PROCEEDINGS OF SCIENCE

Joint Variational Auto-Encoder for Anomaly Detection in High Energy Physics

Lorenzo Valente,[b,*] **Luca Anzalone**[*,a,b] **Marco Lorusso**[a,b] **and Daniele Bonacorsi**[a,b]

[a] *INFN Sezione di Bologna,*
viale Berti Pichat 6/2, Bologna, Italy
[b] *Deparment of Physics and Astronomy (DIFA), University of Bologna,*
viale Berti Pichat 6/2, Bologna, Italy
E-mail: lorenzo.valente3@studio.unibo.it, luca.anzalone2@unibo.it, marco.lorusso11@unibo.it, daniele.bonacorsi@unibo.it

Despite providing invaluable data in the field of High Energy Physics, the LHC may encounter challenges in obtaining interesting results through conventional methods applied in previous run periods. Our proposed approach involves a Joint Variational Autoencoder (JointVAE) model, trained on known physics processes to identify anomalous events corresponding to previously unidentified physics signatures. By doing so, this method does not rely on any specific new physics signatures, and it can detect anomaly events in an unsupervised manner, complementing the traditional LHC search tactics relying on model-dependent hypothesis testing. This paper also presents a study on the implementation feasibility of the JointVAE model for real-time anomaly detection in general-purpose experiments at CERN LHC. Low latency and reduced resource consumption are among the challenges faced when implementing machine learning models in fast applications, such as the trigger system of the LHC experiments. Therefore, the JointVAE model has been studied for its implementation feasibility in Field-Programmable Gate Arrays (FPGAs), utilizing a tool based on High-Level Synthesis named HLS4ML. The code that supports this work is publicly available at https://github.com/LorenzoValente3/JointVAE4AD.

International Symposium on Grids and Clouds (ISGC) 2023,
19 - 31 March 2023
Academia Sinica Taipei, Taiwan

*Speaker
*Corresponding author

PoS(ISGC&HEPiX2023)014

1. Introduction

If new physics does exist at the scales investigated by the Large Hadron Collider, it is more elusive than expected [1, 2, 3]. Finding interesting results may be challenging using conventional methods without substantially increasing the number of analyses. Thus, standard signal-driven search strategies could fail in reaching new results, and unsupervised machine learning techniques could fill this critical gap. Such applications, running in the trigger system of the LHC experiments, could spot anomaly events that would otherwise go unnoticed, enhancing the LHC's scientific capabilities.

In unsupervised machine learning, the most basic technique is an Auto-Encoder (AE) with a bottleneck [4]. It is constructed using a network that translates a high-dimensional data representation onto itself to create an average or typical entity. Standard autoencoders are recommended for unsupervised jet classification, but they are known to have problems in more general applications. The AE learns to compress and rebuild the training data very well, but when new, untrained data is run through the trained AE, it will produce a considerable loss or reconstruction error. By using the AE, it is possible to search for data that differs significantly from training data or even training data that is a small subclass of anomalous instances. In addition, the AE fails if the anomalous data is structurally simpler than the dominant class because the AE can encode simpler data with fewer features more efficiently. It is possible to overcome the disadvantages of AE by substituting a different classification measure for the reconstruction error. A possible alternative approach to the reconstruction error in the case of Variational Autoencoders (VAE) [5] is to derive a metric from the latent space embedding.

To this end, we propose the use of the JointVAE model [6], capable of learning richer latent representations, along with an implementation feasibility study targeted at FPGA acceleration to achieve the lowest latency and resource consumption without compromising model accuracy. In particular:

- We assume an *unsupervised setting* in which the JointVAE learns to discriminate the anomalous top jets from the QCD ones, just by being trained on the background data: it is designed to capture both continuous and discrete latent variables, being able to compress high-dimensional image data to a much lower-latent representation.
- We compare the effectiveness of the *latent-based* anomaly detection scores, to determine which of them enhances class separation the most. Anomaly searches in the latent space can be particularly useful at the LHC, to reduce the amount of data to process thanks to learning compact representations that preserve the topology of the events.
- We provide insights about the challenges and trade-offs required to compress a convolutional auto-encoder model, targeting a deployment on an FPGA board. Compression has been done by *quantization* to reduce the model size, latency, and energy consumption [7]. For the FPGA implementation we adopt a tool based on High-Level Synthesis (HLS) named HLS4ML [8], capable of accelerating small CNNs architectures for real-time machine vision tasks [9].

We expect VAEs will find many uses in science, outperforming classical or standard deep learning baselines and even being able to solve challenges for physics beyond the Standard Model (BSM) that were previously unsolvable. Ideally applicable in a wide spectrum of signal background

discrimination through anomaly detection, this application is expected to produce excellent results in a variety of fields.

1.1 Background

Auto-Encoders. Auto-Encoders (AEs) are *self-supervised* models made of two neural networks: an *encoder* network, q_ϕ, that compresses an input x to a lower-dimensional latent representation z (also called the bottleneck), and a *decoder* network, p_θ, that tries to reconstruct x from the latent codes, z. During training, AEs are supervised by their inputs (thus the name self-supervised) to reconstruct them as closely as possible: $\hat{x} = p_\theta(q_\phi(x))$ such that $\hat{x} \approx x$. For this purpose, common loss functions are the *mean squared error* (MSE) or the *binary cross-entropy*[1] (BCE), both summed over pixel values and averaged over the batch dimension.

