PROCEEDINGS OF SCIENCE

Optimization of ML-Based BSM triggering with Knowledge Distillation for FPGA implementation

Dr. Marco Lorusso*

CNAF Division, Istituto Nazionale di Fisica Nucleare,
Viale Berti Pichat 6/2, Bologna, Italy

E-mail: marco.lorusso@cnaf.infn.it

To enhance the discovery potential of the Large Hadron Collider (LHC) at CERN in Geneva and improve the precision of Standard Model measurements, the High Luminosity LHC (HL-LHC) Project was initiated in 2010 to extend its operation by another decade and increase its luminosity by approximately tenfold beyond the design value.
In this context, the scope of applications for Machine Learning, particularly Artificial Neural Network algorithms, has experienced an exponential expansion due to their considerable potential for elevating the efficiency and efficacy of data processing, especially for innovative trigger-level event selection in Beyond Standard Model (BSM) research. This study explores Autoencoders (AEs), unbiased algorithms that select events based on abnormality without theoretical assumptions. However, the stringent latency and energy constraints of a HEP Trigger system require tailored software development and deployment strategies. These strategies aim to optimize the utilization of on-site hardware, with a specific focus on Field-Programmable Gate Arrays (FPGAs). This is why a technique called Knowledge Distillation (KD) is studied in this work. It consists in using a large and well trained "teacher", like the aforementioned AE, to train a much smaller student model which can be easily implemented on an FPGA. The optimization of this distillation process involves exploring different aspects, such as the architecture of the student and the quantization of weights and biases, with a strategic approach that includes hyperparameter searches to find the best compromise between accuracy, latency and hardware footprint.
The strategy followed to distill the teacher model will be presented, together with consideration on the difference in performance when applying the quantization before or after the best student model has been found.

International Symposium on Grids and Clouds (ISGC2025)
16 -21 March 2025
Academia Sinica Grid Computing Centre (ASGC), Taipei, Taiwan

*Speaker

 ISSN 1824-8039 . Published by SISSA Medialab.

PoS(ISGC2025)018

1. Introduction

In order to increase the discovery potential of the Large Hadron Collider (LHC) at CERN, as well as to improve the precision of Standard Model physics measurements, the High Luminosity LHC (HL-LHC) Project was setup in 2010 to extend its operability by another decade and to increase its luminosity (and thus collision rate) by a factor of ~ 10 beyond its design value.

With the purpose of fully exploiting the HL-LHC running period, major consolidations and upgrades of all four main detectors at LHC are planned. The collision rate and level of expected pileup imply very high particle multiplicity and an intense radiation environment and imposes serious challenges to the Trigger system requirements in order to maintain performance, which pushes the need for technological advances at the hardware level of the data acquisition system, as well as new software ones to increase the physical acceptance of interesting events, while intensifying the efforts to identify and analyze events which are not explainable with the Standard Model theory.

One of the most popular type of algorithms proposed to tackle these needs is the one commonly known as Machine Learning, with a major focus on Artificial Neural Networks. In recent years, Machine Learning has become one of the pillars of Computer and Data Science and it has been introduced in almost every aspect of everyday life. This spread of learning algorithms in many sectors finds its roots mainly in an increased quantity of data available, combined with a technological progress in storage and computational power, which can nowadays be exploited with lower maintenance and material costs.

However, the latency and energy constraints of the first line of data acquisition at LHC experiments are quite unique and create the necessity of specific software development and strategies to deploy Machine Learning models efficiently on the hardware available on-site, like FPGAs.

Field Programmable Gate Arrays (FPGAs) combine the benefits of hardware and software by implementing circuits for high performance and efficiency while being reprogrammable for various tasks. They perform millions of operations simultaneously across a silicon chip, making them significantly faster than microprocessor-based designs, and can be reprogrammed multiple times, unlike ASICs.

