Autoencoders on field-programmable gate arrays 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, Sioni Summers, Jennifer Ngadiuba, Thong Q. Nguyen, Javier Duarte and Zhenbin Wu

To study the physics of fundamental particles and their interactions, the Large Hadron Collider was constructed at CERN, where protons collide to create new particles measured by detectors. Collisions occur at a frequency of 40 MHz, and with an event size of roughly 1 MB it is impossible to read out and store the generated amount of data from the detector and therefore a multi-tiered, real-time filtering system is required. In this paper, we show how to adapt and deploy deep-learning-based autoencoders for the unsupervised detection of new physics signatures in the challenging environment of a real-time event selection system at the Large Hadron Collider. The first-stage filter, implemented on custom electronics, decides within a few microseconds whether an event should be kept or discarded. At this stage, the rate is reduced from 40 MHz to about 100 kHz. We demonstrate the deployment of an unsupervised selection algorithm on this custom electronics, running in as little as 80 ns and enhancing the signal-over-background ratio by three orders of magnitude. This work enables the practical deployment of these networks during the next data-taking campaign of the Large Hadron Collider.

Proton–proton collision events occur 40 million times per second at the particle detectors at the CERN Large Hadron Collider (LHC). The largest general-purpose particle detectors at the LHC, ATLAS and CMS, discard most of the collision events with online selection systems, as a result of bandwidth limitations. These systems consist of two stages: the level-1 trigger (L1T), 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 event rate is reduced from 40 MHz to around 100 kHz within a few microseconds at the first selection stage, L1T. When designing searches for collisions containing new physics (for example, 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. 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. These investigate the use of autoencoders (AEs) and variational autoencoders (VAEs) for offline processing, and therefore do not consider constraints such as resource use and latency. Early suggestions to use AEs in HEP for anomaly detection are not easily adapted to an L1T environment. For instance, refs. require access to the momenta of all jet particle constituents, something that is not available now and will only be partly available (for example, first eight candidates) in the future. Refs. propose integrating unsupervised learning algorithms in the online selection system of the CMS and ATLAS experiments, in order to preserve rare events that would not otherwise be selected, in a special data stream.

While the primary focus for online unsupervised learning so far has been for the HLT, this strategy could be more effective if deployed in the L1T, that is, 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 a new-physics model agnostic algorithm that selects events based on their degree of abnormality, 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.

Many recent efforts for translating ML algorithms into FPGA firmware are reviewed extensively in refs. However, many of these toolflows result in implementations that are not optimized for the L1T systems or do not apply to HEP AE architectures. For example, FINN focuses on dataflow-style implementations of convolutional neural networks (CNNs), which may not achieve the low latency and high throughput required for L1T applications. It is by construction limited to Xilinx FPGAs, while hls4ml backend targeting different HLS libraries (Quartus for Intel and Katapult for ASIC design) are under development. Other efforts, Conifer (also developed by the hls4ml team) and fwXmachina, feature a custom implementation of boosted decision trees on FPGAs, which achieves the desired L1T constraints, but does not extend to neural network implementations.

European Organization for Nuclear Research (CERN), Geneva, Switzerland. Institute of Physics Belgrade, Belgrade, Serbia. Politecnico di Milano, Milan, Italy. Imperial College London, London, UK. Fermi National Accelerator Laboratory, Batavia, USA. California Institute of Technology, Pasadena, USA. University of California San Diego, La Jolla, USA. University of Illinois at Chicago, Chicago, USA. E-mail: katya.govorkova@cern.ch
Recent developments of the hls4ml library allow us to consider the possibility of deploying AE-based AD algorithms on the FPGAs mounted on the L1T boards. The hls4ml library is an open-source software, developed to translate neural networks into FPGA firmware. A fully on-chip implementation of the ML model is used in order to stay within the 1μs latency budget imposed by a typical L1T system. Additionally, the initiation interval (II) of the algorithm should be within 150ns, 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, hls4ml supports quantization-aware training (QAT), 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.