According to the dimensionality of the bottleneck, z, we distinguish among: *over-complete*, *complete*, and *under-complete* auto-encoders, respectively, when $|z|$ is larger, equal, or smaller than the dimensionality of the inputs. Furthermore, the original AE formalism can be extended to cover different data types, but also to ensure specific properties:

- **Convolutional Auto-Encoder (CAE)** [10]: able to handle rank-3 input tensors $H \times W \times D$, having a height H, width W, and depth D axes. The CAE makes use of convolutional layers: *down-sampling* occurs in the encoder (either by stridden convolutions or pooling), and *up-sampling* in the decoder (by transposed convolutions, or un-pooling.) This results in a *convolutional bottleneck*.

- **Sparse Auto-Encoder (SAE)** [11]: trained to produce sparse latent vectors. To ensure *sparsity*, i.e. various components are learned to be zero, an l_1-regularization (or Kullback-Leibler divergence, KLD) penalty on the bottleneck is added to the total loss function, i.e. $\mathcal{L}(\theta) = \mathcal{L}_{\text{reco}}(x, \hat{x}) + \lambda \|z\|_1$, where $\mathcal{L}_{\text{reco}}$ is the AE reconstruction loss, and λ is a (usually small) hyper-parameter controlling the strength of the penalty.

- **Denoising Auto-Encoder (DAE)** [12]: in case the inputs are corrupted from Gaussian noise, i.e. $\tilde{x} = x + \epsilon$ where $\epsilon \sim \mathcal{N}(0, 1)$, the DAE can recover the original, clean, input x by cancelling out the noise component, i.e. $\hat{x} \approx x$. This type of autoencoder is useful for data cleaning and denoising tasks, assuming an i.i.d. noise with zero-mean.

- **Variational Auto-Encoder (VAE)** [5]: is a probabilistic formulation of the AE through the variational Bayesian method. Our proposed model for anomaly detection builds on VAEs, which are discussed in more detail in section 3.1.

Anomaly Detection. The task of distinguishing *anomalies* or outliers that depart from the *normal* (i.e. regular) data distribution is called anomaly detection (AD) [13]. Anomalies are data points either corrupted (by malfunctioning sensors, or erroneous measurement or labelling), or rare, or interesting events. Anomaly detection is a fundamental problem in many fields, including finance, healthcare, and cybersecurity, as well as HEP.

Regarding HEP, the ideal data being the normality is represented by the set of *background* events (assuming no signal contamination), whereas the *signal(s)* is treated as anomalous. To identify the

[1] Note that using BCE as a loss function requires each pixel to take a value between zero and one.

signal, AEs are a suitable class of models to achieve that: the assumption is that the normal data is expected to be reconstructed with a *low error*, while the anomalous, rare, or corrupted events get a sufficiently higher prediction error. This setup is *self-supervised* in the sense that no class labels are required, and in particular, we make use of only the normal data (i.e. our QCD background; see section 2) to train our models with a reconstruction objective. In particular, autoencoders enable us to follow two complementary strategies to implement anomaly detection:

- **Reconstruction-based:** Anomaly scores are defined from distance measures in the pixel space, between the input image x and its reconstruction $\hat{x} = p_\theta(q_\phi(x))$. Examples are scores built from the MSE or BCE between x and $\hat{x}$.
- **Latent-based:** Scores for AD are defined by some notion of distance in the latent space, in particular, such measures are now defined over latent vectors, which can also represent learned probability distributions. In the context of VAEs and variants, a suitable approach is to treat the KLD between the learned distribution and its prior as an anomaly score.

Both strategies have their advantages and disadvantages. For example, reconstruction-based scores can be easier to interpret (e.g., via visual inspection of the reconstructions, or through image quality metrics) and define (e.g., a common loss function or metric can be readily employed as an anomaly score), but these are usually more expensive to compute since a forward pass of the *whole* model is required: to compute the score, we have to first encode the image, and then decode it from its latent vector. Conversely, latent-based anomaly scores only require a prediction of the encoder, being faster and less resource intensive, although more difficult to design (an analytical formula may be required) and interpret: even if lower dimensional than the input, the latent space can easily exceed the number of dimensions that can be directly visualized, in such case dimensionality reduction techniques like t-SNE [14] and UMAP [15, 16] can be quite helpful. In terms of discrimination power both kinds of scores can be equally effective, in particular, KLD-based scores also have a sound probabilistic interpretation.

2. Data Samples

For the following analysis, we utilized the same jet image representation used in a previous AE study [1], the QCD and top jet samples were generated for the top-tagging challenge described in the Reference [17]. The jets are generated using Pythia8 [18] with a centre-of-mass energy of 14 TeV, without considering pile-up and multi-parton interactions. A fast detector simulation is performed using Delphes [19]. The jets are defined using the anti-k_T algorithm [20, 21] in FastJet [22] with a radius of $R = 0.8$. In each event, only the leading jet with p_T ranging from 550 to 650 GeV and $|\eta| < 2$ is kept, and the top jets are required to be matched to a parton-level top within the jet radius and all parton-level decay products to be within the jet radius. The jet constituents are defined using the Delphes energy flow algorithm and the top 200 constituents from each jet are used for analysis. The empty entries are zero-padded. We do not include particle ID or tracking information in our analysis. For pre-processing, we follow a similar procedure as described in References [1, 23]. The pre-processing is done at the constituent level before pixelization. The jet is first centred using the k_T-weighted centroid, such that the major principal axis points upwards. Then, the image is flipped

in both axes so that the majority of p_T is located in the lower left quadrant. Next, the image is pixelized into a 40×40 array, where the intensity is defined by the sum of p_T of the constituents per pixel. The pixel sizes are $[\Delta\eta, \Delta\phi] = [0.029, 0.035]$ and the image is cropped during pixelization to reduce the sparsity of information. Figure 1 shows the average of the whole QCD and top jet image datasets after pre-processing. Our data consists of 200K QCD jets and 200K top jets, and the maximum number of jets possible is used. If equal numbers of QCD and top jets are required, 200K of each are used, and if the QCD jets are treated as the background, the total 200K QCD jets are kept, and the number of top jets is varied. All results presented in this paper use a 75/25 split for training and testing. The data samples are shuffled, and the testing data is selected randomly for each run.

