Autoencoders on FPGAs for real-time, unsupervised new physics detection at 40 MHz at the Large Hadron Collider

Ekaterina Govorkova,* Ema Puljak, Thea Aarrestad, Thomas James, Vladimir Loncar,† Maurizio Pierini, Adrian Alan Pol, Nicolò Ghielmetti,‡ Maksymilian Graczyk,§ and Sioni Summers
European Organization for Nuclear Research (CERN) CH-1211 Geneva 23, Switzerland

Jennifer Ngadiuba¶
Fermi National Accelerator Laboratory, Batavia, IL 60510, USA

Thong Q. Nguyen
California Institute of Technology Pasadena, CA 91125, USA

Javier Duarte
University of California San Diego La Jolla, CA 92093, USA

Zhenbin Wu
University of Illinois at Chicago Chicago, IL 60607, USA

(Dated: August 19, 2021)

In this paper, we show how to adapt and deploy anomaly detection algorithms based on deep autoencoders, for the unsupervised detection of new physics signatures in the extremely challenging environment of a real-time event selection system at the Large Hadron Collider (LHC). We demonstrate that new physics signatures can be enhanced by three orders of magnitude, while staying within the strict latency and resource constraints of a typical LHC event filtering system. This would allow for collecting datasets potentially enriched with high-purity contributions from new physics processes. Through per-layer, highly parallel implementations of network layers, support for autoencoder-specific losses on FPGAs and latent space based inference, we demonstrate that anomaly detection can be performed in as little as 80 ns using less than 3% of the logic resources in the Xilinx Virtex VU9P FPGA. Opening the way to real-life applications of this idea during the next data-taking campaign of the LHC.

I. INTRODUCTION

The CERN Large Hadron Collider (LHC) [1] generates 40 million proton-proton collision events per second. Particles produced in these events are detected in the sensors of detectors located around the LHC ring, producing hundreds of terabytes of data per second. The largest general-purpose particle detectors at LHC, ATLAS [2] and CMS [3], discard most of the collision events with online selection systems, as a result bandwidth limitations. These systems consist of two stages; the level-1 trigger (L1T) [4–7], where algorithms are deployed as programmable logic on custom electronic boards equipped with field-programmable gate arrays (FPGAs), and the High Level Trigger (HLT), where selection algorithms asynchronously process the events accepted by the L1T on commercially available CPUs. The largest fraction of events are discarded at the first selection stage, the L1T, which has the task of reducing the event rate by 2.5 orders of magnitude within a few microseconds. The trigger selection algorithms running in the L1T and HLT are designed to guarantee a high acceptance rate for certain physics processes under study. When designing searches for new physics kinds of collisions (e.g., dark matter production), physicists typically consider specific scenarios motivated by theoretical considerations. This *supervised* strategy has proven to be successful when dealing with theory-motivated searches, as was the case with the search for the Higgs boson [8, 9]. Conversely, this approach may become a limiting factor in the absence of a strong theoretical prior. For this reason, there are several community efforts to investigate unsupervised machine learning (ML) techniques for new physics searches [10, 11]. These investigate the use of autoencoders (AEs) and variational autoencoders (VAEs) for offline processing [24, 25], and therefore do not consider constraints such as resource usage and latency. Ref. [12, 13] propose to integrate unsupervised learning algorithms in the online selection system of the CMS and ATLAS experiments, in order to preserve rare events which would not otherwise be selected, in a special data stream. A similar, albeit non-learning, approach was pursued in CMS with the *exotica hotline* [14, 15] during the first year of LHC data taking and in similar efforts during experiments at the Super Proton–Antiproton Synchrotron and Tevatron.

While the primary focus for online unsupervised learn-

* E-mail: katya.govorkova@cern.ch
† Also at Institute of Physics Belgrade, Serbia
‡ Also at Politecnico di Milano, Italy
§ Also at Imperial College London, UK
¶ Also at California Institute of Technology, USA

ing so far has been for the HLT, this strategy could be more effective if deployed in the L1T, i.e. before any selection bias is introduced. Due to the extreme latency and computing resource constraints of the L1T, only relatively simple, mostly theory-motivated selection algorithms are currently deployed. These usually include requirements on the minimum energy of a physics object, such as a reconstructed lepton or a jet, effectively excluding lower-energy events from further processing. Instead, 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, we can collect data 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 L1T.

Recent developments of the **hls4ml** library allow us to consider, for the first time, the possibility of deploying an AD algorithm on the FPGAs mounted on the L1T boards. The **hls4ml** library is an open-source software, developed to translate neural networks [16–20] and boosted decision trees [21] into FPGA firmware. A fully on-chip implementation of the machine learning model is used in order to stay within the 1 μs latency budget imposed by a typical L1T system. Additionally, the initiation interval (II) of the algorithm should be within 150 ns, which is related to the bunch-crossing time for the upcoming period of the LHC operations. Since there are several L1T algorithms deployed per FPGA, each of them should use much less than the available resources. With its interface to QKERAS [22], **hls4ml** supports quantization-aware training (QAT) [23], which makes it possible to drastically reduce the FPGA resource consumption while preserving accuracy. Using **hls4ml** we can compress neural networks to fit the limited resources of an FPGA.

In this paper, we discuss how to adapt and improve the strategy presented in Ref. [12] to fit the L1T infrastructure. We focus on AEs, with specific emphasis on VAEs [24, 25]. We consider both fully-connected and convolutional architectures, and discuss how to compress the model through pruning [26–28], the removal of unnecessary operations, and quantization [17, 29–36], the reduction of the precision of operations, at training time.

As discussed in Ref. [12], one can train (V)AE on a given data sample by minimizing a measure of the distance between the input and the output (the loss function). This strategy, which is very common when using (V)AEs for anomaly detection [37], comes with practical challenges when considering a deployment on FPGAs. The use of high-level features is not optimal because it requires time-consuming data preprocessing. The situation is further complicated for VAEs, which require a random sampling from a Gaussian distribution in the latent space. Furthermore, one has to buffer the input data on chip while the output is generated by the FPGA processing in order to compute the distance afterwards. To deal with all of these aspects, we explore different approaches and compare the accuracy, latency and resource consumption of the various methods.

In addition, we discuss how to customize the model compression in order to better accommodate for unsupervised learning. Previously, we showed that QAT can result in a large reduction in resource consumption with minor accuracy loss for supervised algorithms [19, 23]. In this paper, we extend and adapt that compression workflow to deal with the specific challenge of compressing autoencoders used for AD. Several approaches are possible:

- Post-training quantization (PTQ) [16, 27, 38–41], consisting of applying a fixed-point precision to a floating-point baseline model. This is the simplest quantization approach, typically resulting in good algorithm stability, at the cost of losing performance. More aggressive PTQ (lower precision) is usually accompanied by a larger reduction in accuracy.

- QAT, consisting of imposing the fixed-point precision constraint at training time, e.g., using the QKERAS library. This approach typically allows one to limit the accuracy loss when imposing a higher level of quantization, finding a better weight configuration than what one can get with PTQ. However, applying QAT to VAE models for AD can result in unstable performance because QAT would return the best input-to-output reconstruction performance, but the best reconstruction does not necessarily guarantee the best AD performance. Ultimately, the stability of the result depends on the nature of the detected anomaly.

