Knowledge is Overrated: A zero-knowledge machine learning and cryptographic hashing-based framework for verifiable, low latency inference at the LHC

Pratik Jawahar
University of Manchester
`pratik.jawahar@cern.ch`

Caterina Doglioni
University of Manchester
`caterina.doglioni@cern.ch`

Maurizio Pierini
CERN
`maurizio.pierini@cern.ch`

Abstract

Low latency event-selection (trigger) algorithms are essential components of Large Hadron Collider (LHC) operation. Modern machine learning (ML) models have shown great offline performance as classifiers and could improve trigger performance, thereby improving downstream physics analyses. However, inference on such large models does not satisfy the 40MHz online latency constraint at the LHC. In this work, we propose `PHAZE` [1], a novel framework built on cryptographic techniques like hashing and zero-knowledge machine learning (zkML) to achieve low latency inference, via a certifiable, early-exit mechanism from an arbitrarily large baseline model. We lay the foundations for such a framework to achieve nanosecond-order latency and discuss its inherent advantages, such as built-in anomaly detection, within the scope of LHC triggers, as well as its potential to enable a dynamic low-level trigger in the future.

1 Introduction

Accelerating ML inference to satisfy the LHC's operational constraints is currently approached as an ad-hoc optimization problem driven by an accuracy-acceleration tradeoff. Algorithms like AXOL1TL and CICADA, recently tested [2] by the CMS experiment, combine this type of an approach with hardware-specific optimization techniques, using frameworks like `hls4ml` [3]. These algorithms successfully achieve $O(\text{ns})$ latencies for anomaly detection triggers [2]. However, we hypothesize that, to achieve $O(\text{ns})$ latencies for a larger set of ML-based trigger decisions, we need a novel approach to trigger design compatible with dynamic performance-improvement strategies.

We start with a short primer on polynomial fingerprinting, zkML, and early-exiting (EE) that form the building blocks of our framework.

1.1 Polynomial fingerprinting

Hash function: Any function that maps a mathematical object, like a vector of arbitrary size, to a fixed-size value [4, 5]. We restrict the definition to non-cryptographic hash functions [6] since we do not require security-related properties in the hashing algorithm for our framework. Say we have a family of hash functions $\mathcal{H}\{h : \mathcal{M} \to \mathcal{Y}\}$. We can then define a probabilistic hashing algorithm as a scheme, that for a given arbitrary-size object $M_i \in \mathcal{M}$, chooses a random hash function $h_i \in \mathcal{H}$ to evaluate the hash.

For a probabilistic hash to serve as a near-unique fingerprint for all objects in $\mathcal{M}$ we require that the collision probability [5] over a random h_i, for any 2 arbitrary values $x_1, x_x \in \mathcal{M}$, be bounded by a small, well-defined value ϵ, written as,

$$\Pr_{h_i \in \mathcal{H}}[h_i(x_1) = h_i(x_2)] \leq \epsilon \tag{1}$$

Rabin fingerprinting [7]: A hash defined over a finite, Galois field, $\mathrm{GF}(2)$, that maps an n-bit vector represented as a polynomial (using techniques like polynomial interpolation [8]), $P(x)$ of degree $(d-1)$, to the remainder, $r(x)$, obtained on dividing $P(x)$ by $g(x)$; where $g(x)$ is an irreducible polynomial of degree k over $\mathrm{GF}(2)$[7]. Thereby the Rabin Fingerprinting algorithm reduces a large d-bit vector to a unique k-bit fingerprint.

So the Rabin fingerprint over a large finite field of prime order, $\mathbb{F}_p$, is given by,

$$h(x) = P(x)(\mathrm{mod}\ g(x)) \tag{2}$$

where the coefficients of $h(x)$ (degree $k-1$) are the reduced k-bit hash.