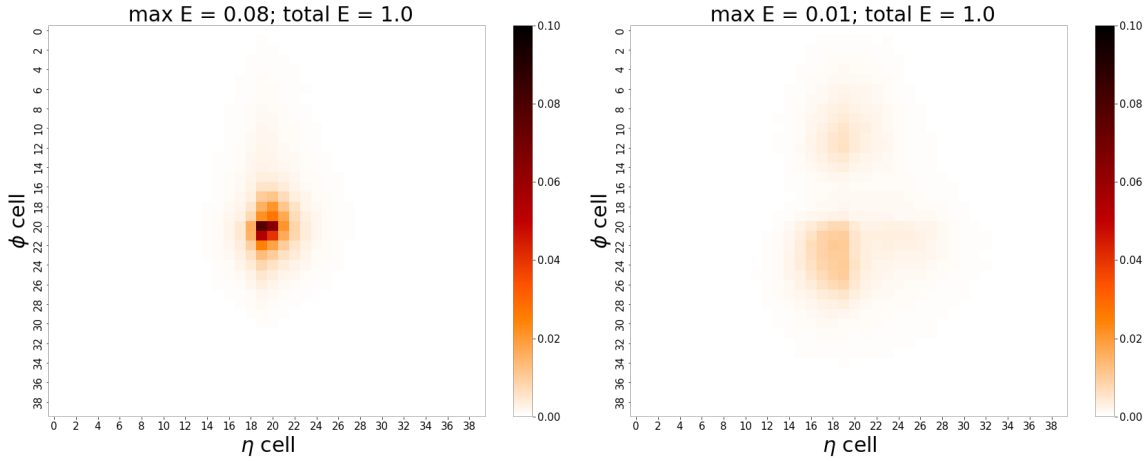

Figure 1: The averaged 200k pre-processed QCD and top jet images: each plot's title displays the brightest pixel value and the total sum of these values. (*Left*) Averaged on Calorimeter QCD jets and (*Right*) on Calorimeter tops jets.

3. Results with JointVAE

In this section we first introduce the model that we apply for anomaly detection, and then we present the discrimination results of our anomaly scores regarding the search of a top signal. Particular attention has been given to the JointVAE [6], which is a deep generative model that combines the strengths of Variational Autoencoders and discrete latent variable models. The JointVAE model can handle both continuous and discrete data variables, which is a key advantage over traditional VAEs that are limited to only continuous latent variables. The JointVAE model consists of two main components: an encoder and a decoder. The encoder maps the input data to a latent space representation and the decoder maps the latent space representation back to the original input data. The encoder outputs the parameters of two different distributions: one for continuous latent variables and one for discrete latent variables. The reparameterization trick is used to sample values from these distributions. The discrete variables are modelled using a discrete distribution, such as the Gumbel-Softmax [24], while the continuous variables are modelled using a Gaussian distribution. The JointVAE model is trained by maximizing the likelihood of the input data given

the latent representation. This is done by minimizing the difference between the generated data from the decoder and the original input data.

3.1 Variational Auto-Encoders

A variational auto-encoder (VAE) [5] can be seen as a probabilistic variant of an AE that aims to learn a compact and continuous latent representation of the data; now, both the encoder $q_\phi(z \mid x)$ and the decoder $p_\theta(x \mid z)$ are probabilistic models learned by amortized variational inference. The most common setup for a VAE is to assume the data can be described by a multivariate *Gaussian distribution*. In this regard, an isotropic Gaussian prior, $p(z) = \mathcal{N}(0, I)$, is usually assumed where I represents the identity matrix. Similarly, the encoder now outputs the means $\mu_\phi = q_\phi(x)$ and covariance matrix $\Sigma_\phi = q_\phi(x)$ that parameterize a Gaussian distribution, from which the latent vectors z are sampled from $z \sim \mathcal{N}(\mu_\phi, \Sigma_\phi)$. Moreover, to contain the growth of the number of free parameters of the learned distribution the covariance Σ_ϕ is usually restricted as $\Sigma_\phi = \sigma_\phi^2 I$, where σ_ϕ^2 is a learned scalar variance: in this way $|\sigma_\phi^2|$ will grow linearly, and not quadratically as a full Σ_ϕ. Compared to the AE, the VAE is said to learn a *smooth* latent space thanks to the sampling of z, occurring during training, which is then mapped to be as close as possible to the corresponding input sample x by the decoder: $z \sim q_\phi(x)$, then $\hat{x} = p_\theta(z)$. In addition, sampling from the prior distribution $p(z)$, allows the VAE to *generate* new data: $\tilde{x} = p_\theta(\tilde{z})$, where $\tilde{z} \sim p(z)$.