A candidate for using Machine Learning as a triggering mechanism at LHC is the research for Beyond Standard Model events. The trigger selection algorithms are designed to guarantee a high acceptance rate for certain physics processes under study. When designing ways to search for new physics kinds of collisions (e.g., dark matter production), physicists typically consider specific scenarios motivated by theoretical considerations. This approach may become a limiting factor in the absence of a strong theoretical prior. This is why unsupervised ML techniques, like Autoencoders, can be useful for new physics mining. By deploying an unbiased algorithm which selects events based on their degree of abnormality, rather than on the amount of energy present in the event, data can be collected in a signal-model-independent way. Such an anomaly detection (AD) algorithm is required to have extremely low latency because of the restrictions imposed by the frequency of new events at LHC, and this is why there is a need to optimize and compress these kind of algorithm to make them suitable for trigger environments.

2. Anomaly Detection with Autoencoders

Anomaly detection (AD) aims to identify instances containing patterns that deviate from those observed in normal instances [1]. This task is crucial in various vision applications, such as manufacturing defect detection, medical image analysis, and video surveillance. Unlike typical supervised classification problems, anomaly detection presents unique challenges. Primarily, it is difficult to obtain a substantial amount of anomalous data, whether labeled or unlabeled. Additionally, the differences between normal and anomalous patterns are often fine-grained, as defective areas can be small and subtle in high-resolution images. Since the distribution of anomaly patterns is unknown in advance, models are trained to learn the patterns of normal instances.

In practice, an instance is determined to be anomalous if it is not well-represented by these models. Considering the peculiarities of this kind of task, AD is one of the popular application for Autoencoders [2].

2.1 Autoencoders

An Autoencoder (AE) is a type of Neural Network that is trained to attempt to copy its input to its output [3]. Internally, it usually comprises of a hidden layer h that describes a code used to represent the input. The network can be viewed as made up of two parts: an encoder function $\mathbf{h} = f(\mathbf{x})$ and a decoder that produces a reconstruction $\mathbf{r} = g(\mathbf{h})$. This architecture is presented in Figure 1. If an Autoencoder succeeds in simply learning to set $g(f(\mathbf{x})) = \mathbf{x}$ everywhere, then it is not especially useful. Instead, Autoencoders are designed to be unable to learn to copy perfectly. Usually they are restricted in ways that allow them to copy only approximately, and to copy only input that resembles the training data. Because the model is forced to prioritize which aspects of the input should be copied, it often learns useful properties of the data.

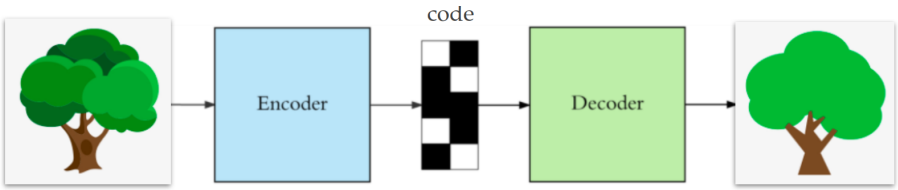

Figure 1: The general structure of an autoencoder, mapping an input to an output (called reconstruction) through an internal representation or code. The autoencoder has two components: the encoder and the decoder.

Autoencoders with nonlinear encoder functions f and nonlinear decoder functions g can learn a more powerful nonlinear generalization of Principal Component Analysis [4]. In other words, it is able to simplify the data while preserving its essential patterns and structures.

3. Knowledge Distillation

Deploying large, accurate deep learning models to resource-constrained environments like FPGAs, mobile phones, and smart cameras presents significant challenges. These models often have millions of parameters requiring substantial storage, while on-device memory is limited.

Additionally, a single model inference can involve billions of memory accesses and arithmetic operations, which consume power, generate heat, and drain battery life, or test the device's thermal limits. To address these issues, research is focused on compressing neural network models to reduce memory and computation demands while maintaining model quality. Model compression not only reduces energy-intensive memory accesses but also improves inference time by increasing effective memory bandwidth. Knowledge Distillation, a key approach in this research, involves training a smaller student model to mimic a larger, complex teacher model, aiming to achieve competitive performance with fewer resources [5, 6].

