentum thresholds) by offering greater ($\sim$ 200%) efficiency and purity with respect to the 2022G Run data taking period dataset. It meets all resources and latency constraints (no more than 50 ns) and avoids unsustainable increases in background rate.

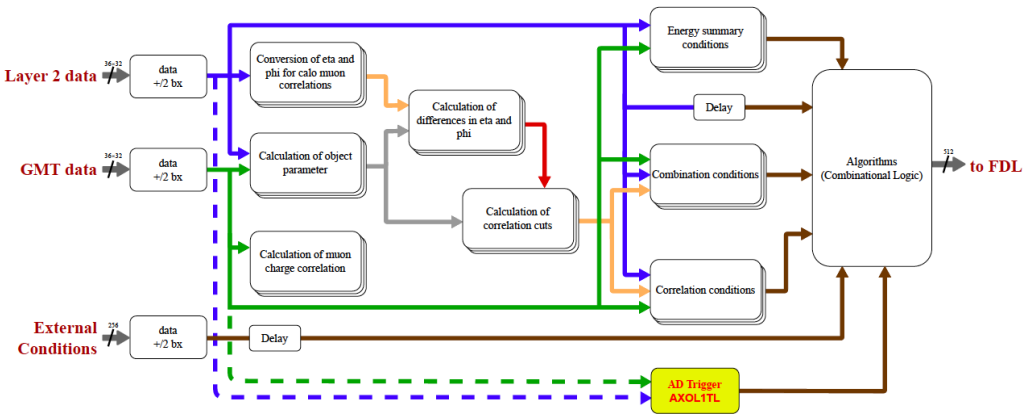


Figure 5: AXOL1TL operates in the CMS Global Trigger (μGT, which has access to all reconstructed L1 physical objects) as an additional trigger algorithm, outputting a binary trigger bit as an anomaly indicator (1 if the event is anomalous, 0 if not) based on events received from the Muon Global Trigger and the Calorimeter Trigger, sending its decision to the HLT.

The entire logic of the μGT in the L1T is known as the L1T menu. In this context, AXOL1TL runs in parallel with other trigger algorithms to allow an event (from at least one of these algorithms) to be considered anomalous (see Figure 5). This is reflected in a fixed acceptance bandwidth per unit of time, referred to as the L1 stream. The μGT provides AXOL1TL, in its current baseline, with L1 physical objects such as E_T^{miss}, 4 e/γ, 4μ, and 10 jets (out of a maximum of 12 e/γ, 8 μ, 12 τ, and 12 jets) as (p_T, η, ϕ) hardware integer vectors , achieving significant signal gains at a rate of 5 kHz in the current L1T system Sun (2023).

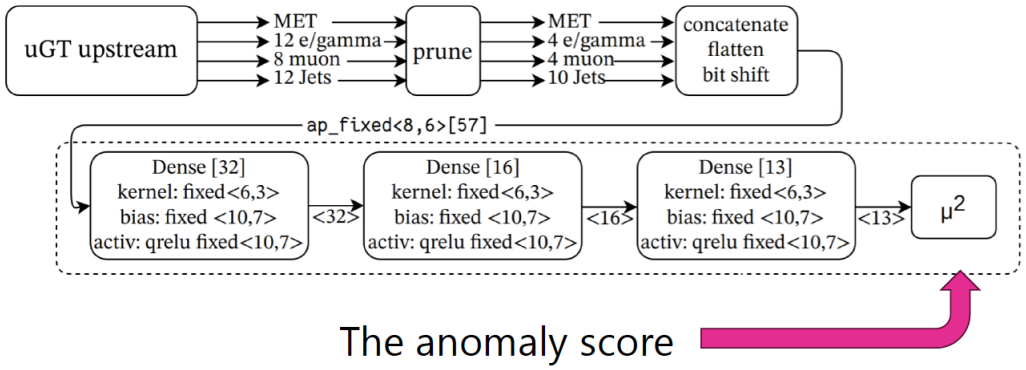


Figure 6: Input Filtering and Encoding Part of AXOL1TL. To emulate the limited bandwidth of a typical L1T system, the inputs to the uGT are pruned to the first 4 μ's, 4 e's, and 10 jets and E_T^{miss}. These objects are then concatenated, flattened, scaled, and biased using bit shifts, and quantized based on `fixed<8,6>`. The result is passed to the quantized NN, where the square of the norm of the mean vector of the latent space (μ^2) is used as the anomaly score.

AXOL1TL, driven by the L1T latency constraints , is designed as a decoder-asymmetric-VAE for increased stability. Its latent space discriminates between *signal* and *background* data thereby calculating the degree of abnormality (guided by *signals* as potential final states, imposing higher performance), making the encoder the only crucial component (allowing low latency and resource savings with a miniminal performance degradation) after the μGT firmware deployment via `hls4ml`, while the decoder is used exclusively during the initial training phase, based on `TensorFlow` and `QKeras`. The AXOL1TL model is based on an numerically stable feed-forward ReLU-activated DNN architecture (except for the last AXO-VAE layers, that are linear activated) that maps onto an 8-dimensional latent space for the prediction of μ and $\log \sigma^2$ through the quantized encoder, which is bottlenecked with a floating-point decoder which is not deployed on hardware and applies batch normalization in each of its layers, except the final one, for more optimal convergence (see Figure 7).

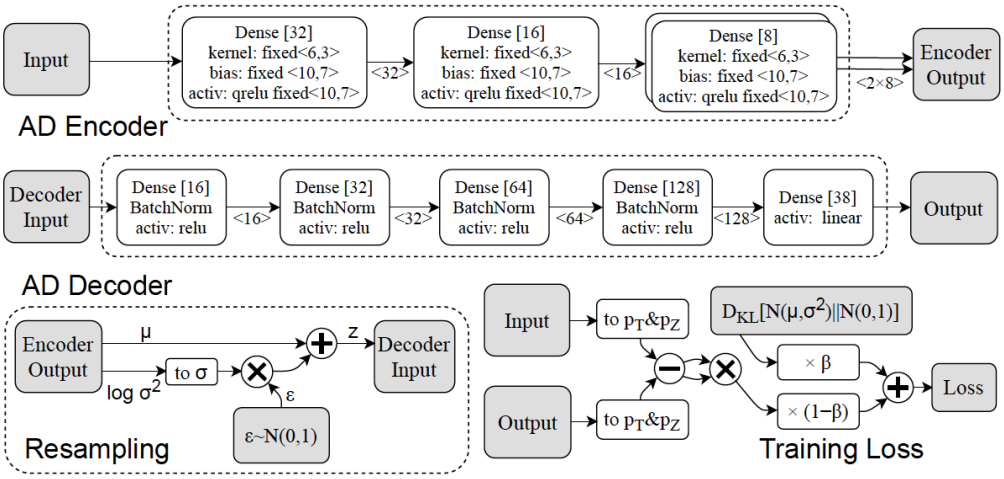


Figure 7: AXOL1TL architecture presented by Sun (2023). Quantized Encoder: 3 layers (`32-16-8` neurons, with quantization parameters `<6,3>`, `<10,7>`, and `<10,7>` for the kernel, bias, and ReLU). The latent space consists of two 1D vectors of dimension 8. Decoder: 4 layers (`16-32-64-128-38` neurons). A 1D vector of dimension 57 is considered as input for AXOL1TL (3 preprocessed generalized momenta for 4 e's, 4 μ's, 10 jets, and E_T^{miss}). An 1D vector output of dimension $57 - 19 = 38$ is produced, with ϕ ignored due to its minimal contribution to L_{recon}. The *reparameterization trick* is adopted during resampling. L_{recon} is defined as the MSE in the $p_T p_Z$ space. The total loss for training corresponds to that of a β-VAE, which includes L_{recon} and D_{KL}. `fixed <b, i>` is a fixed-point data type represented by b bits, of which i are dedicated to the integer part.

5 AXO Metrics

To assess the performance of AXOL1TL, the model will be tested and optimized on benchmark BSM processes, focusing on the improvement of these signals in relation to the efficiency of L1, called η_{pure}^{axo}, given a trigger rate operating point in kHz.

The η_{pure}^{axo} metric is defined as the proportion of *signals* captured exclusively by the additional trigger algorithm, AXOL1TL, compared to the total *signal* events. Based on this definition, it is possible to introduce the efficiency of the new L1 menu, where $\text{L1}^{\text{new}} = \text{L1}^{\text{current}} + \text{AXOL1TL}$, implying that $\eta_{L1}^{new} = \eta_{L1}^{current} + \eta_{pure}^{axo} =$. η_{raw} is a more inclusive metric, also of interest in anomaly detection, as it measures the efficiency of AXOL1TL over all *signals*, regardless of whether the events are captured by the current L1 menu (unlike η_{pure}^{axo}, which is strongly exclusive). The $\eta_{pure}^{axo}/\eta_{L1}^{current}$ ratio is evidently important. It is crucial to focus on BSM signals where the current L1 menu is not robust.

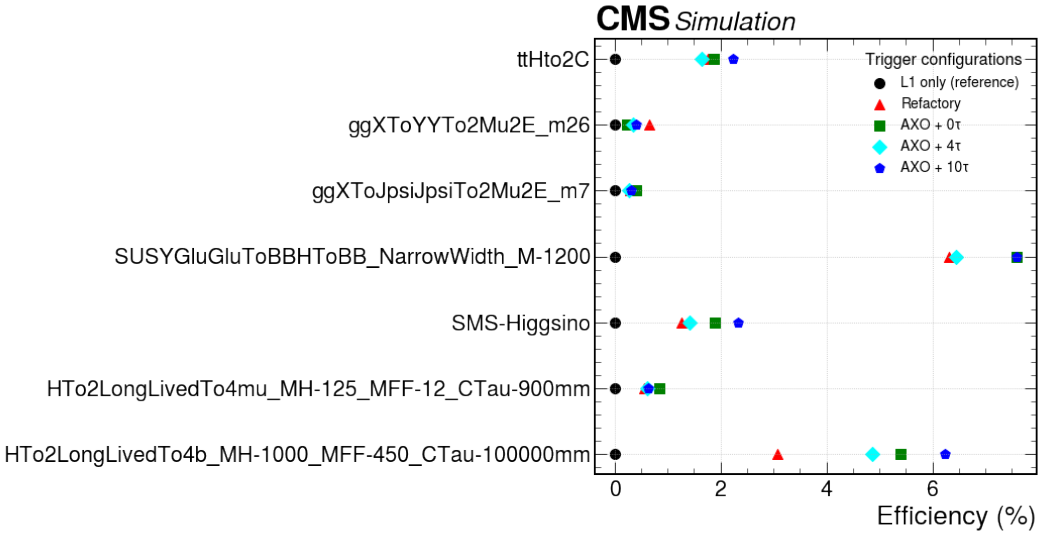


Figure 8: Maximum performance of the AXOL1TL-type triggers, trained and tested on background events (**Run1381148**), in inference on selected benchmark BSM *signals*. The overall improvement of the extended AXOL1TL (including τ's) in event triggering can be significantly higher compared to the results obtained by the standard L1 menu and the latest version of AXOL1TL (*refactory*).

5.1 D_{KL} as an Anomaly Metric

AXOL1TL sets thresholds of $O(1000)$, corresponding to a trigger rate of $O(1000)$ Hz, where an anomaly score can determine whether an event is considered anomalous or not. This is linked to the latent space of the AXO architecture. Sun (2023) demonstrated that the $\mu^2 = \sum_{i=1}^{n} \mu_i^2$ approximation is a convenient and effective mathematical tool for calculating $D_{KL}[N(\mu, \log\sigma^2)||N(0,1)]$ during inference time, where $\vec{\mu} = (\cdots, \mu_i, \cdots)$ represents the mean tensor in latent space. This approximation is particularly useful as it enables efficient development in FPGA hardware.

For the calculation of these anomaly thresholds, the process followed in the `axo` module (see Section 8) involved passing the test events `x_test`, quantized using the `quantized_bits` method from `QKeras`, passing the unbiased samples (see Section 6) into the AXOL1TL VAE architecture. This yielded the latent space of the samples. Significant deviations from the latent space quantify the anomaly score for each sample which is calculated as μ^2. Sun (2023) suggested using the false positive rate as a target rate for determining the anomaly thresholds to obtain a percentile dependent on the LHC bunch crossing rate (approximately 28.61 MHz).

score('SCORE')(thres)

- **raw − axo:** `_raw_rate`
- **pure − axo:** `_pure_rate`
- **L1_rate:** `_l1_rate`
- **HT_rate:** `_ht_rate`
- **AXO_Improvement:** `_axo_improv_rate`

Figure 9: Metrics dictionary for the improvement of the AXOL1TL model relative to the current L1 menu (**L1_rate**), evaluated for each anomaly threshold (kHz). **HT_rate** denotes the performance in triggering by the L1 Transverse Hadronic Energy seed.

5.2 The AXO score

To activate the AXOL1TL hardware bit and thus determine whether an event is anomalous or not, the anomaly threshold must be compared to the anomaly score of the trigger algorithm, `AXO_SCORE`, for each signal sample. The function `get_axo_score` from the `axo_threshold_manager` class in `l1_anomaly_ae/axo/metrics` calculates the AXOL1TL metrics for different BSM signals specified by `signal_names`, for each defined threshold, storing the results in a dictionary specific to that BSM signal within a list of dictionaries called `score`.

For each signal, according to `signal_names`, the event content is retrieved from the data matrix and metadata, such as `ET`, `HT`, `L1_bits`, and `PU`. Based on the anomaly score μ^2 predicted by the model for the given signal (`y_axo_qk`), the events triggered by AXOL1TL are identified as those whose indices satisfy the condition `y_axo_qk > self.threshold[thres]` (where `thres` is a key in the `threshold` dictionary). Similarly, the events triggered by the L1 system and those with an `HT` greater than the `HT_THRESHOLD` are identified. These tensors define a raw event trigger rate for AXOL1TL (`raw_rate` or η_{raw}), the L1 trigger rate (`_l1_rate`), the trigger rate exclusive to AXO (`axo_pure_rate` or η^{axo}_{pure}), the rate of additional events detected by AXO compared to L1 (`axo_improv_rate`, related to $\eta^{axo}_{pure}/\eta^{current}_{L1}$), and the `HT` trigger rate (`_ht_rate`). All these results are stored in the `score` dictionary for each threshold (`thres`). See Figures [8,9].

6 Data Samples

Reconstructed L1 physical objects, such as E_T^{miss}, H^T, E^T, and the reconstructed particles (p_T, η, ϕ) momenta, are available in the μGT (see Figure 10). A significant part of the state-of-the-art datasets for AD at 40 MHz allows for pruning the maximum number of particles (completing the input with zeros), typically μ, jets, and e/γ (see Figure 6). Additionally, E_T^{miss} has been considered as an extra input, thereby providing a cocktail of p-p SM collision events at 13 TeV in the LHC, pre-filtered to include a reconstructed μ or e with a p_T of at least 23 GeV (within a $|\eta|< 2.1$ and $|\eta|< 3$, respectively), ten jets with $p_T > 15\,\text{GeV}$ (within $|\eta| < 4$), along with up to four muons with $|\eta| < 2.1$ and $p_T > 3\,\text{GeV}$, as well as up to four electrons with $|\eta| < 3$ and $p_T > 3\,\text{GeV}$, which is typical in LHC L1 selection algorithms Govorkova, Puljak, Aarrestad, Pierini, et al. (2021).

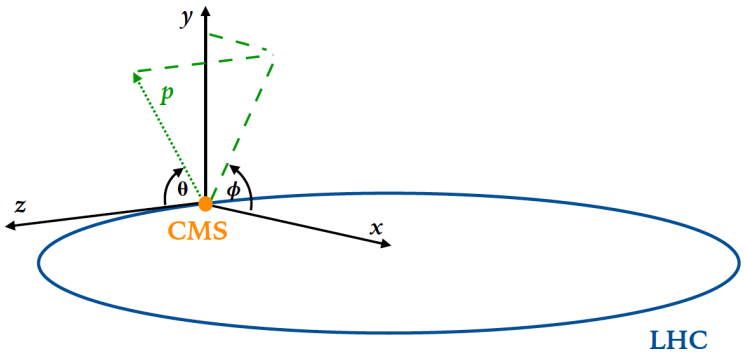


Figure 10: Reference system used to define the generalized momenta. In particular, $\phi \in [-\pi, \pi]$, $\eta = -\log(\tan(\theta/2))$, and $p_t = p\sin(\theta)$ (projection of p onto the xy plane), in units such that $c = h/2\pi = 1$ (c is the speed of light and h is the Planck constant). Each particle in the event (μ's, e/γ's, jets, and potentially τ's) and the E_T^{miss} are represented by these momenta.

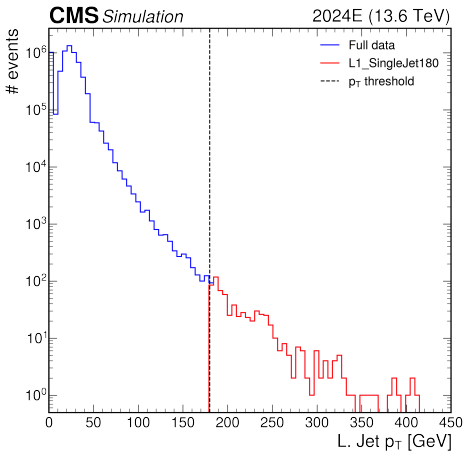
In fact, AXOL1TL was presented by Sun (2023) Sun (2023) following Gorkova et al. (2021) Govorkova, Puljak, Aarrestad, et al. (2021) approach regarding the dataset provided by Puljak et al. (2021) on *Zenodo* for the *Anomaly Detection Data Challenge 2021* Govorkova, Puljak, Aarrestad, Pierini, et al. (2021). This initiative aimed to engage the HEP community in designing AD algorithms to find a-priori unknown rare cases of New Physics hidden in a given SM dominated dataset. Gorkova et al. (2021) *Zenodo* dataset was focused on e's and μ's events using a lepton filter, given their stability in the detector and their detectability through penetration, in their attempt to represent a truly *unbiased* data stream from the L1T, which is currently not possible due to computational resource requirements. The current baseline of AXOL1TL is set in this particle environment.