The aim of this work is the development of a fast algorithm to define a dataset enriched in anomalies, without using physics-motivated expectations about new physics to define the anomaly. Once collected, these data could be visually inspected or analysed with model-agnostic techniques, for example, those proposed in refs. 20–24, or even with traditional model-dependent searches (provided an understanding of the bias imposed by the online selection on the offline event distribution). We focus on AEs, with specific emphasis on VAES. We consider both fully connected and convolutional architectures, and discuss how to compress the model through pruning, the removal of unnecessary operations, and quantization, the reduction of the precision of operations.

As discussed in ref. 25, one can train (V)AEs 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, brings 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. 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)** consists 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.
- **QAT**, consists of imposing the fixed-point precision constraint at training time, for example, using the QKERAS or Brevitas libraries. 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.
- **Knowledge distillation (KD) with QAT** changes the quantized-model optimization strategy by reframing the problem as knowledge distillation.
- **Anomaly classification with QAT**; approximated loss regression with QAT could be turned into a classification problem.

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

### Data samples

This study follows the setup of refs. 16,56. The dataset (with its definition and limitations) is taken from ref. 16. We adapt the data format to make it more consistent with inputs received in the L1T (as opposed to the HLT) and show that one can do at L1T what ref. 16 proposed for the HLT. Perhaps surprisingly, this is indeed possible due to recent progress made on deploying neural networks on FPGAs. 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 $p_T > 23$ GeV and a pseudo-rapidity $|η| < 3$ (electron) and $|η| < 2.1$ (muon). These requirements were introduced to reduce the dataset size to a manageable level, such that we could generate it with our limited computing resources. In a real-life application, no $p_T$ requirement of this kind would be applied. The $η$ requirements would stay since they are intrinsic consequences of the detector geometry. In addition to the background-like sample, we consider the four benchmark new physics scenarios discussed in ref. 16:

- A leptoquark (LQ) with a mass of 80 GeV, decaying to a $b$ quark and a $r$ lepton $\rightarrow br$.
- 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: $A \rightarrow 4\ell^\pm$.
- 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: $h^0 \rightarrow \tau\nu$.

These four processes are used to evaluate the accuracy of the trained models. A detailed description of the dataset can be found in ref. 16. In total, the background sample consists of 8 million events. Of these, 50% are used for training, 40% for testing and 10% for validation.

### Autoencoder models

We consider two classes of architecture: one based on dense feed-forward neural networks (DNNs) and one using CNNs. Both start from the $(p_T, η, φ)$ values for 18 reconstructed objects (ordered as 4 muons, 4 electrons and 10 jets), 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 as the first layer for each model. 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 $\mu$ and the standard deviation $\sigma$ of the $N$-dimensional Gaussian distribution that represents the latent-space probability density function associated with a given event.

For the DNN model (shown on the top plot in Extended Data Fig. 1), all of the inputs are batch-normalized and passed through a stack of three 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. The decoder consists of a stack of three layers, with 16, 32 and 57 nodes. As for the encoder, we use a batch normalization layer between the fully connected layers and its activation. The last layer has no activation function, while leaky ReLU is used for the others.

The CNN AE architecture is shown on the bottom plot in Extended Data Fig. 1. The encoder takes as input the single-channel 2D array of three-vector including the two MET-related features.
ReLU6 activation function. The first layer has 16 3×3 CNN blocks, each including a 2D convolutional layer followed by a batch normalization layer and then processed by a stack of two it possible to explore CNN architectures. The input is scaled by a using image data, rather treating tabular data as a 2D image to make

Both are followed by an upsampling layer, in order to mimic the same number of filters as in the encoder and with ReLU activation.