To learn the model parameters (ϕ, θ) the VAE is trained to minimize the *evidence lower-bound* (ELBO) plus a regularization term:

$$\mathcal{L}_{\phi,\theta}(x) = -\mathbb{E}_{z\sim q_\phi(x)}\big[log\, p_\theta(x \mid z)\big] + \beta D_{\mathrm{KL}}\big(q_\phi(z \mid x) \parallel p(z)\big). \tag{1}$$

The first term in the loss function encourages the decoder to learn to reconstruct the data accurately (by placing probability mass on the true data), while the second KLD term penalizes the learned latent space distribution to be diverse (or far) from the prior: this ensures learned representations, z, to be meaningful rather than just arbitrary. Moreover, the coefficient β allows controlling of such a term by increasing or decreasing the capacity of the latent codes, introducing a rate-distortion trade-off that can lead to disentangled representations [25].

As can be noted, the first term in Eq. 1 is about computing an expectation that involves the sampling of latent vectors. Such sampling operation is *stochastic*, meaning that we cannot back-propagate through it to compute the gradient of the loss. In order to solve the issue we have to rewrite the expression such that the sampling part is not directly involved in back-propagation, this is known as the *reparameterization trick*. To achieve this, we rewrite the latent, $z \sim \mathcal{N}(\mu_\phi(x), \Sigma_\phi(x))$, to be a deterministic function of the encoder plus some sampling *noise*; the reparametrized expression is the following: $z = \mu_\phi(x) + \Sigma_\phi(x) \odot \epsilon$, where $\epsilon \sim \mathcal{N}(0, I)$ is some Gaussian noise that is multiplied to add stochasticity to the expression. Although there is the sampling of ϵ involved, this stochastic operation is now outside the scope of back-propagation, and so the chain of derivatives would not break any more since there is no dependency of ϵ on the parameters (ϕ, θ) to be learned.

3.2 Categorical VAEs

A categorical variational auto-encoder [26, 24] is a variant of the VAE formalism able to learn *discrete* latent variables, instead of continuous ones. The core idea is to describe the latent space

via up to K Categorical distributions[2], each of them over C categories. In this way, the categorical VAE is able to encode discrete features, capturing finite and enumerable quantities like counts that cannot be learned by any continuous distribution.

In practice, the C categories are represented as C-dimensional one-hot encoded vectors, and so if we learn K distributions, the latent space z will have $K \times C$ components, being a rank-2 tensor: in our work, we found sufficient to learn a single categorical distribution ($K = 1$), so for each encoded sample i its latent vector has dimensionality $|z^{(i)}| = C$.

Actually, it is not possible to directly learn a categorical distribution: due to the fact that is discrete, the gradients would explode, resulting in non-differentiable behaviour. To enable learning, we instead learn a *relaxation* called the Gumbel-Softmax [24] (or the Concrete [26]) distribution: the degree of approximation is controlled by a *temperature* (hyper-)parameter, τ, allowing to interpolate (in the limit) between a uniform ($\tau \to \infty$) and a true categorical distribution ($\tau \to 0$). For ease of notation, we denote the Gumbel-Softmax distribution as $\mathrm{Cat}(\alpha, \tau)$, where α is a C-dimensional vector specifying unnormalized log-probabilities (or *logits*) for each category to happen and τ the temperature, instead, we use $Cat(\alpha)$ for regular categorical distribution.

Differently from the VAE, the encoder $\alpha_\phi = q_\phi(x)$ now outputs the logits of the Gumbel-Softmax, also the prior $p(z) = \mathrm{Cat}(\log 1/C)$ is assumed to be categorical with uniform probabilities over the C categories; these new elements can be directly plugged in Eq. 1 to train the Categorical VAE model. The last thing we need to account for is how to reparametrize the sampling of the latent from the relaxed categorical: this operation rewriting is known as the *Gumbel-Max trick* [26, 24]. So, the latent vectors are rewritten as follows:

$$z = \mathrm{softmax}\left(\frac{\alpha_\phi + g}{\tau}\right), \quad \text{where } g \sim \mathrm{Gumbel}(0, 1). \tag{2}$$

In detail, we first forward the encoder to get the logits, $\alpha_\phi = q_\phi(x)$, then we apply some *Gumbel noise* $g \sim \mathrm{Gumbel}(0, 1)$[3] to them, next we divide by the temperature τ parameter. Finally, the obtained logits are subject to a `softmax` operation that also represents a continuous approximation of the `max` operation (hence the name Gumbel-Max trick.)

3.3 Joint-VAE

For this analysis, it is used a modification of the standard VAE framework, it allowed us to model a joint distribution of the discrete latent variables jointly to the continuous ones pursued with the standard VAE. Letting z denote a set of continuous latent variables and c denote a set of categorical or discrete latent variables, we define a joint posterior $q_\phi(z, c|x)$, prior $p(z, c)$ and likelihood $p_\theta(x|z, c)$. Assuming the latent distribution is jointly continuous and discrete, and latent variables conditionally independent, we can rewrite the KL divergence term in the VAE loss Eq. 1 as:

$$D_{KL}\big(q_\phi(z, c \mid x) \parallel p(z, c)\big) = D_{KL}\big(q_\phi(z \mid x) \parallel p_(z)\big) + D_{KL}\big(q_\phi(c \mid x) \parallel p(c)\big), \tag{3}$$

[2]The categorical distribution generalizes the Bernoulli distribution over C categories instead of two.