6.1 Background Data

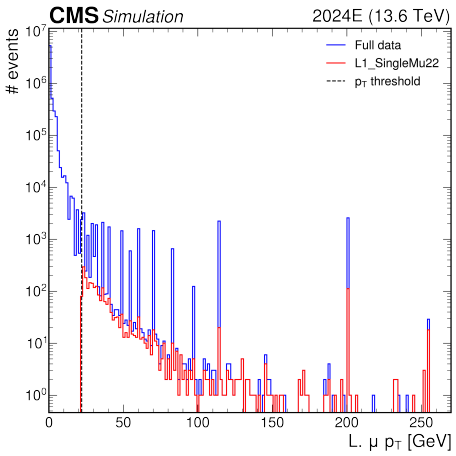
The *background* refers to a realistic data stream populated by known SM processes, useful for the model's training in order to learn the nearly *unbiased* nature of events at the LHC, and to identify a new physics signature (the *signal*) as a metrics deviation outside the *background* learned distribution Govorkova, Puljak, Aarrestad, Pierini, et al. (2021).

In its most current pipeline, AXOL1TL has been trained and tested using background data referred to as *Zero Bias*, collected by the CMS Experiment during 2023, with p-p collisions at $\sqrt{s} = 13.6$ TeV center-of-mass energy. On the other hand, Sun (2023) presented AXOL1TL trained with data from Run number 362616 of the CMS Run 2022G, which consists of 12,108,978 events and has an average pile-up of 65 Sun (2023), while Zipper (2023) developed AXOL1TL in the CMS L1 μGT test crate, trained on Zero Bias events from Run 3 (Run 367883) with 10.5 million events Zipper (2023).

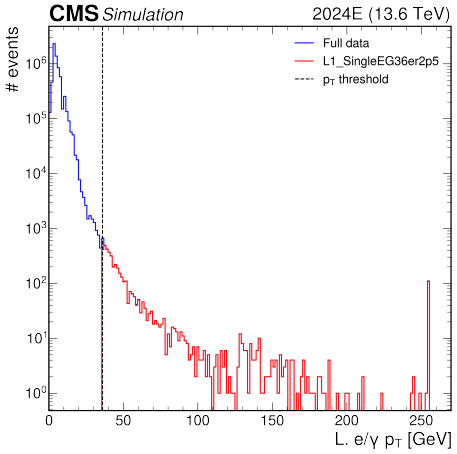
`EOS` is CERN's big data storage system, which meets the LHC computational requirements and backed by the *World LHC Computing Grid*. `CERNBox`, the cloud synchronization service for CERN-associated end users, is implemented on top of EOS. EOS is accessible from `LXPLUS` through paths under the `/eos/` directive. This system hosts the data to be processed by the LHC data reduction instances; in particular, we are interested in the CMS L1T relevant data, making the `/eos/cms/` path suitable for our purposes./**eos**/**cms**/$\cdots$/**EphZB_2024E_run381148**$\cdots$/ is the ROOT (`.root`) data (*background*) home path of interest.



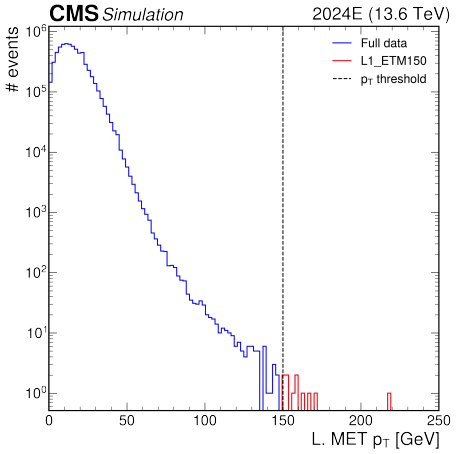
(a) ($p_T > 180$ GeV)-jet spectrum



(b) ($p_T > 22$ GeV)-μ spectrum



(c) ($p_T > 36$ GeV/$|\eta| < 2.5$)-e/γ spectrum



(d) ($p_T > 22$ GeV)-E_T^{miss} spectrum

Figure 11: Selected events from the standard inputs of the latest AXOL1TL baseline (μ, e/γ, E_T^{miss} and jets), triggered by single reconstructed L1-object L1-algorithms. The preprocessed (and raw) background dataset was used (see Subsection 7.2).

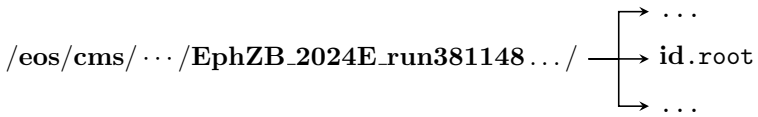


Figure 12: Schematic of the `/eos/` path for background samples. The events are partitioned into approximately 1700 .root files, optimized to facilitate parallelization and concurrency in distributed computing environments. These files are organized into specific trees: `event_tree_name` (records of the collected events), `gen_tree_name` (validation data generated from Monte Carlo simulations), `tree_name` (emulation of L1 trigger objects), and `uGT_tree_name` (μGT decisions emulation).

6.2 Signal Data

The *signal* refers to the data stream from benchmark new physics scenarios, used solely for evaluating the AD model's performance. These BSM samples are generated through Monte Carlo simulation using `GEANT4` under realistic energy conditions of the current Run 3 of the LHC, using `PYTHIA8` or `Madgraph` depending on the physical process and with a defined average pile-up.

Signal data in AD algorithms like AXOL1TL provide numerical performance metrics for the ML model, optimizing its performance based on the BSM benchmark signals through efficiency gains in L1 menu non-saturated signals, reflecting the additional detection power relative to the efficiency of only the L1 menu.

In particular, for the purposes of our study, we will employ **SUSY**, **VBF**, **LL**, **Leptonic**, **Top-quark**, **J/ψ**, **X-Y**, **Higgs** and **Higgsino** types BSM benchmark processes as signal samples to evaluate the performance in anomaly detection and efficiency improvement of the generalized AXOL1TL model. The signal samples are located at the path **/eos/cms/.../jngadiub/**. The names of the BSM processes (for example, **ggXToJpsiJpsiTo2Mu2E_m7**) are defined as *keys* in a configuration file named `config.yaml`. Each key is associated with a *value* that extends the base path specified in the *path* key of the `.yaml` file (**config**(`'path') = /eos/cms/.../jngadiub/`), thereby providing the exact location of the signal data. Therefore, the path of the `.root` files is **config**(`'path')`+**config**(`'ggXToJpsiJpsiTo2Mu2E_m7')`, or **/eos/cms/**$\cdots$**/jngadiub/**$\cdots$**133XWinter24/ggXToJpsiJpsiTo2Mu2E_m7**$\cdots$**/0000/**, corresponding to the first BSM process listed above (note that it follows the `L1NTuple_id.root` format).

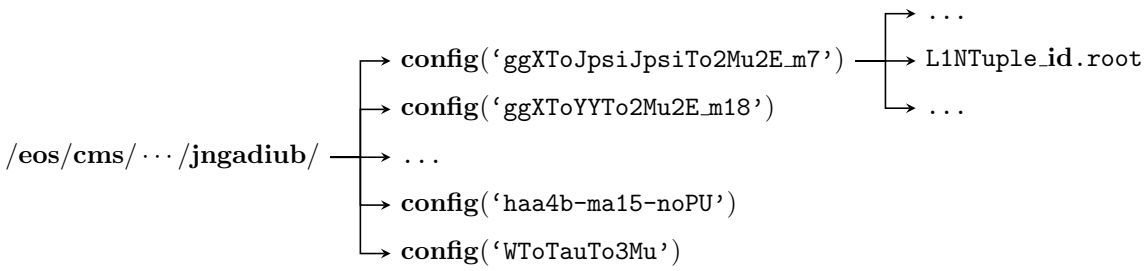
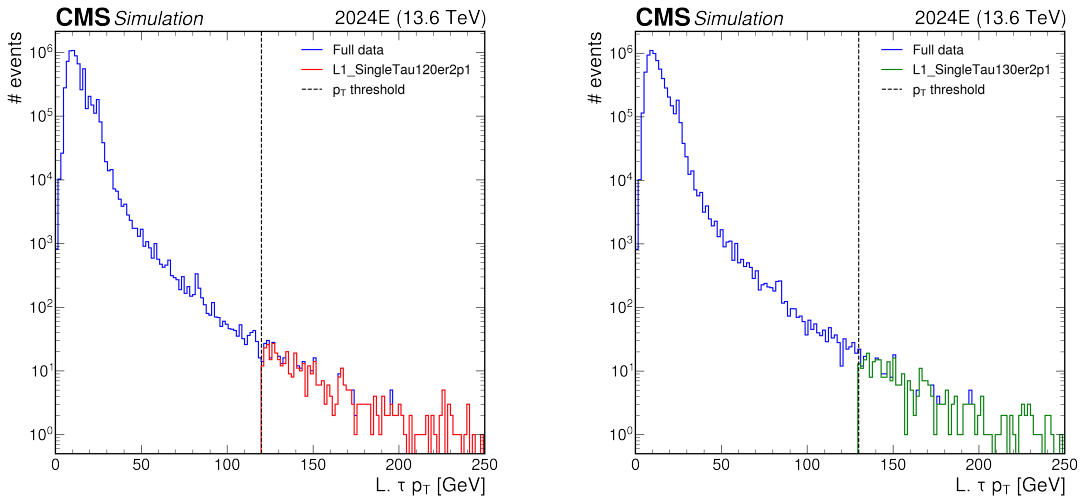


Figure 13: Schematic of the `/eos/` paths for the signal samples. The complete paths specify the event generator (such as `pythia8`, `powheg`, `amcatnlo`, and `jhugen`), the generator tune (`CP5`), the center-of-mass energy for the collision (13.6 TeV), the type of simulation (Monte Carlo), the simulation campaign (Run 3), and the *timestamp*.

6.3 New Inputs

Including tau (τ) leptons as input, completing the lepton family along with electrons and muons, will improve the robustness of AXOL1TL. Unlike e's and μ's , τ leptons, primarily produced by the decay of W and Z bosons at the LHC, are heavier and tend to decay into electrons and muons. While the energy deposit of a tau lepton is greater than that of an electron, the dynamic clustering system developed for e/γ's can reconstruct the products of its decay and consolidate its signature, making them an ideal choice for expanding AXOL1TL Bhowmik (2018).



(a) $p_T > 120$ GeV/$|\eta| < 2.1$ L1T selected events. (b) $p_T > 130$ GeV/$|\eta| < 2.1$ L1T selected events.

Figure 14: p_T spectrum of `L1_SingleTau120er2p1` and `L1_SingleTau130er2p1` triggers relative to the raw total spectrum of the leading (most energetic) τ-lepton.

In this study, we utilize data from run 381148 for the training and validation of our new trigger. Figure 14 presents the p_T spectra of the most energetic τ lepton, along with the band of events accepted by the L1 trigger `L1_SingleTau120er2p1` (with a p_T threshold of 120 GeV, restricted to $|\eta| < 2.1$), and similarly for `L1_SingleTau130er2p1`. It is observed that there are peaks of high p_T events that are not triggered, which may be attributed to limitations in the tau isolation criteria implemented by the current trigger. Figure 15 Y shows a 2D histogram scaled in ϕ and η (according to the scaling factors of the CMS calorimeters, see Listing 5) and weighted by p_T. A high concentration of events around $\eta \sim 0$ is observed, which is expected due to their short lifetime and greater mass compared to electrons and muons, favoring their detection near the central axis of the detector.

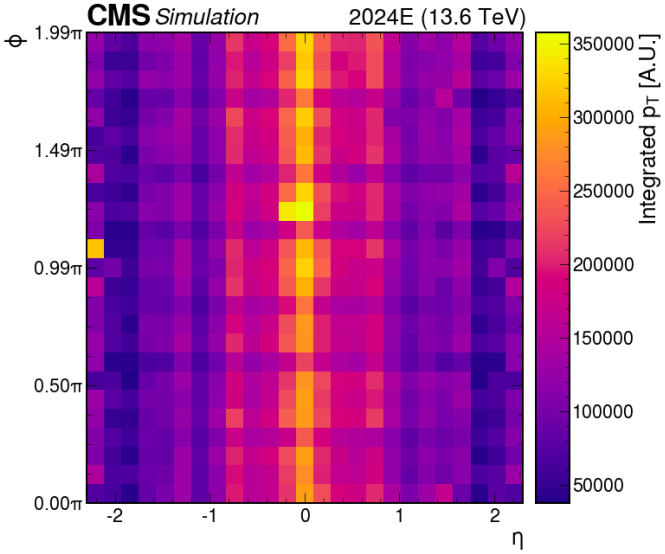


Figure 15: Normalized η-ϕ distribution of the highest-energy τ-lepton according to the scalers for L1 reconstructed objects in the L1T calorimeter. The hot-spot-like pattern is found at $|\eta| \sim 0$.

7 Data Processing

7.1 Signal Data Processing

The signal sample processing stage follows a merge cycle across datasets in `.h5` format. Given a predefined `config.yaml`, the **config**('path') can be utilized to convert the `.root` files in parallel, generating outputs at **config**('`path`') + **config**('{`bsm_type`}'). This conversion is performed for each **id** in `L1Ntuple_id`. The GitLab repository `L1AnomalyDetection/l1tntuple-maker`, part of the *CMS collaboration*, provides the **convert_to_h5.py** script to execute the conversion. The `output` key in `config.yaml` specifies the `/eos/` directory path under the **cms** − **axol1tl** project, where the converted files are stored. Note that the mentioned {`bsm_type`} are `config.yaml` keys that specify each BSM process within the configuration file:

Figure 16: ROOT to HDF5 conversion scheme for signal samples based on `bsm_type` (keys corresponding to `HHTo6B` and `ttHto2B` are shown). The `.root` and `.h5` files are generally not located in the same directory; their locations are defined in `config.yaml`.

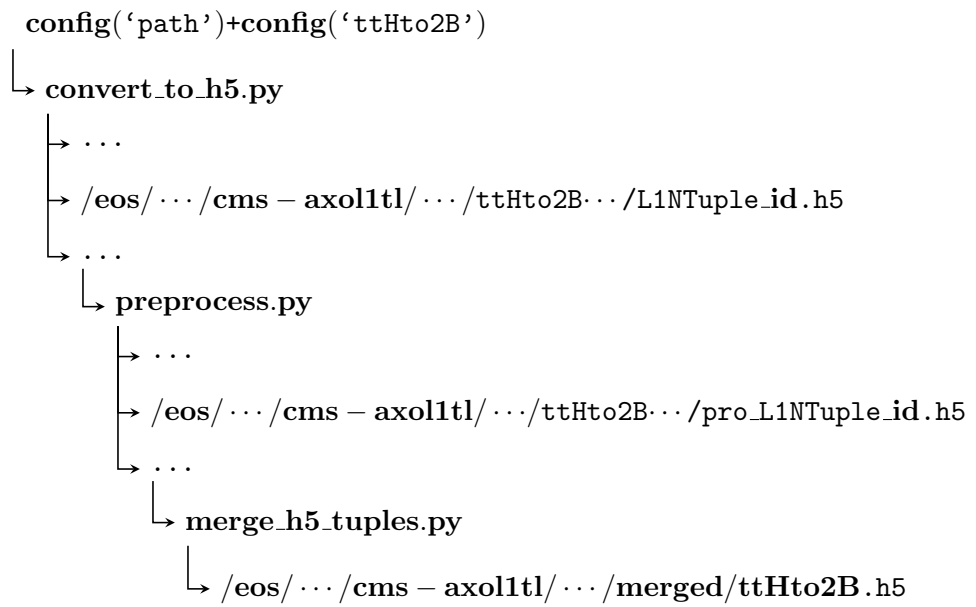


Figure 17: Conversion-preprocessing-merging scheme for signal samples based on `bsm_type` (`ttHto2B` in the example). The conversion results are located in the `/eos/` directory of the **cms** − **axol1tl** project. GitLab `L1AnomalyDetection/l1tntuple-maker` provides a *Snakemake* file, `Snakefile`, to perform this entire process (alternating merging with preprocessing) using the command `snakemake merge_all_bsm_types --cores 8`.

Consider the **ttHto2B** process as an example. Once the datasets in `.h5` format are obtained in the path of the `cms-axol1tl` project, they are preprocessed using the **preprocess.py** script for each **id** in `L1Ntuple_id` (available in `L1AnomalyDetection/l1tntuple-maker`). The resulting file is stored in the directory corresponding to the `bsm_type` (in this case, `ttHto2B`). The subsequent step involves merging

the `pro_ttHto2B.h5` files using **merge_h5_tuples.py** to consolidate the processed data for this BSM process into **ttHto2B.h5**. Finally, the files are transferred from **config**(`'output'`)+/**bsm_type**/ to **config**(`'output'`)+/**merged**/ for storing the merged files.

Therefore, we obtain a .h5 file representing each BSM signal, located in a different path from the original .root files:

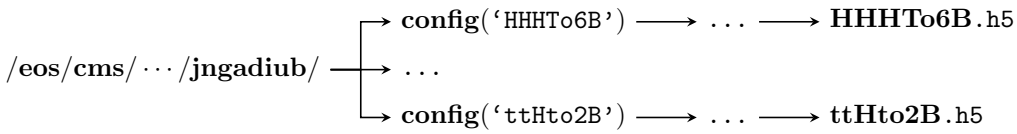


Figure 18: .h5 files resulting from the preprocessing-merging process for each `bsm_type`.

The final step of the preprocessing involves merging these BSM files into a single `.h5` file for each signal. To achieve this, **merge_h5_tuples.py** is used again, generating a unique and correctly preprocessed file, stored in the /**merged**/ directory of the **cms – axol1tl** project. This resulting file, `BSM.h5`, will be used to evaluate the performance of the extended AXOL1TL model against the new inputs.

