Ephemeral learning – Augmenting triggers with online-trained normalizing flows

Anja Butter[1,2], Sascha Diefenbacher[3⋆], Gregor Kasieczka[3], Benjamin Nachman[4,5], Tilman Plehn[1], David Shih[6] and Ramon Winterhalder[7]

1 Institut für Theoretische Physik, Universität Heidelberg, Germany
2 LPNHE, Sorbonne Université, Université de Paris, CNRS/IN2P3, Paris, France
3 Institut für Experimentalphysik, Universität Hamburg, Germany
4 Physics Division, Lawrence Berkeley National Laboratory, Berkeley, CA, USA
5 Berkeley Institute for Data Science, University of California, Berkeley, CA, USA
6 NHETC, Department of Physics & Astronomy, Rutgers University, Piscataway, NJ USA
7 Centre for Cosmology, Particle Physics and Phenomenology (CP3), Université catholique de Louvain, Belgium

⋆ sascha.daniel.diefenbacher@uni-hamburg.de

Abstract

The large data rates at the LHC require an online trigger system to select relevant collisions. Rather than compressing individual events, we propose to compress an entire data set at once. We use a normalizing flow as a deep generative model to learn the probability density of the data online. The events are then represented by the generative neural network and can be inspected offline for anomalies or used for other analysis purposes. We demonstrate our new approach for a toy model and a correlation-enhanced bump hunt.

Received 08-03-2022
Accepted 26-08-2022
Published 07-10-2022
doi:10.21468/SciPostPhys.13.4.087

Check for updates

Contents

1 Introduction

The ATLAS and CMS experiments at the Large Hadron Collider (LHC) produce data rates around 40 terabytes per second and per experiment [1,2], a number that will increase further for the high-luminosity upgrades [3,4]. These rates are far too large to record all events, so these experiments use triggers to quickly select potentially interesting collisions, while discarding the rest [5–8]. The first two trigger stages are a hardware-based low-level trigger, selecting events with μs-level latency, and a software-based high-level trigger with 100 ms-level latency. After these two trigger stages, some interesting event classes, such as events with one highly-energetic jet, still have too high rates to be stored. They are recorded using *prescale* factors, essentially a random selection of events to be saved. An additional strategy to exploit events which cannot be triggered on systematically is data scouting, or trigger-level analysis [9–12]. Through fast online algorithms, parts of the reconstruction are performed at trigger level, and significantly smaller, reconstructed physics objects are stored instead of the entire raw event. This physics-inspired compression increases the number of available events dramatically, with the caveat that the raw events will not be available for offline analyses.

Using machine learning (ML) to increase the trigger efficiency is a long-established idea [13], and simple neural networks for jet tagging have been used, for example, in the CMS high-level trigger [14]. The advent of ML-compatible field-programmable gate arrays (FPGAs) has opened new possibilities for employing such classification networks even at the low-level trigger [15–21]. ML-inference on FPGAs is making rapid progress, but the training of e.g. graph-based networks on such devices is still an active area of research. At the same time, the available resources limit the size and therefor complexity of possible ML models.

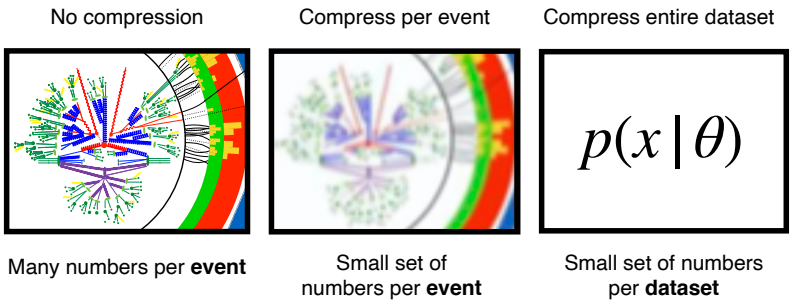

Figure 1: Illustration of data compression at the LHC. Most analyses are performed offline, based on entire events and lossless compression (left). Data scouting employs lossy compression per event (center). Our method compresses an entire data set by learning a generative model for events x in terms of network parameters θ (right).

We propose a new strategy, complementary to current trigger strategies and related methods, where instead of saving individual events, an online-trained generative ML-model learns the underlying structure of the data. The advantage of our strategy, illustrated in Fig. 1, is its fixed memory and storage footprint. While in a traditional trigger setup more events always

require more storage, the size of the generative model is determined by the number of parameters. Additional data increases the accuracy of these parameters at fixed memory size, until the capacity of the model is reached. In practice, we envision an online generative model to augment data taking at the HLT level[†] and act as a scouting tool in regions currently swamped by background. However, a sufficiently optimized version of this approach could transform data taking by instead learning the overall distribution of data without the need for triggers altogether.

The viability of our novel approach rests on the assumption that the relevant physics of LHC collisions can be described, statistically, by far fewer parameters than are necessary to record an entire event. An intuitive example is a set of N Gaussian random numbers. Recording the entire data set requires N numbers, but the sample mean and variance are sufficient statistics, so that only storing them contains all the information about the entire data set. Using generative models for data compression is also an established method [22–26]. However, it usually means encoding a given data point into a more storage-efficient representation. This is different from our proposal which aims to encode the full underlying distribution in the network parameters [27] and to represent all aspects of the LHC training data in the generative network. Modulo limits of expressivity, the online-trained network output can, in principle, replace the training data completely.

This paper is structured as follows: In Sec. 2 we discuss the challenges of running an online trained generative model and strategies by which these can be overcome. In Sec. 3 we perform a first proof-of-concept study using a simple 1-dimensional data set. Section 4 expands this test to a standard benchmark data set. We present our conclusions in Sec. 5.

2 Online training

Trigger systems work through a chain of consecutively stricter requirements. Directly following the sensor measurements are the low-level or level-1 (L1) triggers. They have to make decisions fast enough to keep up with the rate of incoming collisions, in our case 40 MHz, so they are hardware-based and at most perform low-complexity reconstruction. The passing events then reach the high-level trigger (HLT). The reduced data rate output by the L1 triggers, 100 kHz for ATLAS and CMS, allows for a software-based HLT running on a dedicated server farm. Its primary purpose is to reduce the event rate to the point where everything can be stored for offline analysis. In some cases it can be beneficial to have an HLT channel with low thresholds, such that the number of triggered events is still too large to be stored completely. In such cases one randomly decides which event is stored or discarded. The chance to store an event and hence the data reduction is given by a tuneable prescale suppression factor.