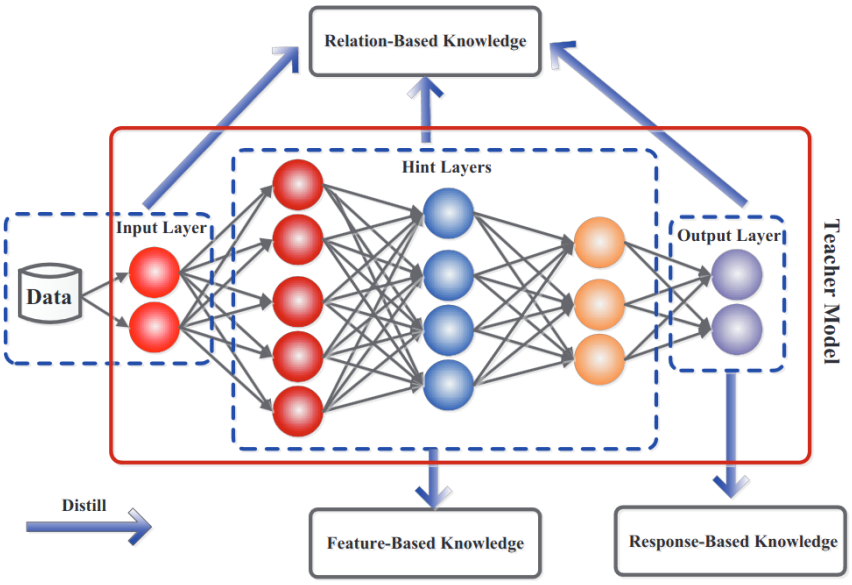

Figure 2: A schematic illustration of three different types of knowledge that can be transferred from a deep teacher network: response-based knowledge, feature-based knowledge and relation-based knowledge.

Knowledge types, distillation strategies and the teacher-student architectures play a crucial role in the student learning [7]. Indeed, there are three different categories of knowledge (see Figure 2):

Response-based It usually refers to the neural response of the last output layer of the teacher model. The main idea is to directly mimic the final prediction of the teacher model. The response-based knowledge distillation is simple yet effective for model compression, and has been widely used in different tasks and applications;

Feature-Based Deep neural networks are good at learning multiple levels of feature representation with increasing abstraction. This is known as representation learning. Therefore, both the output of the last layer and the output of intermediate layers, i.e., feature maps, can be used as the knowledge to supervise the training of the student model. Specifically, feature-based knowledge from the intermediate layers is a good extension of response-based knowledge, allowing the creation of networks with more layers but less nodes per layer, still preserving a good distillation;

Relation-Based Both response-based and feature-based knowledge use the outputs of specific

layers in the teacher model. Relation-based knowledge further explores the relationships between different layers or data samples.

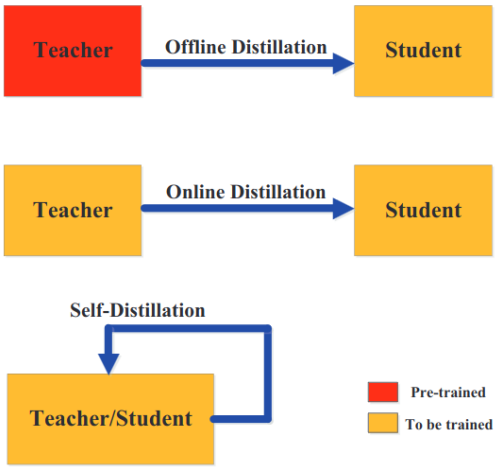

Figure 3: A schematic illustration of three different Knowledge Distillation strategies: Offline, Online and Self-Distillation.

In Figure 3 the three main ways to transfer knowledge from a teacher to a student model are shown. Offline Distillation involves transferring knowledge from a pre-trained teacher model to a student model in two stages: the teacher is first trained on a dataset, then it guides the student model's training using extracted knowledge. Online Distillation updates both the teacher and student models simultaneously in an end-to-end trainable framework: in this way it is possible to exploit the more complex and bigger architecture of a teacher, while instructing the student at the same time. Self-Distillation uses the same network for both teacher and student models by transferring knowledge, for example, from deeper sections of a network to its initial sections.