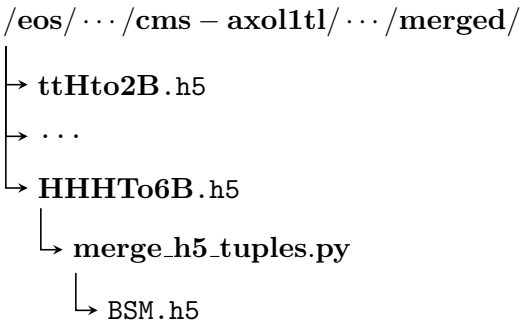
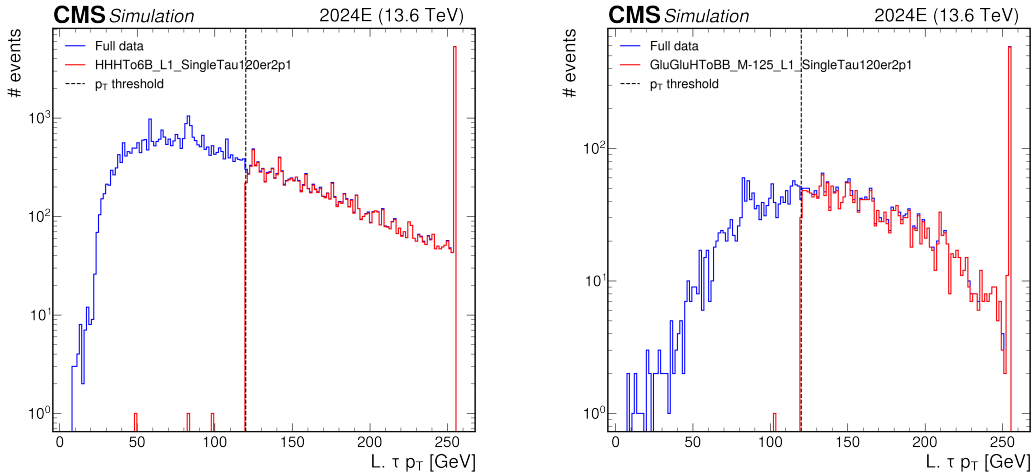


Figure 19: Merging of the .h5 files representing each BSM process signal.



(a) $HHH \to 6b$ ($p_T > 120$ GeV/$|\eta| < 2.1$)-selected events.

(b) $gg \to H \to bb$ ($p_T > 120$ GeV/$|\eta| < 2.1$)-selected events.

Figure 20: Spectrum of the `GluGluHToBB_M-125` and `HHHTo6B` signals over the preprocessed signal sample set `BSM.h5` as a function of the transverse momentum of the most energetic tau lepton. The region triggered by the L1 menu (`L1_SingleTau120er2p1`) is shown.

7.2 Background Data Processing

As shown in Figure 12, the base directory **/eos/cms/⋯/EphZB_2024E_run381148⋯/** contains the `.root` files corresponding to the *background* samples.

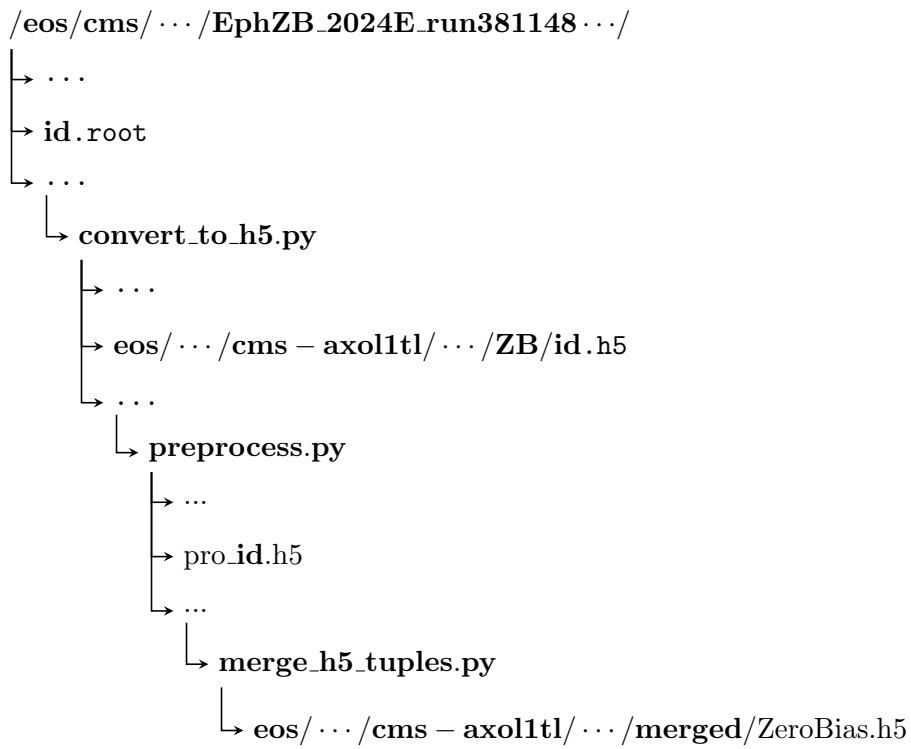


Figure 21: Total processing scheme for background samples. Analogous to the processing logic of signal samples, after a conversion step, preprocessing is performed to obtain the `pro_` files, followed by a single merging step using `merge_h5_tuples.py`. The final result is stored in **cms – axol1tl**.

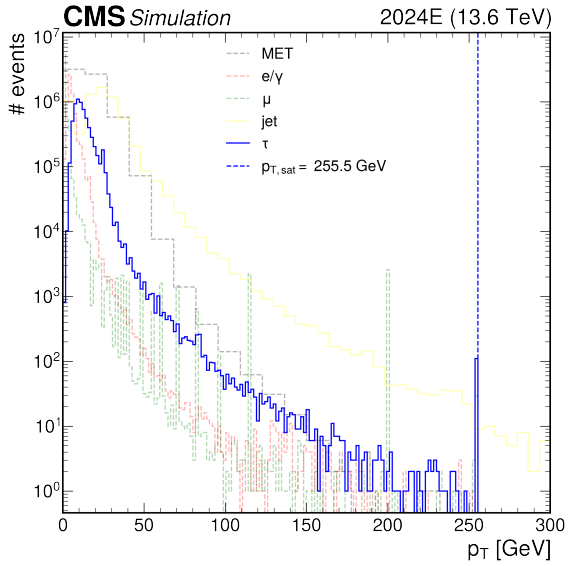


Figure 22: Saturated p_T determination for τ leptons using the last anomalous bin method in the most-energetic-τ spectrum.

The data processing workflow for the background samples follows a logical sequence of `conversion-preprocessing-merging`. For each **id**`.root`, a converted file **id**`.h5` is generated using `Python` or `Snakemake` commands, and is stored in the `/eos/` directory of the `axol1tl` project under the `zh` *egroup* (CMS collaboration's `LXPLUS` computational group), specifically in the /**ZB**/ folder. These **id**`.h5` files are preprocessed and saved as **pro_id**`.h5` in the same directory. Finally, they are merged into a single file, `ZeroBias.h5`, which is stored in the /**merged**/ directory within the axo's project path.

An essential parameter for data post-processing, prior to training the extended AXOL1TL model, is the transverse momentum saturation threshold (see Listing 1). This threshold is determined by analyzing the presence of a peak in the last bin of the background event distribution for the most energetic τ lepton. As shown in Figure 22, this value is 255.5 GeV, similarly observed for e/γ's and μ's. Additionally, the corresponding saturation energies for MET and jets were confirmed to be 2047.5 GeV and 1023.5 GeV, respectively.

8 Redefining the `axo` module

The GitLab repository `L1AnomalyDetection/l1_anomaly_ae` corresponds to the AXOL1TL anomaly detection project. The latest `branch`, *refactory*, contains the updated machine learning training pipeline for the AXOL1TL model, designed to be modular, customizable, and well-structured. The `axo` module hosts directories for data reading, manipulation, and preprocessing, including normalization, saturation handling, and quantization, organized in the `data_util` submodule. The architectures of the AXOL1TL autoencoder models (AE and VAE) are defined in the `models` submodule, which builds the AE for anomaly detection using parameters from a configuration file in `config.yml` format, specified in the `utilities` submodule. The `losses` submodule contains the loss functions used during training, derived from the `L1ADBaseLoss` class defined in `base.py`. The `data_util`, `models`, `utilities`, and `losses` submodules are strongly tied to the shape of the input tensor (currently related to the number of e's, μ's, and jets defined in `config.yml`), meaning their *Python* files will need to be modified to extend the model to include tau leptons (See Figure 23).

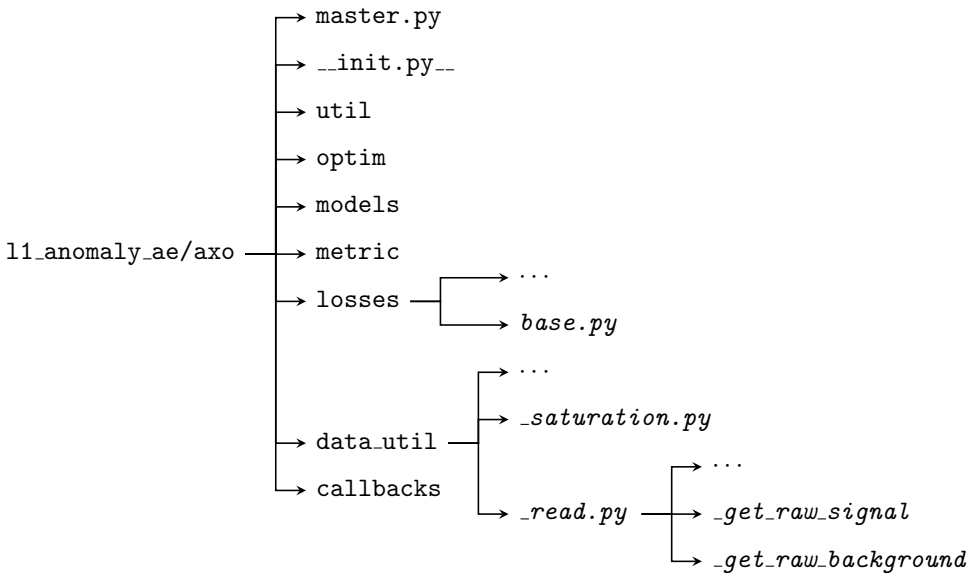


Figure 23: Directory structure of the module `l1_anomaly_ae/axo` in its latest branch, *refactory*. The `Python` files written in italics are those that depend on the nature of the reconstructed L1 objects that are inputs for AXOL1TL. The goal is to modify these scripts to implement tau leptons as new inputs. `_get_raw_background` and `_get_raw_signal` are functions defined within the script `_read.py`.

8.1 The `data_util` submodule

`data_util` is responsible for the reading, creation, and processing of data according to the L1T standards, ensuring that the information is properly prepared for AXOL1TL training. In this context, the mod-

ule `data_util` follows a *reading-saturation treatment-normalization configuration-quantization-packaging* data process for both *background* and *signal* samples. `data_util/data.py` details the complete data treatment procedure.

8.1.1 Saturation Treatment

As indicated in Figure 23, the script `data_util/_saturation.py`, as well as the functions `_get_raw_signal` and `_get_raw_background` functions in `data_util/_read.py`, directly depend on the L1 reconstructed physics objects inputs. In particular, `_saturation.py` is used to manage overflow values, which correspond to data from physics objects reconstructed with energy exceeding a maximum threshold due to the inherent limitations of the detectors. This adjustment ensures that their energy is compatible with the current trigger bandwidth or the maximum number of bits that can be transmitted. This situation may lead to inaccurate data and generate peaks in the energy distribution bins of the reconstructed objects (as is the case for jets, which are fixed at 1023.5 GeV and are regularly generated due to the p_T threshold for event triggering in the $O(100)$ GeV range). Filtering saturated events facilitates the learning of NNs in AXOL1TL. See Listing 1.

```
#...
def _remove_saturation(data_dict,constituents,saturation_mode):
    ET_thres = 4095
    nof_MET = np.sum(constituents['MET'])
    nof_egamma = np.sum(constituents['EGAMMA'])
    nof_mu = np.sum(constituents['MUON'])
    nof_jet = np.sum(constituents['JET'])
    nof_tau = np.sum(constituents['TAU'])

    pt_thres = np.array([2047.5] * nof_MET + [255.5] * (nof_egamma + nof_mu +
        nof_tau)
           + [1023.5] * nof_jet) * 2
#...
```

Listing 1: The script `_saturation.py` defines the function `_remove_saturation` as a filter for the `BACKGROUND-SIGNAL` events dictionary `data_dict` to remove entries that are saturated in E^T and p_T based on threshold values evaluated using `mask` (not shown). The `constituents` dictionary in `utilities/config.file` makes `_saturation.py` dependent on the inputs, generating an array `pt_thres` that defines specific p_T thresholds based on the current bit width for each L1 particle.

Note that `nof_tau` had to be added to account for the active constituents that will be considered among the 12 possible level-1 reconstructed τ leptons. `pt_thres_tau` is the bitwise representation of the maximum energy of these L1 tau leptons, serving as a saturation indicator. The corresponding conditions for tau leptons in the CMS detector were added.

8.1.2 Data Loading

The `_read.py` file serves as the starting point for AXOL1TL training. Specifically, the `get_data` functions (`get_data_background` and `get_data_signal`) define the entire workflow for loading and reading data, and are designed for use by end-users. These functions are linked to `_get_raw_background` and `_get_raw_signal`, respectively, which process the final `.h5` files (`ZeroBias.h5` and `BSM.h5`) resulting from the data processing described in X, for the background and signal samples.

As of the current version of AXOL1TL, `_get_raw_background` is designed to extract data for E_T^{miss}, e/γ's, μ's, and jets, as well as the metadata for ET, HT, PU, and L1bits from the background samples. These data are organized into a dictionary with `'Train'` and `'Test'` keys, generated by splitting the `ZeroBias.h5` file into training and validation portions.

```
#...
def _get_raw_background(file_path,train_sector,test_sector,object_ranges,verbose
    = False):
    f = h5py.File(file_path,"r")
    dataset = f["full_data_cyl"]
    dat_met = dataset[train_sector[0]:train_sector[1],
```

```
                object_ranges["met"], :]
    dat_egs = dataset[train_sector[0]:train_sector[1],
                object_ranges["egs"][0]:object_ranges["egs"][1],:]
    dat_muons = dataset[train_sector[0]:train_sector[1],
                object_ranges["muons"][0]:object_ranges["muons"][1],:]
    dat_jets = dataset[train_sector[0]:train_sector[1],
                object_ranges["jets"][0]:object_ranges["jets"][1],:]
    dat_taus = dataset[train_sector[0]:train_sector[1],
                object_ranges["taus"][0]:object_ranges["taus"][1],:]
    #...
```

Listing 2: The `_get_raw_background` function reads, extracts, and splits (into training and testing) data from the HDF5 file representing the background samples. The function's dependency on the constituents dictionary in `config.yml` is not shown. The `full_data_cyl` extracts the sections for each type of object, which includes τ leptons.

From Listing 2, note that the data corresponding to the training sector for the L1 reconstructed objects (`data_met`, `data_egs`, `data_muons`, and `data_jets`) have been extracted up to the current pipeline, as well as for the added tau object, through `dat_tau`.

```
    #...
    ET = f["ET"][train_sector[0]:train_sector[1]]
    HT = f["HT"][train_sector[0]:train_sector[1]]
    PU = f["event_info"][train_sector[0]:train_sector[1],:]
    L1_bits = f["L1bit"][train_sector[0]:train_sector[1]]
    #...
    return_dict = {}
    return_dict["Train"]={
    "DATA":{
        "MET":dat_met,
        "EGAMMA":dat_egs,
        "MUON": dat_muons,
        "JET": dat_jets,
        "TAU": dat_taus},
     "META":{
        "ET":ET,
        "HT":HT,
        "PU":PU,
        "L1bits":L1_bits}}
#...
```

Listing 3: A `return_dict` dictionary is returned with the keys `Train` and `Test` (the latter is not shown). A `TAU` key is illustrated as being added to the `return_dict` dictionary. The metadata `ET`, `HT`, `PU`, and `L1_bits` are also stored.

The dictionary shown in Listing 3, due to the introduction of taus in Listing 2, had to include the line `"TAU": data_taus` to incorporate these new objects into the training data structure description dictionary.

Similarly, `_get_raw_signal` processes the signal samples but without applying a split, as the metadata is extracted for each BSM process up to the maximum number of signals defined by the parameter `MAX_NUM_SIGNALS`. This clear dependency on the current objects highlights the need to modify `_read.py` to include reconstructed τ's.

```
def _get_raw_signal(file_path,MAX_NUM_SIGNALS,object_ranges,verbose = False):
    #...
    for signal_name in signal_names:
        dataset = data[signal_name]
        dat_met = dataset[:MAX_NUM_SIGNALS,object_ranges["met"],:]
        dat_egs = dataset[:MAX_NUM_SIGNALS,object_ranges["egs"][0]:object_ranges
            ["egs"][1],:]
        #... (Similarly with _muons and _jets)
        dat_taus = dataset[:MAX_NUM_SIGNALS, object_ranges["taus"][0]:
            object_ranges["taus"][1],:]
```

```
        ET = data[f'{signal_name}_ET'][:MAX_NUM_SIGNALS]
        #... (Similarly with HT,PU,L1Bits)
        return_dict[signal_name] = {
            "DATA":{
                "MET":dat_met,
                "EGAMMA":dat_egs,
                "MUON": dat_muons,
                "JET": dat_jets,
                "TAU": dat_taus},
             "META":{
                "ET":ET,
                "HT":HT,
                "PU":PU,
                "L1bits":L1_bits}}
    #...
```

Listing 4: The `_get_raw_signal` function in `_read.py` is constructed in the same way as `_get_raw_background`, with the difference that for each `bsm_type`, an additional dictionary is generated within the list of dictionaries in `return_dict`. `MAX_NUM_SIGNALS` is defined in `utilities/config.yml`.