- Knowledge distillation with QAT: one could change the quantized-model optimization strategy, reframing the problem as knowledge distillation [42–45]. Rather than fixing the quantized weights to minimize the VAE loss, the difference between the loss from the quantized model and the floating-point model for the same input could be minimized. Rather than training a quantized copy of a given floating-point model, one could train a different model to predict this floating-point output, starting from the same input. Doing so, one could aim at targeting the floating-point AD performance with a completely different network (e.g., an MLP regression) that could better meet the constraints of a L1T environment, e.g. being faster or consuming less computing resources.

- Anomaly classification with QAT: the approximated loss regression with QAT could be turned into a classification problem. Rather than approximating the floating-point decision, one could try to obtain a yes/no answer to a different question: would the floating-point algorithm return an AD score larger than a threshold for this event? In this way, one could set the threshold on the accurate floating-point model and could obtain good accuracy (in terms of anomaly acceptance) without having to predict the exact AD score value across multiple orders of magnitude.

In this paper, we focus on the first two approaches, leaving the investigation of the other approaches to future work.

This paper is structured as follows: in Section II we describe the benchmark dataset. In Section III a detailed description of the autoencoder models is given, followed by Section IV in which the definition of the quantities used as anomaly detection scores is presented. In Section V results of the uncompressed and unquantized model are presented. In the next part, Section VI, model compression is detailed. Section VII describes the strategy to compress the models and deploy them on FPGAs, including an assessment of the required FPGA resources.

II. DATA SAMPLES

This study follows the setup of Ref. [12, 46]. We use a data sample 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 $p_T > 23\text{GeV}$ and a pseudo-rapidity $|\eta| < 3$ (electron) and $|\eta| < 2.1$ (muon). This is representative of a typical L1T selection algorithm of a multipurpose LHC experiment. In addition to this, we consider the four benchmark new physics scenarios discussed in Ref. [12]:

- A leptoquark (LQ) with a mass of 80 GeV, decaying to a b quark and a τ lepton [47],
- 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: $A \to 4\ell$ [48],
- A scalar boson with a mass of 60 GeV, decaying to two tau leptons: $h^0 \to \tau\tau$ [49],
- A charged scalar boson with a mass of 60 GeV, decaying to a tau lepton and a neutrino: $h^{\pm} \to \tau\nu$ [50].

These four processes are used to evaluate the accuracy of the trained models. A detailed description of the dataset can be found in Ref. [51].

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.

The training dataset together with signal datasets for testing are published on Zenodo [47–50, 52].

III. AUTOENCODER MODELS

We consider two classes of architectures: one based on dense feed-forward neural networks (DNNs) and one using convolutional neural networks (CNNs). Both start from the same inputs, namely the (p_T, η, ϕ) 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, as commonly done in the L1T algorithm logic.

In order to account for resource consumption and latency of the data pre-processing step, we use a batch normalization layer [53] as the first layer for each model. As all processing is done on-chip, the resource and latency measurements will be consistent with those of a real L1T implementation. For both architectures, CNN and DNN, we consider both a plain AE and a VAE. In the AE, the encoder provides directly the coordinates of the given input, projected in the latent space. In the VAE, the encoder returns the mean values $\vec{\mu}$ and the standard deviation $\vec{\sigma}$ of the N-dimensional Gaussian distribution which represents the latent-space probability density function associated with a given event.

For the DNN model, the four-vector of each reconstructed object is flattened and concatenated into a 1D array, resulting in a 57-dimension input vector. The DNN AE architecture is shown on the top plot in Figure I. All of the inputs are batch-normalized and passed through a stack of 3 fully connected layers, with 32, 16, and 3 nodes. The output of each layer is followed by a batch normalization layer and activated by a leaky ReLU function [54]. The 3-dimensional output of the encoder is the projection of the input in the latent space. The decoder consists of a stack of 3 layers, with 16, 32, and 57 nodes. As for the encoder, we use a batch normalization layer between the fully connected layer and its activation. The last layer has no activation function, while leaky ReLU is used for the others. The DNN VAE follows the same architecture, except for the latent-space processing, which follows the usual VAE prescription: two 3-dimensional fully connected layers produce the $\vec{\mu}$ and $\vec{\sigma}$ vectors from which Gaussian latent quantities are sampled and injected in the decoder.

The CNN AE architecture is shown on the bottom plot in Figure I. The encoder takes as input the single-channel 2D array of four-momenta including the two MET-related features (magnitude and ϕ angle) and zeros for MET η, resulting in a total input size of $19 \times 3 \times 1$. It should be emphasised that we are not using image data, rather treating tabular data as a 2D image to make it possible to explore CNN architectures. The input is first zero-padded in order to resize the image to $20 \times 3 \times 1$, which is required in order to parallelize the network processing in the following layer on the FPGA. For the Conv2D FPGA implementation, we control how many iterations of outer loop (over the rows of the image array) are running in parallel. To simplify the implementation we run the same number of iterations in parallel, which requires that the number of rows in the input image is an integer multiple

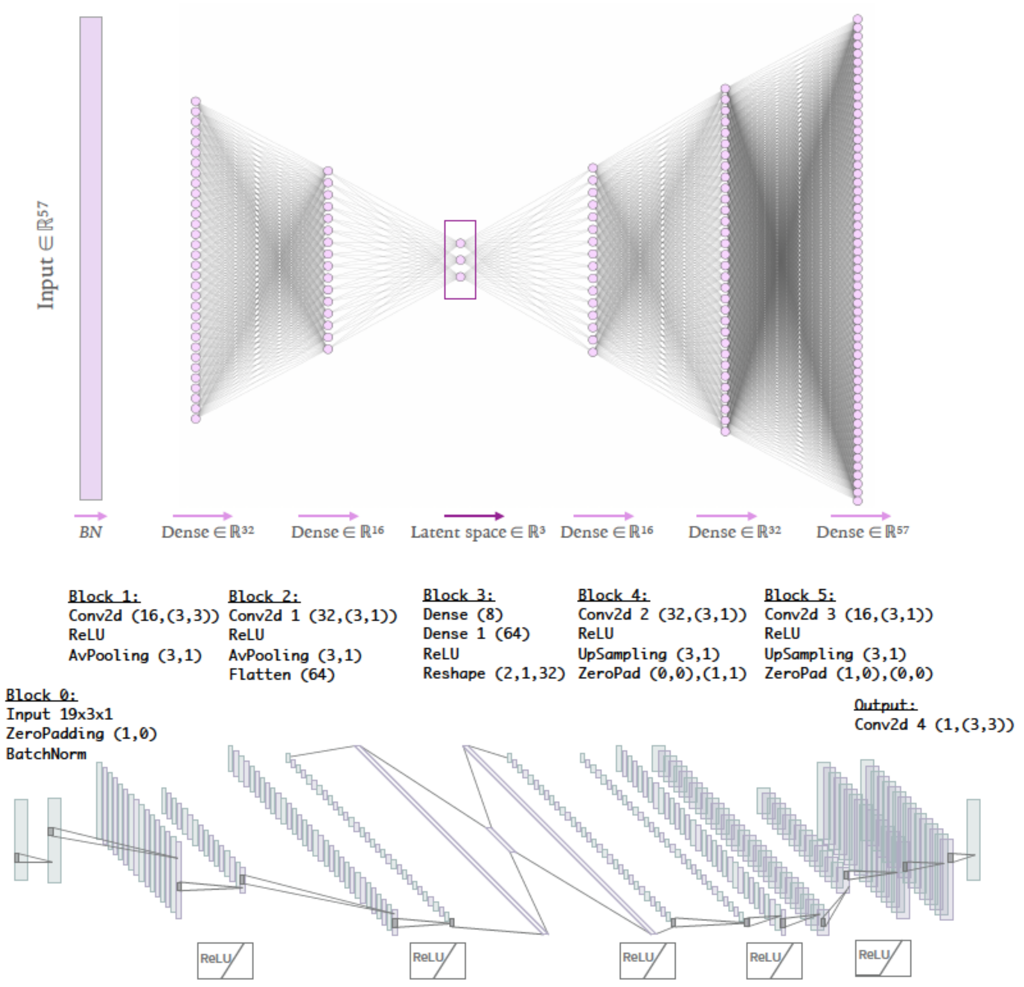

FIG. I. Network architecture for the DNN AE (top) and CNN AE (bottom) models. The corresponding VAE models are derived introducing the Gaussian sampling in the latent space, for the same encoder and decoder architectures (see text).

