Joint Variational Autoencoder for Anomaly Detection in HEP

Lorenzo Valente\textsuperscript{1*}, Luca Anzalone\textsuperscript{1,2}, Marco Lorusso\textsuperscript{1,2}, Daniele Bonacorsi\textsuperscript{1,2}

\textsuperscript{1}University of Bologna \\
\textsuperscript{2}INFN Bologna \\
*Corresponding author
Introduction

Problem & Data
Anomaly Detection
Auto-Encoders
The problem is framed as **anomaly detection**: no assumption of signal form!

- Knowledge about the QCD background is only necessary.
- The model learns the QCD features, instead of adapting to the signal (as a classifier would do.)
- Must define an AD score.
- The signal samples are only used for evaluating the final performance.

⇒ The AD model is **signal agnostic**!

Average of 200k images each.
Dataset: QCD vs Top Jets

Data from the **Top-tagging challenge**: 200k QCD and 200k Top jets.

- Simulated jets with Pythia8 at 14TeV.
- **Selection**: jets with $p_T \in [550,650]$ GeV and $|\eta| < 2$.
- **Pre-processing**: each jet has at most 200 4-vectors; these are centered, rotated, and flipped in both axes. Then pixelization occurs with a slight crop, yielding $40 \times 40$ images. Finally, the pixels are normalized to sum to one.
- Pixel size is $[\Delta \eta, \Delta \phi] = [0.029,0.035]$.
- Train-test split: 75/25 for QCD, 0/100 Tops.

A random sample.
Is about **distinguishing anomalies** (e.g. Top jets) from normal data (e.g. QCD Jets):

- **Idea**: normal samples have either low error or high-likelihood.
- Anomalies can be either erroneous, rare, or interesting events.
- Can be solved either by: estimating data density, thresholding distances or errors, clustering, or classification.
- Our approach is *self-supervised* and assumes a normal-only (N) set of samples: QCD jets.

Figure from [Google AI Blog](https://ai.googleblog.com/2023/08/anomaly-detection.html)
Auto-Encoders (AEs) are a neural network model trained to reconstruct its inputs:

- The encoder has to compress the input into a meaningful latent space $Z$ (bottleneck)
- The decoder reconstructs from the compressed representation.

- AEs and variants are suitable for anomaly detection, since the training doesn’t require labels!
- In context of AD, the AE have to capture the background peculiarities.
Joint-VAE for Anomaly Detection

Variational AE
Discrete VAE
Joint-VAE
Variational Auto-Encoders (VAEs) are probabilistic models:

- The encoder $q_\phi$, parameterizes a **Gaussian distribution** $z \sim N(\mu(x), \Sigma(x))$.
- The decoder $p_\theta$, reconstructs from the sampled latent vectors.

- The latent space encodes **continuous features**.
- VAEs can generate new samples that look like the inputs.
Reparameterization Trick

**Issue:** cannot backpropagate through stochastic (sampling) nodes

\[
z \sim q_\phi(z \mid x)
\]

\[
z = \mu_\phi(x) + \Sigma_\phi(x) \odot \epsilon
\]
**Categorical VAEs** are discrete latent variable models:

- The **encoder** $q_\phi$, parameterizes up to $K$ **Categorical distributions** with $C$ classes each.
- The **decoder** $p_\theta$, reconstructs from the sampled latent vectors.

- The latent space encodes **discrete features**: finite and enumerable quantities, like counts.
- The Categorical is relaxed by a **temperature** parameter, $\tau$: $\tau \to 0$ (categorical), $\tau \to \infty$ (uniform).
Gumbel-Softmax Trick

**Issue:** the categorical distribution is not differentiable

To reparameterize:
1. Forward encoder to get logits
   \[ \alpha = q_\phi(x) \]
2. Sample from Uniform distribution
   \[ u \sim U(0,1) \]
3. Compute Gumbel noise \( g \)
   \[ g = -\log(-\log u) \]
4. Get the Gumbel-Softmax samples
   \[ q = \text{softmax}(\frac{\alpha + g}{\tau}) \]
Joint-VAE

Can encode both **continuous** and **discrete** features:

How the model is trained:

\[
L_{\phi, \theta}(x) = L_{\text{reco}}(x, \hat{x}) + \beta D_{\text{KL}}(N(\mu_{\phi}, \Sigma_{\phi}) \mid N(0, I)) + \beta D_{\text{KL}}(\text{Cat}(\alpha_{\phi}, \tau) \mid \text{Cat}(1/C))
\]

**Continuous KL**

**Discrete KL**

Luca Anzalone
Anomaly Detection

Reconstruction-based
Latent-based
Pros & Cons
Reconstruction-based Anomaly Scores