We propose a new, ML-based scouting strategy in the form of an online-trained generative model. A generative model trained on events, which are not triggered and stored, can extract interesting information without saving the events. We therefore consider augmenting the HLT as a first possible application of the new technique. The proposed workflow, also shown in Fig. 2, is

- (online) Train a generative model on all incoming events;
- (offline) Use the trained model to generate data;
- (offline) Analyze this generated data for indications of new physics;
- (offline) If an interesting feature appears, adjust the trigger to take data accordingly;

[†] as training (as opposed to inference) models on FPGA hardware deployed at earlier trigger stages is currently not possible

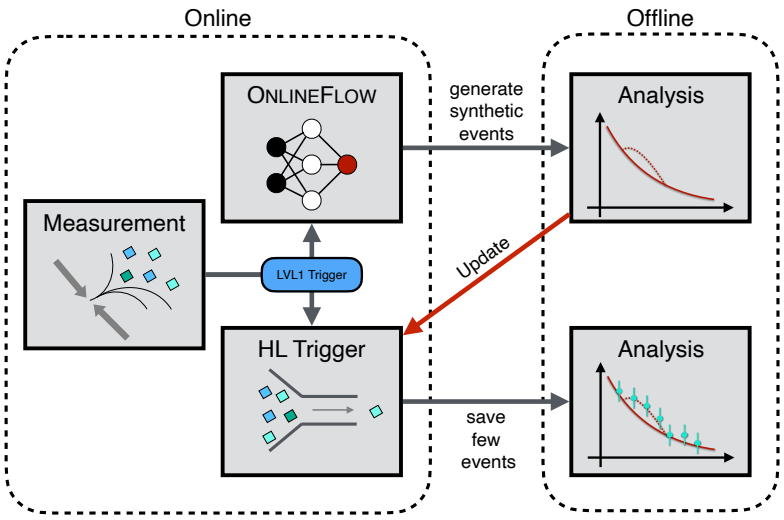

Figure 2: Illustration of the proposed workflow. First, we train a generative model on all incoming events (online). Then, we use the trained model to generate data and analyze the generated data for signs of new physics (offline). If necessary, we adjust the trigger to take new data accordingly (online) and analyze that data (offline).

- (online) Collect data with new triggers;
- (offline) Statistically analyze that recorded data.

This online model is compatible with most current trigger setups[†]. As a first step we assume operation in the HLT, but the concept is also applicable to the first trigger stage. The workflow can also be extended to measurements rather than searches.

While our idea is not tied to specific generative models, normalizing flows (NF) [28–31] are especially well suited due to their stable training. This allows us to train our OnlineFlow without stopping criterion, a property well suited for training online. Furthermore, NFs have been shown to precisely learn complex distributions in particle physics [32–42]. The statistical benefits of using generative models are discussed in Ref. [43], for a discussion of training-related uncertainties using Bayesian normalizing flows see Refs. [44, 45].

The properties of online training, specifically seeing every event independently and only once, are in tension with training generative models. Such models perform best when they have the option to look at data points more than once. Additionally, processing several events at the same time should allow the model to train significantly faster through the use of GPU-based parallelization and stochastic gradient descent. This is why we follow a hybrid approach: incoming events are collected in a buffer with size N_{buff}. Once this buffer is full, it is passed to the network, which processes the information in batches of size N_{batch}. This process is iterated over N_{iter} times. After this, the buffer is discarded and replaced by the next buffer. We visualize this scheme in Fig. 3. In addition to aiding the network training, this hybrid training also decouples the network training rate from the data rate, as we can continuously adapt N_{iter} to ensure the network is done with the current buffer by the time the next is filled. Additional technical details, including the estimation of uncertainties, of our approach are discussed in the context of the examples presented below.

[†]It may also be possible to discover new physics directly with the generated data, in contrast to adjusting the triggers. However, we take a more conservative perspective here.

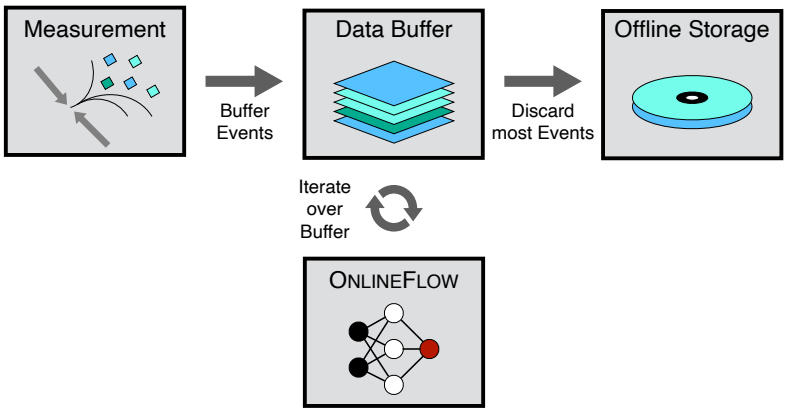

Figure 3: Illustration of our test of the online training. We collect events (left) into a buffer (top center). In the classic prescale approach most events are discarded, only a small fraction is saved for offline analysis (right). Our OnlineFlow (bottom center) is trained online on the data in the buffer and iterated until the buffer is replaced.

3 Parametric example

We first illustrate our strategy for a 1-dimensional parametric example. While in practice it would be straightforward to store at least a histogram for any given 1-dimensional observable, this scenario still allows us to explore how generative training and subsequent statistical analysis approaches need to be modified for the ephemeral learning task.

3.1 Data, model, training

The 1-dimensional data is inspired by a typical invariant mass spectrum with a resonance. In a sample with N events, or points, every event is randomly assigned to be either signal or background with a probability of λ or $1-\lambda$, respectively. On average, this gives $\lambda \times N$ signal and $(1-\lambda) \times N$ background events. The values x of the signal and background events are drawn from their respective distributions,

$$p_B(x) = \frac{1}{b} e^{-bx} \quad \text{and} \quad p_S(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}. \tag{1}$$

We use $b = 1$, $\mu = 1$, $\sigma = 0.04$ and $\lambda = 0.005$ and show the truth distribution in Fig. 4.

As our generative model we choose a masked auto-regressive flow (MAF) [46]. It comprises five MADE [47] blocks, each with two fully connected layers with 32 nodes, resulting in 7560 network weights. To aid the flow in learning the sharp threshold, we train on the logarithm ($\log x$) of each 1-dimensional event. Furthermore, since NFs such as the MAF are not designed for 1-dimensional inputs, we add three dummy dimensions drawn from a normal distribution, quadrupling the total number of input and output dimensions to four. The MAF model is implemented using PyTorch [48] and trained using the Adam optimizer [49] with a learning rate of 10^{-5} and a batch size of $N_{\text{batch}} = 250$. Finally, we use a buffer size of $N_{\text{buff}} = 10,000$ and $N_{\text{iter}} = 100$ iterations per buffer. Our training sample includes 5M events altogether, stored for evaluation purposes.