of the number of parallel processors. Since 19 is a prime number, we choose to extend the input size to 20 before passing it through the Conv2D layer. After padding, the input is scaled by a batch normalization layer and then processed by a stack of two CNN blocks, each including a 2D convolutional layer followed by a ReLU [55] activation function. The first layer has 16 3×3 kernels, without padding to ensure that p_T, η and ϕ inputs do not share weights. The second layer has 32 3×1 kernels. Both layers have no bias parameters and a stride set to one. The output of the second CNN block is flattened and passed to a DNN layer, with 8 neurons and no activation, which represents the latent space. The decoder takes this as input to a dense layer with 64 nodes and ReLU activation, and reshapes it into a $2 \times 1 \times 32$ table. The following architecture mirrors the encoder architecture with 2 CNN blocks with the same number of filters as in the encoder and with ReLU activation. Both are followed by an upsampling layer, in order to mimic the result of a transposed convolutional layer.

Finally, one convolutional layer with a single filter and no activation function is added. Its output is interpreted as the AE reconstructed input. The CNN VAE is derived from the AE, including the $\vec{\mu}$ and $\vec{\sigma}$ Gaussian sampling in the latent space.

All models are implemented in TensorFlow, and trained on the background dataset by minimizing a customized mean squared error (MSE) loss with the Adam [56] optimizer. In order to aid the network learning process, we use a dataset with standardized p_T as a target, so that all the quantities are $\mathcal{O}(1)$. To account for physical boundaries of η and ϕ, for those features a re-scaled `tanh` activation is used in the loss computation. In addition, the sum in the MSE loss is modified in order to ignore the zero-padding entries of the input dataset and the corresponding outputs. When training the VAE, the loss is changed to:

$$\mathcal{L} = (1-\beta)\mathrm{MSE}(\mathrm{Output}, \mathrm{Input}) + \beta \mathrm{D}_{\mathrm{KL}}(\vec{\mu}, \vec{\sigma}) \ , \quad (1)$$

where MSE labels the reconstruction loss (also used in the AE training), D_{KL} is the Kullback-Leibler regularization term [57] usually adopted for VAEs

$$\mathrm{D_{KL}}(\vec{\mu},\vec{\sigma}) = -\frac{1}{2}\sum_i \left(\log(\sigma_i^2) - \sigma_i^2 - \mu_i^2 + 1\right) \ , \qquad (2)$$

and β is a hyperparameter defined in the range $[0,1]$ [58].

Both models are trained for 100 epochs with a batch size of 1024, using early stopping if there is no improvement in the loss observed after ten epochs. All models are trained with floating point precision on an NVIDIA RTX2080 GPU. We refer to these as the baseline floating-point (BF) models.

IV. ANOMALY DETECTION SCORES

An autoencoder is optimized to retain the minimal set of information needed to reconstruct a accurate estimate of the input. During inference, an autoencoder might have problems generalizing to features it was not exposed to during training. Selecting events where the autoencoder output is far from the given input is often seen as an effective AD algorithm. For this purpose, one could use a metric that measures the distance between the input and the output. The simplest solution is to use the same metric that defines the training loss function. In our case, we use the MSE between the input and the output. We refer to this strategy as *input-output* (IO) AD.

In the case of a VAE deployed in the L1T, one cannot simply exploit an IO AD strategy since this would require sampling random numbers on the FPGA. The trigger decision would not be deterministic, something usually tolerated only for service triggers, and not for triggers serving physics studies. Moreover, one would have to store random numbers on the FPGA, which would consume resources and increase the latency. To deal with this problem, we consider an alternative strategy by defining an AD score based on the $\vec{\mu}$ and $\vec{\sigma}$ values returned by the encoder (see Eq. (1)). In particular, we consider two options: the KL divergence term entering the VAE loss (see Eq. (2)) and the z-score of the origin $\vec{0}$ in the latent space with respect to a Gaussian distribution centered at $\vec{\mu}$ with standard deviation $\vec{\sigma}$ [10]:

$$R_z = \sum_i \frac{\mu_i^2}{\sigma_i^2}. \qquad (3)$$

These two AD scores have several benefits we take advantage of: Gaussian sampling is avoided; we save significant resources and latency by not evaluating the decoder; and we do not need to buffer the input data for computation of the MSE. During the model optimization, we tune β so that we obtain (on the benchmark signal models) comparable performance for the $\mathrm{D_{KL}}$ AD score and the IO AD score of the VAE.

V. PERFORMANCE AT FLOATING-POINT PRECISION

The model performance is assessed using the four new physics benchmark models. The anomaly-detection scores considered in this paper are IO AD for the AE models, R_z and $\mathrm{D_{KL}}$ ADs for the VAE models. For completeness, results obtained from the IO AD score of the VAE models are also shown. The receiver operating characteristic (ROC) curves in Figures II and III show the true positive rate (TPR) as a function of the false positive rate (FPR), computed by changing the lower threshold applied on the different anomaly scores. We further quantify the AD performance quoting the area under the ROC curve (AUC) and the TPR corresponding to a FPR working point of 10^{-5} (see Table I), which on this dataset corresponds to the reduction of the background rate to approximately 1000 events per month.

From the ROC curves, we conclude that $\mathrm{D_{KL}}$ can be used as an anomaly metric for both the DNN and CNN VAE. This has the potential to significantly reduce the inference latency and on-chip resource consumption as only half of the network (the encoder) needs to be evaluated and that there no longer is a need to buffer the input in order to compute an MSE loss. The R_z metric performs worse and is therefore not included in the following studies.

VI. MODEL COMPRESSION

We adopt different strategies for model compression. First of all, we compress the BF model by pruning the dense and convolutional layers by 50% of their connections, following the same procedure as Ref. [19]. Pruning is enforced using the polynomial decay implemented in TensorFlow pruning API, a Keras-based [59] interface consisting of a simple drop-in replacement of Keras layers. A sparsity of 50% is targeted, meaning only 50% of the weights are retained in the pruned layers and the remaining ones are set to zero. The pruning is set to start from the fifth epoch of the training to ensure the model is closer to a stable minimum before removing weights deemed unimportant. By pruning the BF model layers to a target sparsity of 50%, the number of floating-point operations required when evaluating the model, can be significantly reduced. We refer to the resulting model as the baseline pruned (BP) model. For the VAE, only the encoder is pruned, since only that will be deployed on FPGA. The BP models are taken as a reference to evaluate the resource saving of the following compression strategies, including QAT and PTQ.