The second layer has 32 3×3 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 eight 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 2×1×32 table. The following architecture mirrors the encoder architecture with two 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 and DNN VAEs are derived from the AEs, including the latent space. The decoder takes the input size of 19×3×1. It should be emphasized that we are not using image data, rather treating tabular data as a 2D image to make

All models are implemented in TENSORFLOW, and trained on the background dataset by minimizing a customized mean squared error (MSE) loss with the Adam optimizer. In order to aid the network learning process, we use a dataset with standardized parameters and a stride set to one. The output of the second CNN block is flattened and passed to a DNN layer, with eight 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 2×1×32 table. The following architecture mirrors the encoder architecture with two 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 and DNN VAEs are derived from the AEs, including the latent space. The decoder takes the input size of 19×3×1. It should be emphasized that we are not using image data, rather treating tabular data as a 2D image to make

Both models are trained for 100 epochs with a batch size of 1,024, 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.

![Fig. 1: Model performance at floating-point precision](image)

ROC curves of four AD scores (IO AD for AE and VAE models, R and DKL ADs for the VAE models) for the CNN (left) and DNN (right) models, obtained from the two new physics benchmark models: LQ → br (top) and A → 4f (bottom).

\[
\mathcal{L} = (1 - \beta)\text{MSE}(\text{Output}, \text{Input}) + \beta \mathcal{D}_{KL}(\mu, \sigma),
\]

where MSE labels the reconstruction loss (also used in the AE training), \(\mathcal{D}_{KL}\) is the Kullback–Leibler regularization term usually adopted for VAEs

\[
\mathcal{D}_{KL}(\mu, \sigma) = -\frac{1}{2} \sum_i \left( \log (\sigma_i^2) - \sigma_i^2 - \mu_i^2 + 1 \right),
\]

and \(\beta\) is a hyperparameter defined in the range [0, 1].
Anomaly detection scores
An autoencoder is optimized to retain the minimal set of information needed to reconstruct an accurate estimate of the input. During inference, an autoencoder might have problems generalizing to topologies 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. The simplest solution is to use the same metric that defines the training loss function. In our case, we use the modified 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. One could generate pseudo-random numbers exploiting meta information (for example, the event number) or symmetries in data (for example, the $\phi$ coordinate of one of the objects). This might imply a limitation on the dimensionality of the latent space, which might impact performance. Moreover, one would have to store random numbers on the FPGA, which would consume resources and increase the latency. We did not explore this possibility further. Instead, we consider an alternative strategy by defining an AD score based on the $\mu$ and $\sigma$ values returned by the encoder (see equation (1)). In particular, we consider two options: the KL divergence term entering the VAE loss (see equation (2)) and the $z$-score of the origin $\mu$ in the latent space with respect to a Gaussian distribution centred at $\mu$ with standard deviation $\sigma$ (ref. 1):

$$R_z = \sum \frac{\mu_i^2}{\sigma_i^2}. \quad (3)$$

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 $\beta$ so that we obtain (on the benchmark signal models) comparable performance for the DKL AD score and the IO AD score of the VAE. In practice, one should train the model using real data, which might contain a very small fraction of signal. Previous studies have verified that small rates of signal contamination have little effect on the training. One would use simulated signals in the same manner as in this paper to tune model parameters. Such a procedure would not bias the architecture choice towards specific signals, given the low dependence of the optimal $\beta$ value on the nature of the anomaly.

Performance at floating-point precision
The model performance is assessed using the four new physics benchmark models: $h^\tau \to \tau\nu$ (top) and $h^0 \to \tau\tau$ (bottom).

Fig. 2 | Model performance at floating-point precision. 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: $h^\tau \to \tau\nu$ (top) and $h^0 \to \tau\tau$ (bottom).
this paper are IO AD for the AE models, $R_i$ and $D_{KL}$ 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 Figs. 1 and 2 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 an FPR working point of $10^{-5}$ (see Table 1), which on this dataset corresponds to the reduction of the background rate to approximately 1,000 events per month.