The collision probability is then described by the Schwartz-Zippel Lemma [9]. The lemma states that for any two distinct polynomials $P_1(x), P_2(x)$, each of degree at most $(d-1)$, the collision probability for a randomly chosen point $q \in \mathbb{F}_p$ is bounded by,

$$\Pr_{q \in \mathbb{F}_p}[P_1(q) = P_2(q)] \leq \frac{(d-1)}{|\mathbb{F}_p|} \tag{3}$$

Thereby, choosing a finite field ordered by a large enough prime, and constraining the highest polynomial degree to be low, leads to statistically negligible collision probabilities.

1.2 Zero-knowledge machine learning

Zero-Knowledge Proof: A ZKP is a cryptographic protocol that allows one party, the prover, to convince another party, the verifier, that a given statement is true, without revealing any information beyond the fact of the statement's truth [10]. A powerful application of modern ZKP systems lies in their ability to guarantee computational integrity, making them tools for fast verifiability. The core properties of ZKPs are completeness, soundness and privacy-preservation.

zkML: An application of ZKPs for generating a succinct proof, π, of the correct execution of an ML model's inference computation, $C(x, w) = y$, where $x \in L$ is a public input in a corpus L (e.g., training dataset), w is the private witness (e.g., model parameters, intermediate activations) and y is the public model output. For our usage, privacy is not a major factor and we focus on verifiability, which for a given prover $\mathcal{P}$ and verifier $\mathcal{V}$ is defined, as in [11], by the completeness,

$$\Pr[\langle \mathcal{P}(x, w) \leftrightarrow \mathcal{V}(x) \rangle = \mathrm{accept}] = 1 \tag{4}$$

and the soundness, for $x' \notin L$ and a random prover $\mathcal{P}^*$,

$$\Pr[\langle \mathcal{P}^*(x', w) \leftrightarrow \mathcal{V}(x') \rangle = \mathrm{accept}] \leq \epsilon \tag{5}$$

where $\langle \rangle$ represents prover-verifier interactions.

The foundational step of most zkML systems is arithmetization; the process of transforming computation C into a family of algebraic equations. These equations serve as polynomial constraints that are tractable when defined over a finite field $\mathbb{F}_p$, and the validity of the computation is reduced to proving that these constraints hold. The complexity of a zkML system is largely concentrated in the arithmetization stage. The subsequent steps involve using cryptographic primitives such as hashing and Polynomial Commitment Schemes (PCS) [10] to compress the constraints down to unique, tractable commitments (e.g., fingerprints), used for faster verification. The Schwartz-Zippel lemma (Eq.3) again serves as a bound for the integrity of the proof.

We focus on a specific type of zkML system based on zk-STARK (Zero-Knowledge Scalable Transparent ARguments of Knowledge) [12]. zk-STARKs are typically constructed under minimal cryptographic assumptions which gives rise to scalability and transparency. The complexity, for a given computation of size $T(n)$, scales quasi-linearly, $O(T(n)\text{poly}\log T(n))$ for proof generation and poly-logarithmically, $O(\text{poly}\log T(n))$ for proof verification [12]. The transparency makes the system auditable and reproducible.

1.3 Early-Exit

EE [13] strategies are a popular approach for significantly reducing inference time and computation of AI algorithms running on low power edge-devices. A typical implementation of this method involves multiple exit-branches at different stages of the forward pass of a large ML model, where each branch is composed of a smaller classifier [14], with preset exit-logic. While this approach significantly reduces computation for some online samples, multiple exit-branches and complex exit-logic likely do not satisfy LHC trigger latency constraints [14].

2 PHAZE: Probabilistic Hashing And ZkML-based Early-exit

We present PHAZE [1], a novel EE framework designed for a new generation of low latency ML-based triggers, built with the cryptographic primitives described in Sec.1. For the LHC, feasibility studies for such a framework should broadly cover: computational constraints for online trigger systems; operational constraints laid by online hardware resources; physics performance in terms of trigger efficiency and other related parameters; verifiability for data quality monitoring and reproducing trigger decisions for downstream analyses.

Scope and assumptions: In this work we restrict ourselves to accelerating the inference process of a performant ML classifier that acts as the main triggering algorithm with the following assumptions:

1. Representative dataset: We assume that a large dataset corresponding to the input-space $\mathcal{I}$, is available to train on and serves as an acceptable representation of multiple classes of real events (backgrounds, noise and signals) commonly encountered at the LHC.