One challenge with this model is a bias towards the more recent buffers, for which the network is optimized on last. To cure this, we use stochastic weight averaging [50]. During training, we keep track of the running average of the model weights. Once the network is done with a given buffer, the average weights are updated, scaled by the number of average-weight updates so far. At the end, we use the averaged weights for generation.

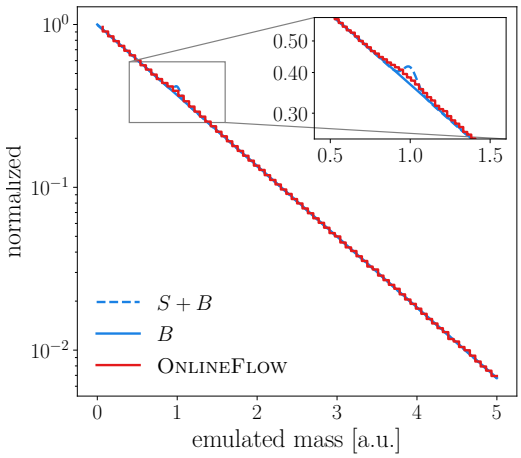

Figure 4: Illustration of our 1-dimensional, exponentially falling, mass spectrum. B denotes background events, S is the signal, following the truth distributions of Eq.(1). The OnlineFlow histogram shows 10M events generated after training on $S+B$.

In Fig. 4 we show how OnlineFlow, trained on signal plus background attempt to reproduce the signal mass peak. While the width of the peak is not correctly described we do see a noticeable abundance. As the goal is in finding evidences of new physics in a subsequent offline analysis of the OnlineFlow data, this abundance may still, however, be sufficient for our purpose even if the peak width is mismodeled. To estimate the statistical uncertainty[§] (described in more detail below), needed to compute p-values, we train 20 flows in parallel. As training samples for the statistical uncertainty, we use bootstrapping. Creating classical bootstrapped ensembles is not possible because we do not have the full dataset to resample from with replacement. Instead, we use an online-compatible version whereby each event in each bootstrapped ensemble is given a weight that is Poisson distributed with unit mean. These weights are independent for each flow in the ensemble and are kept constant as the flow iterates over one buffer. We leave the further exploration of this ad hoc solution and alternative methods such as Bayesian flows [44, 51, 52] to the future.

3.2 Classical bump hunt benchmark

Our goal is to compare how well a potential signal can be extracted from flow-generated events, vs. a range of classical offline analyses, consisting of standard bump hunts on the training data reduced by different levels of prescale triggers. First, we describe our procedure for the latter.

We use SciPy [53] to fit a background model to the mass histogram of $N_{\text{pre}} = N_{\text{data}}/f_{\text{pre}}$ events, where $N_{\text{data}} = 5 \times 10^6$ and f_{pre} is the prescale factor. We find that the following background model fits the data well:

$$p(x) = \alpha\, e^{-\beta x+\gamma x^2+\delta x^3+\epsilon x^4+\zeta x^5}\,, \tag{2}$$

with best fit parameters that depend on f_{pre}.

We supply this background model to BumpHunter [54] to identify the most likely signal region. Our BumpHunter-setup scans the emulated mass ranging from 0 to 5, divided into 50

[§]Assuming that the variations in the output from fixed inputs and the stochastic nature of the network initialization and training are negligible compared to the data statistical uncertainty. If this is not the case, one could reduce these with further ensembling.

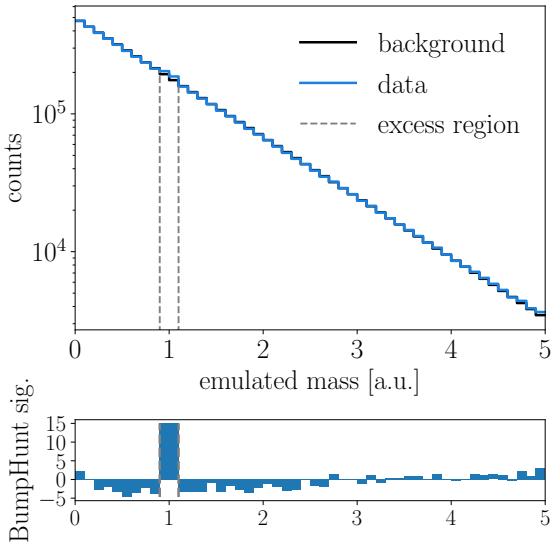

Figure 5: Example of the BUMPHUNTER used on the training data, based on the background model of Eq.(2). Dotted lines indicate the upper and lower bounds of the signal region. The lower panel shows the significance.

bins, with a minimal signal window size of two bins and maximal window size of six. Given the data, BUMPHUNTER extracts a lower and an upper bound on the most likely bump position and defines our signal region. We then extract the local significance as

$$\text{significance} = \frac{\mathcal{O}-B}{\sqrt{B}} \equiv \frac{S}{\sqrt{B}}, \tag{3}$$

where B is the number of background events predicted in the signal region, and S is defined as the number of observed events $\mathcal{O}$ minus background model. As the prescale factor is increased, we expect the local statistical significance of the bump hunt to decrease as $1/\sqrt{f_{\text{pre}}}$. We illustrate an example of the classical analysis in Fig. 5, corresponding to $f_{\text{pre}} = 1$ (i.e. using the full 5M training data). We see that the background model agrees well with the data over the entire range, except for the identified signal region (and the surrounding regions, which must compensate for the excess).

3.3 ONLINEFLOW performance

For ONLINEFLOW, we train (in the online fashion described above) on the full 5M events, and use an ensemble of networks to generate 100M events, combined into one large set. This large number is used to make the statistical uncertainty from sampling negligible compared with other sources of uncertainty. We again fit the same background model, Eq.(2), to the mass histogram of these 100M events, and find the best fit parameters to be

$$\begin{aligned} &\alpha = 1.0099194(5), && \beta = 1.020952(15), && \gamma = 0.03216(4), \\ &\delta = -0.017450(14), && \epsilon = 0.003913(1), && \zeta = -0.00031395(1). \end{aligned} \tag{4}$$

While the background model is trained on a large sample of flow-generated data, its χ^2 for the smaller set of training events and a smaller set of flow-generated events is excellent ($\chi^2/\text{dof} \approx 1$) and consistent with each other. We have also checked that changing the analytic form of the background model has little effect on our results.

As for the classical analysis, the best-fit background model is then input to BUMPHUNTER in order to identify the most likely signal region. To reduce the chance of mistaking imperfect

network trainings for a signal, we randomly split our model ensemble into two equal parts, other splits being possible as well (such as k-folding; see e.g., Refs. [55,56]). The first ensemble of ten networks defines the signal region using BUMPHUNTER. Given the signal region, we then use the second ensemble of ten networks to compare the number of generated events in the signal region to the predicted background.