4. The Case study

In order to test Offline Response-Based Distillation, the aforementioned use of an Autoencoder to perform physics mining of events not explainable using the Standard model was chosen [8]. A data sample was selected that represents a typical proton-proton collision dataset that has been pre-filtered by requiring the presence of an electron or a muon with a transverse momentum $p_T > 23$ GeV and a pseudo-rapidity $|\eta| < 3$ and $|\eta| < 2.1$, respectively. This is representative of a typical trigger selection algorithm of a multipurpose LHC experiment. In addition to this, four benchmark new physics scenarios discussed were considered [9]:

- A *Leptoquark* (LQ) with a mass of 80 GeV, decaying to a b quark and a τ lepton;
- A Neutral scalar boson (A) with a mass of 50 GeV, decaying to two off-shell Z bosons, each forced to decay to two leptons: $A \rightarrow 4\ell$;
- A Scalar boson with a mass of 60 GeV, decaying to two tau leptons: $h_0 \rightarrow \tau\tau$;

- A charged scalar boson with a mass of 60 GeV, decaying to a tau lepton and a neutrino: $h_{\pm} \to \tau\nu$.

In total, the background sample consists of 8 million events. Of these, 50% are used for training, 40% for testing and 10% for validation. The new physics benchmark samples are only used for evaluating the performance of the models.

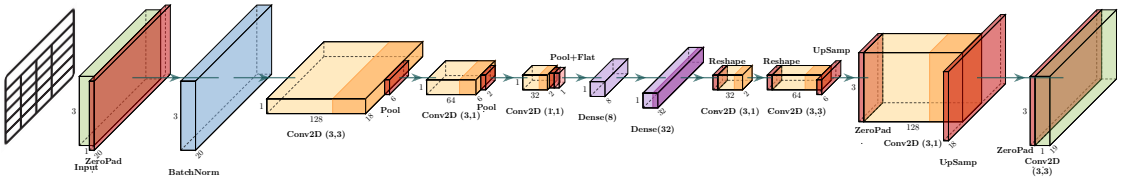

Figure 4: Network architecture of the teacher AE model performing Anomaly Detection in the case under study.

The architecture of the teacher model was an AE using convolutional layers (more details can be found in [8]), shown in Figure 4. The inputs consisted in p_T, η, ϕ (azimuthal angle w.r.t. the LHC beam pipe) values for 18 reconstructed objects (ordered as 4 muons, 4 electrons, and 10 jets), and the ϕ and magnitude of the missing transverse energy (MET), forming together an input of shape (19, 3) where MET η values are zero-padded by construction (η is zero for transverse quantities). For events with fewer than the maximum number of muons, electrons, or jets, the input is also zero-padded.

4.1 Quantization vs architecture

The work in this paper was done not only to simply test KD, but also to try to answer a question related to the actual procedure to follow when trying to obtain a small model implementable on an FPGA. In particular, when optimizing a NN for hardware inference, one must consider not only finding the optimal architecture but also identifying the best quantization. Here for quantization is intended the conversion of all parameters, e.g. weights, of a model to fixed-point numbers, better handled by FPGAs. In this case the quantization is considered before performing the training of the students, falling into the Quantization Aware Training category [10], achieved using QKeras [11].

Thus, the question arises: Is there a difference between searching for the best candidate with the quantization process in mind versus without it? In other words, do the results differ when first identifying the optimal architecture and then determining the best quantization compared to performing a hyperparameter search that simultaneously considers both aspects?

The strategy to answer to this was simple. Firstly a simple hyperparameter search with no quantisation was performed to get the best student architecture; then, another search was done for the quantization using the model obtained (PhaseSearch quantization). On the other hand, a single search was set up to search the architecture and quantization parameters at the same time (CoSearch quantization).