2. Baseline model: Design choices for this large, high-accuracy model, $\mathbb{M}_{\text{full}}$, like a single, large, general-purpose, physics foundation model versus multiple small specialized models; the model architecture; its training recipe etc., are left out of our scope.

3. Early predictive sufficiency: We assume that early-layer activations, $\mathcal{A} \in \mathbb{R}^k$ of $\mathbb{M}_{\text{early}} \subset \mathbb{M}_{\text{full}}$, act as EE signals and are a strong proxy to the model decisions $\mathcal{D} = \{0,1\}^t$ (for t different trigger classes), for a large enough subset $I \subseteq \mathcal{I}$. Here $\mathbb{R}^k$ is a k-dimensional real, latent activation vector space such that $\mathbb{M}_{\text{early}} \colon I \to \mathbb{R}^k$. Design choices to fulfill this requirement may involve knowledge distillation (KD), fine-tuned adapter layers on top of a pre-trained $\mathbb{M}_{\text{full}}$ or other such strategies and this choice is out of scope for our work.

4. Injectivity of polynomial interpolation: We assume that a polynomial interpolation scheme that maps a vector $A^* \in \mathbb{R}^k$ to a polynomial $P_{A^*}(x)$ is available and injective. Here, $P_{A^*}(x)$ belongs to the space $\mathcal{P}_d(\mathbb{F}_p^m)$ of all m-variate polynomials of degree at most d.

2.1 Method

PHAZE is designed to be a two-stage algorithm: a computation-intensive build phase that is not constrained by latency requirements or hardware choices and a low latency online phase that is expected to be a part of the online trigger algorithm.

Build phase: We start with a trained, large baseline model that gives the mapping $\mathbb{M}_{\text{full}} \colon \mathcal{I} \to \mathcal{D}$. Based on the assumption above, this trained model can be approximated by the map $\text{EE} \colon \mathcal{A} \to \mathcal{D}$, defined as the activation extraction step. While this EE can be statistically equivalent to KD, PHAZE does not smear over unseen events, and that is essential for discoveries. Each activation $A_j \in \mathcal{A}$ corresponding to $I_j \in \mathcal{I}$ is quantized to $A_j^* \in \mathbb{F}_p^k$. The choice between quantization during versus after training is left to experimental evaluation. We then generate an injective map for each A_j^* to polynomial $P_{A_j^*}$ that passes through a predefined set of k evaluation nodes, $\{x_1, ..., x_k\} \subset \mathbb{F}_p^m$ such

that $P_{A_j^*}(x_i) = A_{j,i}^*$ for all $i \in \{1, ..., k\}$. A rabin fingerprint is then generated as a hash for each such $P_{A_j^*}$ as described in Eq.2 with each polynomial evaluated at a constant, random challenge point $r = (r_1, ..., r_m)$ sampled from $\mathbb{F}_p^m$. This fingerprint h_j is then used as a key to populate a decision map , the Verifiable Decision Map (VDM), whose value is $D_j \in \mathcal{D}$. Next, we start the zk-STARK based proof generation, by reusing the polynomial interpolation and hashing steps. The computational integrity (CI) statement to be proven is formulated as a compound assertion S_j for each I_j given by:

> *"The inference of $\mathbb{M}_{\text{full}}$ on I_j yields D_j AND inference of $\mathbb{M}_{\text{full}}$ on I_j yields a quantized A_j^*, whose canonical polynomial $P_{A_j^*}$ evaluates to the hash value h_j at the public challenge point r."*

This statement is converted into an Algebraic Intermediate Representation (AIR), which consists of an execution trace of the whole system with corresponding polynomial constraints that hold if and only if the computations were executed correctly as explained in Sec.1.2. The zkML prover $\mathcal{P}$ then uses the AIR to generate a succinct proof π_j to certify S_j. As discussed above, this proof generation, though computation-intensive, scales to reasonably large models, with the proofs being transparent for ad-hoc downstream auditing and verification. π_j now certifies the validity of the decision map VDM: $h_j \to D_j$. The proofs π_j make the decision map resistant to corruptions such as accidental bit-flips and can then be archived.