8.2 The `losses` submodule

In `base.py`, the class `L1ADBaseLoss` is defined, inheriting from `tensorflow.keras.losses.Loss`, as a base tool from which custom loss functions to be added in the `losses` module, useful for the training process of AXOL1TL, should be derived. The file `axo/losses/base.py` primarily handles the preparation of scales and biases for adjusting values that will be used in the loss calculation, and for defining the normalization conditions of the physical variables (`PT`, `ETA`, and `PHI`) to later apply them in the definition of specific scalers (`PT_SCALER`, `ETA_SCALER`, `PHI_SCALER`).

```
#...
K = tf.keras.backend
from tensorflow.keras.losses import Loss
class L1ADBaseLoss(Loss):
    def __init__(self, norm_scales, norm_biases, mask, unscale_energy = False,
        name="L1ADLOSS"):
        MUON_PHI_SCALER = 2 * np.pi / 576
        CALO_PHI_SCALER = 2 * np.pi / 144
        MUON_ETA_SCALER = 0.0870 / 8
        CALO_ETA_SCALER = 0.0870 / 2
        PT_CALO_SCALER = 0.5
        PT_MUON_SCALER = 0.5
#...
```

Listing 5: Definition of scalers at the calorimeter and muon triggers level in the L1T. These `*_SCALER` variables will be used to normalize and prepare the three momenta for each high-energy event in both background and signal samples.

The following `_SCALER` magnitudes strictly depend on the number of reconstructed objects being considered (currently 1 E_T^{miss}, 12 e/γ's, 8 μ's, and 12 jets). This should be extended to include τ leptons (12) as new inputs. See Listing 6.

```
#...
        PT_SCALER: np.ndarray = np.array([PT_CALO_SCALER] * 13
                                + [PT_MUON_SCALER] * 8
                                + [PT_CALO_SCALER] * 12
                                + [PT_CALO_SCALER] * 12)
        ETA_SCALER: np.ndarray = np.array([CALO_ETA_SCALER] * 13
                                + [MUON_ETA_SCALER] * 8
                                + [CALO_ETA_SCALER] * 12
                                + [CALO_ETA_SCALER] * 12)
```

```
        PHI_SCALER: np.ndarray = np.array([CALO_PHI_SCALER] * 13
                                  + [MUON_PHI_SCALER] * 8
                                  + [CALO_PHI_SCALER] * 12
                                  + [CALO_PHI_SCALER] * 12)
#...
```

Listing 6: `Numpy` arrays of p_T, η, and ϕ scalers, for each constituent reconstructed in the Calorimeter Trigger and in the Muon Trigger. [`CALO_ETA_SCALER`]`*12` is explicitly added to introduce taus as CALO-L1 objects..

Note that from the listing above (Listing 6), arrays of the form `[CALO_*_SCALER]*12` have been added to cover the 12 possible new τ objects for normalizing the constituent features. These `NumPy` arrays will then pass through a filter defined by the active components specified in the mask parameter (which stores the `constituents` key of `config.yml`).

8.3 The `utilities` submodule

The `utilities` folder contains the master configuration file `config.yml`, which holds the default settings for training. AXOL1TL can be run using this default configuration file, which defines the training and testing sectors on `ZeroBias.h5` (`train_sector` and `test_sector`) and the constituents from the total number of reconstructed L1 objects to be used (currently 1 of 1 E_T^{miss}, 4 of 12 e/γ's, 4 of 8 μ's, and 10 of 12 jets). It also defines the bits for quantization, the nodes for the AXOL1TL VAE architecture, and other variables such as `batch_size`, `n_epochs`, the optimizer and α and β loss parameters. We are particularly interested in extending the constituents to include τ objects in both the `BACKGROUND` and `SIGNAL` configurations in `config.yml`.

```
data_config:
  Read_configs:
    BACKGROUND:
      file_path: ../../L1NTuple-133XWinter24/ZeroBias.h5
      #...
      object_ranges:
        met: 0
        egs: !!python/tuple
        - 1
        - 13
        muons: !!python/tuple
        - 13
        - 21
        jets: !!python/tuple
        - 21
        - 33
        taus: !!python/tuple
        - 33
        - 45
      constituents:
        MET:
        - true
        EGAMMA:
        - true
        - true
        #...
        - false
        MUON:
        - true
        - true
        #...
        - false
        JET:
        - true
        #..
        - false
        TAU:
```

```
        - true
        #..
        - false
    #..
```

Listing 7: AXOL1TL configuration file, `config.yml`. It defines the `BACKGROUND` and `SIGNAL` sample dictionaries, along with settings for saturation, normalization, quantization, determinism, VAE architecture nodes, and training configurations. τ leptons were added as constituents in `BACKGROUND` and `SIGNAL` dictionaries, associated to an additional range in `object_ranges`.

Note that in Listing 7, the corresponding keys associated with the 12 additional possible τ leptons were added to `object_ranges` and `constituents`. Specifically, the boolean lists for the reconstructed objects correspond to the number of active particles of each type for the model, which serves as an initial filter on the `_SCALER` functions via the mask in **base.py**, where the physical space for training is defined. Similarly to the `BACKGROUND` key, the `SIGNAL` key must be extended. The remaining keys in `config.yml` are independent of the L1 inputs.

9 Redifining `l1tntuple-maker` for τ addition

`L1AnomalyDetection/l1tntuple-maker` is the GitLab repository where the `.py` scripts for data preprocessing are located. Defined by a `conda` environment specified in `converter.yml`, we will use the `133XWinter24_newInputs` branch, which is a promising extension of `133XWinter24` that now includes new L1 objects, τ's, as well as new features of the existing objects up to the latest version of AXOL1TL, as in the case of reconstructed jets. Figure 24 shows the module diagram of interest for `l1tntuple-maker`, highlighting the modifications dependent on the nature of the L1 inputs.

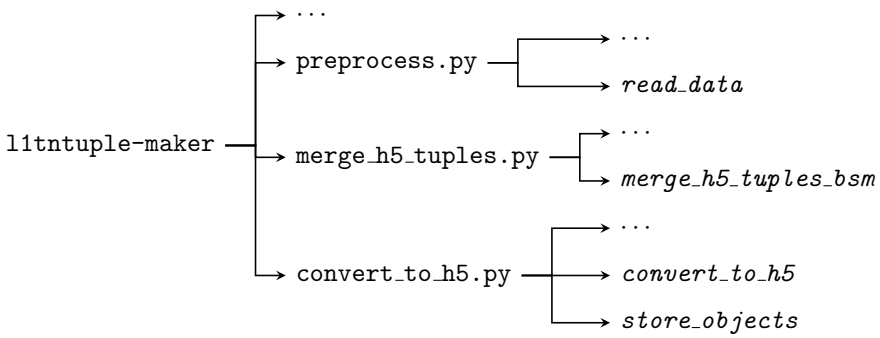


Figure 24: Directory structure of the `l1tntuple-maker` project. `Python` files used to preprocess background and signal samples are shown. Names in italics indicate functions within `.py` scripts that depend directly on reconstructed L1 objects and their high-energy attributes (`AtVtx`, `Qual`, `Iso`, `Bx`, etc.).

9.1 The `convert_to_h5.py` script

9.1.1 From ROOT to HDF5

The function `convert_to_h5` in `convert_to_h5.py` is key for converting data from ROOT format (`.root`) to HDF5 (`.h5`). As described in Figure 12, the trees `tree_name`, `uGT_tree_name`, `event_tree_name`, and `gen_tree_name` are necessary for reading L1 information and are accessed through the uproot library. `prescale_file` contains information about the prescaling of algorithms in the trigger system for reducing the rate of processed events by adjusting the probability of an event being recorded according to the relevant algorithms selected using the `filterAlgoMap` method, while the `getAlgoMap` method extracts information about trigger decisions from the trees. Listing 8 shows the script added to the `convert_to_h5` function for the new AXOL1TL baseline.

```
def convert_to_h5(input_file, output_file, prescale_file, tree_name,
    uGT_tree_name, event_tree_name, gen_tree_name, mc, hw):
```

```
    #...
    ntaus = 12
    cylNames = ['pT', 'eta', 'phi']
    evtInfoList = ["run","lumi","event","bx","orbit","time","nPV_True"]
    if hw == True:
        varList = [...,'jetHwQual',
                   ..., 'muonIEtaAtVtx','muonIPhiAtVtx', ... ,
                   'tauIEt', 'tauIEta', 'tauIPhi', 'tauIso', 'tauBx']
    else:
        varList = [..., 'jetHwQual',
                    , 'muonEtaAtVtx', 'muonPhiAtVtx',...,
                    , 'tauEt', 'tauEta', 'tauPhi', 'tauIso', 'tauBx']
    #...
    if hw == True:
        l1Jet_cyl, _, _, _, l1Jet_qual = store_objects(arrays, nentries, nobj=
            njets, obj='jetI')
        #...
        l1tau_cyl, l1tau_iso, _, _, _ = store_objects(arrays, nentries, nobj=
            ntaus, obj='tauI')
    else:
        #...
        l1tau_cyl, l1tau_iso, _, _, _ = store_objects(arrays, nentries, nobj=
            ntaus, obj='tau')
    #...
```

Listing 8: Definition of L1 configuration parameters (adding the number of τ's) and description variables (`pT`, `eta`, and `phi` in cylindrical coordinates, stored in `cyl`) in **convert_to_h5.py**. The newly added variables compared to the 133XWinter24 version are displayed in `varList`, featuring attributes for taus and new features for muons and jets. `store_objects` stores information for each L1 object, now including τ's.

Note that `ntaus` is added in the initialization of L1 object variables, while maintaining the characteristics pT, eta, and `phi`, as well as the event information variables that will be extracted from the `evtTree` through `evtInfoList`. Data is extracted from `varList` from l1Tree, whether from software or hardware (`hw`), for specific L1 particles in each event (jets, μ's, $e\gamma$'s, and τ's). The variables `jetHwQual`, `muonEtaAtVtx`, `muonPhiAtVtx`, `tauEt`, `tauEta`, `tauPhi`, `tauIso`, and `tauBx` (software), along with their corresponding I-variables, such as `tauIEt` (hardware), were added, which will enable a more robust analysis of events by considering quantities like E^T, η ϕ, isolation, and bunch crossing of τ objects.

```
    #...
    outFile = h5py.File(output_file, 'w')
    #...
    outFile.create_dataset('l1Jet_Qual', data=l1Jet_qual, compression='gzip')
    #...
    outFile.create_dataset('l1Tau_cyl', data=l1tau_cyl, compression='gzip')
    outFile.create_dataset('l1Tau_Iso', data=l1tau_iso, compression='gzip')
    #..
```

Listing 9: HDF5 file creation statement (containing L1 attributes). The new attributes `l1Jet_Qual`, `l1Tau_cyl`, and `l1Tau_Iso` are presented, corresponding to datasets related to jet quality, as well as the cylindrical momentum representations and isolation of τ leptons. The file size is reduced using the `gzip` method. For each `.root` file (both for background and signal samples), an `.h5` file is created containing this information.

9.1.2 Storing L1 objects

To facilitate the analysis of L1-reconstructed objects' data in particle collision events stored in ROOT trees, transforming these hierarchical structures (`awkward` arrays) into standard `NumPy` arrays is an optimal solution. The `store_objects` function handles this conversion and organizes the objects based on their p_T. Its inputs include: `arrays = l1Tree.arrays(varList)` (the data arrays initialized), `nentries = l1Tree.num_entries` (the number of events), `nobj = {njets, ntaus, nelectrons, nmuons}` (the

number of objects to store per event), `obj = {jet, muon, eg, tau}` (the object type), and the sorting option by p_T, `sort`. These values are defined inside `convert_to_h5`.

```
def store_objects(arrays, nentries, nobj=10, obj='jet', sort=False):
    #...
    if maxobj>nobj:
            l1Obj_cyl = np.zeros((nentries, maxobj, 3), dtype=np.float16)
            #_iso, _dxy, _upt, _qual
            pt = to_np_array(arrays['{}Et'.format(obj)][mask], maxN=maxobj)
            if 'muon' in obj:
                eta = to_np_array(arrays['{}EtaAtVtx'.format(obj)][mask], maxN=
                    maxobj)
                phi = to_np_array(arrays['{}PhiAtVtx'.format(obj)][mask], maxN=
                    maxobj)
            else:
                eta = to_np_array(arrays['{}Eta'.format(obj)][mask], maxN=maxobj
                    )
                phi = to_np_array(arrays['{}Phi'.format(obj)][mask], maxN=maxobj
                    )
        else:
            l1Obj_cyl = np.zeros((nentries, nobj, 3), dtype=np.float16)
            #_iso, _dxy, _upt, _qual
            pt = to_np_array(arrays['{}Et'.format(obj)][mask], maxN=nobj)
            #...
            if 'muon' in obj:
                eta = to_np_array(arrays['{}EtaAtVtx'.format(obj)][mask], maxN=
                    nobj)
                phi = to_np_array(arrays['{}PhiAtVtx'.format(obj)][mask], maxN=
                    nobj)
            else:
                eta = to_np_array(arrays['{}Eta'.format(obj)][mask], maxN=nobj)
                phi = to_np_array(arrays['{}Phi'.format(obj)][mask], maxN=nobj)
#...
```

Listing 10: Acquisition of L1 moments (p_T, η, ϕ) from `awkward` to `NumPy` (via the `to_np_array` method) for each object. `maxobj` is the maximum number of L1 objects, defined as the maximum number of entries for the Et array of a specific object `obj`. A boolean mask is created based on `Bx=0` for filtering (not shown). **max**(`maxobj`, `obj`) defines a dimension for the generated datasets. Unlike the 133XWinter branch, the `AtVtx` attributes are now retrieved, given their improved precision and resolution in events with muons.

The first step for improving the accuracy of the training data is found in the condition `if 'muon' in obj:`. This suggests that, in the presence of μ's, one should access the attributes at the vertex `AtVtx` (`EtaAtVtx`and `PhiAtVtx`), which refer to the corrections of η and ϕ *at the event vertex*, as these are not direct measurements from the detectors. Objects such as jets or e's, having more direct trajectories, do not require this type of correction, unlike μ's, which are deflected by magnetic fields. The variable `mask` is a boolean filter that has been defined within the `store_objects` function (it does not appear in Listing 10) to select events with null bunch crossing (`mask = arrays['Bx'.format(obj[:-1])] == 0`), focusing on the current beam crossing.

```
    #...
    if obj in ['eg', 'muon', 'tau'] or obj in ['egI', 'muonI', 'tauI']:
        if maxobj>nobj:
            l1Obj_iso = to_np_array(arrays['{}Iso'.format(obj.replace("I",''))][
                mask], maxN=maxobj)
        else:
            l1Obj_iso = to_np_array(arrays['{}Iso'.format(obj.replace("I",''))][
                mask], maxN=nobj)
    if 'muon' in obj:
        if maxobj>nobj:
            l1Obj_dxy = to_np_array(arrays['muonDxy'.format(obj)][mask], maxN=
                maxobj)
```

```
            l1Obj_upt = to_np_array(arrays['{}EtUnconstrained'.format(obj)][mask
                ], maxN=maxobj)
            l1Obj_qual = to_np_array(arrays['muonQual'.format(obj)][mask], maxN=
                maxobj)
        else:
            l1Obj_dxy = to_np_array(arrays['muonDxy'.format(obj)][mask], maxN=
                nobj)
            l1Obj_upt = to_np_array(arrays['{}EtUnconstrained'.format(obj)][mask
                ], maxN=nobj)
            l1Obj_qual = to_np_array(arrays['muonQual'.format(obj)][mask], maxN=
                nobj)
    if 'jet' in obj:
        if maxobj>nobj:
            l1Obj_qual = to_np_array(arrays['jetHwQual'.format(obj)][mask], maxN
                =maxobj)
        else:
            l1Obj_qual = to_np_array(arrays['jetHwQual'.format(obj)][mask], maxN
                =nobj)

#...
```

Listing 11: The `l1Obj_*` datasets are completed for each object (hardware or software), corresponding to each high-energy attribute. `133XWinter24_NewInputs` includes the attributes `Iso`, `muonDxy` (particle transverse impact distance in the XY plane), `EtUnconstrained` (absolute E^T of the particle), and `MuonQual` in the presence of μ's, as well as `JetHwQual` in the presence of jets.

Note that in the conditional `if obj in ['eg', 'muon', 'tau'] or obj in ['egI', 'muonI', 'tauI']`, which now includes τ leptons, the isolation attribute `Iso` is obtained for the corresponding object. If the object is a muon, the attributes `muonDxy` (the transverse distance of the muon from the primary vertex), `EtUnconstrained` (the unconstrained transverse energy of the muon), and `muonQual` (the quality of the muon) are obtained. On the other hand, if the object is a jet, its quality `jetHwQual` is obtained (which was not available in the branch `133XWinter24`). The inclusion of the τ lepton and the quality of the jet allows for a more comprehensive analysis of collision events.

```
    if sort:
        #...
        if not np.allclose(sort_indices, check_indices):
            l1Obj_cyl[:,:,0] = np.take_along_axis(l1Obj_cyl[:,:,0], sort_indices
                , axis=1)
            #[:,:,1], [:,:,2]
            if obj in ['eg', 'muon', 'tau'] or obj in ['egI', 'muonI', 'tauI']:
                l1Obj_iso = np.take_along_axis(l1Obj_iso, sort_indices, axis=1)
            if 'muon' in obj:
                l1Obj_dxy = np.take_along_axis(l1Obj_dxy, sort_indices, axis=1)
                #_upt, _qual
            if 'jet' in obj:
                    l1Obj_qual = np.take_along_axis(l1Obj_qual, sort_indices,
                        axis=1)
    return l1Obj_cyl[:,:nobj], l1Obj_iso[:,:nobj], l1Obj_dxy[:,:nobj], l1Obj_upt
        [:,:nobj], l1Obj_qual[:,:nobj]
#...
```

Listing 12: Sorting of all datasets `l1Obj_*` according to p_T, in descending order using the method `np.allclose(sort_indices, check_indices)`. The sorted datasets are returned up to the first `nobj` objects associated with the particle `obj`. This is used, per L1 object, in the function `convert_to_h5`.

