Creating Simple, Interpretable Anomaly Detectors for New Physics in Jet Substructure

Layne Bradshaw\(\ast\) and Spencer Chang\(\dagger\)

Department of Physics and Institute for Fundamental Science, University of Oregon, Eugene, OR 97403

Anomaly detection with convolutional autoencoders is a popular method to search for new physics in a model-agnostic manner. These techniques are powerful, but they are still a “black box,” since we do not know what high-level physical observables determine how anomalous an event is. To address this, we adapt a recently proposed technique by Faucett et al., which maps out the physical observables learned by a neural network classifier, to the case of anomaly detection. We propose two different strategies that use a small number of high-level observables to mimic the decisions made by the autoencoder on background events, one designed to directly learn the output of the autoencoder, and the other designed to learn the difference between the autoencoder’s outputs on a pair of events. Despite the underlying differences in their approach, we find that both strategies have similar ordering performance as the autoencoder and independently use the same six high-level observables. From there, we compare the performance of these networks as anomaly detectors. We find that both strategies perform similarly to the autoencoder across a variety of signals, giving a nontrivial demonstration that learning to order background events transfers to ordering a variety of signal events.
similar, one has to work to exploit differences in the underlying probability distributions, and many techniques have been developed to highlight those differences \[2\]–\[24\]. However, one often expects there to be qualitative differences between signal and background. In that case, there are a variety of methods that can determine whether events are anomalous or not on an event-by-event basis \[25\]–\[55\].

Machine learning (ML) techniques, including unsupervised anomaly detection, typically make use of low-level, high-dimensional data. This is in contrast to human-engineered strategies, which tend to use high-level, low-dimensional data. When the two perform equally well on a given task, we tend to assume that the ML strategy must have used some combination of its low-level inputs to create an approximation of the high-level variables used by humans. It could be, however, that the ML strategy has found an alternative that is just as efficient. Unfortunately, the “black box” nature of ML techniques make it difficult to understand what the machine is actually learning. This problem is only amplified when the ML strategy outperforms the human-engineered one. Has the machine learned a simple observable humans didn’t consider or has it perhaps found something new?

There have been efforts to understand a neural network by using existing high level observables \[56\]–\[60\], as well as “knowledge distillation” techniques to gain insights about complex networks by analyzing simpler ones \[61\]–\[64\]. In a recent paper (Ref. \[65\]), a promising iterative technique was introduced to build an interpretable classifier. This classifier mimics a “black box” deep neural network classifier, where the mimicker’s inputs consists of a limited set of human-interpretable high-level variables (see also \[66\]–\[67\]). In this paper, we extend this technique to anomaly detectors by presenting two strategies for mapping the low-level information utilized by an anomaly detector into a handful of simple to understand high-level observables. As a concrete example, we attempt to mimic both the decisions and performance of an anomaly detector based on a convolutional autoencoder, which is trained on background jet images. The convolutional autoencoder then helps to iteratively select high-level observables that serve as the inputs to the mimicker networks. As our pool of high-level observables, we use the Energy Flow Polynomials \[68\] because they form a basis for all infrared- and collinear-safe observables.

We introduce two strategies to mimic an autoencoder. The first strategy, the High-Level Network, uses a small number of high-level observables to match the autoencoder’s anomaly score on an event-by-event basis. The other strategy, the Paired Neural Network, is tasked with using a potentially different set of observables to learn to make the same ordering decisions as the autoencoder. Given a pair of events, the Paired Neural Network learns which of the two was deemed to be less anomalous by the autoencoder. Note that like the convolutional autoencoder we want to mimic, both the Paired and High-Level neural networks are only trained on background events and so are unsupervised with respect to signal events. Despite their philosophical differences, we find that both strategies agree on which high-level observables are useful for ordering background events like the autoencoder. These two strategies also have comparable performance, where we find that they both make the same ordering decisions as the autoencoder ∼83% of the time.