Even if the VAE-$D_{KL}$ TPR is smaller than the corresponding full-precision model for certain benchmark signals, the TPR values are similar after pruning. So, we conclude that $D_{KL}$ can be used as an anomaly metric for the rest of this work. The $R_i$ metric performs worse and is therefore not included in the following studies.

**Model compression**

We compress the BF model by pruning the dense and convolutional layers by 50% of their connections, following the previously reported procedure. Pruning is enforced using the polynomial decay implemented in TENSORFLOW pruning API, a KERAS-based 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 ‘Autoencoder models’, implementing them in the QKERAS library. 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 of QAT are compared to results obtained by applying a fixed-point precision to a BP floating-point model (that is 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 Fig. 3 and Extended Data Fig. 2 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. 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 that minimized during the QAT training (VAE IO + $D_{KL}$). Therefore, we use PTQ compression for both DNN and CNN VAEs because they show stable results as a function of the bit width. For autoencoders, both quantization approaches show stable results, and therefore we choose quantization-aware training. For all the models a bit width of 8 is chosen, except from the CNN VAE for which a bit width of 4 is found to be the best. The performance numbers for the chosen models are summarized in Table 2.

**Porting the algorithm to FPGAs**

The models described above are translated into firmware using hls4ml, then synthesized with Vivado HLS 2020.1, targeting a Xilinx Virtex UltraScale+ VU9P (xcu9p-flgb2104-2-c) 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 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 3. 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 365ns for all models except the CNN AE, which has a latency of 1,480 ns. The II for all models is within the required 115ns, 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-fkw infrastructure firmware for LHC trigger boards, 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 arrive from optical fibres in the LIT 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.

---

**Table 1 | 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 $\rightarrow b\tau$ | $A \rightarrow A\ell$ | $h^i \rightarrow \tau\nu$ | $h^i \rightarrow \tau\tau$ | LQ $\rightarrow b\tau$ | $A \rightarrow A\ell$ | $h^i \rightarrow \tau\nu$ | $h^i \rightarrow \tau\tau$ |
| CNN VAE   | IO       | 0.09                    | 6.19    | 0.10    | 0.11    | 92       | 95       | 95       | 85       |
| $D_{KL}$  |          | 0.03                    | 1.63    | 0.08    | 0.09    | 93       | 93       | 93       | 92       |
| $R_i$     |          | 0.01                    | 0.48    | 0.04    | 0.04    | 93       | 93       | 93       | 92       |
| CNN AE    | IO       | 0.06                    | 3.89    | 0.08    | 0.09    | 96       | 97       | 96       | 88       |
| DNN VAE   | IO       | 0.08                    | 5.33    | 0.08    | 0.10    | 93       | 95       | 95       | 85       |
| $D_{KL}$  |          | 0.05                    | 3.78    | 0.08    | 0.10    | 93       | 95       | 94       | 84       |
| $R_i$     |          | 0.07                    | 4.90    | 0.07    | 0.10    | 85       | 91       | 87       | 74       |
| DNN AE    | IO       | 0.05                    | 3.47    | 0.06    | 0.09    | 95       | 96       | 96       | 88       

The best-performing autoencoder model for each anomaly is highlighted in bold.

---
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 or convolutional neural network AE on a FPGA using the hls4ml library, within a $O(1)$ μs 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 AD with a 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.

The DNN (V)AE models use less than 5% of the Xilinx VU9P resources and the corresponding latency is within 130ns, while the CNN VAE uses less than 12% and the corresponding latency is 365ns. All three models have the initiation interval within the strict limit imposed by the frequency of bunch crossing at the LHC. With this work, we have identified and finalized the necessary ingredients to deploy (V)AEs in the LIT of the LHC experiments for Run 3 to accelerate the search for unexpected signatures of new physics.