Furthermore, we perform a QAT of each model described in Section III, implementing them in the QKeras library [23]. The bit precision is scanned between 2 and 16 with a 2-bit step. When quantizing a model, we also impose a pruning of the dense (convolutional) layers by 50%, as done for the DNN (CNN) BP models. The results

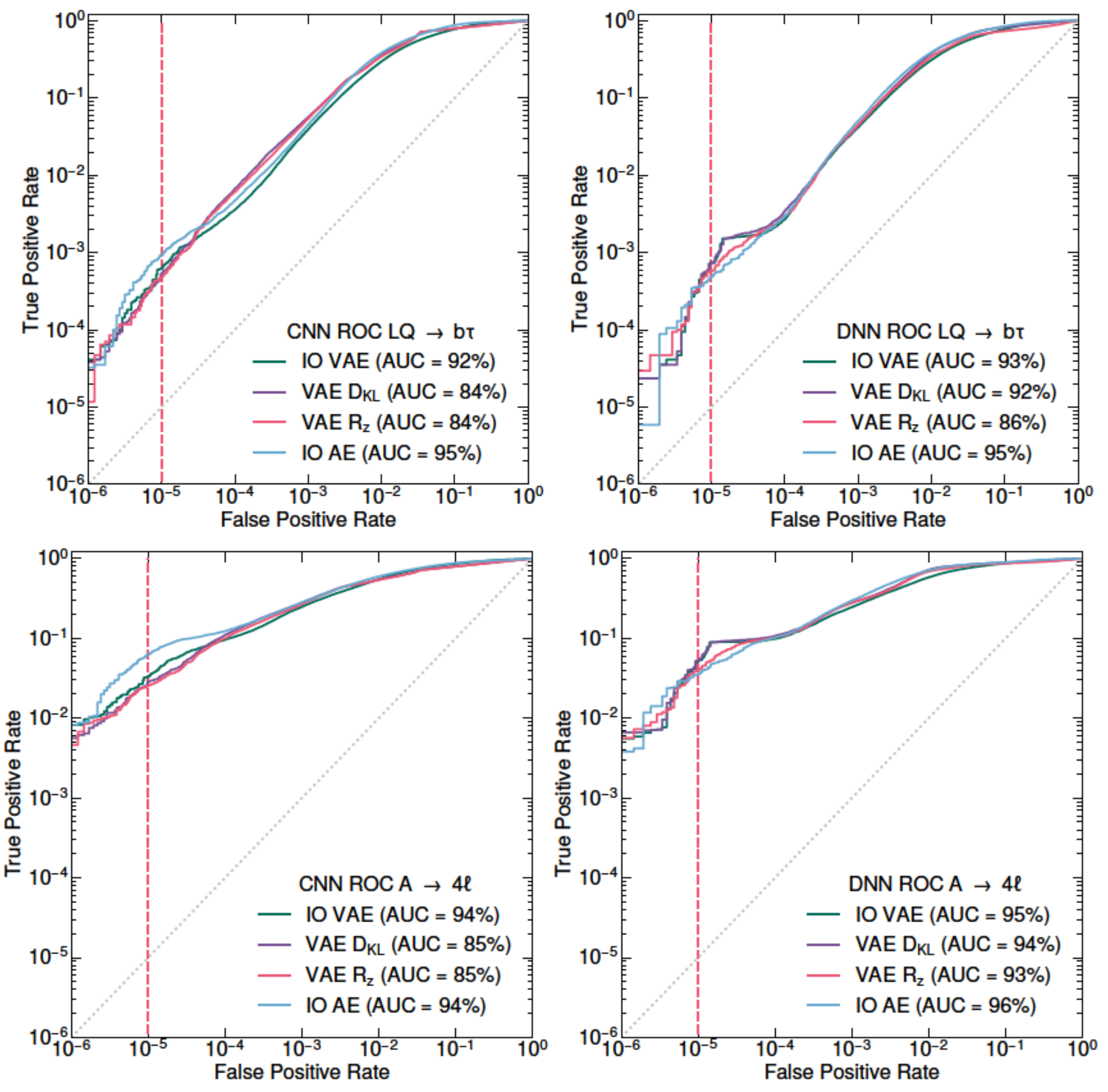

FIG. II. ROC curves of four AD scores (IO AD for AE and VAE models, R_z and D_{KL} ADs for the VAE models) for the CNN (left) and DNN (right) models, obtained from the two new physics benchmark models: $LQ \to b\tau$ (top) and $A \to 4\ell$ (bottom).

of QAT are compared to results obtained by applying a fixed-point precision to a BP floating-point model (i.e. using PTQ), using the same bit precision scan.

Performance of the quantized models, both for QAT and PTQ, is assessed using the TPR obtained for an FPR of 10^{-5} for the given precision. The bottom plots in Figures IV and V show ratios of QAT performance quantities obtained for each bit width with respect to the BP model performance of the AE and VAE, respectively. The top plots show ratios of PTQ performance quantities obtained in the same manner as for QAT.

Based on these ratio plots, the precision used for the final model is chosen. As expected, the performance of the VAEs is not stable as a function of bit width, since the AD figure of merit used for inference (D_{KL}) is different from those minimized during the QAT training (VAE IO). Therefore, we use PTQ compression for both DNN and CNN VAEs because they show stable results as a function of the bit width. For DNN and CNN VAE both PTQ and QAT show stable results, and therefore we choose QAT for AEs. For the QAT CNN, the QAT DNN AE and the PTQ DNN VAE a bit width of 8 is chosen, and for the PTQ CNN VAE a bit width of 4 is used. The performance numbers for the chosen models are summarized in Table II.

VII. PORTING THE ALGORITHM TO FPGAS

The models described above are translated into firmware using `hls4ml`, then synthesized with Vivado HLS 2020.1 [60], targeting a Xilinx Virtex UltraScale+ VU9P (`xcvu9p-flgb2104-2-e`) FPGA with a clock frequency of 200 MHz. In order to have fair resource and latency estimations, obtained from the HLS C Simulation we have implemented custom layers in `hls4ml`, which in