Online phase: This phase is deployed at the trigger level and it starts with the same activation extraction step as in the build phase to produce A_{new}^* for each new event I_{new}, ideally optimized for specialized hardware like FPGAs. Dedicated hardware cores then perform an on-the-fly (OTF) hashing step to directly evaluate $h_{\text{new}} = P_{A_{\text{new}}^*}(r)$ using the numerically stable Barycentric Lagrange interpolation and evaluation method [15] which skips a major portion of the polynomial definition in the build phase since we do not need to generate a new proof online. Then a fast VDM lookup is performed using h_{new} as the key to retrieve the trigger decision, $D' = \text{VDM}[h_{\text{new}}]$.

A map-miss occurs when no VDM key matches h_{new}. This could indicate either a physics anomaly or an unknown detector anomaly. In either case, PHAZE inherently serves as an anomaly detector, with potential for incorporating specialized, downstream anomaly handling algorithms by buffering such events or by setting a default action (e.g., discard the event).

Table 1: Breakdown of computational tasks in the PHAZE framework.

Phase	Task	Latency per event	Computational complexity
Build	Full Model Inference	$O(\text{ms - s})$	$O(|\mathbb{M}_{\text{full}}|)$
	EE Activation Extraction	$O(\mu s)$	$O(|\mathbb{M}_{\text{early}}|)$
	Polynomial Interpolation	$O(\text{ms - s})$	$O(\text{poly}(k \log k))$
	Probabilistic Hashing	$O(\mu s)$	$O(d)$
	zk-STARK Proof Generation	$O(\text{min})$	$O(|\mathbb{M}_{\text{full}}|\text{poly-log}(|\mathbb{M}_{\text{full}}|))$
	ad-hoc Proof Verification	$O(\mu s - ms)$	$O(\text{poly-log}(|\mathbb{M}_{\text{full}}|))$
	VDM Population	$O(\text{ns})$	$O(1)$ per event
Online	**EE Activation Extraction**	$O(\textbf{ns})$ **(FPGA)**	$O(|\mathbb{M}_{\text{early}}|)$
	On-the-Fly Hashing	$O(\textbf{ns})$ **(FPGA)**	$O(\sqrt{d})$ **Erstin's method**
	VDM Lookup	$O(\textbf{ns})$ **(FPGA)**	$O(1)$ **from BRAM**

Latency estimates per event: The net theoretical latency for the online trigger is thereby given by $T_{\text{online}} = T_{\text{EE}} + T_{\text{OTF-hashing}} + T_{\text{VDM-lookup}}$. A conservative estimate for T_{EE}, for an optimized $\mathbb{M}_{\text{early}}$ on an FPGA would be 100-200ns [16, 17]. OTF-hashing using fast polynomial evaluation (e.g. Estrin's method [18]) takes as low as ~ 20 clock cycles on modern FPGAs, giving $T_{\text{OTF-hashing}} \approx 50\text{ns}$, while a VDM lookup can be performed in a single clock cycle, giving $T_{\text{VDM-lookup}} \approx 2.5\text{ns}$. Thereby a conservative estimate of T_{online} for a single event across a reasonable number of trigger classes would be in the range of 152.5-252.5ns on modern FPGAs.

3 Software Package and Benchmarking

The `PHAZE` [1] software package (hosted on `GitHub`[1]), is written mainly in `PYTHON` with custom `PYTHON-RUST` bindings to integrate contemporary cryptography packages generally written in `RUST`. `PHAZE` is designed for a wide range of benchmarking tasks on `PHAZE`-like trigger systems. The package (`v0.14.4`) contains the full build-phase pipeline. We are actively expanding towards full feasibility studies for LHC triggers.