Like Sun (2023) and Govorkova, Puljak, Aarrestad, et al. (2021), sorting the data corresponding to each object in descending order based on their energy, represented by p_T, is a common practice in trigger AD algorithms. In Listing 12, it is noted that the branch `133Xwinter24_newInputs` has expanded this section of the function `store_objects` to include the case of τ leptons and jets. Depending on the type

of object, the corresponding attributes are also sorted following the same p_T order, using `sort_indices = np.argsort(-pt, axis=1)`.

9.2 The `preprocess.py` script

9.2.1 Reading `.h5` data

Since attributes have been added to some L1 objects (such as `JetHwQual` for jets), and tau leptons have been introduced as a new input, the `read_data` function has been modified accordingly to reflect this update in the reconstructed object space used by AXOL1TL.

```
def read_data(input_file):
    f = h5py.File(input_file, 'r')
    #...
    mydata = np.array(f.get('l1Tau_cyl'))
    t_cyl = mydata
    #...
    mydata = np.array(f.get('l1Tau_Iso'))
    t_iso = mydata
    #...
    mydata = np.array(f.get('l1Jet_Qual'))
    j_qual = mydata
    #...
    print('The jet data shape is: ', j_cyl.shape, j_qual.shape)
    print('The tau data shape is: ', t_cyl.shape, t_iso.shape)
   #...
```

Listing 13: Reading datasets from an HDF5 file `input_file` (for each `.h5` in the already converted ROOT files, both background and signal samples) and extracting particle physics objects and metrics into NumPy arrays. `t_cyl` corresponds to the new input, while `t_iso` and `j_qual` correspond to the new attributes introduced in `133XWinter24_NewInputs`.

From Listing 13, it is evident that the datasets corresponding to the attributes `l1Tau_cyl` and `l1Tau_Iso` in the input `.h5` dataset are included as a result of integrating tau leptons into the model. Similarly, the attribute `l1Jet_Qual` for jets has also been incorporated. `t_cyl`, `t_iso`, and `j_qual`, along with the datasets corresponding to the attributes of the other L1 objects, will be prepared for preprocessing through concatenation.

```
    full_data_cyl = np.concatenate([met_cyl, e_cyl, m_cyl, j_cyl, t_cyl],
                    axis=1)
    full_data_iso = np.concatenate( [..., t_iso],...)
    full_data_dxy = np.concatenate([...,
                    np.zeros([t_cyl.shape[0],t_cyl.shape[1]],dtype=np.float16)],
                    ...)
    full_data_upt = np.concatenate([...,
                    np.zeros([t_cyl.shape[0],t_cyl.shape[1]],dtype=np.float16)],
                    ...)
    full_data_qual = np.concatenate([..., j_qual,
                    np.zeros([t_cyl.shape[0],t_cyl.shape[1]],dtype=np.float16)],
                    ...)
    #...
    return full_data_cyl, full_data_iso, full_data_dxy, full_data_upt,
        full_data_qual, ET, HT, genHT, l1bit, seedinfo, evt_info
```

Listing 14: Update of the creation statement for the `full_*` datasets (which contain information on `_cyl`, `_iso`, `_dxy`, `_upt`, and `_qual` of the L1 objects) from the `133XWinter24` branch. Zeros are inserted in the positions of particles where no attribute has been specified. The `NumPy` arrays `full_data_*`, event energy information, L1T decision bits, and trigger system activation seeds are returned.

Note from Listing 14 that data is being manipulated using `np.concatenate`. `full_data_cyl` organizes the cylindrical description of e's, μ's, τ's, E_T^{miss}, and jets into columns. `full_data_iso` combines the

isolation values of e's, μ's, and τ's, introducing zeros for quantities without associated isolation, such as E_T^{miss} and jet. `full_data_upt` does the same with the `upt` attribute for μ's, and `full_data_qual` handles the quality data for μ's and jets. τ's, as well as other objects not mentioned in each case, are filled with a matrix of zeros. Notably, the isolation of τ is of particular interest, as it provides information about non-noisy events.

10 Set-Up

We will define the data post-processing configurations (*saturation-normalization-quantization*) and the model architecture to establish a common ultra-fast environment and perform a fair comparison between the latest version of AXOL1TL (*refactory* branch of `L1AnomalyDetection/l1_anomaly_ae`) and the extended version (with τ leptons as new inputs). A computing *container* in `CERN SWAN` linked to `CERNBox` will be used, with the compiler and flags configured in `gcc111 AlmaLinux9`, featuring a `Tesla T4 GPU` with 15,360 MiB of GDDR6 memory and driver version 550.54.15 at a temperature of 38°C, supporting `CUDA` 12.4, and equipped with 4 Intel®Xeon®Silver 4216 CPUs at a base frequency of 2.10 GHz, supporting virtualization and a single `NUMA` node.

For both models, the data range [0, 2000000] will be selected as the training sector for all L1 objects, and the range [2000000, 8000000] will be used as the testing sector. The `object_ranges` key from Listing 7 will be followed, where (0) corresponds to E_T^{miss}, (1-13) to e/γ's, (13-21) to μ's, (21-33) to jets, and (now) (33-45) to τ's. The total amount of training and testing data will depend on how many events pass the saturation filter on E^T and p_T defined in Figure 23 in **_saturation.py**, and extended in Listing 1 for the new baseline.

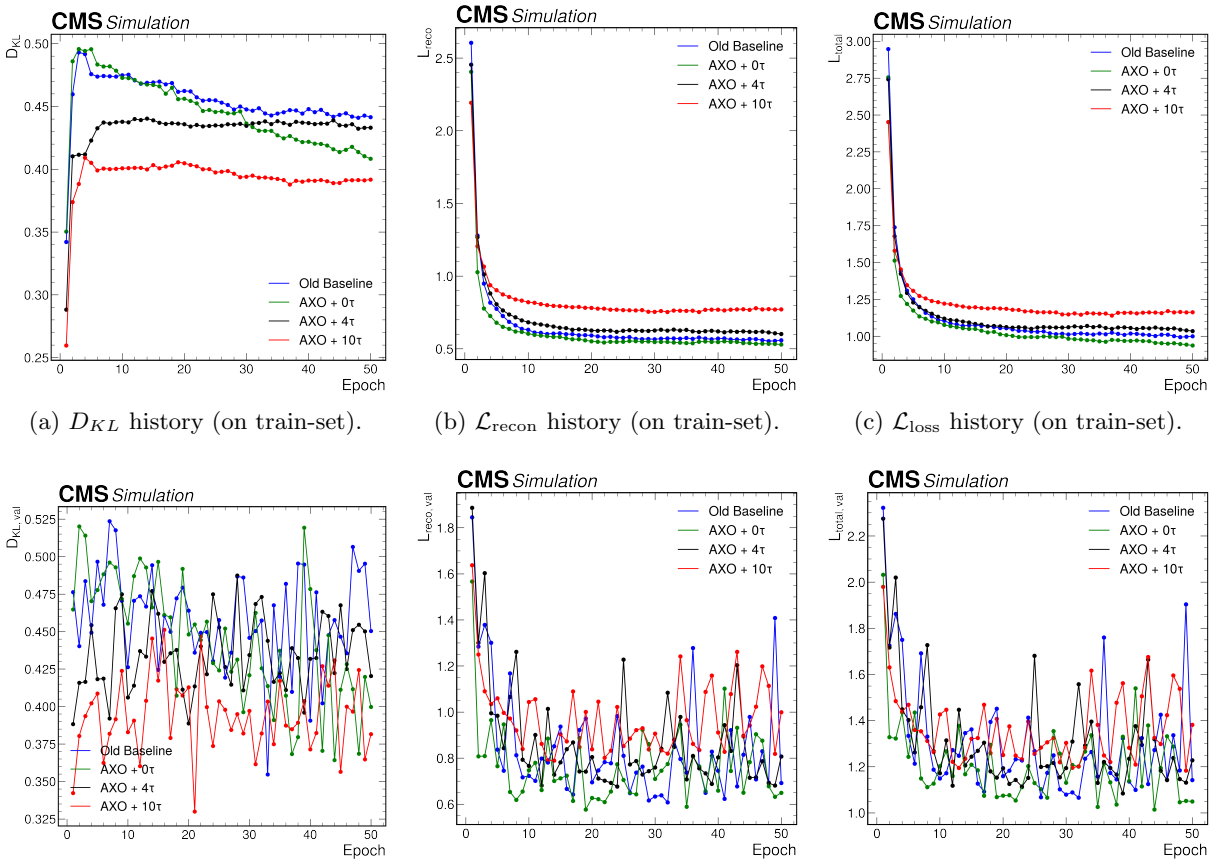
The postprocessing of both raw datasets (**2024E-Run1381148** standard and with added τ leptons) is initiated using normalization based on the `RobustScaler_pow2(5,95,-2,2)` scaling method, where $[5, 95]$ specifies the percentile range and $[-2, 2]$ defines the scaling/transformation bounds. The `norm_ignore_zeros=True` flag is applied to filter out rows where the first value is zero in the percentile calculation. For quantization, a bit configuration of $[8, 5]$ (8 bits for quantization and 5 bits for the integer part) will be employed to clip the normalized background (training and testing) and signal data (for each BSM process).

As specified in Figure [6-7], AXOL1TL is VAE-based. To compare the latest baseline with the new baseline, we will use an encoder with two hidden layers of 28 and 15 neurons, which is reduced to a latent space of 8 dimensions. For the decoder (asymmetric to the encoder), we will use 24, 32, 64, 128, and 57 (87) neurons for reconstruction from the latent space. Note that the last layer of the encoder matches the number of input features, which is $57 = 19 \times 3$ for the current AXOL1TL model, and $87 = 29 \times 3$ (where 19 and $29 = 19 + 10$ correspond to the number of constituents, considering the 10 most energetic τ's for the new baseline). We will also implement this for a configuration where the four most energetic τ leptons are selected (resulting in an input dimensionality to our VAE of $29 + 4 \times 3 = 69$), as well as for the scenario where no tau leptons are selected. A QAT-based workflow is configured using fixed-point representations for the model weights (`ap_fixed_kernel = [6,2]`), neuron biases (`ap_fixed_bias = [10,6]`), neuron activations (`ap_fixed_activation = [10,6]`), and the network's input and output data (`ap_fixed_data = [8,5]`). See Listing 15.

```
model_config["latent_dim"] = 8
model_config["decoder_config"] = {"nodes":[24,32,64,128,57]}
model_config["features"] = 57
```

Listing 15: AXOL1TL baseline architecture configuration. The truncated model up to the encoder is used for inference.

For training the VAE we set the hyperparameters $\alpha = 1$ and $\beta = 0.604108559135001$ for the reconstruction and D_{KL} regularization term weights, respectively. We have chosen a batch size of 10 000, 50 epochs, and an *Adam* optimizer with a learning rate of 0.001. The MSE over the cylindrical space $p_T p_z$ across the batches is used as the reconstruction loss (`axo/losses/cyl_PtPz.py` handles this). To introduce the out-of-distribution abnormality degree, $D_{KL}\left[N(\mu, \sigma^2)||N(0, 1)\right]$, in the total VAE loss, `axo/losses/kld.py` is used. Both formulas are implemented in `TensorFlow` and passed to the model for simultaneous optimization via $\mathcal{L}_{\text{loss}} = \alpha(1-\beta)\mathcal{L}_{\text{recon}} + \beta D_{KL}\left[N(\mu, \sigma^2)||N(0, 1)\right]$. The metrics `kl_loss` ($D_{KL}$ loss), `reco_loss` (reconstruction loss), and `total_loss` (total loss) are calculated per epoch during training to guide the optimization process. `val_kl_loss`, `val_reco_loss`, and `val_loss` evaluate the performance, during validation, on unseen data, helping to prevent overfitting.



(a) D_{KL} history (on train-set). (b) $\mathcal{L}_{\text{recon}}$ history (on train-set). (c) $\mathcal{L}_{\text{loss}}$ history (on train-set).

(d) D_{KL} history (on test-set). (e) $\mathcal{L}_{\text{recon}}$ history (on test-set). (f) $\mathcal{L}_{\text{loss}}$ (on test-set)

Figure 25: Complete training and validation metrics history (50 epochs) for AXOL1TL (full AE). The four scenarios described in this section are compared. The architecture and number of intermediate nodes (encoder and decoder) remain consistent, as does the dimension of the latent space.

Figure 25 demonstrates that D_{KL} stabilizes rapidly across all configurations, with an increasing preference for regularizing the adjustment of the latent space toward a Gaussian distribution as tau leptons are introduced. This trend is particularly pronounced in the model with 10 τ's, indicating that tau leptons play a significant role in shaping the latent distribution. Additionally, $\mathcal{L}_{\text{reco}}$ exhibits a slight increase per epoch as the number of taus included in the model increases (0 taus, 4 taus, and 10 taus). This behavior can be attributed to the fact that incorporating more τ leptons as input features expands the complexity of the feature space, resulting in dimensionalities of 57, 69, and 87, respectively. The fixed 8-dimensional bottleneck in the latent space may be insufficient to fully encode the underlying structure of the input, thereby leading to a greater loss of information during reconstruction. The model utilizing 4 taus emerges as the most optimal in terms of both reconstruction and latent regularization losses. This suggests that the 4 most energetic taus likely contain the most relevant information about the event, while the additional 6 taus may introduce noise or less relevant data, thereby diminishing overall performance.

11 Results

11.1 Signal efficiency gains

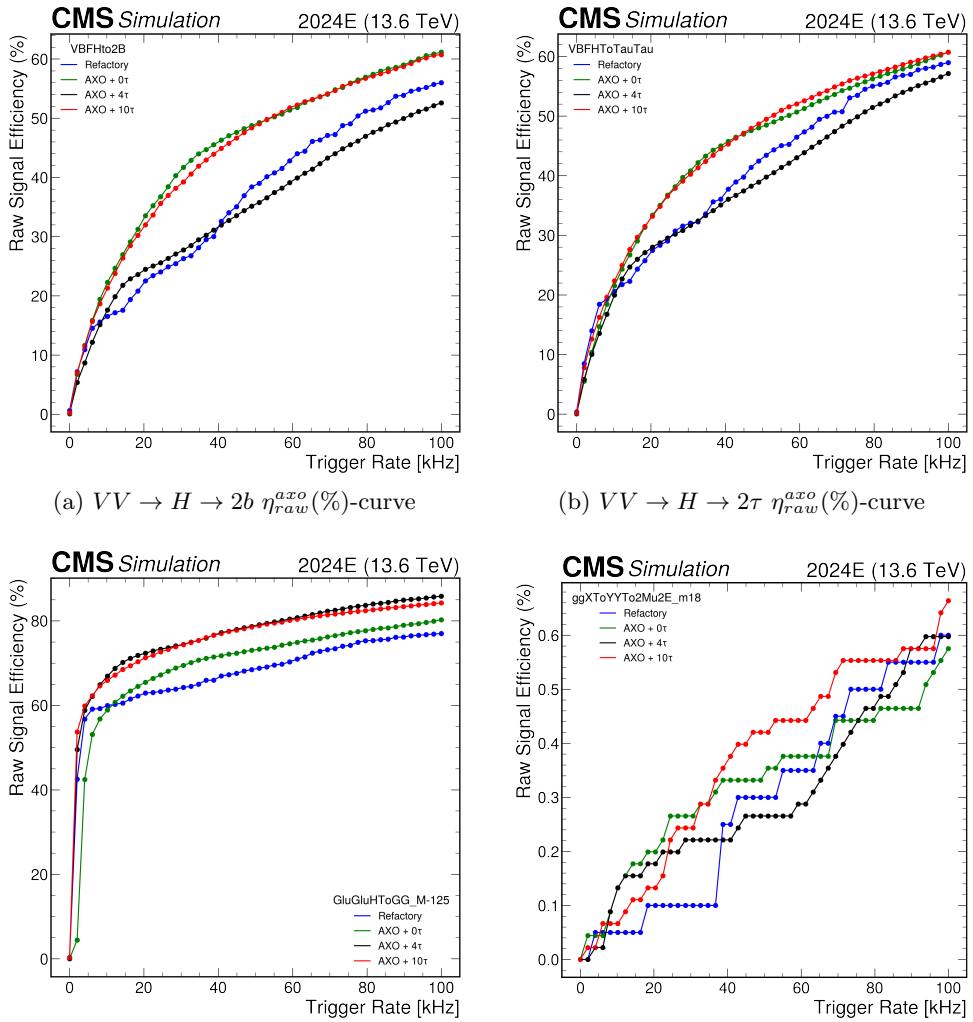
To evaluate the sensitivity of the extended AXOL1TL model to the signals introduced in Section 6.2, we will use the strategy described in Subsection 5.2. In particular, following Sun (2023), we will employ the approximation $D_{KL}[N(\mu, \log\sigma^2)||N(0,1)] \sim \mu^2$, calculated over the latent representations of the signals, as an anomaly score. This score will be compared against a predefined threshold based on the latent representations of background samples. It has been shown that implementing the full definition of D_{KL} is computationally expensive in FPGA firmware and does not provide significant additional information.

The scores presented in Figure 9 are normalized relative to the total number of events in the signal, specifically, for **raw** – **axo**:

```
    latent_axo_qk = self.model.predict(self.input_quantizer(signal_data),...)
    y_axo_qk = np.sum(latent_axo_qk**2, axis=1)
    nsamples = y_axo_qk.shape[0]
    axo_triggered = np.where(y_axo_qk > self.threshold[thres])[0].tolist()
    raw_rate = len(axo_triggered)/nsamples
```

Listing 16: `axo_raw` metric formula (η^{axo}_{raw}) on-code definition.

Which results in:



(a) $VV \to H \to 2b$ $\eta^{axo}_{raw}(\%)$-curve

(b) $VV \to H \to 2\tau$ $\eta^{axo}_{raw}(\%)$-curve

(c) $gg \to H \to \gamma\gamma$ $\eta^{axo}_{raw}(\%)$-curve

(d) $ggX \to YY \to 2\mu 2e$ $\eta^{axo}_{raw}(\%)$-curve

Figure 26: The dependence of raw efficiency on the trigger rate for **VBF**, **Higgs**, and **X-Y** signals. As the trigger rate increases, the addition of tau leptons enhances the signal efficiency in most cases, given a fixed rate.