To estimate the significance of the signal encoded in the ONLINEFLOW, it is a bit more subtle than just using Eq,(3), since we are essentially taking the statistical uncertainty of the generated events to be zero by generating 100M of them. Instead, the statistical uncertainty on the training data is translated into a systematic uncertainty in the generated events. To quantify this, we use the second network ensemble to compute bootstrap statistics in the usual way. First, combining all flow-generated events in the signal region, $\mathcal{O}$, and the event count from the respective background fit, B, gives us the total signal and background in the signal region and the corresponding uncertainties

$$\begin{aligned} B &= \frac{2}{N_\text{ens}} \sum_i^{N_\text{ens}/2} B_i\,, & \delta_B &= \sqrt{\frac{2}{N_\text{ens}}}\,\sigma(B)\,, \\ \mathcal{O} &= \frac{2}{N_\text{ens}} \sum_i^{N_\text{ens}/2} \mathcal{O}_i\,, & \delta_\mathcal{O} &= \sqrt{\frac{2}{N_\text{ens}}}\,\sigma(\mathcal{O})\,, \end{aligned} \tag{5}$$

where $\sigma(X)$ is the standard deviation of X over the ensemble and, in our case, $N_\text{ens}/2 = 10$. The signal rate and uncertainty,

$$S = \mathcal{O} - B\,, \qquad \delta_S^2 = \delta_\mathcal{O}^2 + \delta_B^2\,, \tag{6}$$

define the signal significance

$$\text{significance} = \frac{S}{\sqrt{\delta_S^2 + (\sqrt{B})^2}}\,. \tag{7}$$

The contribution $\sqrt{B}$ represents the flow statistical uncertainty from the finite amount of generated samples. It will usually be negligible compared to the data statistical uncertainty, δ_S, but we still include it in our calculation for consistency.

Because for our parametric example we can save the training data, as well as the weights of all models after each buffer, we can track the signal significance during the online training. In the left panel of Fig. 6 we compare the significance of the ONLINEFLOW and the complete training data. As expected, the data-derived significance scales with the square root of the number of events. The ONLINEFLOW significance initially rises at a similar rate, and then increases at a slower rate. Asymptotically, the significance may saturate if the network approaches the limits of its expressiveness.

In the right panel of Fig. 6, we compare the performance of ONLINEFLOW vs the classical bump hunt with different prescale factors f_pre. We see that, as expected, the significance of the latter decreases as $1/\sqrt{f_\text{pre}}$. Meanwhile the significance returned by ONLINEFLOW is constant since the network is always trained on the full data. Above $f_\text{pre} \approx 4$, we see that ONLINEFLOW starts to outperform the classical approach.

Finally, we need to check how susceptible our ONLINEFLOW setup is to fake signals. We run our setup with the same parameters as before, but for zero signal fraction. The result is shown in Fig. 7. We see that there is a larger error margin for the ONLINEFLOW significance, owing to the larger fluctuations between individual ONLINEFLOW training, however the average fake rate for the flow stays well below the signal significance we achieve for the 0.5% signal contamination. The average also stays consistent with the fake rate of the classical bump hunt.

The precise behavior for intermediate signal rates presents an interesting question that exceeds the scope of this work, but would warrant further investigation.

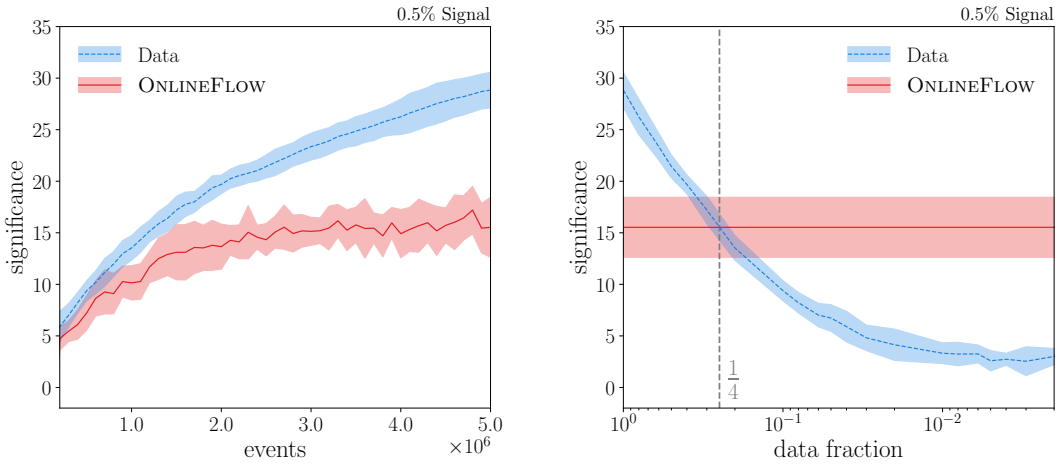

Figure 6: Left: signal significance as a function of the amount of training data for the classical approach, based on all training data, and the OnlineFlow. Right: signal significance as a function of the prescale factor. A prescale factor of one corresponds to 500×10^4 events. The shaded regions estimate the uncertainty based on five executions of the experimental setup. The dotted grey line indicates the crossover point at a datafaction of $\frac{1}{4}$, which corresponds to a prescale factor of 4.

4 LHCO dataset

After illustrating our novel approach for the 1-dimensional toy model, we move to the more realistic, higher dimensional LHC Olympics anomaly challenge R&D dataset [57].

4.1 Data, model, training

The dataset comprises a background of dijet events including a new-physics signal. This signal, which we assume to contribute 1% of our events, originates from a W' decaying into two heavy particles, which in turn decay into quarks,

$$W' \to X(\to qq)Y(\to qq)\,. \tag{8}$$

The respective particle masses are $m_{W'} = 3.5$ TeV, $m_X = 500$ GeV, and $m_Y = 100$ GeV. All events are generated using Pythia8 [58] and Delphes3.4.1 [59–61]. The jets are clustered using FastJet [62] with the anti-k_T algorithm [63] using $R = 1$. Finally, all events are required to have at least one jet with $p_T > 1.2$ TeV.

While this dataset features high mass resonances that are not perfectly in line with the intended application range of OnlineFlow, we feel that the proven and well known nature of the LHCO data, as well as its availability make up for this shortcoming.