The aim is to use these algorithms in the trigger in order to create a catalogue of anomalous events that researchers could explore, for example, with clustering techniques. Furthermore, one could perform traditional data analysis, provided a (non-trivial) understanding of the effect of the trigger selection on the kinematic distribution. In presence of a good description of the loss distribution, the approach used in ref. 72 could be adopted.

### Data availability

The data used in this study are openly available at Zenodo57–60,62.

### Code availability

The QKeras library is available at 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 available at https://github.com/fastmachinelearning/hls4ml/tree/AE_L1_paper.

Received: 12 August 2021; Accepted: 6 January 2022; Published online: 23 February 2022

---

**Table 2** | 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 | LQ $\rightarrow b\tau$ | A $\rightarrow 4\ell$ | $h^0 \rightarrow \tau\tau$ |
|---------------------|----------|-----------|------------------------|----------------------|---------------------------|
| CNN VAE PTQ 8 bits  | D_{ril}  | 0.05      | 2.56                   | 0.06                 | 0.12                      |
| CNN VAE QAT 8 bits  | IO       | 0.08      | 5.48                   | 0.09                 | 0.11                      |
| DNN VAE QAT 8 bits  | IO       | 0.08      | 3.41                   | 0.09                 | 0.08                      |
| DNN VAE PTQ 8 bits  | D_{ril}  | 0.05      | 2.56                   | 0.06                 | 0.12                      |

**Fig. 3** | Compressed model performance. 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.
Table 3 | Resource utilization and latency for the quantized and pruned DNN and CNN (V)AE models

| Model             | Hardware  | DSP [%] | LUT [%] | FF [%] | BRAM [%] | Latency [ns] | II [ns] |
|-------------------|-----------|---------|---------|--------|----------|--------------|--------|
| DNN AE QAT 8 bits | Xilinx VU9P | 2       | 5       | 1      | 0.5      | 130          | 5      |
| CNN AE QAT 4 bits | Xilinx VU9P | 8       | 47      | 5      | 6        | 1,480        | 895    |
| CNN VAE PTQ 8 bits| Xilinx VU9P | 1       | 3       | 0.5    | 0.3      | 80           | 5      |
| CNN VAE PTQ 8 bits| Xilinx V7-690 | 3       | 9       | 3      | 0.4      | 205          | 5      |
| CNN VAE PTQ 8 bits| Xilinx VU9P | 10      | 12      | 4      | 2        | 365          | 115    |

Resources are based on the Vivado estimates from Vivado HLS 2020.1 for a clock period of 5ns on Xilinx VU9P. aFor the DNN VAE model, resources estimation is also provided based on Xilinx V7-690.