[3]In practice, this is done through *inverse sampling*: a random sample from a Gumbel distribution is obtained by sampling from a uniform U one, then transforming the samples with a deterministic operation. To obtain $g \sim$ Gumbel(0,1) we do: $g = -\log(-\log u)$, where $u \sim U(0, 1)$.

i.e. we can separate the discrete and continuous KL divergence terms. Under this assumption, the JointVAE loss finally becomes

$$\mathcal{L}_{\theta,\phi}(x) = \mathbb{E}_{z,c\sim q_\phi(x)}\left[log p_\theta(x|z,c)\right] - \beta D_{KL}\big(q_\phi(z|x) \parallel p(z)\big) - \beta D_{KL}\big(q_\phi(c|x) \parallel p(c)\big), \quad (4)$$

where β is a positive constant. When $\beta > 1$ it is theorized that the combination of the increased force exerted by the joint posterior $q_\phi(z, c \mid x)$ aligns with the prior $p(z, c)$, combined with maximizing the likelihood term, leads to efficient and well-separated representations of the data [25].

3.4 Anomaly Scores and Results

In the context of JointVAE, several metrics can be used to evaluate the performance of the model in detecting anomalies. To ensure that the network can be synthesized with Vivado HLS and run on an FPGA, the number of filters in the network has to be reduced. For Vivado HLS 2020.1, the parameter size for synthesis is limited to 2048 per layer. Despite the reduction, the network is still able to detect anomalies with high accuracy. The section also notes that, given the limited space available on the FPGA, the decision is made to only synthesize the encoder model and exclude the decoder. However, during the training phase, the output of the decoder is crucial for properly training the network weights, as it is responsible for reconstructing the input data used to calculate the loss during training. By excluding the decoder during synthesis, only the KL divergence metric could be used for anomaly detection, which is effective at calculating anomaly detection scores. The best performances have been achieved for the metrics:

- **KL Continuous**: The KL divergence between the estimated continuous distribution $q_\phi(z \mid x)$ and the prior distribution $p(z)$. This metric is particularly useful for evaluating the performance of the model in capturing the continuous structure of the data.
- **KL Discrete**: similarly, the divergence between the estimated discrete distribution $q_\phi(c \mid x)$ and the prior distribution $p(c)$ is particularly useful for evaluating the performance of the model in capturing the discrete structure of the data.
- **KL Total**: The total KL divergence is the sum of the KL discrete and KL continuous divergences, which is a measure of the overall dissimilarity between the estimated and prior distributions.

These metrics can be used to identify instances that deviate significantly from the majority of the data, and thus are anomaly candidates.

Table 1: The best performance of the compete floating-point precision JointVAE encoder model, for the different AD scores used.

Model	**AD Scores**	**TPR@FPR** 10^{-1} [%]	**AUC** [%]
Complete	KL Discrete	52.74	73.63
Encoder	KL Continuous	58.10	86.46
	KL Total	58.34	86.63

The ROC curves depict the relationship between the true positive rate (TPR), representing the proportion of correctly identified anomalies, and the false positive rate (FPR), representing the

proportion of non-anomalous events incorrectly classified as anomalies, by adjusting the lower threshold for various anomaly scores. To assess the performance of AD, we measure the area under the ROC curve (AUC) and the TPR at an FPR of 10^{-1} (refer to Table 1). Looking at the obtained performance, we can conclude that the three KL divergences can be used as anomaly metrics for the rest of this work. The next sections provide a detailed description of the synthesis process, including the optimization techniques employed to achieve the best possible performance. Additionally, the results of the evaluation of the hardware implementation feasibility study are discussed, and a comparison is made with the software implementation.

3.5 Compression by Quantization

In this section we discuss the process of reducing the model footprint to optimize its computational efficiency, specifically when deploying the model on an FPGA board. To achieve this, the *encoder* described previously is drastically reduced in size and complexity by decreasing both the length and the number of filters; an additional convolutional layer (`conv_fin`) is added without calling the activation function to preserve the linearity and to help reduce the number of parameters. After this, the final flatten layer is applied, followed by the output of three dense layers for **mean**, **variance**, and **categorical** values, which correspond to the latent vectors. In addition, it is discussed the results after quantizing the model with the quantization-aware training (QAT) technique. In particular, it has been imposed a precision of `<16,6>` to all the convolutional, activation functions and the last final dense layers. We also anticipate that the quantized model will consume fewer resources compared to the floating-point precision model. Furthermore, we expect the quantized model to maintain a high level of accuracy in detecting anomalies. To evaluate the performance of the quantized model, we will use the same dataset and metrics in both the quantized and non-quantized models. Finally, we expect that the synthesized model will have a similar accuracy in detecting anomalies as the software implementation. However, we expect also that the FPGA hardware will be able to process the data at a much faster rate with lower resource consumption due to the hardware acceleration. We foresee these techniques to result in a faster and more resource-efficient implementation of the model on the FPGA accelerator.

Table 2: Performance assessment of the floating-point precision and quantized JointVAE, for different AD scores.

Model	**AD Scores**	**TPR@FPR** 10^{-1} [%]	**AUC** [%]
Reduced Encoder	KL Discrete	31.46	68.29
	KL Continuous	63.48	84.52
	KL Total	63.63	84.64
Quantized Encoder	KL Discrete	51.67	77.38
	KL Continuous	59.79	80.88
	KL Total	60.03	81.17

Table 2 presents the final AD performance for both the reduced encoder model and the quantized version. Although there is a slight decrease in the performance in the total KL AD scores compared to the complete model, as shown in Table 1, this can be attributed to the significant reduction in the model size, as explained earlier. Furthermore, the quantization resulted in a reduction in the total

KL AD performance, but an increase in the KL discrete performance. Nevertheless, a performance of approximately 81.2% for the AUC of the total KL AD score can still be considered satisfactory for our intended purposes.