The same input format used for the anomaly detection [32, 40, 64] is also used for the OnlineFlow. Specifically, there are five input features, the dijet mass, the mass of the leading jet, the mass difference between the leading and sub-leading jets, and the two n-subjettiness ratios [65, 66],

$$\left\{ m_{jj}, m_1, m_2 - m_1, \tau_{21}^{(1)}, \tau_{21}^{(2)} \right\}. \tag{9}$$

All observables except for m_{jj} are subjet observables and at most weakly correlated with m_{jj}. We show distributions of all observables in Fig. 8, for the training data and the OnlineFlow

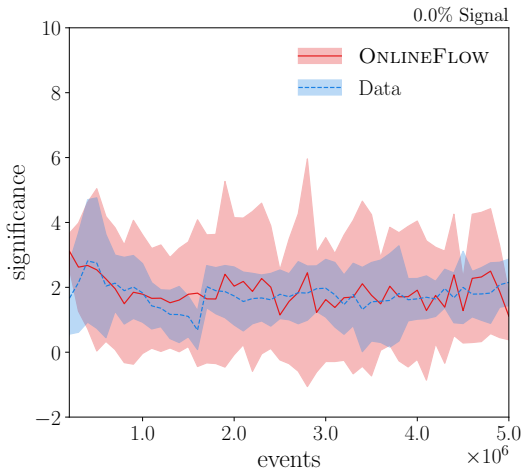

Figure 7: Fake signal significance as a function of the amount of training data for the classical approach, based on all training data, and the OnlineFlow. The analysis is identical to that of Fig. 6, but with zero signal fraction.

output. We also show a 10-fold enhanced signal, relative to the 1% signal rate we will use for our actual analysis, to illustrate the narrow kinematic patterns of the W' resonance.

The LHCO version of the OnlineFlow network is slightly modified compared to the parametric setup to accommodate a 10-dimensional input. These comprise five features and five additional noise dimensions, the additional noise was found to increase the performance, although no systematic scan over this hyperparameter was performed. The number of MADE blocks is now 10, and the number of nodes in the fully connected layers is quadrupled to 128. The buffer size is still 10,000, and the number of iterations per buffer is increased to 1,000. Learning rate and optimizer are identical to those used before.

For an assumed signal rate of 1% we split a total of around 300k LHCO events into 80% training, 10% evaluation, and 10% validation data. The limited evaluation data is further supplemented with additional signal and background events, to smooth out the ROC and SIC curves. This gives us a new set of 330k evaluation events, roughly equally split between signal and background, but only to present our results.

The red lines in Fig. 8 demonstrate the flow's ability to reproduce the five input features. The leading jet mass and the mass difference are learned well, as are the n-subjettiness ratios. The invariant dijet mass distributions shows some deviations at the sharp boundaries, a typical effect for neural networks which can be cured using a range of standard methods [45].

4.2 CWoLa benchmark

Just like for the parametric example, we need to determine how well the OnlineFlow captures the signal features for an anomaly detection setup. As a simple benchmark we choose the Classification Without Labels (CWoLa) [55,56,67] setup, implemented following Ref. [40]. In the CWoLa framework, a classifier is trained to distinguish between two samples with different relative amounts of signal. For the LHCO dataset these two samples are the signal region defined in terms of the dijet invariant mass and indicated in Fig. 8,

$$m_{jj} = m_{W'} \pm 200 \text{ GeV} = 3.3 \ldots 3.7 \text{ TeV}, \tag{10}$$

and the control regions away from the W'-peak. While this definition of a signal region has no effect on the training of the OnlineFlow, it implies that we cannot include m_{jj} as a training

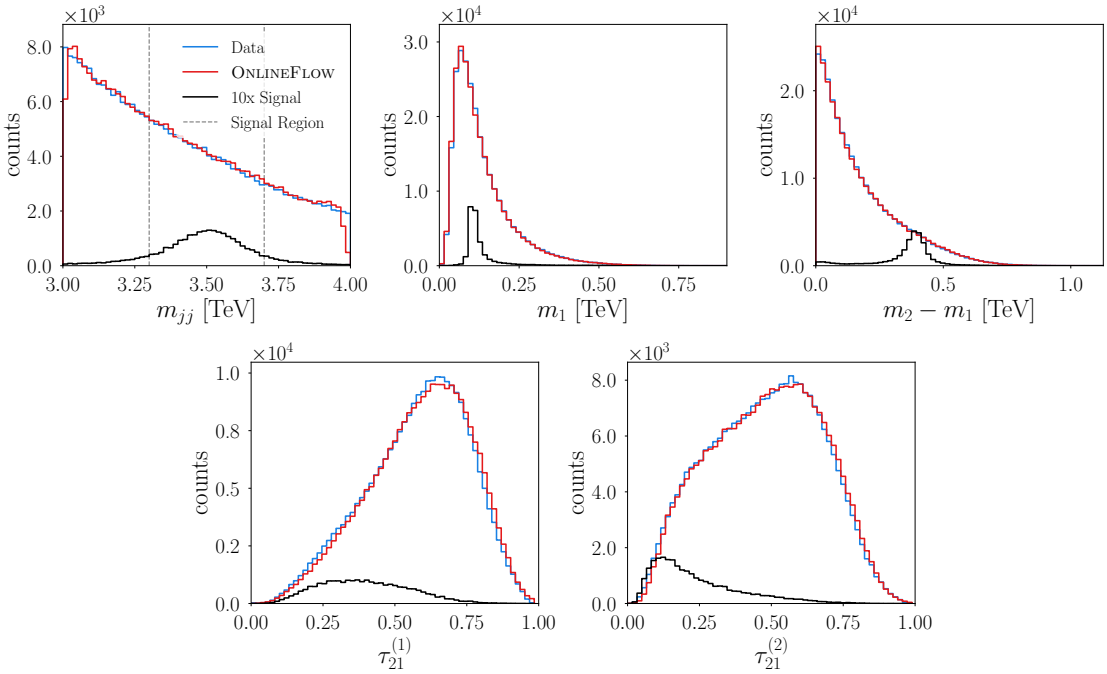

Figure 8: Observables for the LHCO data set, as listed in Eq.(9). We show the original training data, with 1% signal contamination, and the data generated by the flow. The signal region in m_{jj} is indicated by dotted lines.

variable for the CWoLa network. In a realistic anomaly search one would use a sliding mass window, but since we are only interested in benchmarking the OnlineFlow we assume a signal region around the resonance.

As the signal and control regions should only differ in the amount of potential signal, the classifier learns to distinguish signal from background while separating the two regions. We can define the likelihood ratio for a given event x to be signal as

$$R_{\text{CWoLa}}(x) = \frac{p(x|\text{SR})}{p(x|\text{CR})}, \tag{11}$$

where $p(x|\text{SR})$ and $p(x|\text{CR})$ are the classifier outputs for signal and control regions. By scanning different thresholds on this ratio we can enrich the relative amount of signal-like events.

Our CWoLa classification network is implemented using PyTorch [48] and consists of three fully connected layers, each with 64 nodes. The training uses a binary cross entropy loss, the Adam [49] optimizer with a learning rate of 10^{-3}, and runs for 100 epochs. As the signal and control region contain an unequal amount of data, the two classes are reweighted to correct for this imbalance. The final classifier comprises an ensemble of the ten network states with the lowest validation loss during training.