The results of the two search strategies are presented in Figure 5 for $\approx$ 1000 student candidates, where the distributions of the mean squared errors (MSE) of the models with respect to the teacher are compared across different anomalies the students are expected to detect. In these box plots,

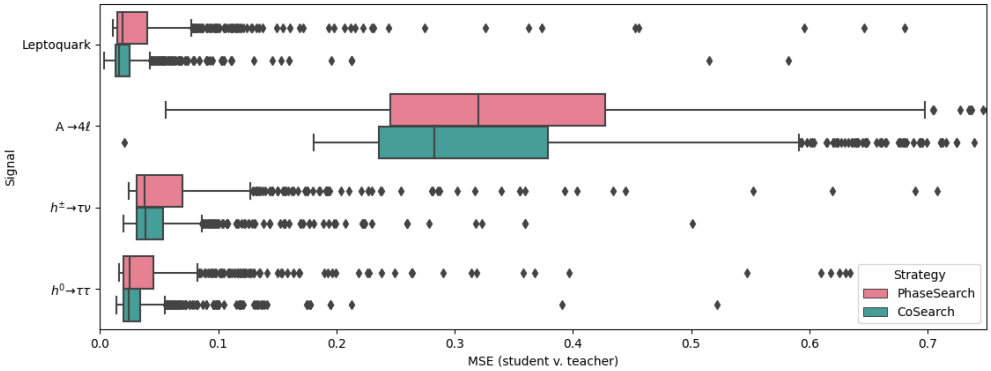

Figure 5: Distribution of MSE scores with respect to the teacher model of the students produced by the PhaseSearch (pink) and CoSearch (green) procedures for the 4 BSM signals under study.

the boxes extend from the first quartile (Q1) to the third quartile (Q3), while the whiskers reach out to the most extreme data points within 1.5 times the interquartile range (IQR), offering a visual summary of the performance variability across strategies and anomaly types.

It is evident how the CoSearch quantization produces students with lower MSEs more consistently, with minimum values of both searches that are basically the same.

This means that by performing architecture and quantization optimizations as a single hyperparameter search, it is more likely to achieve a good model with fewer attempts, while also ensuring that both procedures will ultimately yield nearly identical optimal candidates.

5. Results

After selecting the CoSearch procedure, the hyperparameter search identified the network shown in Figure 6 as the best-performing student. The entire architecture is made up of:

1. Flatten layer to transform the tabular data in input in a 1D vector;
2. BatchNormalization layer to make training faster and more stable;
3. A series of densely connected hidden layer with the following number of nodes and bits used to represent each weight and bias:
   (a) 64 nodes, 16 bits (6 bits integer part);
   (b) 32 nodes, 16 bits (6 bits integer part);
   (c) 8 nodes, 16 bits (6 bits integer part);
   (d) 8 nodes, 16 bits (6 bits integer part);
   (e) 16 nodes, 16 bits (6 bits integer part);
   (f) 8 nodes, 16 bits (6 bits integer part);
4. The final layer returning a single value: the reconstruction loss of the distilled AE.

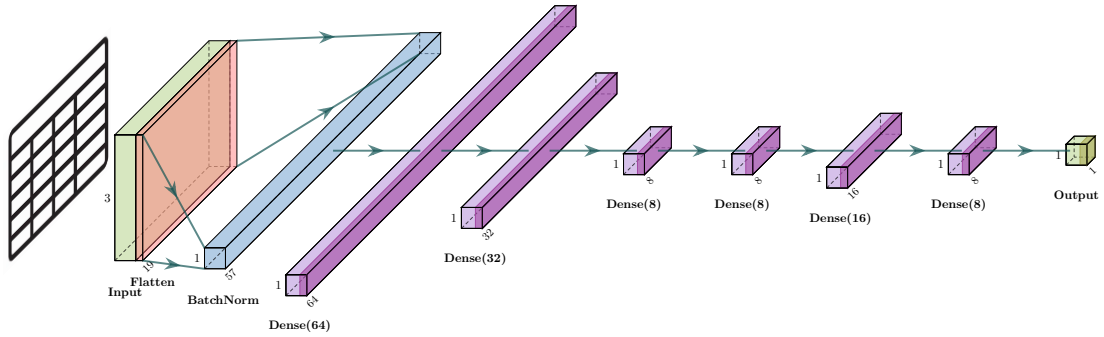

Figure 6: Network architecture for the best performing student using the CoSearch procedure for Knowledge Distillation.