References
1. LHC Machine. JINST 3, S08001 (2008).
2. Aad, G. et al. The ATLAS Experiment at the CERN Large Hadron Collider. J. Instrum. 3, S08003 (2008).
3. Chatrchyan, S. et al. The CMS Experiment at the CERN LHC. J. Instrum. 3, S08004 (2008).
4. Sirunyan, A. M. et al. Performance of the CMS experiment at the CERN-LHCC-2017-020 TR-021 (2020).
5. Aad, G. et al. Operation of the ATLAS trigger system in Run 2. J. Instrum. 15, P10014 (2020).
6. Kasieczka, G. et al. The LHC olympics 2020: A community challenge for neural network inference. Preprint at https://arxiv.org/abs/2111.06141 (2021).
7. Rezende, D. J., Mohamed, S. & Wierstra, D. Stochastic backpropagation and approximate inference in deep generative models. Preprint at https://arxiv.org/abs/1401.4082 (2014).
8. Heimel, T., Kasieczka, G., Pühlmann, T. & Thompson, J. M. QCD or What? SciPost Phys. 6, 30 (2019).
9. Farina, M., Nakai, Y. & Shih, D. Searching for new physics with deep autoencoders. Phys. Rev. D 101, 075021 (2020).
10. Cerri, O. et al. Variational autoencoders for new physics mining at the Large Hadron Collider. J. High Energy Phys. 2019, 36 (2019).
11. Knapp, O. et al. Adversarially Learned Anomaly Detection on CMS Open Data: re-discovering the top quark. Eur. Phys. J. Plus 136, 236 (2021).
12. Venieris, S. I., Kouris, A. & Bougani, C.-S. Toolowls for mapping convolutional neural networks on FPGAs: A survey and future directions. Preprint at https://arxiv.org/abs/1803.05960 (2018).
13. Guo, K., Zeng, S., Yu, J., Wang, Y. & Yang, H. A survey of FPGA-based neural network inference accelerators. https://arxiv.org/abs/1712.08934 (2019).
14. Shawahna, A., Satt, S. M. & El-Maleh, A. FPGA-based accelerators of deep learning networks for learning and classification: a review. IEEE Access 7, 7823 (2019).
15. Umuroglu, Y. et al. FINN: A framework for fast, scalable binarized neural network inference. In Proc. 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays 65 (ACM, 2017).
16. Blott, M. et al. FINN-R: An end-to-end deep-learning framework for fast exploration of quantized neural networks. Preprint at https://arxiv.org/abs/1809.04570 (2018).
17. Summers, S. et al. Fast inference of boosted decision trees in FPGAs for particle physics. J. Instrum. 15, P05026 (2020).
18. Hong, T. M. et al. Nanosecond machine learning event classification with boosted decision trees in FPGA for high energy physics. J. Instrum. 16, P08016 (2021).
19. Duarte, J. et al. Fast inference of deep neural networks in FPGAs for particle physics. J. Instrum. 13, P07027 (2018).
20. Ngadiuba, J. et al. Compressing deep neural networks on FPGAs to binary and ternary precision with HIL3ML. Mach. Learn. Sci. Technol. 2, 2632 (2020).
21. 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).
22. Heintz, A. et al. Accelerated charged particle tracking with graph neural networks on FPGAs. In 34th Conference on Neural Information Processing Systems (2020).
23. Summers, S. et al. Fast inference of boosted decision trees in FPGAs for particle physics. J. Instrum. 15, P05026 (2020).
24. Coelho, C. Qkeras https://github.com/google/qkeras (2019).
25. Blalock, D., Ortiz, J. I. G., Frankle, J. & Guttag, J. What is the state of neural network pruning? In Proc. Machine Learning and Systems Vol. 2, 129 (2020).
26. Micikevicius, P. et al. Mixed precision training. In Adv. Neural Information Processing Systems (eds. Bengio, Y. & LeCun, Y) (2016).
27. Wang, N., Choi, J., Brand, D., Chen, C.-Y. & Gopalakrishnan, K. Training deep neural networks with pruning, trained quantization and Huffman coding. In 4th Int. Conf. Learning Representations (ed. Bengio, Y. & LeCun, Y) (2016).
49. Zhao, R., Hu, Y., Dotzel, J., Sa, C. D. & Zhang, Z. Improving neural network quantization without retraining using outlier channel splitting. In Proc. 36th Int. Conference on Machine Learning (eds. Chaudhuri, K. & Salakhutdinov, R.) Vol. 97, 7543 (PMLR, 2019).

50. Banner, R., Nahshan, Y., Hoffer, E. & Soudry, D. Post-training 4-bit quantization of convolution networks for rapid-deployment. In Adv. Neural Information Processing Systems (eds. Wallach, H. et al.) Vol. 32, 7950 (Curran Associates, 2019).

51. Pappalardo, A. brevitas https://github.com/Xilinx/brevitas (2020).

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

53. Polino, A., Pascanu, R. & Alistarh, D. Model compression via distillation. In Int. Conf. Learning Representations (2018).

54. Gao, M. et al. An embarrassingly simple approach for knowledge distillation. Preprint at https://arxiv.org/abs/1812.01819 (2019).

55. Mishra, A. & Marr, D. Apprentice: using knowledge distillation techniques to improve low-precision network accuracy. In Int. Conf. Learning Representations (2018).