To benchmark the signal extraction of the OnlineFlow, we train CWoLa on the LHCO training data. It contains 240k events and is identical to the data used to train the OnlineFlow. This training is aided by a 30k validation set. To determine how much information the OnlineFlow captures we repeat the CWoLa benchmark training with 50%, 20%, 10%, and 5% of the LHCO training and validation sets.

4.3 OnlineFlow performance

Considering the positive results from our simple toy model, we now compare the CWoLa results based on the OnlineFlow and on the LHCO training data. The relevant numbers of

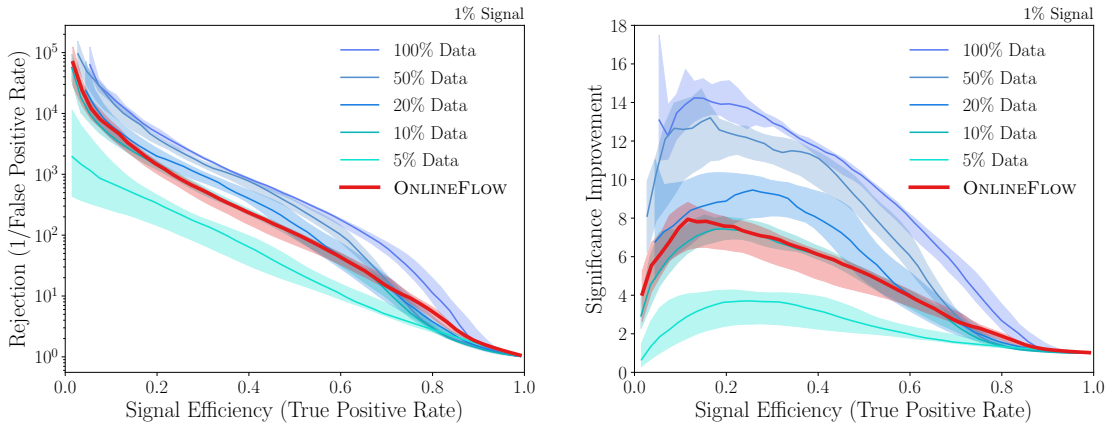

Figure 9: ROC (left) and significance improvement (right) for the CWoLa benchmark approach, based on a decreasing amount of data, compared with the OnlineFlow. The signal fraction is 1%. Vertical order of the Data lines corresponds to their order in the legend.

merit, namely the ROC curve and the signal improvement $\epsilon_S/\sqrt{\epsilon_B}$, are shown in Fig. 9. For the standard CWoLa approach, trained on the LHCO data, the smaller training samples correspond to prescale factors of 2, 5, 10, and 20. The shaded regions indicate the one-sigma range from repeating the CWoLa analysis ten times. We see that, for instance for a constant signal efficiency, the background rejection drops increasingly rapidly for smaller training samples. This illustrates how larger prescale factors seriously inhibit the reach of searches for new physics in non-trivial kinematic regions.

To determine the power of the OnlineFlow we then train the CWoLa network on 500k events generated from the OnlineFlow, with an additional 62500 OnlineFlow events serving as the validation set. This mirrors the split into training-validation-test data of the LHCO data. In both panels of Fig. 9 we can now compare the OnlineFlow results to the different prescalings and find that it performs similarly to 10% of the training data. In a setting where one has to work with a trigger fraction of less then 10%, one could benefit from the OnlineFlow setup.

While the CWoLa results show that the OnlineFlow does not only encode features represented in the input variables, but also describes correlations directly, it remains to be shown that its performance is stable when we decouple the main features more and more from the input variables. This happens when we train the generative networks on low-level event representation, challenging the network both in expressivity and reliability. In line with the conclusions from Fig. 6 this might, for instance, require a larger network and adjustments to the building blocks of the normalizing flow and the bijectional training.

5 Conclusions

Data rates of modern particle colliders are a serious challenge for analysis pipelines. In terms of data compression, triggered offline analyses use lossless data recording per event, but at the price of a huge loss in deciding which event should be recorded. Trigger-level analyses accept losses in the individual event information, to be able to analyze significantly more events. Our strategy is inspired by the statistical nature of LHC measurements and aims at analyzing as many events as possible, but accepting a potential loss of information on the event sample level.

For this purpose, we propose to train a generative neural network, specifically a normalizing flow (OnlineFlow), to learn and represent LHC events for offline analyses.

First, we have studied the performance of this OnlineFlow compression for a 1-dimensional parametric example. We mimic a narrow bump hunt on an exponentially dropping flat background and compare the significance achieved by classic methods with different prescale factors with the significance based on generated events from the trained OnlineFlow. We find that OnlineFlow outperforms a classical, offline analysis for prescale factors larger than 3.5, while fake signal significances remain at the level of the classical analysis once the flow is trained properly.

Second, we looked at a more realistic example, specifically a simulated W'-signal available as the LHCO R&D dataset. We use the CWoLa method to extract the signal from correlated phase space observables and to define the benchmark based on the training events with a variable prescale factor. Again, we find that OnlineFlow outperforms the offline CWoLa method for prescale factors above a certain threshold (in this case ~ 10).

Implementing OnlineFlow into an existing trigger system will require further work to scale up the networks input dimensionality as well as its expressiveness to handle the more complex data structures of real LHC events. Further, the challenge of integrating the model training into the infrastructure of a real experiment will require further work and exploration. However, regardless of these challenges, we believe the examples demonstrated here serve as a proof of concept for the proposed OnlineFlow, warranting further investigation.

Acknowledgements

The research of AB and TP is supported by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under grant 396021762 – TRR 257 Particle Physics Phenomenology after the Higgs Discovery. AB would like to thank the LPNHE for their hospitality. This work was supported by the Deutsche Forschungsgemeinschaft under Germany's Excellence Strategy EXC 2181/1 - 390900948 (the Heidelberg STRUCTURES Excellence Cluster). SD is funded by the Deutsche Forschungsgemeinschaft under Germany's Excellence Strategy – EXC 2121 Quantum Universe – 390833306. BN is supported by the U.S. Department of Energy, Office of Science under contract DE-AC02- 05CH11231. The work of DS was supported by DOE grant DOE-SC0010008. RW is supported by FRS-FNRS (Belgian National Scientific Research Fund) IISN projects 4.4503.16. This research was supported in part through the Maxwell computational resources operated at Deutsches Elektronen-Synchrotron DESY, Hamburg, Germany.

References

[1] ATLAS collaboration, *Operation of the ATLAS trigger system in Run 2*, J. Instr. **15**, P10004 (2020), doi:10.1088/1748-0221/15/10/P10004.