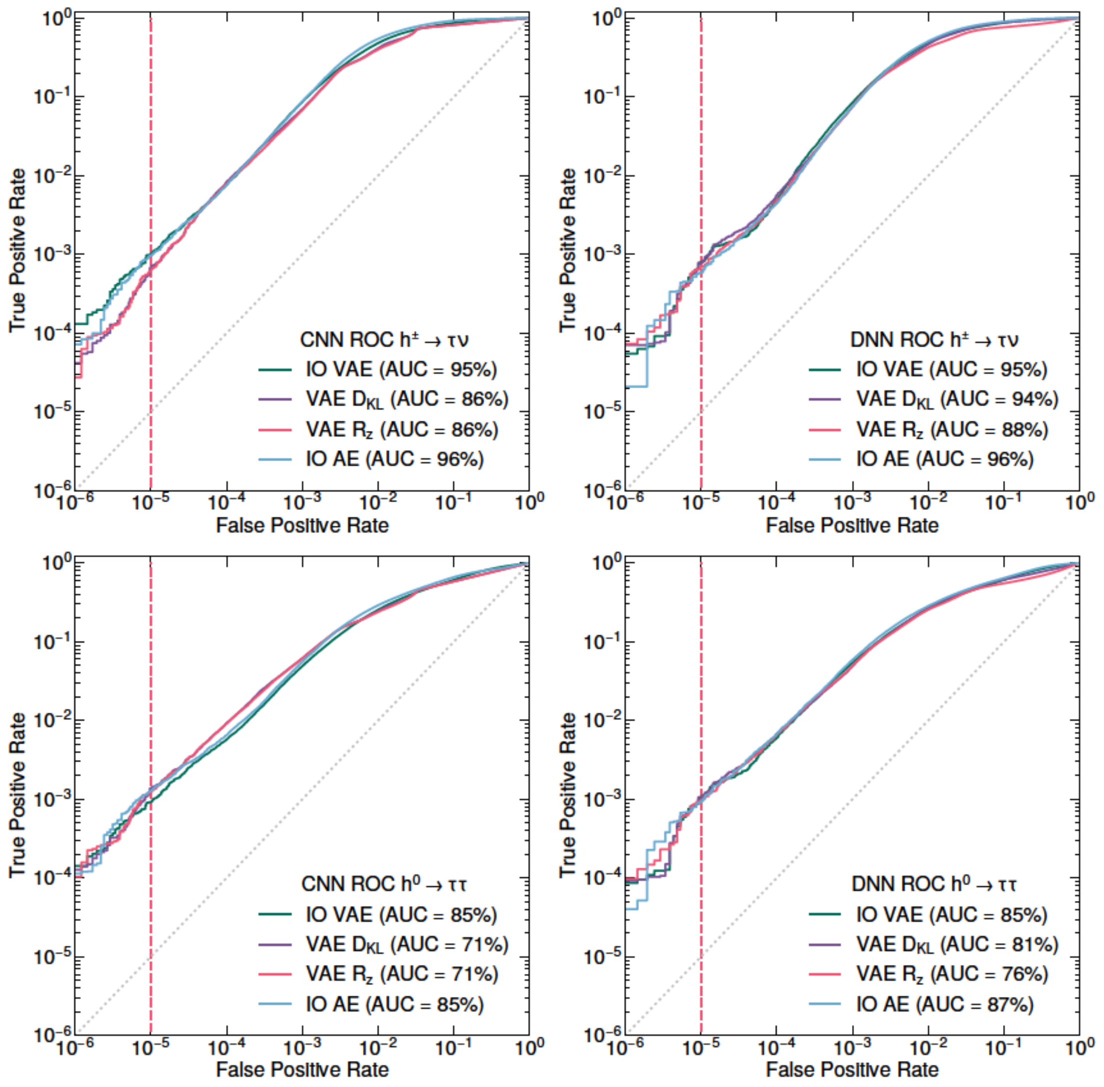

FIG. III. ROC curves of four AD scores (IO AD for AE and VAE models, R_z and D_{KL} ADs for the VAE models) for the CNN (left) and DNN (right) models, obtained from two new physics benchmark models: $h^{\pm} \to \tau\nu$ (top) and $h^0 \to \tau\tau$ (bottom).

the case of AE computes the loss function between the input and network output and for VAE computes the D_{KL} term of the loss.

A summary of the accuracy, resource consumption, and latency for the QAT DNN and CNN BP AE models, and the PTQ DNN and CNN BP VAE models is shown in Table III. Resource utilization is quoted as a fraction of the total available resources on the FPGA. We find the resources are less than about 12% of the available FPGA resources, except for the CNN AE, which uses up to 47% of the look-up tables (LUTs). Moreover, the latency is less than about 365 ns for all models except the CNN AE, which has a latency of 1480 ns. The II for all models is within the required 115 ns, again except the CNN AE. Based on these, both types of architectures with both types of autoencoders are suitable for application at the LHC L1T, except for the CNN AE, which consumes too much of the resources.

Since the performance of all the models under study are of a similar level, we choose the "best" model based on the smallest resource consumption, which turns out to be DNN VAE. This model was integrated into the **`emp-fwk`** infrastructure firmware for LHC trigger boards [61], targeting a Xilinx VCU118 development kit, with the same VU9P FPGA as previously discussed. Data were loaded into onboard buffers mimicking the manner in which data arrives from optical fibres in the L1T system. The design was operated at 240 MHz, and the model predictions observed at the output were consistent with those captured from the HLS C Simulation. For this model we also provide resource and latency estimates for a Xilinx Virtex 7 690 FPGA, which is the FPGA most widely used in the current CMS trigger. The estimates are given in Table IV.

TABLE I. Performance assessment of the CNN and DNN models, for different AD scores and different new physics benchmark scenarios.

Model	AD score	TPR @ FPR 10^{-5}[%]				AUC[%]			
		LQ $\rightarrow$ bτ	$A \rightarrow 4\ell$	$h^{\pm} \rightarrow \tau\nu$	$h^0 \rightarrow \tau\tau$	LQ $\rightarrow$ bτ	$A \rightarrow 4\ell$	$h^{\pm} \rightarrow \tau\nu$	$h^0 \rightarrow \tau\tau$
CNN VAE	IO	0.06	3.28	0.10	0.09	92	94	95	85
	D_{KL}	0.05	2.85	0.07	0.14	84	85	86	71
	R_z	0.05	2.53	0.06	0.12	84	85	86	71
CNN AE	IO	0.09	6.29	0.10	0.13	95	94	96	85
DNN VAE	IO	0.07	5.23	0.08	0.11	93	95	95	85
	D_{KL}	0.07	5.27	0.08	0.11	92	94	94	81
	R_z	0.06	4.05	0.07	0.10	86	93	88	76
DNN AE	IO	0.05	3.56	0.06	0.09	95	96	96	87

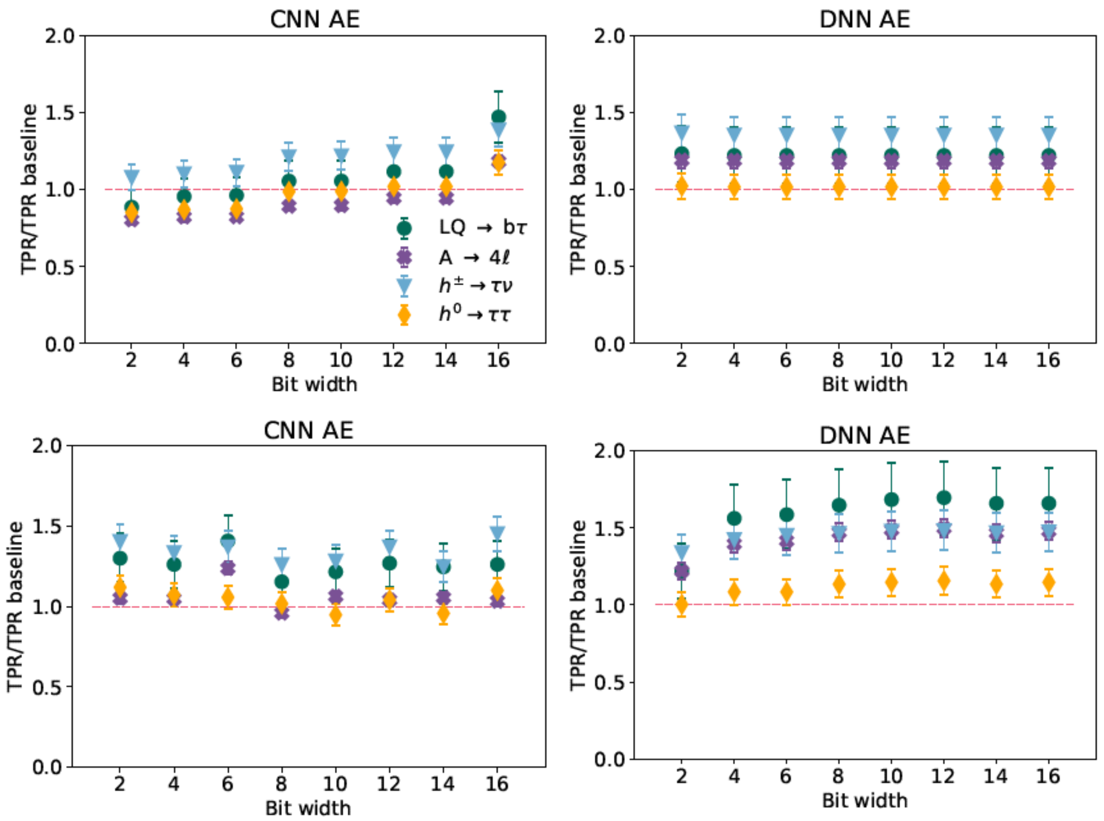

FIG. IV. TPR ratios versus model bit width for the AE CNN (left) and DNN (right) models tested on four new physics benchmark models, using mean squared error as figure of merit for PTQ (top) and QAT (bottom) strategies.