56. 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).

57. Govorkova, E. et al. Unsupervised new physics detection at 40 MHz: B^+ -> b \tau signal benchmark dataset. Zenodo https://doi.org/10.5281/zenodo.5055454 (2021).

58. Govorkova, E. et al. Unsupervised new physics detection at 40 MHz: \Lambda^b \rightarrow 4 leptons signal benchmark dataset. Zenodo https://doi.org/10.5281/zenodo.5046446 (2021).

59. Govorkova, E. et al. Unsupervised new physics detection at 40 MHz: H^0 \rightarrow \tau\tau signal benchmark dataset. Zenodo https://doi.org/10.5281/zenodo.5061633 (2021).

60. Govorkova, E. et al. Unsupervised new physics detection at 40 MHz: H^0 \rightarrow \tau\tau signal benchmark dataset. Zenodo https://doi.org/10.5281/zenodo.5061688 (2021).

61. Govorkova, E. et al. LHC physics dataset for unsupervised new physics detection at 40 MHz. Preprint at https://arxiv.org/abs/2107.02157 (2021).

62. Govorkova, E. et al. Unsupervised new physics detection at 40 MHz: training dataset. Zenodo https://doi.org/10.5281/zenodo.5046389 (2021).

63. Ioffe, S. & Szegedy, C. Batch normalization: accelerating deep network training by reducing internal covariate shift. In Proc. 32nd International Conference on Machine Learning (eds. Bach, F. & Blei, D.) Vol. 37, 448 (PMLR, 2015).

64. 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).

65. Nair, V. & Hinton, G. E. Rectified linear units improve restricted boltzmann machines. In ICML (eds. Fürnkranz, J. & Joachims, T.) 807 (Omnipress, 2010).

66. Kingma, D. P. & Ba, J. Adam: A Method for Stochastic Optimization. Preprint at https://arxiv.org/abs/1412.6980 (2014).

67. Joyce, J. M. in International Encyclopedia of Statistical Science 720–722 (Springer, 2011); https://doi.org/10.1007/978-3-642-04898-2_327

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

69. Chollet, F. et al. Keras https://keras.io (2015).

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

71. Emp Collaboration. emp-fwk homepage. https://serenity.web.cern.ch/serenity/emp-fwk/ (2019).

72. D’Agnolo, R. T. & Wulzer, A. Learning new physics from a machine. Phys. Rev. D 99, 015014 (2019).

Acknowledgements

This work is supported by the European Research Council (ERC) under the European Union’s Horizon 2020 research and innovation programme (grant agreement no. 772369) and the ERC-POC programme (grant no. 996696).

Author contributions

V.L., M.P., A.A.P., N.G., M.G., S.S., J.D. and Z.W. conceived and designed the hls4ml software library. M.P., T.Q.N. and Z.W. designed and prepared the dataset format. E.G., E.P., T.A., T.J., V.L., M.P., J.N., T.Q.N. and Z.W. designed and implemented autoencoders and the ERC-POC programme (grant no. 996696).

Competing interests

The authors declare no competing interests.

Additional information

Extended data is available for this paper at https://doi.org/10.1038/s42256-022-00441-3. Correspondence and requests for materials should be addressed to Ekaterina Govorkova.

Peer review information Nature Machine Intelligence thanks the anonymous reviewers for their contribution to the peer review of this work.

Reprints and permissions information is available at www.nature.com/reprints.

Publisher’s note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

© The Author(s), under exclusive licence to Springer Nature Limited 2022, corrected publication 2022
Extended Data Fig. 1 | Network architectures. 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).
Extended Data Fig. 2 | TPR ratios for different bit width. TPR ratios versus model bit width for the VAE CNN (left) and DNN (right) models tested on four new physics benchmark models, using $D_k$ as figure of merit for PTQ (top) and QAT (bottom) strategies.