3.1 Early benchmark results: Build-scaling

We construct $\mathbb{M}_{\text{full}}$ as a DNN with $\sim 7M$ parameters and train it on the `MNIST` dataset [19]. We then extract multiple $\mathbb{M}_{\text{early}}$ models at different parameter ratios from $5 - 95\%$ of $\mathbb{M}_{\text{full}}$ parameter count. We use the `EZKL` package built on zk-SNARKs [20], for any ML model exported to `ONNX` [21]. We then compare Rabin fingerprinting and a competing algorithm Shamir Secret Sharing [22], to generate 64-bit hashes of the different M_{early} outputs. We run experiments on an Apple `MAC-M1` `CPUs`. No optimizations have been attempted for these early benchmarks.
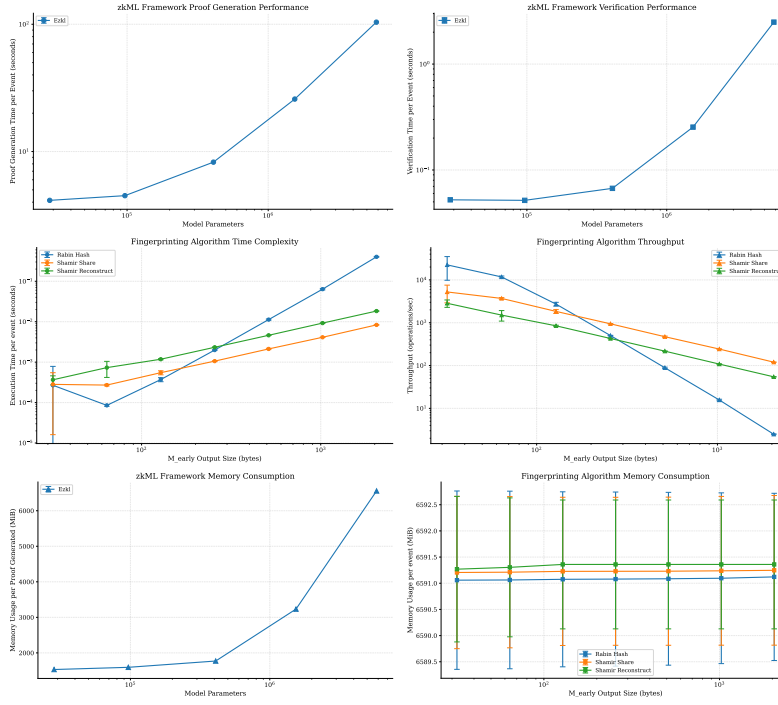

Figure 1: Early build-phase feasibility plots showing time complexity scaling for `ezkl` proof generation (top-left), verification (top-right) and fingerprinting (middle-left); net fingerprinting throughput (middle-right); peak memory consumption for `ezkl` proof generation (bottom-left) and fingerprinting (bottom-right). All values are reported per event, across 10 independent trials.

4 Conclusions

We present the `PHAZE` framework to potentially achieve nanosecond-order trigger latencies at the LHC using foundation model-scale ML algorithms. To the best of our knowledge, this is the first physics framework to utilize cryptographic primitives to enable fast inference and ensure verifiability of trigger algorithms, which is essential for dynamic, ML-based event selection at the LHC. We include further comments on feasibility in the appendices and the current assumptions are part of our ongoing efforts in realizing `PHAZE` as an experimental proof-of-concept system. `PHAZE` is also compatible as a building block for a larger distributed algorithm that is dynamically updated, while remaining verifiable, for flexible low level selections as a new paradigm for trigger design (App.C).

---

[1]Code available as the Zenodo-DOI for PHAZE in [1]

References

[1] P. Jawahar, "Github:praktikal24/phaze: Probabilistic hashing and zero-knowledge-ml for early-exit models." `https://doi.org/10.5281/zenodo.17370252`, Oct., 2025. 10.5281/zenodo.17370252.

[2] CMS collaboration, *Realtime Anomaly Detection at the L1 Trigger of CMS Experiment*, *PoS* **ICHEP2024** (2025) 1025 [2411.19506].