Table 3: The energy consumption is estimated assuming a 45 nm process using `QTOOLS`. The quantized model has successfully achieved a reduction in energy consumption.

Model	**Total Energy** [μJ]
Reduced Encoder	5.4957
Quantized Encoder	3.3434

Table 3 indicates that the quantized model has achieved a reduction in total energy consumption by 39.2%. As a result, we anticipate that deploying the JointVAE model for anomaly detection, which has been quantized and implemented on an FPGA, will result in a highly accurate and efficient implementation. We will present these results in the next section.

4. Study of the FPGA implementation feasibility

Field Programmable Gate Arrays (FPGAs) [27] are specialized devices that can implement circuits with hardware-like efficiency, providing benefits in power consumption, size, and performance compared to software, making them well-suited for real-time data processing. Additionally, FPGAs can be easily reconfigured for various tasks without significant cost. To create an FPGA-based circuit, memory bits that determine routing decisions must be configured with appropriate values, achieved by generating a bitstream that loads into the device. FPGAs are used in the context of High Energy Physics (HEP), such as the CMS experiment, to process the large amounts of data generated by the detectors and increase the speed of the data processing, filtering and compressing of the data (such as using AEs and VAEs) from the detectors, reducing the amount of data that needs to be transferred over the network and stored on disk.

Traditionally, FPGAs are designed using hardware description languages such as VHDL or Verilog. However, this study adopted a "higher-level" approach by utilizing tools and libraries that allow FPGA design to be accomplished from a *behavioural description* written in C++ or Python for neural networks. The first step required for the implementation of a neural network on an FPGA is the conversion of the high-level code used for the creation of the model (Python + TensorFlow & QKeras) into high-level synthesis (HLS) code. To accomplish this task, the *hls4ml* package [28] has been used. This tool has been developed by members of the HEP community to translate ML algorithms, built using frameworks like TensorFlow2, into HLS code, enabling firmware development times to be drastically reduced. With its interface to QKeras, hls4ml supports quantization-aware training [7], which makes it possible to reduce the FPGA resource consumption while preserving accuracy drastically. Through the use of this relatively new technique, hls4ml can integrate resource-intensive models without sacrificing performance and resulting in efficient interference. In this case, a fixed numerical representation is employed for the entire model, and the model is trained with this constraint during weight optimization. Using hls4ml we can compress neural networks to fit the limited resources of an FPGA. In the context of the CMS experiment, hls4ml is used in the High-Level Trigger system to improve the event selection rate. A fully on-chip

implementation of the machine learning model is required to stay within the $1\mu s$ latency budget imposed by a typical L1T system. Since there are several L1T algorithms deployed per FPGA, each should use much less than the available resources. Compared to the previous software-based version, the ML algorithm hardware implementation can process data way faster. This allows for a higher event selection rate and improved overall performance of the HLT system.

Results obtained from the implementation feasibility study of the quantized JointVAE encoder model on an FPGA are presented in this section. The study focuses on the performance of the model when trained on data from the QCD and top jets datasets. During the conversion of the QKeras model to an HLS project, the model's quantization configuration is passed to hls4ml and enforced on the FPGA firmware. The firmware is designed to use specific arbitrary precision based on the quantization configuration to ensure that the same precision is maintained during inference. The precision of the model parameters is critical in preserving the accuracy of the JointVAE model. This procedure ensures that the use of specific arbitrary precision in the QKeras model is maintained during the inference [7]. To improve the implementation of the JointVAE encoder model, various strategies such as modifying the reuse factor and testing resource or latency strategies were investigated using hls4ml. However, the model still proves too complex for low-level implementation. This can be attributed to the computational intensity of the model's multidimensional convolutional layers, which cannot be easily simplified for implementation in hardware algorithms. The neural network architecture model used for synthesis required significant computing resources on the development machine used to compile the firmware.

Table 4 provides a comprehensive overview of the expected resource of the FPGA needed to implement the different layers in the quantized JointVAE encoder model. To examine the resource utilization of the model in detail, we split the blocks and analyzed the resource usage for each layer. Specifically, for each `dconv` block, we considered a `QConv2D`, an instance normalization layer and `QActivation`. For instance, `conv2_b1_1` represents the first `QConv2D` layer, instance normalization layer, and activation block with an input and output shape of $(10, 10, 20)$ and $(5, 5, 4)$ respectively. A block similar to the previous one named `conv2_b2`, but ingesting an image with dimensions $(5, 5, 4)$ and generating an output with the same size. Moreover, the final block in the quantized JointVAE encoder model comprises `conv_fin`, which is a `QConv2D` layer with instance normalization but no activation, and three vectors of `QDense`, which provide the final output of the encoder and the vector defining the latent space of the JointVAE. The quantized JointVAE encoder is a complex neural network that requires careful optimization for efficient implementation. To this end, resource utilization metrics play a critical role in evaluating the performance of the model and optimizing its implementation. In particular, the number of DSPs, LUTs, FFs, and BRAM used by each layer in the model provides valuable insights into the resource requirements of the model and helps identify potential performance bottlenecks that need to be addressed. It is worth noting that in the actual model intended to be implemented on the FPGA, the blocks present in Table 4 are present multiple times with different and bigger inputs and outputs. This architectural choice is made for better encoding computation, as previously described. It is important to note that the first `conv2_b0_0` layer in the table, which takes an input of $(20, 20, 16)$ and has an output shape of $(10, 10, 20)$, was impossible to simulate with Vivado HLS due to its large shape in convolutional computation. Assuming that the resource consumption would scale proportionally with the input size, the resource usage of this layer was estimated based on a linear relationship