[2] V. Khachatryan et al., *The CMS trigger system*, J. Instr. **12**, P01020 (2017), doi:10.1088/1748-0221/12/01/P01020.

[3] ATLAS collaboration, *Technical design report for the phase-II upgrade of the ATLAS TDAQ system*, CERN (2017), doi:10.17181/CERN.2LBB.4IAL.

[4] CMS collaboration, *The phase-2 upgrade of the CMS DAQ interim technical design report*, CERN (2017), https://cds.cern.ch/record/2283193.

[5] S. Cittolin, A. Rácz and P. Sphicas, *CMS the TriDAS project: Technical design report, volume 2: Data acquisition and high-level trigger. CMS trigger and data-acquisition project*, CERN (2002), http://cds.cern.ch/record/578006.

[6] ATLAS collaboration, *ATLAS level-1 trigger: Technical design report*, CERN (1998), https://cds.cern.ch/record/381429.

[7] LHCb collaboration, *LHCb trigger and online upgrade technical design report*, CERN (2014), https://cds.cern.ch/record/1701361.

[8] P. Antonioli, A. Kluge and W. Riegler, *Upgrade of the ALICE readout & trigger system*, CERN (2013), https://cds.cern.ch/record/1603472.

[9] R. Aaij et al., *Tesla: An application for real-time data analysis in high energy physics*, Comput. Phys. Commun. **208**, 35 (2016), doi:10.1016/j.cpc.2016.07.022.

[10] V. Khachatryan et al., *Search for narrow resonances in dijet final states at $\sqrt{s} = 8$ TeV with the novel CMS technique of data scouting*, Phys. Rev. Lett. **117**, 031802 (2016), doi:10.1103/PhysRevLett.117.031802.

[11] M. Aaboud et al., *Search for low-mass dijet resonances using trigger-level jets with the ATLAS detector in pp collisions at $\sqrt{s} = 13$ TeV*, Phys. Rev. Lett. **121**, 081801 (2018), doi:10.1103/PhysRevLett.121.081801.

[12] R. Aaij et al., *A comprehensive real-time analysis model at the LHCb experiment*, J. Instr. **14**, P04006 (2019), doi:10.1088/1748-0221/14/04/P04006.

[13] L. Lonnblad, C. Peterson and T. Rognvaldsson, *Finding gluon jets with a neural trigger*, Phys. Rev. Lett. **65**, 1321 (1990), doi:10.1103/PhysRevLett.65.1321.

[14] CMS Collaboration, *Identification of heavy-flavour jets with the CMS detector in pp collisions at 13 TeV*, J. Instr. **13**, P05011 (2018), doi:10.1088/1748-0221/13/05/p05011.

[15] J. Duarte, S. Han, Harris et al., *Fast inference of deep neural networks in FPGAs for particle physics*, J. Instr. **13**, P07027 (2018), doi:10.1088/1748-0221/13/07/p07027.

[16] N. Nottbeck, D. C. Schmitt and P. D. V. Büscher, *Implementation of high-performance, sub-microsecond deep neural networks on FPGAs for trigger applications*, J. Instr. **14**, P09014 (2019), doi:10.1088/1748-0221/14/09/p09014.

[17] S. Summers, G. D. Guglielmo, J. Duarte et al., *Fast inference of boosted decision trees in FPGAs for particle physics*, J. Instr. **15**, P05026(2020), doi:10.1088/1748-0221/15/05/p05026.

[18] CMS Collaboration, *The phase-2 upgrade of the CMS level-1 trigger*, CERN (2020), https://cds.cern.ch/record/2714892.

[19] T. Hong, B. Carlson, B. Eubanks, S. Racz, S. Roche, J. Stelzer and D. Stumpp, *Nanosecond machine learning event classification with boosted decision trees in FPGA for high energy physics*, J. Instr. **16**, P08016 (2021), doi:10.1088/1748-0221/16/08/p08016.

[20] T. Aarrestad, V. Loncar, N. Ghielmetti et al., *Fast convolutional neural networks on FPGAs with hls4ml*, Mach. Learn.: Sci. Technol. **2**, 045015 (2021), doi:10.1088/2632-2153/ac0ea1.

[21] A. M. Deiana et al., *Applications and techniques for fast machine learning in science*, arXiv:2110.13041.

[22] J. Townsend, T. Bird and D. Barber, *Practical lossless compression with latent variables using bits back coding*, arXiv:1901.04866.

[23] F. Mentzer, E. Agustsson, M. Tschannen, R. Timofte and L. Van Gool, *Practical full resolution learned lossless image compression*, arXiv:1811.12817.

[24] E. Hoogeboom, J. W. T. Peters, R. van den Berg and M. Welling, *Integer discrete flows and lossless compression*, arXiv:1905.07376.

[25] J. Ho, E. Lohn and P. Abbeel, *Compression with flows via local bits-back coding*, arXiv:1905.08500.

[26] R. van den Berg, A. A. Gritsenko, M. Dehghani, C. Kaae Sønderby and T. Salimans, *IDF++: Analyzing and improving integer discrete flows for lossless compression*, arXiv:2006.12459.

[27] S. Carrazza, J. M. Cruz-Martinez and T. R. Rabemananjara, *Compressing PDF sets using generative adversarial networks*, Eur. Phys. J. C **81**, 530 (2021), doi:10.1140/epjc/s10052-021-09338-8.

[28] D. J. Rezende and S. Mohamed, *Variational inference with normalizing flows*, arXiv:1505.05770.

[29] L. Ardizzone et al., *Analyzing inverse problems with invertible neural networks*, arXiv:1808.04730.

[30] G. Papamakarios, E. Nalisnick, D. J. Rezende, S. Mohamed and B. Lakshminarayanan, *Normalizing flows for probabilistic modeling and inference*, arXiv:1912.02762.

[31] I. Kobyzev, S. J. Prince and M. A. Brubaker, *Normalizing flows: An introduction and review of current methods*, IEEE **43**, 3964 (2021), doi:10.1109/tpami.2020.2992934.

[32] B. Nachman and D. Shih, *Anomaly detection with density estimation*, Phys. Rev. D **101**, 075042 (2020), doi:10.1103/PhysRevD.101.075042.

[33] E. Bothmann, T. Janßen, M. Knobbe, T. Schmale and S. Schumann, *Exploring phase space with neural importance sampling*, SciPost Phys. **8**, 069 (2020), doi:10.21468/SciPostPhys.8.4.069.

[34] C. Gao, J. Isaacson and C. Krause, *i-flow: High-dimensional integration and sampling with normalizing flows*, Mach. Learn. Sci. Tech. **1**, 045023 (2020), doi:10.1088/2632-2153/abab62.