[3] F. Fahim, B. Hawks, C. Herwig, J. Hirschauer, S. Jindariani, N. Tran et al., *hls4ml: An open-source codesign workflow to empower scientific low-power machine learning devices*, *arXiv preprint arXiv:2103.05579* (2021) .

[4] B. Preneel, *Cryptographic hash functions*, *European Transactions on Telecommunications* **5** (1994) 431.

[5] P. Rogaway and T. Shrimpton, *Cryptographic hash-function basics: Definitions, implications, and separations for preimage resistance, second-preimage resistance, and collision resistance*, in *International workshop on fast software encryption*, pp. 371–388, Springer, 2004.

[6] C. Estébanez, Y. Saez, G. Recio and P. Isasi, *Performance of the most common non-cryptographic hash functions*, *Software: Practice and Experience* **44** (2014) 681.

[7] M.O. Rabin, *Fingerprinting by random polynomials*, Ph.D. thesis, Cambridge, MA, USA, 1981.

[8] D. Bleichenbacher and P.Q. Nguyen, *Noisy polynomial interpolation and noisy chinese remaindering*, in *International conference on the theory and applications of cryptographic techniques*, pp. 53–69, Springer, 2000.

[9] Ø. Ore, *Über höhere Kongruenzen*, Norsk Matematisk Forenings skrifter, Grøndahl (1921).

[10] O. Goldreich and H. Krawczyk, *On the composition of zero-knowledge proof systems*, *SIAM Journal on Computing* **25** (1996) 169.

[11] E. Ben-Sasson, A. Chiesa, M. Green, E. Tromer and M. Virza, *Secure sampling of public parameters for succinct zero knowledge proofs*, in *2015 IEEE Symposium on Security and Privacy*, pp. 287–304, IEEE, 2015.

[12] E. Ben-Sasson, I. Bentov, Y. Horesh and M. Riabzev, *Scalable, transparent, and post-quantum secure computational integrity*, *Cryptology ePrint Archive* (2018) .

[13] S. Teerapittayanon, B. McDanel and H.-T. Kung, *Branchynet: Fast inference via early exiting from deep neural networks*, in *2016 23rd international conference on pattern recognition (ICPR)*, pp. 2464–2469, IEEE, 2016.

[14] Y. Matsubara, M. Levorato and F. Restuccia, *Split computing and early exiting for deep learning applications: Survey and research challenges*, *ACM Computing Surveys* **55** (2022) 1.

[15] J.-P. Berrut and L.N. Trefethen, *Barycentric lagrange interpolation*, *SIAM review* **46** (2004) 501.

[16] E. Govorkova, E. Puljak, T. Aarrestad, M. Pierini, K.A. Woźniak and J. Ngadiuba, *Lhc physics dataset for unsupervised new physics detection at 40 mhz*, *Scientific Data* **9** (2022) 118.

[17] P. Odagiu, Z. Que, J. Duarte, J. Haller, G. Kasieczka, A. Lobanov et al., *Ultrafast jet classification at the hl-lhc*, *Machine Learning: Science and Technology* **5** (2024) 035017.

[18] M. Bodrato and A. Zanoni, *Long integers and polynomial evaluation with estrin's scheme*, in *2011 13th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing*, pp. 39–46, IEEE, 2011.

[19] L. Deng, *The mnist database of handwritten digit images for machine learning research [best of the web]*, *IEEE signal processing magazine* **29** (2012) 141.

[20] T. Chen, H. Lu, T. Kunpittaya and A. Luo, *A review of zk-snarks*, *arXiv preprint arXiv:2202.06877* (2022) .

[21] O.R. developers, "Onnx runtime." `https://onnxruntime.ai/`, 2021.

[22] A. Shamir, *How to share a secret*, *Communications of the ACM* **22** (1979) 612.

[23] S. Xu, *Efficient polynomial evaluation algorithm and implementation on fpga*, 2013.