VIII. CONCLUSIONS

We discussed how to extend new physics detection strategies at the LHC with autoencoders deployed in the L1T infrastructure of the experiments. In particular, we show how one could deploy a deep neural network (DNN) or convolutional neural network (CNN) AE on a field-programmable gate array (FPGA) using the **hls4ml** library, within a $\mathcal{O}(1)\mu$s latency and with small resource utilization once the model is quantized and pruned. We show that one can retain accuracy by compressing the model at training time. Moreover, we discuss different strategies to identify potential anomalies. We show that one could perform the anomaly detection (AD) with a variational AE (VAE) using the projected representation of a given input in the latent space, which has several advantages for an FPGA implementation: (1) no need to sample Gaussian-distributed pseudorandom numbers (preserving the deterministic outcome of the trigger decision) and (2) no need to run the decoder in the trigger, resulting in a significant resource saving.

As can be seen from Table III, the latency when using only the encoder as opposed to full VAE is reduced by a factor of two, while the performance is of a similar level (see Table II). The DNN (V)AE models use less than 5% of

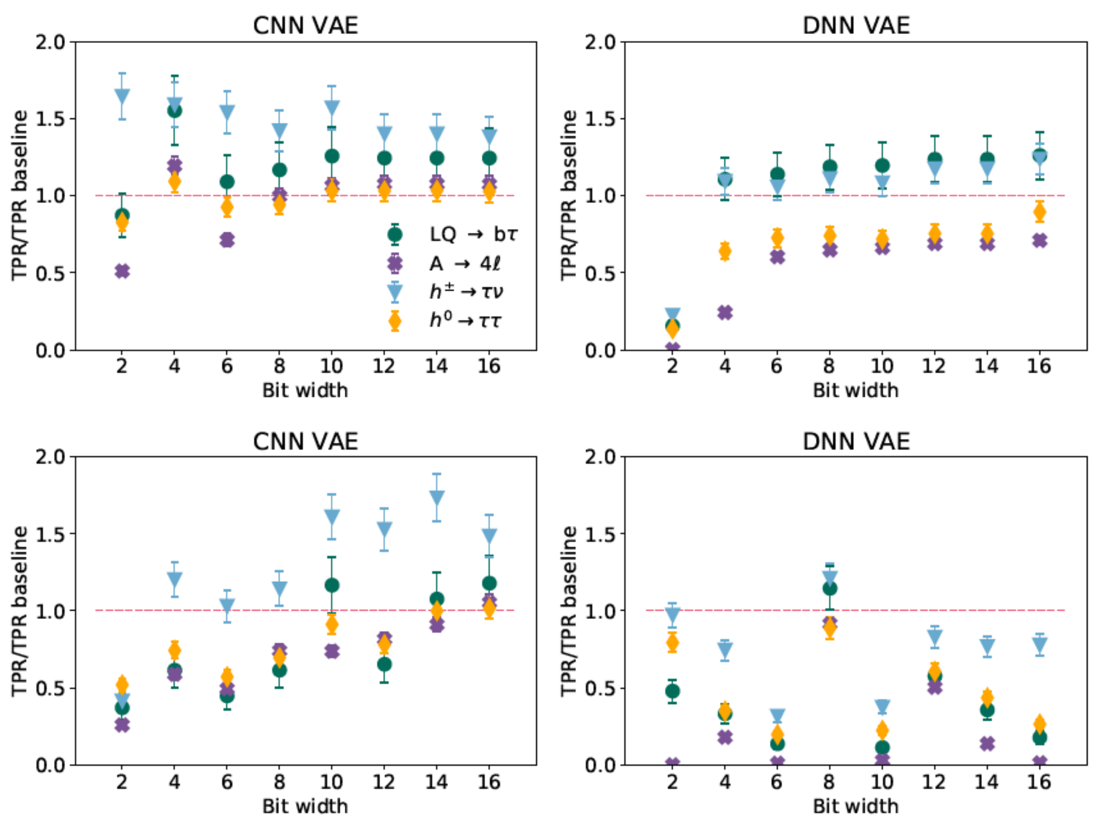

FIG. V. TPR ratios versus model bit width for the VAE CNN (left) and DNN (right) models tested on four new physics benchmark models, using D_{KL} as figure of merit for PTQ (top) and QAT (bottom) strategies.

TABLE II. Performance assessment of the quantized and pruned CNN and DNN models, for different AD scores and different new physics benchmark scenarios.

Model	AD score	TPR @ FPR 10^{-5}[%]				AUC[%]			
		LQ $\to b\tau$	$A \to 4\ell$	$h^{\pm} \to \tau\nu$	$h^0 \to \tau\tau$	LQ $\to b\tau$	$A \to 4\ell$	$h^{\pm} \to \tau\nu$	$h^0 \to \tau\tau$
CNN AE QAT 4 bits	IO	0.09	5.96	0.10	0.13	94	96	96	88
CNN VAE PTQ 8 bits	D_{KL}	0.05	2.56	0.06	0.12	84	84	85	71
DNN AE QAT 8 bits	IO	0.08	5.48	0.09	0.11	95	96	96	88
DNN VAE PTQ 8 bits	D_{KL}	0.08	3.41	0.09	0.08	92	94	94	81

the Xilinx VU9P resources and the corresponding latency is within 130 ns, while the CNN VAE uses less than 12% and the corresponding latency is 365 ns. All three models have the initiation interval within the strict limit imposed by the frequency of bunch crossing at the LHC. Between the two architectures under study, the DNN requires a few times less resources in the trigger, however both DNN and CNN fit the strict latency requirement and therefore both architectures can potentially be used at the LHC trigger. The CNN AE model is found to require more resources than are available.

With this work, we have identified and finalized the necessary ingredients to deploy (V)AEs in the L1T of the LHC experiments for Run 3 to accelerate the search for unexpected signatures of new physics.

IX. CODE AVAILABILITY

The QKeras library is available under **github.com/google/qkeras**, where the work presented here is using QKeras version 0.9.0. The **hls4ml** library with custom layers used in the paper are under **AE_L1_paper** branch and is available at **https://github.com/fastmachinelearning/hls4ml/tree/AE_L1_paper**.

X. DATA AVAILABILITY

The data used in this study are openly available at Zenodo at Ref. [47–50, 52].

TABLE III. Resource utilization and latency for the quantized and pruned DNN and CNN (V)AE models. Resources are based on the Vivado estimates from Vivado HLS 2020.1 for a clock period of 5 ns on Xilinx VU9P.