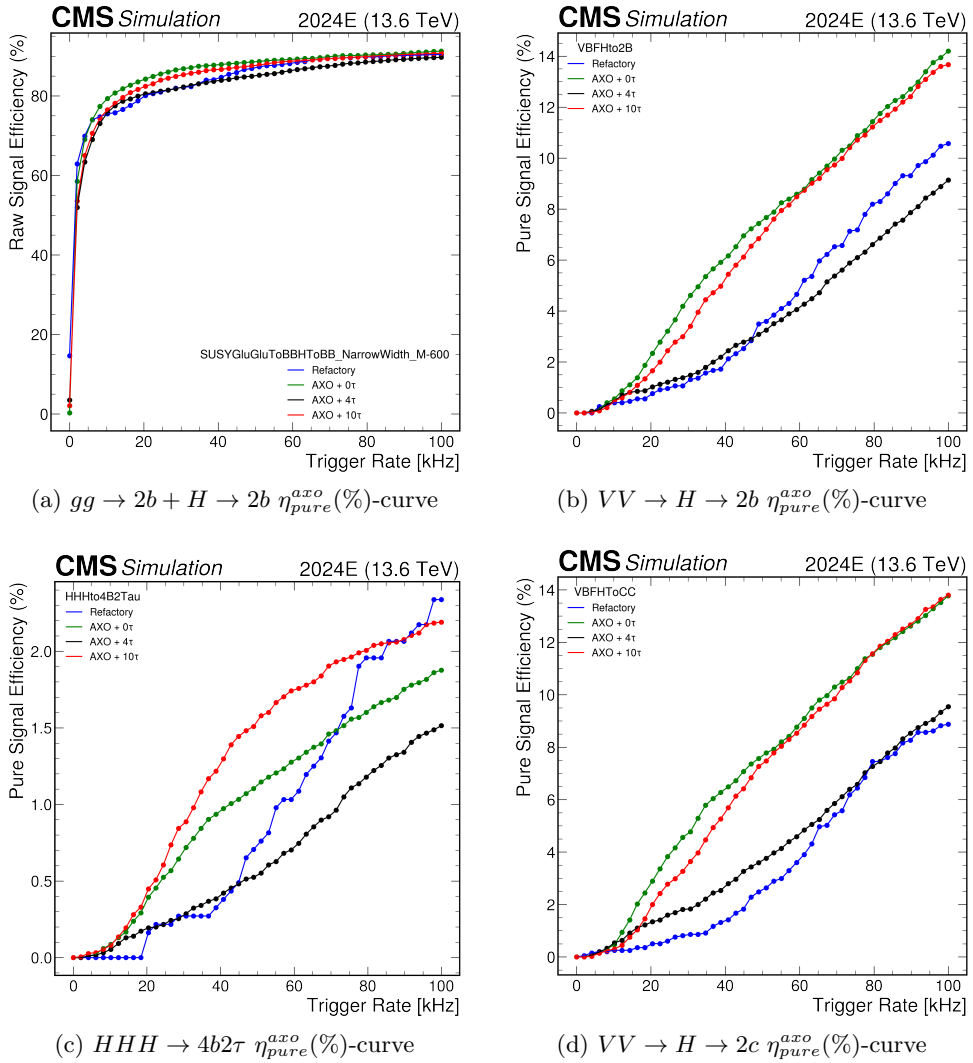
In Figure 26, the performance of the most recent version of AXOL1TL is compared to its tau-extended version. It is observed that, in general, for all signal types, the inclusion of tau leptons induces a higher rate of potentially anomalous events in the overall L1 trigger system, thereby enhancing its performance.

Analogously, for **pure** − **axo**:

```
l1_triggered = np.where(signal_L1)[0].tolist()
axo_pure = list(set(axo_triggered)-set(l1_triggered))
axo_pure_rate = len(axo_pure)/nsamples
```

Listing 17: `axo_pure` metric formula (η^{axo}_{pure}) on-code definition.

Where `L1_bits` is a binary (`bool`) array representing signal-like events triggered by the standard L1 menu. Consequently, **axo** − **pure** is defined as the ratio of signal-like events triggered by AXOL1TL but not by the standard L1 menu. It was found that:



(a) $gg \to 2b + H \to 2b$ $\eta^{axo}_{pure}(\%)$-curve

(b) $VV \to H \to 2b$ $\eta^{axo}_{pure}(\%)$-curve

(c) $HHH \to 4b2\tau$ $\eta^{axo}_{pure}(\%)$-curve

(d) $VV \to H \to 2c$ $\eta^{axo}_{pure}(\%)$-curve

Figure 27: The dependence of pure efficiency on the trigger rate for **VBF**, **Higgs**, and **SUSY** signals.

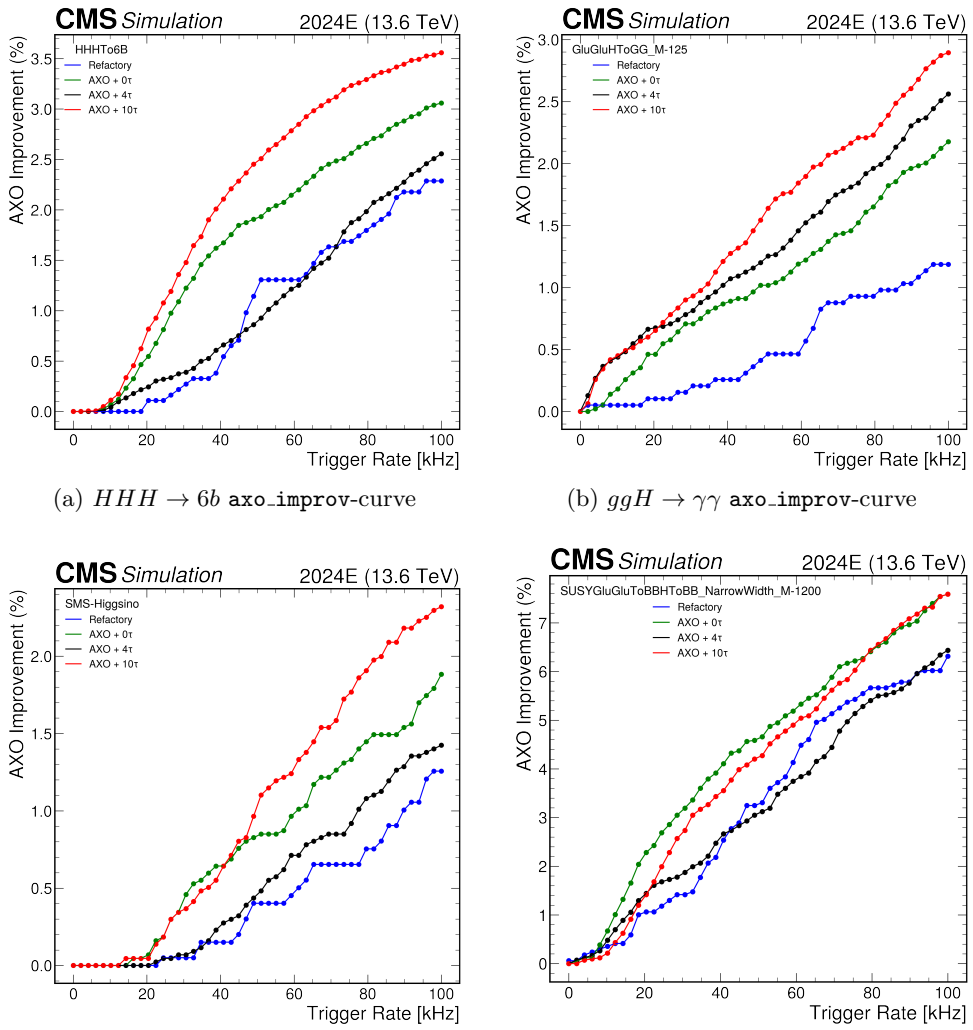
This is consistent with **raw** − **axo**. It can be observed that extending the model to accommodate 10 τ-leptons results in efficiency metrics comparable to those obtained when tau leptons are not included. In contrast, incorporating 4 tau leptons may prove to be counterproductive.

The improvement of AXOL1TL over the L1 menu, **AXO_improvement**, as also defined in Figure 9, is given by:

```
axo_improv = list(set(axo_triggered).union(set(l1_triggered)))
axo_improv_rate = (len(axo_improv) - len(l1_triggered))/nsamples
```

Listing 18: `axo_improv` metric formula on-code definition.

Which is an indicator of the additional signal events triggered by AXOL1TL. This resulted, in our comparative analysis, in:



(a) $HHH \to 6b$ `axo_improv`-curve

(b) $ggH \to \gamma\gamma$ `axo_improv`-curve

(c) `SMS-Higgsino` `axo_improv`-curve

(d) $gg \to 2b + H \to 2b$ `axo_improv`-curve

Figure 28: The dependence of the `axo_improv` efficiency on the trigger rate for **VBF**, **Higgs**, and **SUSY** signals.

The additional (efficiency) gain achievable by the new model is evident in 28. The maximum improvement of the extended models, compared to the L1 menu, over the latest baseline can be seen in Figure 29. It is observed that for high trigger rates, the inclusion of taus significantly increases the efficiency of triggering benchmark signals events. Table 1 presents a comparative analysis of the improvement in event detection that AXOL1TL (extended and baseline) contributes to the L1 menu, for a varied list of signals of each type from our BSM sample cocktail.

AXOL1TL	Refactory	0 τ	4 τ	10 τ
GluGluHToGG_M-125	0.103	0.461	0.675	0.654
GluGluHToTauTau	0.000	0.306	0.166	0.286
GluGlutoHHto2B2WtoLNu2Q	0.153	1.233	0.452	0.966
HHHTo6B	0.109	0.547	0.244	0.818
HHHto4B2Tau	0.163	0.395	0.195	0.449
HTo2LongLivedTo4b[‡1]	0.110	0.380	0.228	0.532
SMS-Higgsino	0.000	0.069	0.000	0.046
SUSYGluGluToBBHToBB[‡2]	1.063	2.281	1.441	1.417
VBFHToCC	0.507	2.891	1.340	1.998
VBFHToInvisible	0.202	0.703	0.304	0.445

Table 1: Relative improvement with respect to the standard L1 menu, evaluated for AXOL1TL τ-extended models in the pure efficiency metric for selected signals. The trigger rate is 22.449 kHz. Additionally, the efficiency achieved by the Refactory version of AXOL1TL is presented. [‡1] _MH-1000_MFF-450_CTau 100000mm. [‡2] _NarrowWidth_M-120.

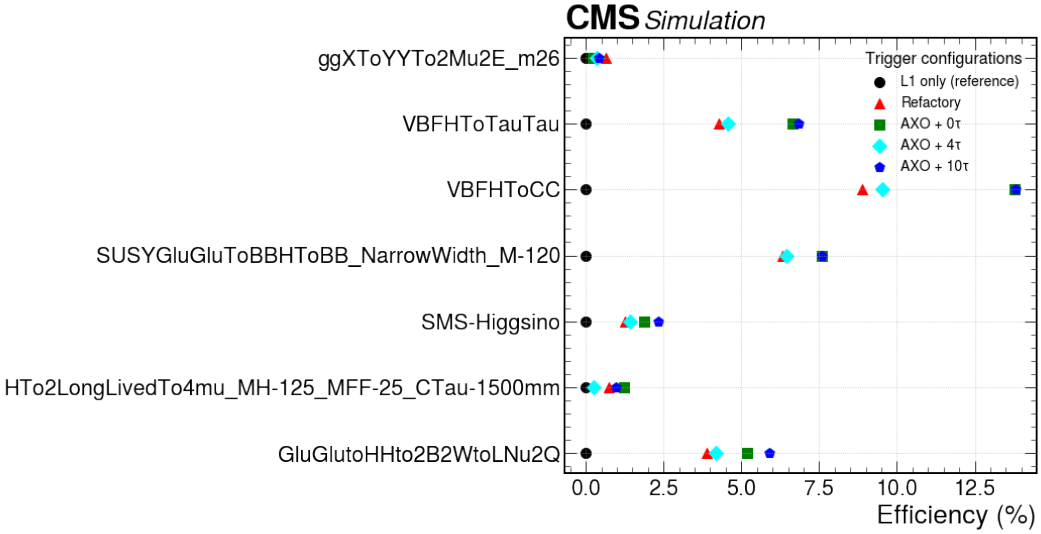
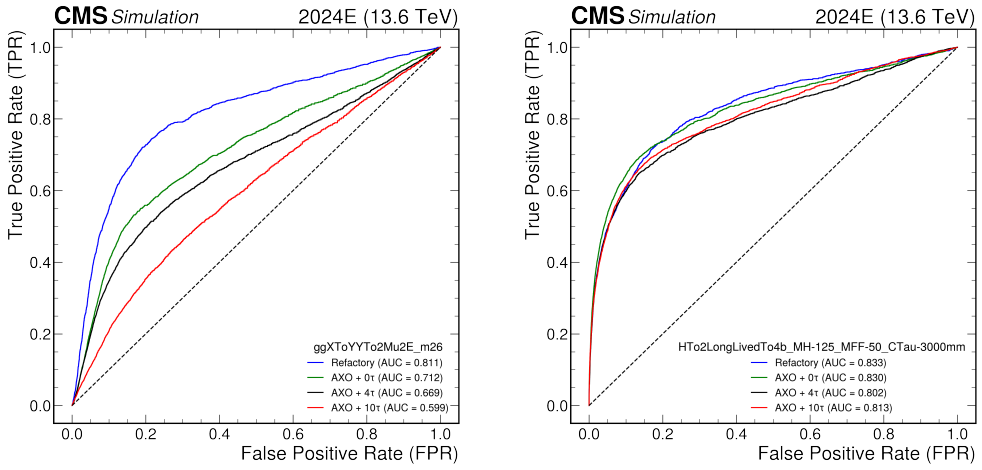


Figure 29: As presented in Figure 8, the model extended to 10 tau leptons can significantly enhance AXOL1TL performance across a considerable number of signals. Using only the new dataset, without the addition of tau leptons via constituents (Figure 7), can produce a comparable effect. The plot illustrates the `axo_improv_rate` relative to the `l1_rate` for each signal type. The difference between the colored markers and the one corresponding to the L1 menu (black circle) represents `axo_pure_rate`.

11.2 Anomaly clasification

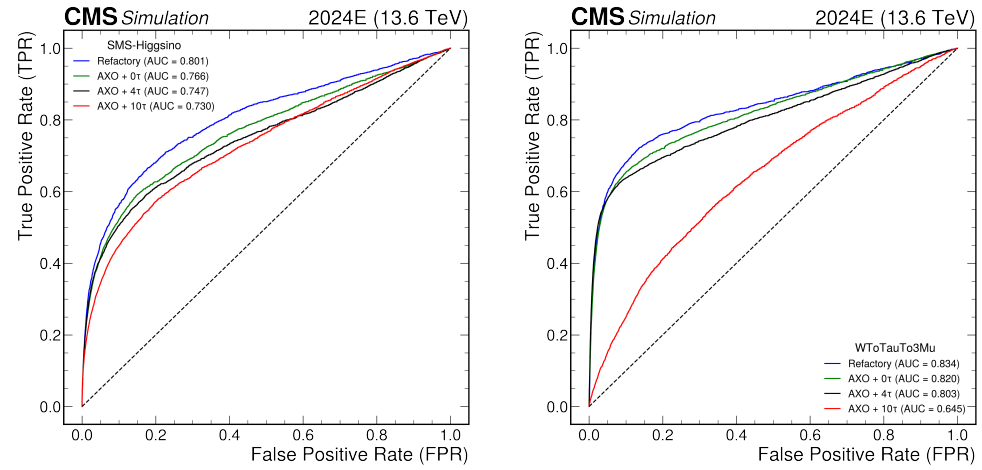
To determine the classification performance of our AXOL1TL model variants (latest baseline and with 0-4-10 τ's) in distinguishing *background* from *signal*, we can construct ROC curves from the background and signal scores obtained during the efficiency computation over signals in `axo/metric/scores.py`. We will calculate the False Positive Rate (FPR) and True Positive Rate (TPR) based on the principle that background samples are expected to have lower anomaly scores than signal samples.



(a) $ggX \to YY \to 2\mu 2e$ ROC curve (b) $H \to 2LL \to 4b$ ROC curve

Figure 30: A classification analysis of anomalies based on signal and background scores is presented. The extended AXOL1TL model is shown to be comparable for specific signals, such as `HTo2LongLivedTo4b_MH-125_MFF-50_CTau-3000mm` (Subfigure 30a). However, it has also been observed that the inclusion of taus (and, more broadly, the new dataset) can negatively impact the model's classification performance, as demonstrated by the signal `ggXToYYTo2Mu2E_m26` (Subfigure 30b).

Two pointers (`ptr`) will be used to traverse the background and signal scores. If the signal score is greater than the background score, a true positive (`TP`) will be identified, and the background pointer will move to the next score in its sorted array (`ptr_bkg`). If the background score is greater than the signal score, a false positive (`FP`) will be recorded, and the signal pointer (`ptr_sig`) will advance to the next score in its sorted array. If the signal and background scores are equal, both pointers will advance by one position. The ratio of `ptr_bkg`/`bkg_n` and `ptr_sig`/`sig_n` at each iteration will determine the TP and FP values for our ROC curve (`bkg_n` and `sig_n` are the length of the background and signal scores arrays).