with the computational complexity of the convolutional layer as described in the Reference [29]. However, it is important to note that this estimate is only an approximation and may not precisely reflect the actual resource demands of the layer. In this configuration, as we can see for the layer `conv2_b0_0` we have LUT consumption of 6189696, which exceeds more than three times and a half the total resource available on the FPGA target, which demonstrates the impossibility of the implementation in a single FPGA for this configuration.

Table 4: Resource utilization estimates and latency for some of the quantized JointVAE encoder layers. Resources are based on the Vivado estimates from Vivado HLS 2020.1 for a target clock period of 5 ns on the Alveo U250. The second column represents the variables used for each layer to estimate resource usage. The percentage is computed against the total available resources on the FPGA target board.

Layer Name	$\mathbf{h_{in}}$, $\mathbf{w_{in}}$, $\mathbf{d_{in}}$, $\mathbf{d_{out}}$	**DSP** (%)	**LUT** (%)	**FF** (%)	**BRAM** (%)
conv2_b0_0	20, 20, 16, 10	*752* (6)	*6189696 (358)*	*229536* (~7)	*288* (5)
conv2_b1_1	10, 10, 20, 5	47 (~0)	386856 (22)	14346 (~0)	18 (~0)
dconv_b2	5, 5, 4, 4	15 (~0)	309544 (17)	9352 (~0)	4 (~0)
final block	5, 5, 4, 2	851 (6)	91763 (5)	28074 (~0)	377 (7)

Overall, the resource utilization metrics and analyses presented in this study demonstrate that the quantized JointVAE encoder model is too large for efficient implementation on the FPGA. The implementation of the network on an FPGA would have allowed for hardware acceleration and parallel processing, resulting in a significant reduction in the inference time compared to the software implementation. This would have been particularly beneficial for real-time applications where low latency is critical. However, due to hardware constraints and design complexities, we were only able to partially implement the network on the FPGA, further research and optimization efforts are needed to compress and optimize the model for a more efficient implementation.

5. Conclusions and Outlook

We propose the use of convolutional neural network joint variational autoencoders (JointVAE) trained on a reference standard model (SM) sample to identify top decay jets, but the same technique can be used for potential beyond standard model (BSM) events at the LHC. This approach can detect recurrent anomalies that may be missed by traditional trigger selection, and it can be applied in general-purpose LHC experiments. Our algorithm has the potential to select datasets enriched with events from challenging BSM models, in a similar approach as demonstrated for top jets samples. We discussed a strategy for detecting potential anomalies, including using a JointVAE for anomaly detection (AD) by projecting the input's representation in the latent space. The final outcome of this application is a list of anomalous scores that experimental collaborations could scrutinize further. The purpose of this application is not to enhance the signal selection efficiency for BSM models but to provide a high-purity sample of potentially interesting events. Repeated patterns in these events could motivate new scenarios beyond the standard model physics and inspire new searches to be performed on future data with traditional supervised approaches. Our proposed approach is general and not sensitive to a particular BSM scenario. While supervised algorithms could give better discrimination capability for a given BSM hypothesis, they would not generalize to other BSM

scenarios. The JointVAE, on the other hand, comes with little model dependence and therefore generalizes, in principle, to unforeseen BSM models. As typical of autoencoders used for AD, our JointVAE model is trained to learn the SM background, but there is no guarantee that the best SM-learning model will be the best anomaly detection algorithm.

During our discussion, we also explored methods to enhance the detection of new physics at the LHC by utilizing the model within the L1T infrastructure of the experiments. We proposed deploying a JointVAE on an FPGA using the hls4ml library. By quantizing the model, we achieved a lower energy consumption with respect to the floating point complete model. It is important to note that the model quantization configuration plays a significant role in ensuring the accuracy of the inference results. This is a crucial aspect since the FPGA has limited resources, and using floating-point arithmetic would result in high resource usage and longer latencies. The strategy we propose can help extend the physics reach of the current and next stages of the CERN LHC, and the strategy demonstrated for the data stream coming from a CMS L1 selection can be generalized to any other data stream from any L1 selection, allowing scrutiny of the full 100 Hz rate entering the HLT system of ATLAS or CMS. While our implementation of the JointVAE model is a novel advancement in the field, further upgrades still need to be made. We found that the computational cost of this large network is significant and requires substantial resources, making it difficult to fully implement it on an accelerator board. One possible solution could be to employ *knowledge distillation* [30], whereby a smaller student network learns from the larger one by minimizing the differences between their outputs. Additionally, we observed that in an FPGA environment where multiple algorithms are running, it might not be feasible for the JointVAE to utilize all available resources for optimal performance. Nonetheless, the JointVAE has the potential to be a valuable tool in L1 trigger systems, enabling more efficient and accurate detection of anomalous signals that could signify new physics signatures.

PoS(ISGC&HEPiX2023)014

References

[1] Theo Heimel et al. "QCD or What?" In: *SciPost Physics* 6.3 (2019), p. 030.