[35] C. Gao, S. Höche, J. Isaacson, C. Krause and H. Schulz, *Event generation with normalizing flows*, Phys. Rev. D **101**, 076002 (2020), doi:10.1103/PhysRevD.101.076002.

[36] M. Bellagente, A. Butter, G. Kasieczka, T. Plehn, A. Rousselot, R. Winterhalder, L. Ardizzone and U. Köthe, *Invertible networks or partons to detector and back again*, SciPost Phys. **9**, 074 (2020), doi:10.21468/SciPostPhys.9.5.074.

[37] B. Stienen and R. Verheyen, *Phase space sampling and inference from weighted events with autoregressive flows*, SciPost Phys. **10**, 038 (2021), doi:10.21468/SciPostPhys.10.2.038.

[38] R. Winterhalder, M. Bellagente and B. Nachman, *Latent space refinement for deep generative models*, arXiv:2106.00792.

[39] C. Krause and D. Shih, *CaloFlow: Fast and accurate generation of calorimeter showers with normalizing flows*, arXiv:2106.05285.

[40] A. Hallin et al., *Classifying anomalies through outer density estimation*, Phys. Rev. D **106**, 055006 (2022), doi:10.1103/PhysRevD.106.055006.

[41] C. Krause and D. Shih, *CaloFlow II: Even faster and still accurate generation of calorimeter showers with normalizing flows*, arXiv:2110.11377.

[42] R. Winterhalder, V. Magerya, E. Villa, S. P. Jones, M. Kerner, A. Butter, G. Heinrich and T. Plehn, *Targeting multi-loop integrals with neural networks*, SciPost Phys. **12**, 129 (2022), doi:10.21468/SciPostPhys.12.4.129.

[43] A. Butter, S. Diefenbacher, G. Kasieczka, B. Nachman and T. Plehn, *GANplifying event samples*, SciPost Phys. **10**, 139 (2021), doi:10.21468/SciPostPhys.10.6.139.

[44] M. Bellagente, M. Haußmann, M. Luchmann and T. Plehn, *Understanding event-generation networks via uncertainties*, SciPost Phys. **13**, 003 (2022) doi:10.21468/SciPostPhys.13.1.003.

[45] A. Butter, T. Heimel, S. Hummerich, T. Krebs, T. Plehn, A. Rousselot and S. Vent, *Generative networks for precision enthusiasts*, arXiv:2110.13632.

[46] G. Papamakarios, T. Pavlakou and I. Murray, *Masked autoregressive flow for density estimation*, arXiv:1705.07057.

[47] M. Germain, K. Gregor, I. Murray and H. Larochelle, *MADE: Masked autoencoder for distribution estimation*, arXiv:1502.03509.

[48] A. Paszke et al., *PyTorch: An imperative style, high-performance deep learning library*, Adv. Neural Inf. Process. Syst. **32**, 8024 (2019).

[49] D. P. Kingma and J. Ba, *Adam: A method for stochastic optimization*, arXiv:1412.6980.

[50] P. Izmailov, D. Podoprikhin, T. Garipov, D. Vetrov and A. Gordon Wilson, *Averaging weights leads to wider optima and better generalization*, arXiv:1803.05407.

[51] S. Bollweg, M. Haußmann, G. Kasieczka, M. Luchmann, T. Plehn and J. Thompson, *Deep-learning jets with uncertainties and more*, SciPost Phys. **8**, 006 (2020), doi:10.21468/SciPostPhys.8.1.006.

[52] G. Kasieczka, M. Luchmann, F. Otterpohl and T. Plehn, *Per-object systematics using deep-learned calibration*, SciPost Phys. **9**, 089 (2020), doi:10.21468/SciPostPhys.9.6.089.

[53] P. Virtanen et al., *SciPy*, Nat. Methods **17**, 261 (2020), doi:10.1038/s41592-019-0686-2.

[54] G. Choudalakis, *On hypothesis testing, trials factor, hypertests and the BumpHunter*, arXiv:1101.0390.

[55] J. Collins, K. Howe and B. Nachman, *Anomaly detection for resonant new physics with machine learning*, Phys. Rev. Lett. **121**, 241803 (2018), doi:10.1103/physrevlett.121.241803.

[56] J. H. Collins, K. Howe and B. Nachman, *Extending the search for new resonances with machine learning*, Phys. Rev. D **99**, 014038 (2019), doi:10.1103/physrevd.99.014038.

[57] G. Kasieczka et al., *The LHC olympics 2020: A community challenge for anomaly detection in high energy physics*, Rep. Prog. Phys. **84**, 124201 (2021), doi:10.1088/1361-6633/ac36b9.

[58] T. Sjöstrand, S. Mrenna and P. Skands, *A brief introduction to Pythia 8.1*, Comput. Phys. Commun. **178**, 852 (2008), doi:10.1016/j.cpc.2008.01.036.

[59] J. de Favereau, C. Delaere, P. Demin, A. Giammanco, V. Lemaître, A. Mertens and M. Selvaggi, *Delphes 3: A modular framework for fast simulation of a generic collider experiment*, J. High Energy Phys. **02**, 57 (2014), doi:10.1007/jhep02(2014)057.

[60] A. Mertens, *New features in Delphes 3*, J. Phys.: Conf. Ser. **608**, 012045 (2015), doi:10.1088/1742-6596/608/1/012045.

[61] M. Selvaggi, *Delphes 3: A modular framework for fast-simulation of generic collider experiments*, J. Phys.: Conf. Ser. **523**, 012033 (2014), doi:10.1088/1742-6596/523/1/012033.

[62] M. Cacciari, G. P. Salam and G. Soyez, *Fastjet user manual*, Eur. Phys. J. C **72**, 1896 (2012), doi:10.1140/epjc/s10052-012-1896-2.

[63] M. Cacciari, G. P. Salam and G. Soyez, *The anti-k_t jet clustering algorithm*, J. High Energy Phys. **04**, 063 (2008), doi:DOI10.1088/1126-6708/2008/04/063

[64] A. Andreassen, B. Nachman and D. Shih, *Simulation assisted likelihood-free anomaly detection*, Phys. Rev. D **101**, 095004 (2020), doi:10.1103/PhysRevD.101.095004.

[65] J. Thaler and K. Van Tilburg, *Maximizing boosted top identification by minimizing n-subjettiness*, J. High Energy Phys. **02**, 93 (2012), doi:10.1007/jhep02(2012)093.

[66] J. Thaler and K. Van Tilburg, *Identifying boosted objects with n-subjettiness*, J. High Energy Phys. **03**, 15 (2011), doi:10.1007/jhep03(2011)015.

[67] E. M. Metodiev, B. Nachman and J. Thaler, *Classification without labels: Learning from mixed samples in high energy physics*, J. High Energy Phys. **10**, 174 (2017), doi:10.1007/jhep10(2017)174.