AD scores can be defined from reconstructed images $x'$:

- **MSE**: sum of squared differences of pixel values $P$
  \[ S_{MSE}(x, x') = \sum_{p \in P} (x_p - x'_p)^2 \]

- **BCE**: sum of binary-cross entropies on pixels (note: only if normalized in $[0, 1]$)
  \[ S_{BCE}(x, x') = -\sum_{p \in P} x_p \log x'_p + (1 - x_p) \log 1 - x'_p \]

- **Dice** (see ref.): measures the overlap between the predicted and original image
  \[ S_{Dice}(x, x') = \frac{\sum_{p \in P} x_p^2 + \sum_{p \in P} x'_p^2}{2 \sum_{p \in P} x_p - x'_p} \]

- **PixelDiff**: difference of pixel sums (*)
  \[ S_{diff}(x, x') = 1 - \sum_{p \in P} x'_p \]

*True image sum to one.
Latent-based Anomaly Scores

AD scores defined from the **joint latent space** $z = (\alpha, \mu, \Sigma)$:

- **Idea**: KL divergence between learned distribution and prior!
- **KL Continuous**: divergence between learned Normal $N(\mu, \Sigma)$ and standard Normal prior $N(0, I)$

$$S_{KL, cont}(\mu, \Sigma) = -\frac{1}{2} \sum_i (1 + \log \Sigma_{ii} - \mu_i^2 - \exp \Sigma_{ii})$$

- **KL Discrete**: divergence between relaxed Categorical $Cat(\alpha, \tau)$ and the uniform Gumbel-Softmax prior $Cat(1/C)$ - where $C$ is the number of classes.

$$S_{KL, disc}(\alpha) = \sum_i (\pi_i \log \pi_i) - (\pi_i \log 1/C)$$  
  where $\pi = softmax(\alpha)$

*Sums are over latent dimensions.*
Discussion: Pros & Cons

**Reconstruction-based AD:**

- Easier to define anomaly scores, e.g. from common loss functions and metrics.
- Scores values can be interpreted by image quality metrics or visual inspection.
- Requires forward pass of whole model (encoder + decoder): slower.

**Latent-based AD:**

- Possibly difficult to interpret: high-dim latent space cannot be visualized.
- Scores can be difficult to design, e.g. analytical KLD - but equally performant.
- Faster: requires only encoder predictions.
- Suitable for model optimization and FPGA deployment.
Model Acceleration

Quantization

FPGA Study
Quantization transforms floating-point arithmetic to **fixed-point precision**:

- Less #bits to reduce memory footprint, and FPGA resources.
- Quantization is applied on both weights, and activations.
- **Quantization-Aware-Training (QAT)** maintains high accuracy at low-precision <16, 6>: total with of 16bits, 10bits for floats and 6bits for integers.
- Yields lower latency and energy consumption [J] (by QTools).

### Table: Energy Consumption

<table>
<thead>
<tr>
<th>Model</th>
<th>Total Energy [μJ]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reduced Encoder</td>
<td>5.4957</td>
</tr>
<tr>
<td>Quantized Encoder</td>
<td>3.3434</td>
</tr>
</tbody>
</table>

Energy consumption reduced by **39%**

Luca Anzalone
Field-Programmable Gate Arrays

FPGAs are **hardware-programmable** devices:

An FPGA is made of many replicated units.

A configurable logic block.
The HLS4ML Python Package

ML models have to be translated to Hardware Description Language (HDL) to deploy on FPGA:

- **HLS4ML** does this.
- Converts layers to High Level Synthesis code, then C++.
- It optimizes also.
- Finally, proprietary SW compilation.
- Synthesized code can be simulated before deployment.
FPGA Implementation Feasibility Study

FPGA are programmable accelerators that can enable **real-time inference**:

- Network synthesis is done via [HLS4ML](https://github.com/hls4ml/hls4ml) toolkit.
- From a synthesized layer or block, we can estimate the **resource factor** $\rho$ via:

$$\rho(a, b) = \frac{(h_{in}^a/s) \times (w_{in}^a/s) \times d_{in}^a \times d_{out}^a}{(h_{in}^b/s) \times (w_{in}^b/s) \times d_{in}^b \times d_{out}^b}$$

- To estimate FPGA resources (e.g., LUTs) multiply $\rho$ by a known layer or block:

**conv2_b0_0** won’t fit in FPGA, so we estimated its resource consumption.

<table>
<thead>
<tr>
<th>Layer Name</th>
<th>$h_{in}$, $w_{in}$, $d_{in}$, $d_{out}$</th>
<th>DSP(%)</th>
<th>LUT(%)</th>
<th>FF (%)</th>
<th>BRAM(%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>conv2_b0_0</td>
<td>20,20,16,10</td>
<td>752(6)</td>
<td><strong>6189696(358)</strong></td>
<td>229536(~7)</td>
<td>288(5)</td>
</tr>
<tr>
<td>conv2_b1_1</td>
<td>10,10,20,5</td>
<td>47(~0)</td>
<td>386856(22)</td>
<td>14346(~0)</td>
<td>18(~0)</td>
</tr>
<tr>
<td>dconv_b2</td>
<td>5, 5, 4,4</td>
<td>15(~0)</td>
<td>309544(17)</td>
<td>9352(~0)</td>
<td>4(~0)</td>
</tr>
<tr>
<td>final block</td>
<td>5, 5, 4,2</td>
<td>851(6)</td>
<td>91763(5)</td>
<td>28074(~0)</td>
<td>377(7)</td>
</tr>
</tbody>
</table>
Results

Reconstructions

Anomaly Performance

Comparison: large vs quantized
Reconstructed Samples

Reconstructed images *averaged* over test-set.

The ground-truth is on the left.

- **QCD** (top-row) are closely reconstructed: *low error*.
- **Tops** (bottom) are predicted to be QCD-like: *high error*.
Joint Latent Space

Learned latent spaces by our Joint-VAE; projected to 2d.

- Spaces: a 32-d Gaussian, and 20-d Categorical.
- We can see the 20 class-clusters for the categorical space.
Reconstruction-based Scores: Large Model

Luca Anzalone

ISGC 2023
By combining both continuous and discrete KL divergences, is possible to further improve the anomaly score.
Reconstruction-based Scores: Quantized Model
Latent-based Scores: Quantized Model

By quantizing we lose performance also on the latent space, so the KL scores. But the trend is maintained.
Comparison: Large vs Quantized Model

Summary of ROC-AUC performance per metric:

- **Large model** has 262k (encoder) + 545k (decoder) params: 6 residual blocks.
- **Quantized model** has 10k (**constraint**: max. 1024 params per layer) + 545k params: 4 residual blocks.
- Latent dimensions for both models are 32 (continuous) and 20 (discrete).
- Decoder is the same ⇒ similar AD performance.
- #params and quantization impacts on encoder, KL-based metrics.

<table>
<thead>
<tr>
<th>AD Score</th>
<th>Large</th>
<th>Quantized</th>
</tr>
</thead>
<tbody>
<tr>
<td>MSE</td>
<td>38.35%</td>
<td>38.33%</td>
</tr>
<tr>
<td>Pixel-diff</td>
<td>84.18%</td>
<td>84.4%</td>
</tr>
<tr>
<td>BCE</td>
<td>84.35%</td>
<td>84.54%</td>
</tr>
<tr>
<td>Total (Dice + BCE)</td>
<td>85.05%</td>
<td><strong>85.25%</strong></td>
</tr>
<tr>
<td>KL Cont.</td>
<td>86.45%</td>
<td>80.88%</td>
</tr>
<tr>
<td>KL Discrete</td>
<td>73.78%</td>
<td>77.46%</td>
</tr>
<tr>
<td>KL Total (Cont. + Disc.)</td>
<td><strong>86.62%</strong></td>
<td>81.17%</td>
</tr>
</tbody>
</table>

*Constraint is due to Vivado synthesis.
Conclusions

Summary
Limitations
Outlook
Variational Auto-Encoders are suitable models for anomaly detection:

- We don’t assume any specific signal $\Rightarrow$ not sensitive to particular BSM scenario.
- The model is only trained to reconstruct the QCD background.
- Combining both continuous and discrete latent spaces achieves better AD performance.
- Latent-based AD is competitive with reconstruction-based scores, allowing to deploy only the encoder model.
- Model compression via weight and activation quantization can be done with Qkeras: saving energy, memory, and accelerator resources.
- Model synthesis for FPGA deployment can be done by HLS4ML.
Limitations and Outlook

General limitations of such kind of approaches:

• Need test samples of different kind of signals to assess generalization to BSM models.

• The VAE method is simple to train, but optimizes a different objective (i.e. reconstruction loss) ⇒ we have little control about maximizing the target AD score (e.g. KL-divergence)

• FPGA deployment can be challenging: accelerator resources are limited while DL layers are costly (like convolutions.), especially on image-like inputs.

• Moreover, vendors can add additional constraints: like maximum #params per layer.

• Limited support of libraries: for example HLS4ML is compatible with few common layers.

• Need better methods that yield very compact models: knowledge distillation?
Thanks for the Attention!

Questions?

Contacts:

lorenzo.valente3@studio.unibo.it
luca.anzalone2@unibo.it
marco.lorusso11@unibo.it