Since these networks are unsupervised, applying these networks as anomaly detectors allows us to test whether the decision ordering on background events transfers to signal events. Interestingly, for seven of the eight different signals we consider, we find that the mimickers perform as well or better as anomaly detectors than the autoencoder. Thus, this shows that it is possible to create interpretable anomaly detectors that have a limited number of high-level inputs without compromising performance. This reduction of complexity is an obvious advantage for experimental applications of anomaly detection, reducing work needed for variable validation and determination of systematic uncertainties. Theoretically, this result gives insights into the features of a QCD jet image which are harder to compress into a lower dimensional latent space.

This paper is outlined as follows. In Sec. II we describe the Monte Carlo generated dataset, as well as the relevant selection criteria and preprocessing. Sec. III starts by describing the details of the convolutional autoencoder. We then review all of the pieces needed to mimic the autoencoder—the pool of high-level observables we use to explain the autoencoder, a metric to determine how similar the decisions of two networks are, the details of our two simplified anomaly detectors, and the iterative procedure we use to construct the mimickers from the pool of high-level observables. We present our results in Sec. IV, detailing the construction and performance of the mimickers. Finally we conclude in Sec. V. Details of the simulated events and network training hyperparameters appear in the appendices.

II. Datasets

In this section, we briefly describe the simulated datasets we use in this study. In particular, our focus is on anomaly detection in boosted jets at the LHC. We utilize the publicly available datasets pro-
vided by Ref. [38], using QCD dijet events [69] as background and W, top, and Higgs jets [70] as the anomalous events. We consider four different W masses, \( m_W = 59, 80, 120, 174 \text{ GeV} \), two different top masses, \( m_t = 80, 174 \text{ GeV} \), and two different Higgs masses, \( m_h = 20, 80 \text{ GeV} \). Note that when \( m_t = 80 \text{ GeV} \), the mass of the decay product \( W \) is set to 20 GeV. The full simulation details are given in App. [A]. These signals give a broad range of signals with varying amounts of substructure (two to four prongs), which will prove useful when testing the ability of our anomaly detectors.

These datasets contain approximately 700,000 QCD dijet events and 100,000 events for each of the W, top, and Higgs signals. After applying a \( p_T \) cut (see App. [A]), we are left with \(~150,000\) QCD events and \(~30,000\) events for each of the anomalous signals. We use \( 2/3 \) of the QCD dijet events for training the autoencoder, with the remaining \( 1/3 \) being reserved for testing and validation. We are not considering training on real data at this point, so we do not include the possibility of contamination in the background set from signal samples when training the autoencoder. However, previous work has shown that autoencoders are robust to up to \(~10\%\) signal contamination [28–30, 71].

Our procedure for preprocessing the raw four vectors into images follows that outlined in Ref. [72] and is implemented with the EnergyFlow package [73]. For the leading jet in each event, we boost and rotate along the beam direction, such that the \( p_T \) weighted centroid lies at \((\eta, \phi) = (0, 0)\). The jet is then rotated about its centroid until its principal axis lies along the vertical. Finally, the jet is reflected about the horizontal and vertical axes so that the maximum intensity lies in the upper-right quadrant. Only after centering, rotating, and reflecting the jet do we pixellate the image. Our final pixellated images are \( 40 \times 40 \), covering \( \Delta \eta = \Delta \phi = 2.0 \). The last step of our preprocessing procedure is to divide by the total \( p_T \) in the image. This final normalization step ensures that each image has the same scale, which helps with training. Figure 1 shows the average jet image for the background, \( 80 \text{ GeV } W \), \( 174 \text{ GeV } \) top, and \( 80 \text{ GeV } \) Higgs.

### III. Methodology

While neural networks have been used for classification and anomaly detection with great success, they are often viewed as black boxes, leading one to wonder what information they are using to match or outperform traditional techniques. With this in mind, the authors of Ref. [65] showed that modern classification networks are able to be mimicked by interpretable networks using a few high level physics variables as inputs. In this work, we adapt this method to the task of anomaly detection. In order to do this, we first need a good anomaly detector to mimic with physics variables.