This model contains ≈ 6200 parameters, reducing this number by ≈ 10% with respect to the Autoencoder used as teacher model.

Finally, in Figure 7, the ROC curves generated by the model are presented. To enable a meaningful comparison between the student and teacher models, the Clopper-Pearson interval was employed to compute upper and lower bounds for the True Positive Rate (TPR) at each point, corresponding to a 99% confidence level. Simultaneously, DeLong's algorithm was applied to estimate the standard error associated with the Area Under the Curve (AUC). The use of these two statistical methods [12] provided robust uncertainty quantification without resorting to the computationally intensive bootstrap approach (i.e., repeating the measurement multiple times). This allowed for more efficient and reliable comparisons between the models' performances.

The student model achieved a performance that closely matches that of the teacher model, making it a strong candidate for future FPGA implementation to evaluate its efficiency in terms of latency and hardware footprint.

6. Conclusions

In conclusion, this study addressed the critical aspect of efficient data processing for online applications when utilizing Artificial Neural Networks, particularly for selecting interesting events at the trigger level, such as Beyond Standard Model (BSM) events. By focusing on Autoencoders (AEs) — unbiased algorithms capable of event selection based on abnormality without theoretical priors — this research tackled the unique latency and energy constraints within the trigger domain, necessitating tailored software and deployment strategies to optimize on-site hardware, specifically Field-Programmable Gate Arrays (FPGAs). The study compared two different strategies for optimizing Neural Networks obtained from the Knowledge Distillation of an Autoencoder, examining the performance differences between applying quantization during the phase of identifying the best architecture and after. The results show that it is more probable to find a good model by performing a combined hyperparameter search for both aspects of the optimization, while also confirming that