[2] Barry Dillon et al. "Better latent spaces for better autoencoders". In: *SciPost Physics* 11.3 (2021), p. 061.

[3] Ekaterina Govorkova et al. "Autoencoders on field-programmable gate arrays for real-time, unsupervised new physics detection at 40 MHz at the Large Hadron Collider". In: *Nature Machine Intelligence* 4.2 (2022), pp. 154–161.

[4] Thorben Finke et al. "Autoencoders for unsupervised anomaly detection in high energy physics". In: *Journal of High Energy Physics* 2021.6 (2021), pp. 1–32.

[5] Diederik P Kingma and Max Welling. "Auto-encoding variational bayes". In: *arXiv preprint arXiv:1312.6114* (2013).

[6] Emilien Dupont. "Learning disentangled joint continuous and discrete representations". In: *Advances in Neural Information Processing Systems* 31 (2018).

[7] Claudionor N Coelho Jr et al. "Automatic heterogeneous quantization of deep neural networks for low-latency inference on the edge for particle detectors". In: *Nature Machine Intelligence* 3.8 (2021), pp. 675–686.

[8] FastML Team. *fastmachinelearning/hls4ml*. 2021. DOI: 10.5281/zenodo.1201549. URL: https://github.com/fastmachinelearning/hls4ml.

[9] Thea Aarrestad et al. "Fast convolutional neural networks on FPGAs with hls4ml". In: *Machine Learning: Science and Technology* 2.4 (2021), p. 045015.

[10] Jonathan Masci et al. "Stacked convolutional auto-encoders for hierarchical feature extraction". In: *Artificial Neural Networks and Machine Learning–ICANN 2011: 21st International Conference on Artificial Neural Networks, Espoo, Finland, June 14-17, 2011, Proceedings, Part I 21*. Springer. 2011, pp. 52–59.

[11] Andrew Ng et al. "Sparse autoencoder". In: *CS294A Lecture notes* 72.2011 (2011), pp. 1–19.

[12] Pascal Vincent et al. "Extracting and composing robust features with denoising autoencoders". In: *Machine Learning, Proceedings of the Twenty-Fifth International Conference (ICML 2008), Helsinki, Finland, June 5-9, 2008*. Ed. by William W. Cohen, Andrew McCallum, and Sam T. Roweis. Vol. 307. ACM International Conference Proceeding Series. ACM, 2008, pp. 1096–1103. DOI: 10.1145/1390156.1390294. URL: https://doi.org/10.1145/1390156.1390294.

[13] Raghavendra Chalapathy and Sanjay Chawla. "Deep learning for anomaly detection: A survey". In: *arXiv preprint arXiv:1901.03407* (2019).

[14] Laurens Van der Maaten and Geoffrey Hinton. "Visualizing data using t-SNE." In: *Journal of machine learning research* 9.11 (2008).

[15] Leland McInnes et al. "UMAP: Uniform Manifold Approximation and Projection". In: *The Journal of Open Source Software* 3.29 (2018), p. 861.

PoS(ISGC&HEPiX2023)014

[16] Leland McInnes et al. "UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction". In: *ArXiv e-prints* (Feb. 2018). arXiv: 1802.03426 [stat.ML].

[17] Gregor Kasieczka et al. "The machine learning landscape of top taggers". In: *SciPost Physics* 7.1 (2019), p. 014.

[18] Torbjörn Sjöstrand et al. "An introduction to PYTHIA 8.2". In: *Computer physics communications* 191 (2015), pp. 159–177.

[19] J De Favereau et al. "DELPHES 3: a modular framework for fast simulation of a generic collider experiment". In: *Journal of High Energy Physics* 2014.2 (2014), pp. 1–26.

[20] Matteo Cacciari and Gavin P Salam. "Dispelling the N^3 myth for the k_t jet-finder". In: *Physics Letters B* 641.1 (2006), pp. 57–61.

[21] Matteo Cacciari et al. "The anti-k_t jet clustering algorithm". In: *Journal of High Energy Physics* 2008.04 (2008), p. 063.

[22] Matteo Cacciari et al. "FastJet user manual: (for version 3.0. 2)". In: *The European Physical Journal C* 72 (2012), pp. 1–54.

[23] Sebastian Macaluso and David Shih. "Pulling out all the tops with computer vision and deep learning". In: *Journal of High Energy Physics* 2018.10 (2018), pp. 1–27.

[24] Eric Jang et al. "Categorical reparameterization with gumbel-softmax". In: *arXiv preprint arXiv:1611.01144* (2016).

[25] Irina Higgins et al. "beta-vae: Learning basic visual concepts with a constrained variational framework". In: *International conference on learning representations*. 2017.

[26] Chris Maddison et al. "The concrete distribution: A continuous relaxation of discrete random variables". In: *arXiv preprint arXiv:1611.00712* (2016).

[27] Scott Hauck and Andre DeHon. *Reconfigurable Computing: The Theory and Practice of FPGA-Based Computation*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2007. ISBN: 9780080556017.

[28] Javier Duarte et al. "Fast inference of deep neural networks in FPGAs for particle physics". In: *Journal of Instrumentation* 13.07 (2018), P07027.

[29] Mark Sandler et al. "Mobilenetv2: Inverted residuals and linear bottlenecks". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 4510–4520.

[30] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. "Distilling the knowledge in a neural network". In: *arXiv preprint arXiv:1503.02531* (2015).

PoS(ISGC&HEPiX2023)014