#### A. Creating a Target Anomaly Detector with a Convolutional Autoencoder

The anomaly detector we chose is a convolutional autoencoder (hereafter referred to as the AE). Given an input image, the AE is tasked with encoding the image down into a smaller latent space, then reconstructing the original image from its latent space representation. The idea behind compressing the data to a smaller representation is that it forces the network to learn what is important about the jet image, while ignoring noisy or less crucial aspects. The hope is that when the autoencoder is applied to anomalous data, the important characteristics will be different, and thus the image will be poorly encoded, leading to a decoded image which is quite different from the initial image. Thus, we can distinguish between the background data and the anomalous signal data by the size of the reconstruction error. AEs were first introduced to the high energy community as anomaly detectors in Refs. [27, 29].

\[ \text{Often, AEs can be improved with Variational Autoencoders (VAEs), in which the latent space representation becomes} \]
The architecture of our AE is shown in Fig. 2 and is described below. The encoder consists of multiple layers. The first two layers are a set of five $3 \times 3$ pixel convolutional filters. We use a stride of one and pad the output to keep the same height and width as the original image. After each convolution we apply an exponential linear unit (ELU) activation \[71\]. Following these convolutions, the representation is down sampled with a $2 \times 2$ max pooling layer, leading to a height and width of 20 pixels. This reduced image is then passed through another two convolutional layers with five filters each. All of the convolutional layers in the decoder use a $3 \times 3$ convolutional kernel and the ELU activation function, with the exception of the last convolutional layer in the decoder, which uses the SOFTMAX activation function along the pixel dimension so that the sum of the pixel intensities is unity. These are then upsampased with a transposed convolutional layer to $40 \times 40$, passed through a convolutional layer with 5 filters, and finally passed through one last convolutional layer to create the output image. We train the AE to reproduce QCD jet images, by minimizing the mean squared error of their reconstruction. Explicitly, this is given as

\[
L_{AE} = \frac{1}{N_iN_p} \sum_{k} \sum_{j} \left( f_A(I^j_k) - I^j_k \right)^2
\]

where $N_i$ is the total number of images, $N_p$ is the number of pixels in each image, $I^j_k$ is the $j$th pixel of the $k$th input image, and $f_A(I^j_k)$ is the AE’s reconstruction of that pixel for that input image. The training details for the AE are provided in App. \[13\]. Our AE, along with all of the other neural network architectures discussed in Sec. \[11\] are implemented with KERAS \[72\] using the TENSORFLOW \[76\] backend.

Figure 3 shows some examples of how the trained AE can act as an anomaly detector. The left panels display the distribution of the reconstruction errors as the anomaly score for the background training data. The right panels display the ROC curves for each signal.
set as well as three different anomalous signals. At first glance, the reconstruction errors are very small, but this is explained by the normalization and the sparsity of our jet images. Because each image is normalized to sum to one, all pixels have a value of less than one. The images are also very sparse, so most pixels are identically 0, and the network is very good at predicting that. When we take the mean squared error over the pixels, we actually average over the number of pixels, so the number of pixels with no intensity leads to a very good average reconstruction. Importantly, we see that the background distribution is at lower scores than the signal distributions. The encoder has never seen jets with inherent substructure from the decay of a heavy resonance, so it doesn’t recognize the important information to encode into the latent space, and the decoder therefore performs worse when reconstructing the images. The right panel displays the Receiver Operating Characteristic (ROC) curves for these three signals. While the W is harder for the AE to distinguish from the background, the top and Higgs jets have decent Area Under the ROC Curve (AUC) scores.

As we’ve seen, our constructed AE is capable of detecting jets which are different from the QCD background it was trained on. In the next section we build up a method to mimic the ordering decisions the AE makes using physics observables.