Model	DSP [%]	LUT [%]	FF [%]	BRAM [%]	Latency [ns]	II [ns]
DNN AE QAT 8 bits	2	5	1	0.5	130	5
CNN AE QAT 4 bits	8	47	5	6	1480	895
DNN VAE PTQ 8 bits	1	3	0.5	0.3	80	5
CNN VAE PTQ 8 bits	10	12	4	2	365	115

TABLE IV. Resource utilization and latency for the quantized and pruned DNN AE model. Resources are based on the Vivado estimates from Vivado HLS 2020.1 for a clock period of 5 ns on Xilinx V7-690.

Model	DSP [%]	LUT [%]	FF [%]	BRAM [%]	Latency [ns]	II [ns]
DNN VAE PTQ 8 bits	3	9	3	0.4	205	5

XI. AUTHOR INFORMATION

Correspondence and material requests can be e-mailed to E. Govorkova (katya.govorkova@cern.ch).

ACKNOWLEDGEMENTS

This work is supported by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation program (Grant Agreement No. 772369) and the ERC-POC programme (grant No. 996696).

[1] LHC Machine. *JINST* **3**, S08001 (2008).

[2] Aad, G. *et al.* The ATLAS Experiment at the CERN Large Hadron Collider. *JINST* **3**, S08003 (2008).

[3] Chatrchyan, S. *et al.* The CMS Experiment at the CERN LHC. *JINST* **3**, S08004 (2008).

[4] Sirunyan, A. M. *et al.* Performance of the CMS Level-1 trigger in proton-proton collisions at $\sqrt{s} = 13$ TeV. *J. Instrum.* **15**, P10017 (2020). 2006.10165.

[5] The Phase-2 upgrade of the CMS Level-1 trigger. CMS Technical Design Report CERN-LHCC-2020-004. CMS-TDR-021 (2020). URL `https://cds.cern.ch/record/2714892`.

[6] Aad, G. *et al.* Operation of the ATLAS trigger system in Run 2. *J. Instrum.* **15**, P10004 (2020). 2007.12539.

[7] Technical Design Report for the Phase-II Upgrade of the ATLAS TDAQ System. ATLAS Technical Design Report CERN-LHCC-2017-020. ATLAS-TDR-029 (2017). URL `https://cds.cern.ch/record/2285584`.

[8] Aad, G. *et al.* Observation of a new particle in the search for the standard model Higgs boson with the ATLAS detector at the LHC. *Phys. Lett. B* **716**, 1 (2012). 1207.7214.

[9] Chatrchyan, S. *et al.* Observation of a new boson at a mass of 125 GeV with the CMS experiment at the LHC. *Phys. Lett. B* **716**, 30 (2012). 1207.7235.

[10] Aarrestad, T. *et al.* The dark machines anomaly score challenge: Benchmark data and model independent event classification for the large hadron collider (2021). 2105.14027.

[11] Kasieczka, G. *et al.* The lhc olympics 2020: A community challenge for anomaly detection in high energy physics (2021). 2101.08320.

[12] Cerri, O. *et al.* Variational Autoencoders for New Physics Mining at the Large Hadron Collider. *JHEP* **05**, 036 (2019). 1811.10276.

[13] Knapp, O. *et al.* Adversarially Learned Anomaly Detection on CMS Open Data: re-discovering the top quark. *Eur. Phys. J. Plus* **136**, 236 (2021). 2005.01598.

[14] CMS Exotica hotline leads hunt for exotic particles (2010). URL `https://www.symmetrymagazine.org/breaking/2010/06/24/cms-exotica-hotline-leads-hunt-for-exotic-particles`.

[15] Poppi, F. Is the bell ringing?. Exotica : à l'affût des événements exotiques 14 (2010). URL `http://cds.cern.ch/record/1306501`.

[16] Duarte, J. *et al.* Fast inference of deep neural networks in FPGAs for particle physics. *JINST* **13**, P07027 (2018). 1804.06913.

[17] Ngadiuba, J. *et al.* Compressing deep neural networks on FPGAs to binary and ternary precision with `hls4ml`. *Mach. Learn.: Sci. Technol.* (2020). 2003.06308.

[18] Iiyama, Y. *et al.* Distance-Weighted Graph Neural Networks on FPGAs for Real-Time Particle Reconstruction in High Energy Physics. *Front. Big Data* **3**, 598927 (2020). 2008.03601.

[19] Aarrestad, T. *et al.* Fast convolutional neural networks on fpgas with `hls4ml`. *Mach. Learn.: Sci. Technol.* **2**, 045015 (2021). 2101.05108.

[20] Heintz, A. *et al.* Accelerated Charged Particle Tracking with Graph Neural Networks on FPGAs. In *34th Conference on Neural Information Processing Systems* (2020). 2012.01563.

[21] Summers, S. *et al.* Fast inference of Boosted Decision Trees in FPGAs for particle physics. *JINST* **15**, P05026 (2020). 2002.02534.

[22] Coelho, C. Qkeras (2019). URL `https://github.com/google/qkeras`.

[23] Coelho, C. N. *et al.* Automatic heterogeneous quantization of deep neural networks for low-latency inference on the edge for particle detectors. *Nat. Mach. Intell.* (2021).

2006.10159.

[24] Kingma, D. P. & Welling, M. Auto-encoding variational bayes (2014). 1312.6114.

[25] Rezende, D. J., Mohamed, S. & Wierstra, D. Stochastic backpropagation and approximate inference in deep generative models. In *ICML*, vol. 32, 1278 (2014). 1401.4082.

[26] LeCun, Y., Denker, J. S. & Solla, S. A. Optimal brain damage. In Touretzky, D. S. (ed.) *Advances in Neural Information Processing Systems*, vol. 2, 598 (Morgan-Kaufmann, 1990). URL `http://papers.nips.cc/paper/250-optimal-brain-damage`.

[27] Han, S., Mao, H. & Dally, W. J. Deep compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding. In Bengio, Y. & LeCun, Y. (eds.) *4th International Conference on Learning Representations, San Juan, Puerto Rico, May 2, 2016* (2016). 1510.00149.

[28] Blalock, D., Ortiz, J. J. G., Frankle, J. & Guttag, J. What is the state of neural network pruning? In Dhillon, I., Papailiopoulos, D. & Sze, V. (eds.) *Proceedings of Machine Learning and Systems*, vol. 2, 129 (2020). URL `https://proceedings.mlsys.org/paper/2020/file/d2ddea18f00665ce8623e36bd4e3c7c5-Paper.pdf`. 2003.03033.

[29] Moons, B., Goetschalckx, K., Berckelaer, N. V. & Verhelst, M. Minimum energy quantized neural networks. In Matthews, M. B. (ed.) *2017 51st Asilomar Conference on Signals, Systems, and Computers, Pacific Grove, CA, USA, October 29, 2017*, 1921 (2017). 1711.00215.

[30] Courbariaux, M., Bengio, Y. & David, J.-P. BinaryConnect: Training deep neural networks with binary weights during propagations. In Cortes, C., Lawrence, N. D., Lee, D. D., Sugiyama, M. & Garnett, R. (eds.) *Advances in Neural Information Processing Systems*, vol. 28, 3123 (Curran Associates, Inc., 2015). URL `https://proceedings.neurips.cc/paper/2015/file/3e15cc11f979ed25912dff5b0669f2cd-Paper.pdf`. 1511.00363.

[31] Zhang, D., Yang, J., Ye, D. & Hua, G. LQ-nets: Learned quantization for highly accurate and compact deep neural networks. In Ferrari, V., Hebert, M., Sminchisescu, C. & Weiss, Y. (eds.) *Proceedings of the European Conference on Computer Vision, Munich, Germany, September 8, 2018*, 373 (2018). 1807.10029.