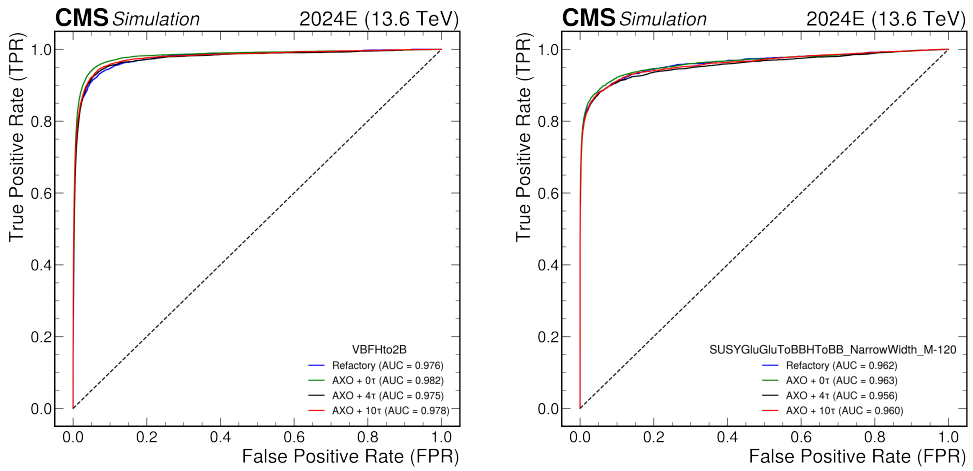


(a) `SMS-Higgsino` ROC curve (b) $W \to \tau \to 3\mu$ ROC curve

Figure 31: The performance analysis for detecting potential anomalies in signals `SMS-Higgsino` and `WToTauTo3Mu` (Figures [31a,31b]) demonstrates a sequential decline in the model's robustness with the inclusion of taus for classification. Specifically, it is observed that as τ leptons are incorporated as constituents in the model, the AUC value decreases progressively.

Figures [30,31] demonstrate that the extended model exhibits inferior classification performance com-

pared to the baseline refactory (latest version of AXOL1TL) under identical hyperparameters. This indicates that hyperparameter optimization is necessary to enhance the performance of the tau-extended models.



(a) $VV \to H \to 2b$ ROC curve

(b) $gg \to bb + H \to bb$ ROC curve

Figure 32: Classification performance for **SUSY** and **VBF** signals.

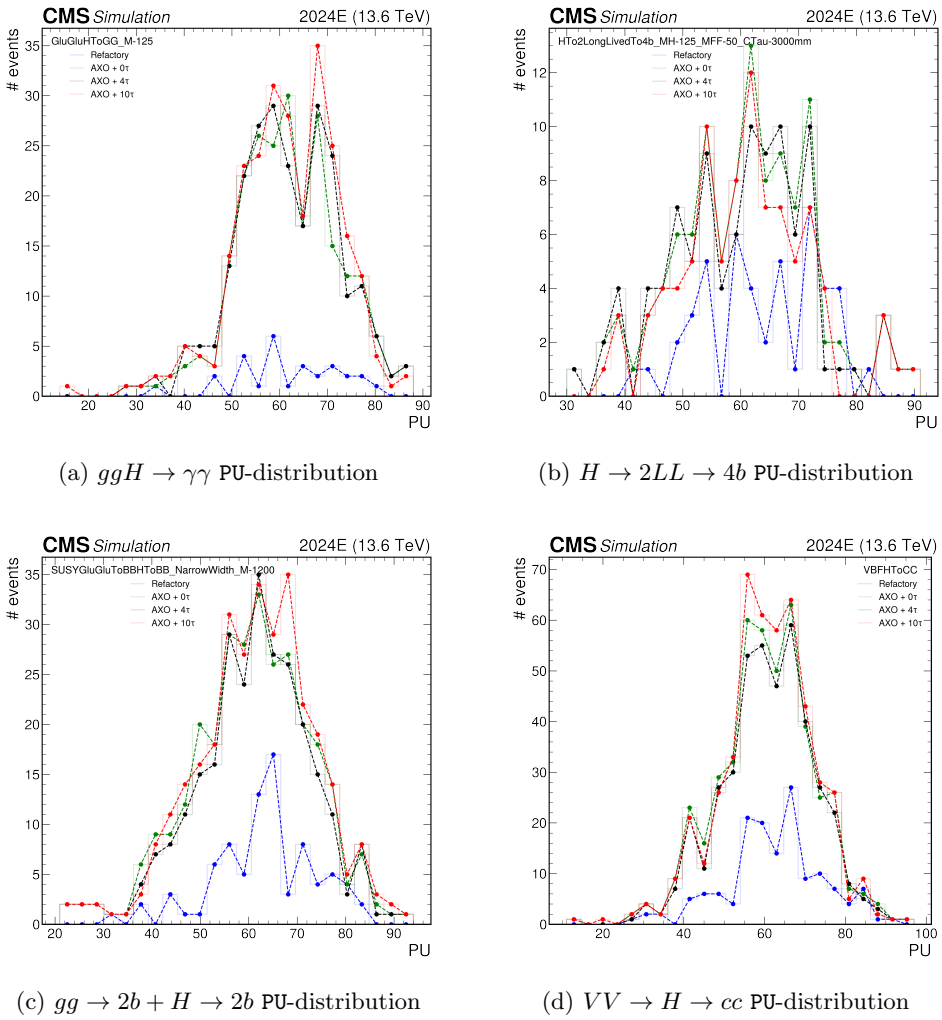
Table 2 shows the AD accuracy on L1T-type events for each selected signal sample, and for each τ-extended model.

AXOL1TL	**Refactory**	**0 τ**	**4 τ**	**10 τ**
GluGluHToGG_M-125	0.987	0.990	0.991	0.990
GluGluHToTauTau	0.924	0.920	0.905	0.890
GluGlutoHHto2B2WtoLNu2Q	0.996	0.996	0.996	0.995
HHHTo6B	0.999	0.999	0.999	0.999
HHHto4B2Tau	0.999	0.999	0.999	0.999
HTo2LongLivedTo4b^{‡_1}	0.946	0.947	0.938	0.942
SMS-Higgsino	0.801	0.766	0.747	0.730
SUSYGluGluToBBHToBB^{‡_2}	0.990	0.991	0.989	0.989
VBFHToCC	0.984	0.986	0.981	0.983
VBFHToInvisible	0.918	0.934	0.917	0.924
VBFHToTauTau	0.979	0.984	0.978	0.976
VBFHto2B	0.976	0.982	0.975	0.978
WToTauTo3Mu	0.834	0.820	0.803	0.645
ggXToJpsiJpsiTo2Mu2E_m7	0.548	0.553	0.513	0.569
haa4b-ma15-noPU	0.867	0.893	0.914	0.870

Table 2: Comparison of anomaly classification performance based on the AUC metric for the τ-extended AXOL1TL models. Additionally, the AUC values obtained for the baseline AXO model (Refactory) are presented. ^{‡_1}_MH-1000_MFF-450_CTau-10000mm.^{‡_1}_NarrowWidth_M-350.

11.3 Pile-Up dependency

To study the sensitivity of the AXOL1TL methods to pile-up, we can determine the number of triggered events for each pile-up value in the inference on our reference signal cocktail. That is, given `axo_triggered` in `axo/metric/axo_score.py`, for each `signal` in `signal_names` (see Subsection 5.2), we can obtain the PU values per event and per BSM sample from `signal_PU = data_file["Signal_data"][signal]["PU"][:]`. We collect the PU values triggered by AXOL1TL for the highest threshold. Specifically, we are interested in studying the distribution over a subset of `axo_triggered`, `axo_pure`, `signal_PU[axo_pure]`. The height of each bin will be proportional to the pure efficiency of the signal for the PU value corresponding to that bin.



(a) $ggH \to \gamma\gamma$ `PU`-distribution

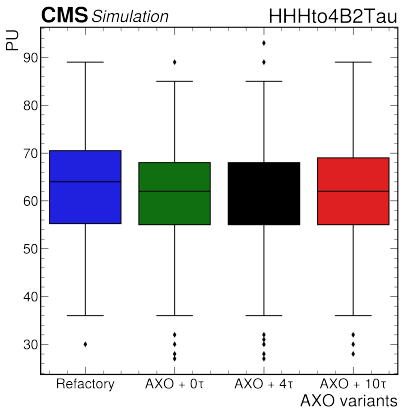
(b) $H \to 2LL \to 4b$ `PU`-distribution

(c) $gg \to 2b + H \to 2b$ `PU`-distribution

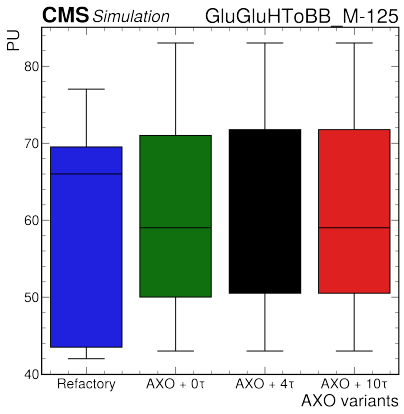
(d) $VV \to H \to cc$ `PU`-distribution

Figure 33: Pile-up distribution of events triggered by AXOL1TL during inference on signals of types `GluGluHToGG`, `HTo2LongLivedTo4b`, `SUSYGluGluToBBHToBB`, and `VBFHToCC` at a trigger rate of 100 kHz.

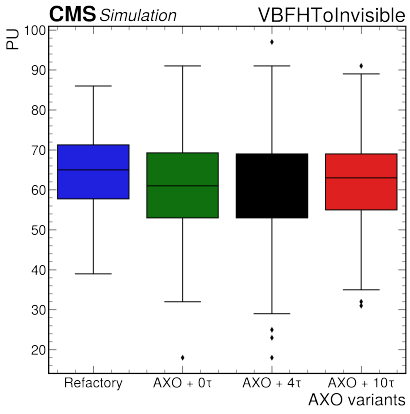
Figure 33 shows the response to pile-up for **LL**, **VBF**, **SUSY** and **Higgs** signals. The peaks of each bin were plotted to highlight the trend of the distribution, and the bin partitioning is fair for comparison. It can be observed that the extended models are more robust to pile-up in a central range of values. The robustness of the four variants of the AXOL1TL model is evaluated against the pile-up of signal events, considering the samples selected in Figure 34. In this context, robustness to pile-up is defined as the model's ability to maintain consistent performance, specifically its capability to detect signals of interest without being affected by additional events generated by pile-up.



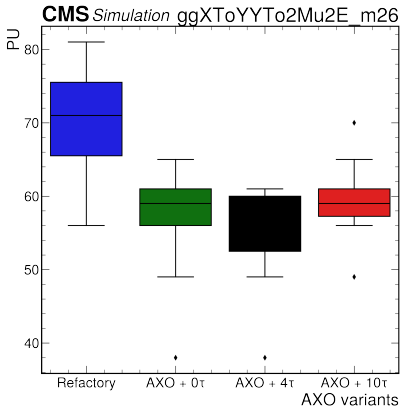
(a) $HHH \to 4b + 2\tau$ `PU`-boxplot



(b) $ggH \to bb$ `PU`-curve



(c) `VBFToInvisible` `PU`-boxplot



(d) $ggX \to YY \to 2\mu 2e$ `PU`-boxplot

Figure 34: Boxplot distribution of the pile-up values for events triggered by AXOL1TL during inference on selected reference signals at 100kHz of trigger rate.

The results indicate a tendency toward reduced dispersion in signals `HHHto4B2Tau`, `GluGluHToBB_M`, `ggXToYYTo2Mu2E_m26` for the τ-extended AXOL1TL models, as evidenced by narrower box widths, as shown in Subfigures [34b,34d]. Additionally, the medians remain consistent across both the extended models and the model trained with the new dataset. The boxplot-whiskers of the newly trained models are shorter in 34d, suggesting lower variability in response to PU. However, it is observed that τ-extended models tend to exhibit a higher number of outliers, which may indicate increased sensitivity to extreme events.

11.4 Hyperparameter Tuning

We will modify the architecture of the AXOL1TL network with the aim of improving efficiency and performance in classification, particularly when τ leptons are added as new inputs to the model. As τ's are incorporated and the input space becomes more complex, the dimensionality of the latent space will be increased to mitigate potential information loss due to the inherent complexity. In this context, we propose a first scenario, which we will refer to as **HPT1**.

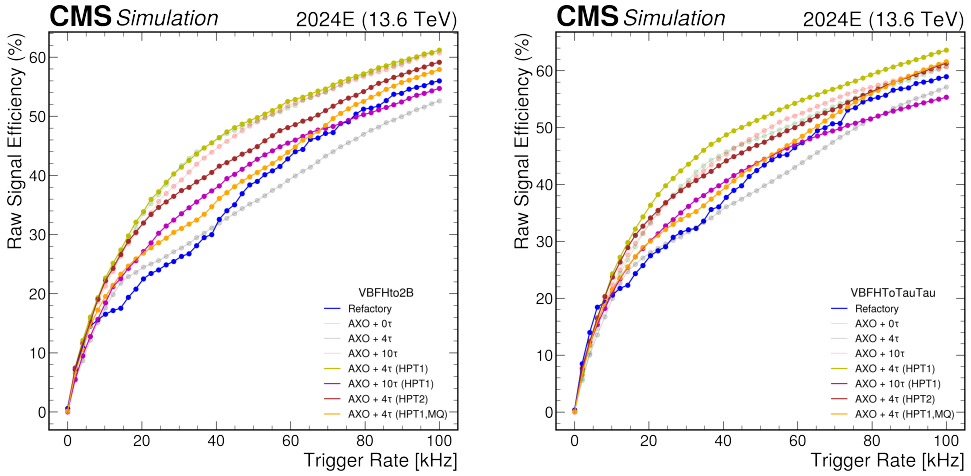
```
    model_config["encoder_config"] = {"nodes": [46, 23, 12]}
    model_config["latent_dim"] = 10
    model_config["decoder_config"] = {"nodes": [16, 32, 64, 96, 128, 69]}
    model_config["features"] = 69
```

Listing 19: First scenario for redefining the AXOL1TL architecture **HPT1**. The last node of the decoder and `model_config("features")` should match the total number of momenta per L1 reconstructed object (69 for 4 taus added, and 87 for 10 taus).

As indicated in Figure 7, the asymmetry of the decoder relative to the encoder is maintained. This model will be trained using a batch size of 10000, with 50 epochs, for the cases where 4 and 10 additional taus are included.

```
model_config["encoder_config"] = {"nodes": [52, 36, 24]}
model_config["latent_dim"] = 16
model_config["decoder_config"] = {"nodes": [32, 64, 128, 69]}
model_config["features"] = 69
```

Listing 20: Second scenario for redefining the AXOL1TL architecture (**HPT2**). The sequence of nodes/neurons in the encoder is smoother compared to the **HPT1** case. The dimensionality of the latent space is larger.



(a) $VV \to H \to 2b$ $\eta^{axo}_{raw}(\%)$-curve

(b) $VV \to H \to \tau\tau$ $\eta^{axo}_{raw}(\%)$-curve

(c) $ggH \to \gamma\gamma$ $\eta^{axo}_{raw}(\%)$-curve

(d) $gg \to 2b + H \to 2b$ $\eta^{axo}_{raw}(\%)$-curve

Figure 35: The dependence of raw efficiency on the trigger rate for signals `VBFHto2B`, `VBFHToTauTau`, `GluGluHToGG` and `SUSYGluGluToBBHToBB` after inference through AXO models with tuned hyperparameters. `PT1` and `PT2` correspond to architectures modified relative to the baseline. MQ indicates that a muon quality filter was applied during the data preprocessing stage prior to training (`qualcut = 2`).

We propose a second case in which the dimensionality of the latent space is increased (to 16), while maintaining the asymmetry of the VAE (Listing 20). The next step will be to evaluate this model using a batch size of 14000, for 50 epochs, in the case where 4 taus are included (**PT2**). Our third scenario involves applying the first adjustment (listing 19) on an extended space of 4 taus, filtered with a muon quality cut (`qualcut`) of 2, and using a batch size of 16000 (**PT1,MQ**).

The raw efficiency results of AXO are consistently improved for the **PT1** adjustment (4 taus) (See 35. Depending on the signal, the **PT1** (10 taus) adjustment can significantly improve or worsen the event trigger rate of AXOL1TL (See Subfigures [35d,35c]). The improvement in **PT2** (4 taus) is significant compared to `AXO + 4 taus` (defined with the baseline architecture), but its improvement does not seem as stable when compared to the **PT1** adjustment for 4 taus. Table 3 provides a comparative analysis of the net impact of the tuned AXOL1TL models on the L1 menu and the AXO's refactory trigger for event selection. It displays the inference results for signals chosen based on their type.