B. Mimicking the Target Anomaly Detector

As shown in the previous section, the AE is able to tag various signals as being different from QCD. However, it is unclear what information in the event image is being used to do this. In order to mimic the behavior of the AE, we need a few ingredients. The first is a wide set of physics observables which could possibly explain the anomaly detector. For these, we use the Energy Flow Polynomials, described in detail in Sec. III B 1. Next, we use the idea of decision ordering to select which observables are important as described in Sec. III B 2. Finally, we need a flexible function which can use the physics observables to produce an anomaly score which mimics that of the AE. We describe two complementary methods which achieve this goal. The first method, a Paired Neural Network, is a neural network which takes in the physics observables from two events at the same time and is trained to determine which event had the worse reconstruction error from the AE. We construct this in such a way that at inference, we can feed in a single event and get an anomaly score. This technique is described in Sec. III B 3. The second method, a High-Level Neural Network, instead takes in only a single event at a time and is trained to regress the reconstruction error of the AE for that event. This second method is described in Sec. III B 4.

1. High Level Observables

Since there is no way to know which human-constructed, high-level observables will be relevant a priori, we need to rely on using a basis of observables. To that end, we make use of the Energy Flow Polynomials (EFPs) [65], a formally infinite set of jet substructure observables inspired by previous work on energy correlation functions [77–82]. The EFPs form a discrete linear basis for all infrared- and collinear-safe (IRC-safe) observables and are defined in terms of the momentum fraction, $z_a$, and pairwise angular distances, $\theta_{ab}$. The EFPs are computed using the four-momentum of each particle in the jet, where $z_a$ is the momentum fraction carried by particle $a$, and $\theta_{ab}$ is the pairwise angular distance between particles $a$ and $b$. Each EFP is conveniently represented by a multigraph, using the following correspondences:

$$\text{each node } a \leftrightarrow \sum_{a=1}^{N} z_a$$  \hspace{1cm} (2)

and

$$\text{each } k\text{-fold edge between nodes } a \text{ and } b \leftrightarrow (\theta_{ab})^k.$$  \hspace{1cm} (3)

As an example, we have

$$\sum_{a=1}^{N} \sum_{b=1}^{N} \sum_{c=1}^{N} \sum_{d=1}^{N} z_a z_b z_c z_d \theta_{ab}^2 \theta_{ac} \theta_{bc} \theta_{cd}^3.$$  \hspace{1cm} (4)

In this example, we’ve labeled the nodes for clarity, but will not do so for future graphs. To build some intuition for this framework, we note that the fully connected graphs with $N$ vertices correspond to the $N$–point energy correlation functions.

The EFPs corresponding to each multigraph can be modified with a pair of parameters, $(\kappa, \beta)$, which determine the precise meaning of $z_a$ and $\theta_{ab}$. More specifically,

$$z_a^{(\kappa)} = \left( \frac{pT_a}{\sum_b pT_b} \right)^\kappa,$$  \hspace{1cm} (5)

$$\theta_{ab}^{(\beta)} = \left( \Delta \eta_{ab}^2 + \Delta \phi_{ab}^2 \right)^{\beta/2}$$  \hspace{1cm} (6)

where $pT_a$ is the transverse momentum of particle $a$, $\Delta \eta_{ab}$ is the difference in pseudorapidity between particles $a$ and $b$, and $\Delta \phi_{ab}$ is the difference in azimuthal angle between particles $a$ and $b$. The original IRC-safe EFPs require $\kappa = 1$.
well-motivated reasons to explore a broader space of observables at the cost of IR and/or C safety [32,33], we restrict ourselves to only IRC-safe observables in this work. For our iterative procedure to mimic the AE, we choose \( \alpha = 1, \beta = 1 \), and consider all EFPs with degree (i.e. the number of edges) \( d \leq 5 \). With these parameters, we have a total of 102 EFPs to explore.