A Probabilistic guarantees and error components

The model accuracy $\varepsilon(\mathbb{M}_{\mathrm{full}})$ is a major component of the net achievable trigger efficiency for `PHAZE`-like frameworks, and is framework-independent by assumption. The most prominent factor that leads to a drop in $\varepsilon(\mathbb{M}_{\mathrm{full}})$ is the probability of predictive insufficiency for the map $\mathrm{EE}\colon a_j \in \mathcal{A} \to d' \in \mathcal{D}$, given by $\Pr[d' \neq d_j]$, for $d_j = \mathbb{M}_{\mathrm{full}}[I_j]$. The operational bounds for this term depend on the EE strategy chosen and should be statistically estimated from experimental validation. The quantization of the latent activation vector can introduce prediction errors as well. However, this can be mitigated to an extent using representation learning techniques, like contrastive loss terms, that force the early layers to place different classes in distinct regions of the latent activation space. Events on the boundaries of these regions would have a higher chance of misclassification and thereby their contributions are encapsulated in the accuracy losses above. The hashing itself introduces an error when a collision occurs. However, for a 64-bit hash and an expressive polynomial degree choice of $d = 100$, the upper bound given by Eq.3 is $100/2^{64}$, which is computationally insignificant, but should be included in the net uncertainties regardless.

B Hardware resource feasibility

The main accelerating factor that `PHAZE` takes advantage of, is moving most of the heavy computation to the build phase where we are not compute/latency constrained. For the online phase, evaluating $\mathrm{EE}\colon \mathcal{A} \to \mathcal{D}$ has shown to be feasible for reasonably large models using `hls4ml`. OTF-hashing using optimizations on Horner's rule or Estrin's method are viable on modern FPGAs (e.g., Xilinx Virtex 6) [23]. The VDM itself is a limiting component since its most efficient use requires storage in FPGA memory. For e.g., AMD UltraScale+ FPGA series provides 455Mb memory. For a 64-bit hash, each VDM entry would require say 72 bits, for the hash and the decision across say 8 trigger classes. This means a single FPGA can hold evaluations for $\frac{455 \times 10^6}{72} \approx 6.3 \times 10^6$ events. This size is not sufficient for a representative dataset and thereby strategies to enable fast distributed look-ups across multiple VDMs would be preferable. This is currently an open research question we look to address in our future work.

C Inherent capabilites and future directions

C.1 Map-miss anomaly detection

Map-misses are an intrinsic tool that can provide "computationally free", low-level anomaly detection. Map-miss rates are strongly tied to representativeness of the dataset description, EE prediction sufficiency and VDM size and can be counter-productive if the above assumptions are not met. However, map-misses can strongly point to anomalous physics events or novel, unknown detector effects, that would otherwise be missed by current trigger algorithms. Such events would generally be discarded, or would require dedicated anomaly detection algorithms at the low-level trigger, which increases the computational overhead. Both these types of anomalies are essential to analyze for better long-term physics, as well as detector, performance. Leveraging map-misses in `PHAZE`-like trigger paradigms and combining it with powerful downstream trigger analyses or a dynamic VDM setup could potentially improve chances for discovery.

C.2 Dynamic VDM

The initial proposal envisions a static VDM, compiled once from a fixed dataset. However, the `PHAZE` framework readily supports dynamic updates, allowing the trigger system to adapt over time without requiring a full hardware redesign or resynthesis. The zkML prover-verifier setup is crucial to maintain the integrity of such a system.

Scheduled updates: The most direct approach involves periodically updating the VDM during planned LHC operational pauses, such as technical stops or shutdowns. During these periods, a new, more comprehensive VDM can be compiled offline. This updated map could incorporate knowledge gained from previous runs, adapt to changes in detector calibration and conditions, or, most significantly, include newly understood event topologies derived from classes of events with high map-miss rates.

Autonomous updates: A more ambitious form of a dynamic VDM would involve using knowledge accumulated online by downstream high-level triggers (HLT), and intermittently use beam downtime to update the VDM. This would require smaller baseline models that can simulate artificial maps between the event hashes and the HLT trigger decisions for event topologies with high map-miss rates. However, this would also require very slow online proof generation before the VDM can be altered. The current technology gap to realize a system like this is high, however, an autonomous, dynamic trigger system can greatly improve physics analyses, specially in the direction of new physics searches.