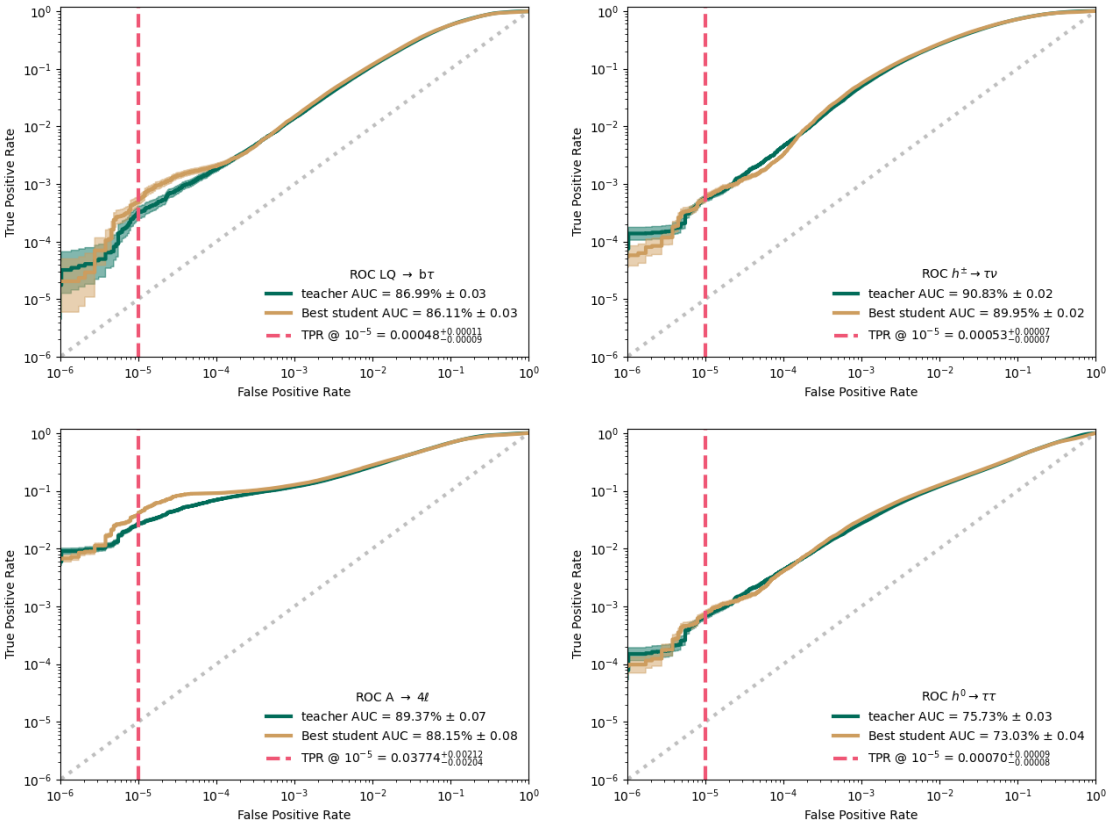

Figure 7: ROCs displaying the ability of the best student model (red) built with CoSearch quantization to detect the 4 test signals, compared with the performance of the teacher model (green).

the best models in both cases are comparable. Finally, the best model obtained by the combined search was compared to the accuracy of detecting BSM events of its teacher, showing a very close match in performance, making it a strong candidate for future FPGA implementation to evaluate its efficiency in terms of latency and hardware footprint.

References

[1] C.L. Li, K. Sohn, J. Yoon and T. Pfister, *Cutpaste: Self-supervised learning for anomaly detection and localization*, *ArXiv e-prints* (2021) [2104.04015].

[2] R. Chalapathy and S. Chawla, *Deep learning for anomaly detection: A survey*, *ArXiv e-prints* (2019) [1901.03407].

[3] I. Goodfellow, Y. Bengio and A. Courville, *Deep Learning*, MIT Press (2016).

[4] I.T. Jolliffe and J. Cadima, *Principal component analysis: a review and recent developments*, *Philos. Trans. A Math. Phys. Eng. Sci.* **374** (2016) .

[5] C. Buciluundefined, R. Caruana and A. Niculescu-Mizil, *Model compression*, *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* **KDD '06** (2006) 535–541.

[6] G. Hinton, O. Vinyals and J. Dean, *Distilling the knowledge in a neural network*, *ArXiv e-prints* (2015) [1503.02531].

[7] J. Gou, B. Yu, S.J. Maybank and D. Tao, *Knowledge distillation: A survey*, *International Journal of Computer Vision* **129** (2021) 1789–1819.

[8] E. Govorkova, E. Puljak, T. Aarrestad, T. James, V. Loncar, M. Pierini et al., *Autoencoders on field-programmable gate arrays for real-time, unsupervised new physics detection at 40 mhz at the large hadron collider*, *Nature Machine Intelligence* **4** (2022) .

[9] O. Cerri, T.Q. Nguyen, M. Pierini, M. Spiropulu and J.-R. Vlimant, *Variational autoencoders for new physics mining at the large hadron collider*, *Journal of High Energy Physics* **2019** (2019) .

[10] C.N. Coelho Jr., A. Kuusela, S. Li, H. Zhuang, T. Aarrestad, V. Loncar et al., *Automatic deep heterogeneous quantization of Deep Neural Networks for ultra low-area, low-latency inference on the edge at particle colliders*, *ArXiv e-prints* (2020) [2006.10159].

[11] QKeras Github Repository https://github.com/google/qkeras, 2024.

[12] M. Lorusso, *Optimization of ML-Based BSM triggering with Knowledge Distillation for FPGA implementation in the CMS Level-1 Trigger*, Ph.D. thesis, Alma Mater Studiorum - University of Bologna, 2025.