2. Decision Ordering

To create an interpretable alternative to the AE, we will iteratively add EFP observables as inputs to the mimicking networks. To compare how well a network (or EFP input) orders events relative to the AE, we use a series of metrics implemented in Ref. [65]. Here we briefly summarize these metrics. Given two decision functions, \( f(x) \) and \( g(x) \), the decision ordering (DO) for a pair of events \( x_1 \) and \( x_2 \) is defined as

\[
\text{DO}[f,g](x_1,x_2) = \Theta\left( \left[ (f(x_1) - f(x_2))^2 + (g(x_1) - g(x_2))^2 \right] \right)
\]

where \( \Theta(x) \) is the Heaviside theta function, and we choose \( \Theta(0) = 1 \). Here, we can think of \( f(x) \) as being the anomaly score/reconstruction error for the AE and \( g(x) \) being the output of one of our methods. Later, we will also use \( f(x) = \text{AE}(x) \) and \( g(x) = \text{EFP}(x) \) to determine which EFP observables to include for our mimickers. A DO of 1 means that \( f \) and \( g \) agree that one event is more anomalous than another; a DO of 0 indicates the two methods disagree on which event is more anomalous. If two decision functions have DO = 1 for all possible pairs \( x_1 \) and \( x_2 \), then the two are effectively identical decision functions on the domain tested.

To create a summary statistic, we then average the DO over all possible pairs, weighted by the underlying distributions that \( x_1 \) and \( x_2 \) are drawn from. The resulting statistic, the average decision ordering (ADO) is given by

\[
\text{ADO}[f,g] = \int dx_1 dx_2 p_1(x_1)p_2(x_2) \text{DO}[f,g](x_1,x_2)
\]

This evaluates to 1 if both decision functions order every possible pair of events in the same manner (making them equivalent decision functions), 0 if they order the pairs in the opposite manner, and \( \frac{1}{2} \) if there is no consistency to the way the decision functions order the events. Due to computing constraints, we could not compute the ADO on the entirety of the background training set. Instead, when computing the ADO, we choose 10,000 events at random, and then evaluate on the \( \binom{10000}{2} \sim 5 \times 10^7 \) pairs of events.

We now follow the Black-Box Guided Search Strategy from Ref. [65] to iteratively construct neural networks whose decision functions should become better and better approximations of the AE's. We start by training a neural network, \( \text{NN}_0 \) on some initial set of observables, \( X_0 = (m_T,p_T) \). We will later describe the two possible architectures for \( \text{NN}_0 \), but for now it is enough to say it aims to produce decision functions that mimic the AE on background events. We then compute the ADO between \( \text{NN}_0 \) and the AE, and isolate all of the pairs of events misordered by \( \text{NN}_0 \). From our set of high-level observables, \( O \), we then want to find the observable \( O_1 \in O \) with the highest ADO on the pairs misordered by \( \text{NN}_0 \). We then train a new neural network, \( \text{NN}_1 \), whose input observables are \( X_1 = X_0 \cup O_1 \). Given its inputs, we would expect \( \text{NN}_1 \) to have a decision function that more closely resembles that of the AE—and consequently, a higher ADO compared to \( \text{NN}_0 \)—since it has access to the same information \( \text{NN}_0 \) had, as well as information that can help order the pairs misordered by \( \text{NN}_0 \).

From here, we continue to iterate using the remaining observables in \( O \). On the \( n \)th iteration, we start by finding the observable \( O_n \in O \) with the highest ADO on the pairs misordered by \( \text{NN}_{n-1}(X_{n-1}) \) that is not already part of \( X_{n-1} \). We then build a new set of inputs, \( X_n = X_{n-1} \cup O_n \), and train a new neural network, \( \text{NN}_n \), on \( X_n \). At each iteration, we expect the ADO between the neural network and AE to increase, since the neural network we construct on the \( n \)th iteration has access to all of the same information available to the previous network, as well as a new observable \( O_n \) that helps order the events misordered by the \((n-1)\)th neural network.