AXOL1TL	Refactory	4τ PT1	10τ PT1	4τ PT2	4τ PT1-MQ
GluGluHToGG_M-125	0.103	0.707	0.922	0.600	0.579
GluGluHToTauTau	0.000	0.281	0.261	0.261	0.190
GluGlutoHHto2B2WtoLNu2Q	0.153	1.007	0.432	1.089	0.555
HHHTo6B	0.109	0.552	0.249	0.607	0.206
HTo2LongLivedTo4b^{‡_1}	0.303	0.974	0.278	0.834	0.695
SMS-Higgsino	0.000	0.092	0.092	0.046	0.092
SUSYGluGluToBBHToBB^{‡_2}	1.307	4.325	2.020	4.610	3.468
VBFHto2B	0.759	2.082	1.126	1.742	1.147
haa4b-ma15-noPU	0.301	0.200	0.601	0.301	0.301

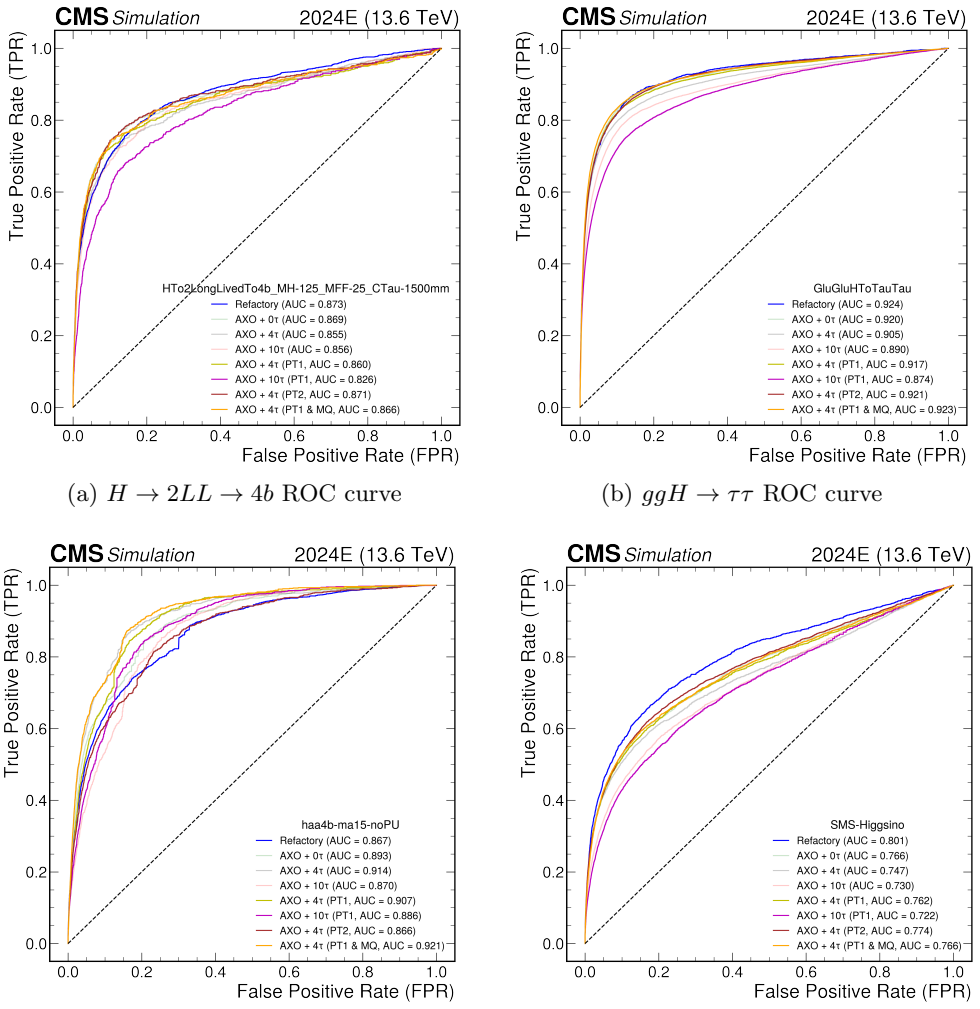
Table 3: Relative improvement with respect to the standard L1 trigger, evaluated for AXOL1TL tuned τ- extended models in the pure efficiency metric for selected signals. The trigger rate is 22.449 kHz. Additionally, the efficiency achieved by the Refactory version of AXOL1TL is presented. ^{†_1}_MH-125_MFF-25_CTau-1500mm.^{†_2}NarrowWidth_M-350.

AXOL1TL	Refactory	4τ PT1	10τ PT1	4τ PT2	4τ PT1-MQ
GluGluHToGG_M-125	0.987	0.992	0.990	0.993	0.992
GluGluHToTauTau	0.924	0.917	0.874	0.921	0.923
GluGlutoHHto2B2WtoLNu2Q	0.995	0.997	0.992	0.997	0.997
HHHto4B2Tau	0.999	0.999	0.998	0.999	0.999
HTo2LongLivedTo4mu^{‡_1}	0.936	0.931	0.776	0.935	0.925
SUSYGluGluToBBHToBB^{‡_2}	0.988	0.991	0.987	0.992	0.992
VBFHToCC	0.984	0.986	0.979	0.986	0.986
VBFHToInvisible	0.918	0.933	0.913	0.930	0.935
VBFHToTauTau	0.979	0.984	0.970	0.984	0.984
VBFHto2B	0.976	0.981	0.972	0.982	0.981
WToTauTo3Mu	0.834	0.808	0.676	0.812	0.809
ggXToJpsiJpsiTo2Mu2E_m7	0.548	0.524	0.541	0.518	0.535
ggXToYYTo2Mu2E_m26	0.811	0.688	0.602	0.730	0.696
haa4b-ma15-noPU	0.867	0.907	0.886	0.866	0.921

Table 4: Comparison of anomaly classification performance based on the AUC metric for the tuned τ-extended AXOL1TL models. Additionally, the AUC values obtained for the baseline AXO model (Refactory) are presented.^{‡_1} _MH-125_MFF-25_CTau-1500mm.^{‡_2} _NarrowWidth_M-600.

Regarding classification performance, it is observed that the adjusted AXO models can improve anomaly detection while maintaining a high event firing rate compared to the baseline tau-extended models of AXOL1TL [35,36]. In Subfigure 36c, it can be seen that for signal $H \to aa \to 4b$, the AUC

has been significantly magnified. Table 4 shows the AD accuracy for each selected signal sample, and for each tuned τ-extended AXO model.



(a) $H \to 2LL \to 4b$ ROC curve

(b) $ggH \to \tau\tau$ ROC curve

(c) $H \to aa \to 4b$ ROC curve

(d) `SMS-Higgsino` ROC curve

Figure 36: Improvement in the classification performance of the extended AXOL1TL models with tuned hyperparameters compared to the AXO models with the baseline architecture. The AUC values corresponding to signals `HTo2LongLivedTo4b`, `GluGluHToTauTau`, `haa4b-ma15` and `SMS-Higgsino` are presented.

12 Conclusions

To determine which variant of AXOL1TL constitutes the most suitable model for anomaly detection, it is essential to analyze classification results from a global perspective that encompasses both a dominance metric based on the AUC obtained per signal (considering all signals and models) and a relative perspective. The latter involves studying the average differences (deviations with respect to the values of other columns) for each column in Tables [2,4], allowing an assessment of the relative magnitude of one column compared to the others. Additionally, the average ratio between each column and the others will be analyzed as a metric for the average proportional relationship between their values. These analyses must be integrated with an evaluation of the trigger capacity of each model, as the objective is to select a model capable of detecting the largest possible number of potentially anomalous events while maintaining high standards of anomaly classification performance.

```
def compare_columns(df):
    columns = [col for col in df.columns if col != 'Signal']
```

```
    avg_diffs = {}
    avg_ratios = {}
    for col in columns:
        total_diff = 0
        total_ratio = 0
        total_comparisons = 0
        for other_col in columns:
            if col != other_col:
                for _, row in df.iterrows():
                    diff = row[col] - row[other_col]
                    ratio = row[col] / row[other_col] if row[other_col] != 0
                        else 0
                    total_diff += diff
                    total_ratio += ratio
                    total_comparisons += 1
        avg_diffs[col] = total_diff / total_comparisons
        avg_ratios[col] = total_ratio / total_comparisons
    return avg_diffs, avg_ratios
```

Listing 21: Definition of our metric comparison system. The metrics obtained by all proposed AXO-type models (organized in a column of a `DataFrame`) are compared across all signals (represented by the rows of the `DataFrame`).

In Listing 21, the parameter `avg_diffs` represents the average differences of each column in table Y relative to the other columns. A positive value of `avg_diffs` indicates that the model associated with that column exhibits, on average, superior classification metrics. Meanwhile, the parameter `avg_ratios` stores the average ratio of the values of each column relative to the others. A value of `avg_ratios` greater than 1 for a given column implies that the corresponding AXOL1TL variant displays, on average, higher values than the other models. In contrast, a value less than 1 indicates that the variant has lower average values.

AXOL1TL	AUC avg_diff	AUC avg_ratio
Refactory	0.0191	1.0279
0τ	0.0116	1.0161
4τ	-0.0005	1.0000
10τ	-0.0251	0.9709
4τ PT1	0.0063	1.0086
10τ PT1	-0.0299	0.9648
4τ PT2	0.0096	1.0133
4τ PT1-MQ	0.0088	1.0119

Table 5: Comparative table of `avg_diff` and `avg_ratio` for the AUC classification metric across all proposed AXO models (tuned and derived from the baseline) considering all signals.

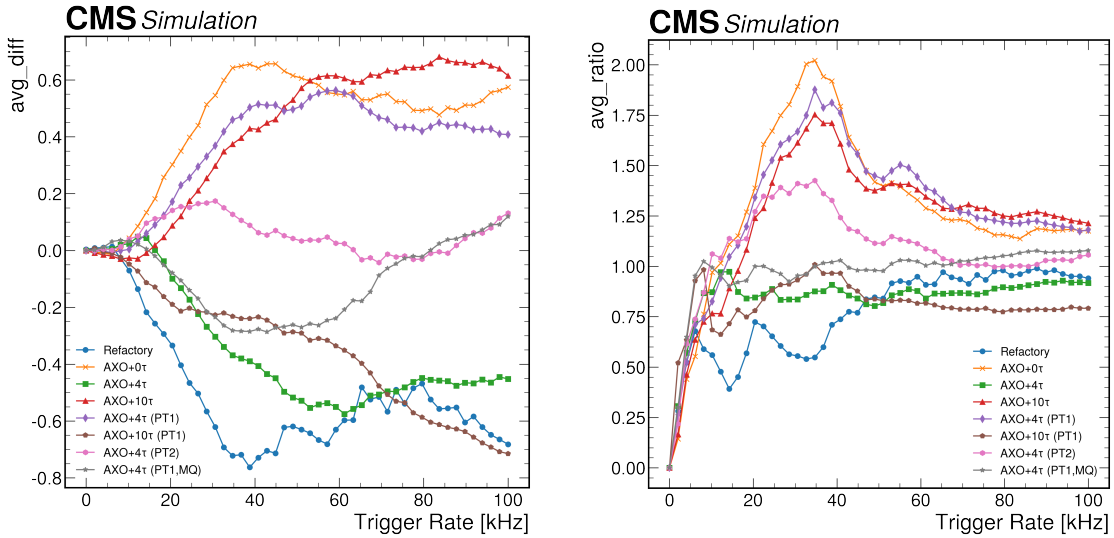
From Table 5, the AXO-type model **Refactory** has an `avg_diff` value of 0.0191, the **0τ**-variant 0.0116, **4τ (PT1)** 0.0063, **4τ (PT2)** 0.0096, and **4τ (PT1-MQ)** 0.0088, all of which represent positive average differences, indicating slightly higher classification accuracy compared to the other AXO variants. In contrast, the **4τ** model shows an almost negligible average difference of -0.0005, remaining well-balanced relative to the other models, while **10τ** and **10τ** models **(PT1)** exhibit the most negative average differences, -0.0251 and -0.0299 respectively, suggesting that their AUC values are, on average, lower than those of the other models. The **Refactory**, **0τ**, **4τ (PT1)**, **4τ (PT2)**, and **4τ (PT1-MQ)** models have average ratios (`avg_ratio`) greater than 1, indicating that, on average, their values are between 0.86% and 2.79% higher than those of the other columns. In contrast, **10τ** and **10τ (PT1)** models have average ratios less than 1, with values that are 2.91% and 3.52% lower, respectively, than those of the other columns. The **4τ** AXO model has an average ratio of 1.0000, meaning its values are equivalent to those of the other columns. It is found that the most dominant models, in terms of their classification power, are **Refactory**, **0τ**, **4τ (PT2)**, **4τ (PT1-MQ)**, and **4τ (PT1)**

Table 6 facilitates the reconciliation between the ability to accurately classify a potentially anomalous event (represented by the AUC metric) and the need to capture more events of this type, potentially

AXOL1TL	n_{pure}^{axo} avg_diff	n_{pure}^{axo} avg_ratio
Refactory	-0.6304	0.9492
0τ	0.5515	1.3297
4τ	-0.5504	0.8781
10τ	0.6146	1.3811
4τ PT1	0.5635	1.4444
10τ PT1	-0.3357	0.8158
4τ PT2	0.0257	1.1130
4τ PT1-MQ	-0.2388	1.0240

Table 6: Comparative table of `avg_diff` and `avg_ratio` for the purity metric (n_{pure}^{axo}) across all proposed AXO models (tuned and derived from the baseline) considering all signals. A working trigger rate of 59.184 kHz was used.

associated with new physics signatures (indicated by the n_{pure}^{axo} metric). Models **Refactory**, **4τ**, **10τ (PT1)**, and **4τ (PT1-MQ)** are not of interest due to their non-dominant nature, as evidenced by their negative `avg_diff` values and `avg_ratio` below one. Our focus is on models **0τ**, **10τ**, **4τ (PT1)** and **4τ (PT2)**. Models **4τ (PT1)** (`avg_diff` = 0.5635 and `avg_ratio` =1.4444), followed by **10τ** (`avg_diff` = 0.6146 and `avg_ratio` = 1.3811), are the most dominant. Along with the results obtained in Table 5, it is confirmed that the most dominant model, capable of reconciling robustness between classification and triggering, is **4τ (PT1)**, despite its slight difference in classification accuracy compared to **Refactory**, **0τ**, **4τ (PT2)** and **4τ (PT1-MQ)**.



(a) n_{pure}^{axo} `avg_diff` evolution with trigger rate.

(b) n_{pure}^{axo} `avg_ratio` evolution with trigger rate.

Figure 37: Evolution of comparison metrics based on differences (`avg_diff`) and ratios (`avg_ratio`) for the anomaly metric n_{pure}^{axo} across a spectrum of trigger frequencies ranging from 0 to 100 kHz. The distinction of the best-performing trigger model is unclear within the first 20 kHz, suggesting the need to optimize the AD metrics at lower trigger rates. All proposed AXOL1TL variants are compared, including those with hyperparameter tuning and those defined using a muon quality filter.

Figure 37 summarizes the information presented in Table 6 by evaluating the improvement in purity of each AXOL1TL model relative to the others AXO-type models, considering a full spectrum of trigger rates ranging from 0 to 100 kHz and across all signals. Subfigure 37a analyzes the overall behavior in the average deviations of pure efficiency for each model, particularly in comparison to other models, over all signals and trigger rates. Meanwhile, Subfigure 37b represents the trend in the average magnitude of the

(pure) efficiency of one AXO-type model compared to the efficiencies achieved by all other AXO models, thus serving as an indicator of the relative advantage of a given model over another.

The distinction in the sign of the trends, particularly in Subfigure 37a, is of particular interest as it serves as a direct indicator of the relative (efficiency) loss of performance of one AXO variant compared to the rest of the models when detecting events potentially rich in new physics. The most prominent AXO-models, identified in both Subfigures [37a,37b], are $\mathbf{4}\tau$ **(PT1)**, $\mathbf{0}\tau$, and $\mathbf{10}\tau$.

Furthermore, `avg_diff` and `avg_ratio` in Table 5 are presented as indicators of the classification accuracy of one model over another, evaluated across all anomaly scores and, consequently, all trigger rates. The $\mathbf{4}\tau$ **(PT1)** and $\mathbf{0}\tau$ AXO-variants stand out as the best implementations of AXOL1TL, achieving an optimal balance between anomaly detection and efficient classification performance.

References

An, J., & Cho, S. (2015). Snu data mining center 2015-2 special lecture on ie variational autoencoder based anomaly detection using reconstruction probability. *Seoul National University*.

Bermot, E., Zoufal, C., Grossi, M., Schuhmacher, J., Tacchino, F., Vallecorsa, S., & Tavernelli, I. (2023). Quantum generative adversarial networks for anomaly detection in high energy physics. In *2023 ieee international conference on quantum computing and engineering (qce)*. DOI: 10.1109/QCE57702.2023.00045

Bhowmik, S. (2018). Triggering on electrons, photons, tau leptons, jets and energy sums with the cms level-1 trigger. Retrieved from https://pos.sissa.it/

Blance, A., & Spannowsky, M. (2021). Unsupervised event classification with graphs on classical and photonic quantum computers. *Journal of High Energy Physics*, *2021*(8). DOI: 10.1007/JHEP08(2021)170

Gemici, M., Hung, C.-C., Santoro, A., et al. (2017). Generative temporal models with memory. *arXiv preprint arXiv:1702.04649*. https://arxiv.org/abs/1702.04649.

Govorkova, E., Puljak, E., Aarrestad, T., et al. (2021). Autoencoders on fpgas for real-time, unsupervised new physics detection at 40 mhz at the large hadron collider. *Nature Machine Intelligence*, *4*(2), 110–116. DOI: 10.1038/s42256-022-00441-3

Govorkova, E., Puljak, E., Aarrestad, T., Pierini, M., Woźniak, K. A., & Ngadiuba, J. (2021). Lhc physics dataset for unsupervised new physics detection at 40 mhz. *arXiv*. Retrieved from http://arxiv.org/abs/2107.02157

Halilovic, A. (2017). *Anomaly detection for the cern large hadron collider injection magnets* (PhD Thesis). CERN.

Mikuni, V., & Canelli, F. (2020). Unsupervised clustering for collider physics. *Physical Review D*, *103*(9), 092007. Retrieved from https://doi.org/10.1103/PhysRevD.103.092007 DOI: 10.1103/PhysRevD.103.092007

Ngairangbam, V. S., Spannowsky, M., & Takeuchi, M. (2021). Anomaly detection in high-energy physics using a quantum autoencoder. *Phys. Rev. D*, *105*, 095004. Retrieved from https://doi.org/10.1103/PhysRevD.105.095004 DOI: 10.1103/PhysRevD.105.095004

Roche, S., Bayer, Q., Carlson, B., Ouligian, W., Serhiayenka, P., Stelzer, J., & Hong, T. M. (2024). Nanosecond anomaly detection with decision trees and real-time application to exotic higgs decays. *Nature Communications*, *15*. Retrieved from https://doi.org/10.1038/s41467-024-47704-8 DOI: 10.1038/s41467-024-47704-8

Schuhmacher, J., Boggia, L., Belis, V., Puljak, E., Grossi, M., Pierini, M., ... Tavernelli, I. (2023). Unravelling physics beyond the standard model with classical and quantum anomaly detection. *Machine Learning: Science and Technology*. Retrieved from https://doi.org/10.1088/2632-2153/ad07f7 DOI: 10.1088/2632-2153/ad07f7

Sun, C. (2023). Axol1tl: Real-time anomaly detection in the cms hardware trigger.

Woźniak, K. A., Belis, V., Puljak, E., Barkoutsos, P., Dissertori, G., Grossi, M., ... Vallecorsa, S. (2023). Quantum anomaly detection in the latent space of proton collision events at the lhc. Retrieved from http://arxiv.org/abs/2301.10780

Zipper, N. (2023). Testing a neural network for anomaly detection in the cms global trigger test crate during run 3. *arXiv*. Retrieved from http://arxiv.org/abs/2312.10009