[32] Hubara, I., Courbariaux, M., Soudry, D., El-Yaniv, R. & Bengio, Y. Quantized neural networks: Training neural networks with low precision weights and activations. *J. Mach. Learn. Res.* **18**, 1 (2018). URL `http://jmlr.org/papers/v18/16-456.html`. 1609.07061.

[33] Rastegari, M., Ordonez, V., Redmon, J. & Farhadi, A. XNOR-Net: ImageNet classification using binary convolutional neural networks. In *14th European Conference on Computer Vision (ECCV)*, 525 (Springer International Publishing, Cham, Switzerland, 2016). 1603.05279.

[34] Micikevicius, P. *et al.* Mixed precision training. In *6th International Conference on Learning Representations, Vancouver, BC, Canada, April 30, 2018* (2018). URL `https://openreview.net/forum?id=r1gs9JgRZ`. https://openreview.net/forum?id=r1gs9JgRZ, 1710.03740.

[35] Zhuang, B., Shen, C., Tan, M., Liu, L. & Reid, I. Towards effective low-bitwidth convolutional neural networks. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, June 18, 2018*, 7920 (2018). 1711.00205.

[36] Wang, N., Choi, J., Brand, D., Chen, C.-Y. & Gopalakrishnan, K. Training deep neural networks with 8-bit floating point numbers. In Bengio, S. *et al.* (eds.) *Advances in Neural Information Processing Systems*, vol. 31, 7675 (Curran Associates, Inc., 2018). URL `https://proceedings.neurips.cc/paper/2018/file/335d3d1cd7ef05ec77714a215134914c-Paper.pdf`. 1812.08011.

[37] An, J. & Cho, S. Variational autoencoder based anomaly detection using reconstruction probability. *Special Lecture on IE* **2**, 1 (2015).

[38] Nagel, M., van Baalen, M., Blankevoort, T. & Welling, M. Data-free quantization through weight equalization and bias correction. In *2019 IEEE/CVF International Conference on Computer Vision, Seoul, South Korea, October 27, 2019*, 1325 (2019). 1906.04721.

[39] Meller, E., Finkelstein, A., Almog, U. & Grobman, M. Same, same but different: Recovering neural network quantization error through weight factorization. In Chaudhuri, K. & Salakhutdinov, R. (eds.) *Proceedings of the 36th International Conference on Machine Learning, June 9, 2019, Long Beach, CA, USA*, vol. 97, 4486 (PMLR, 2019). URL `http://proceedings.mlr.press/v97/meller19a.html`. 1902.01917.

[40] Zhao, R., Hu, Y., Dotzel, J., Sa, C. D. & Zhang, Z. Improving neural network quantization without retraining using outlier channel splitting. In Chaudhuri, K. & Salakhutdinov, R. (eds.) *Proceedings of the 36th International Conference on Machine Learning, June 9, 2019, Long Beach, CA, USA*, vol. 97, 7543 (PMLR, 2019). URL `http://proceedings.mlr.press/v97/zhao19c.html`. 1901.09504.

[41] Banner, R., Nahshan, Y., Hoffer, E. & Soudry, D. Post-training 4-bit quantization of convolution networks for rapid-deployment. In Wallach, H. *et al.* (eds.) *Advances in Neural Information Processing Systems*, vol. 32, 7950 (Curran Associates, Inc., 2019). URL `https://proceedings.neurips.cc/paper/2019/file/c0a62e133894cdce435bcb4a5df1db2d-Paper.pdf`. 1810.05723.

[42] Shin, S., Boo, Y. & Sung, W. Knowledge distillation for optimization of quantized deep neural networks. In *2020 IEEE Workshop on Signal Processing Systems (SiPS)*, 1 (2020).

[43] Polino, A., Pascanu, R. & Alistarh, D. Model compression via distillation and quantization. In *International Conference on Learning Representations* (2018). URL `https://openreview.net/forum?id=S1XolQbRW`.

[44] Gao, M. *et al.* An embarrassingly simple approach for knowledge distillation (2019). 1812.01819.

[45] Mishra, A. & Marr, D. Apprentice: Using knowledge distillation techniques to improve low-precision network accuracy. In *International Conference on Learning Representations* (2018). URL `https://openreview.net/forum?id=B1ae1lZRb`.

[46] Nguyen, T. Q. *et al.* Topology classification with deep learning to improve real-time event selection at the LHC. *Comput. Softw. Big Sci.* **3**, 12 (2019). 1807.00083.

[47] Govorkova, E. *et al.* Unsupervised new physics detection at 40 mhz: LQ $\rightarrow$ b τ signal benchmark dataset. *Zenodo* `https://doi.org/10.5281/zenodo.5055454` (2021).

[48] Govorkova, E. *et al.* Unsupervised new physics detection at 40 mhz: A $\rightarrow$ 4 leptons signal benchmark dataset. *Zen-*

odo https://doi.org/10.5281/zenodo.5046446 (2021).

[49] Govorkova, E. *et al.* Unsupervised new physics detection at 40 mhz: $h^0 \to \tau\tau$ signal benchmark dataset. *Zenodo* https://doi.org/10.5281/zenodo.5061633 (2021).

[50] Govorkova, E. *et al.* Unsupervised new physics detection at 40 mhz: $h^+ \to \tau\nu$ signal benchmark dataset. *Zenodo* https://doi.org/10.5281/zenodo.5061688 (2021).

[51] Govorkova, E. *et al.* LHC physics dataset for unsupervised New Physics detection at 40 MHz (2021). 2107.02157.

[52] Govorkova, E. *et al.* Unsupervised new physics detection at 40 mhz: Training dataset. *Zenodo* https://doi.org/10.5281/zenodo.5046389 (2021).

[53] Ioffe, S. & Szegedy, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In Bach, F. & Blei, D. (eds.) *Proceedings of the 32nd International Conference on Machine Learning*, vol. 37, 448 (PMLR, 2015). URL http://proceedings.mlr.press/v37/ioffe15.html. 1502.03167.

[54] Maas, A. L., Hannun, A. Y. & Ng, A. Y. Rectifier nonlinearities improve neural network acoustic models. In *ICML Workshop on Deep Learning for Audio, Speech and Language Processing* (2013). URL https://sites.google.com/site/deeplearningicml2013/relu_hybrid_icml2013_final.pdf?attredirects=0&d=1.

[55] Nair, V. & Hinton, G. E. Rectified linear units improve restricted boltzmann machines. In Fürnkranz, J. & Joachims, T. (eds.) *ICML*, 807 (Omnipress, 2010). URL http://dblp.uni-trier.de/db/conf/icml/icml2010.html#NairH10.

[56] Kingma, D. P. & Ba, J. Adam: A Method for Stochastic Optimization. *ArXiv e-prints* (2014). 1412.6980.

[57] Joyce, J. M. *Kullback-Leibler Divergence*, 720–722 (Springer Berlin Heidelberg, 2011). URL https://doi.org/10.1007/978-3-642-04898-2_327.

[58] Higgins, I. *et al.* beta-vae: Learning basic visual concepts with a constrained variational framework (2016).

[59] Chollet, F. *et al.* Keras (2015). URL https://keras.io.

[60] Xilinx. Vivado design suite user guide: High-level synthesis (2020). URL https://www.xilinx.com/support/documentation/sw_manuals/xilinx2020_1/ug902-vivado-high-level-synthesis.pdf.

[61] EMP Collaboration. emp-fwk homepage (2019). URL https://serenity.web.cern.ch/serenity/emp-fwk/.