Now that we have described both the physics observables and the general method for choosing which observables to give the networks, we describe the two network architectures in more detail.

3. Paired Neural Network

Our first attempt to mimic the AE is an approach we call the Paired Neural Network (PNN). The aim of the Paired Neural Network is to mimic the AE by learning to predict the relative anomaly score between two events. To do this, the PNN takes pairs of events as its input and classifies which has a larger anomaly score. This is in contrast to other methods such as trying to match the AE’s output or anomaly

\[2\] If the ADO of an observable is less than 0.5, we take 1-ADO, since a highly anticorrelated variable is also useful.
FIG. 4. The architecture of the Paired Neural Network. The interior model consists of 4 hidden layers each with 50 nodes and using the ELU activation function. The interior model outputs a single node for each input and uses the ReLU activation function. The final output of the model is a single node which is the difference between the two interior model outputs and uses a sigmoid activation function. Our input data are the jet’s mass, $p_T$, and up to 14 EFPs.

score on an event-by-event basis. In general, classifiers are easier to train, so this seems like a promising method.

Figure 4 shows the PNN architecture. Both events are fed through the same interior model in parallel. This is shown in the image as the “Common Interior Model.” The interior model consists of four hidden layers with 50 nodes each, and the ELU activation function is used for all layers. The interior model produces a single output for each input event, and this single output node uses the ReLU activation. The motivation for this is to think of the output for each event as its own anomaly score. Within the larger PNN, we then subtract these two output anomaly scores from each other. If the first event is more anomalous, the result should be negative and if the second is more anomalous, the result will be positive. The larger the difference in scores should tell us about the networks confidence in the relative ordering. Finally, to turn this into a classification problem, we apply the sigmoid function to the interior model difference, mapping large negative numbers to 0 and large positive values to 1. If the anomaly scores are the same (the difference is 0) the sigmoid gives a value of 0.5.

To train the network, we continue the idea of classification and minimize the binary cross-entropy given by

$$L_{PNN} = -\frac{1}{N} \sum_k^N \left[ y_k \ln \left( f_P(X_k) \right) + (1 - y_k) \ln \left( 1 - f_P(X_k) \right) \right]$$

where $k$ represents a specific pair of events, where the order matters. The value of $y_k$ is the truth “label” for the pair of events as determined by the AE, i.e. $y_k = 0 (1)$ if the AE determines the event in Input 1 to be more (less) anomalous than the event in Input 2, and $f_P(X_k)$ is the PNN’s output for the pair of events. Appendix B provides the training details for the PNN.

After training the PNN on $\sim$250,000 pairs of events, we extract the interior model for use on single events. Thus, even though the training procedure requires pairs of events and was trained as a classifier, the interior model provides a function which takes in observables from a single event and outputs an anomaly score.

4. High-Level Neural Network

The PNN described in the last section does not attempt to learn the actual anomaly score of the AE, but only the relative difference in the anomaly score between pairs of events. We also introduce a method which specifically aims to mimic the actual anomaly score of the AE. We call this network the High-Level Neural Network (HLN). In practice, the anomaly score (reconstruction error) from the AE spans many orders of magnitude, so we found better results when the HLN is trained to predict the log of the anomaly score rather than the score itself.

We find that a relatively simple neural network is able to achieve the task of reproducing the loss of the AE. Figure 5 shows the architecture we use for the HLN. The HLN consists of 4 hidden layers, with each hidden layer having 50 nodes. The final output of the network is a single node. All of the nodes in the hidden layers use the ELU activation function.

To train the HLN, we minimize the mean squared error between the (log of the) anomaly score of the
AE and the output of the HLN. Specifically, we use a loss function of,

$$L_{\text{HLN}} = \frac{1}{N} \sum_{k} \left[f_H(X_k) - \ln \left( \frac{1}{N_p} \sum_{j} (f_A(I^k_j) - I^k_j)^2 \right) \right]^2$$

(10)

where $f_H(X_k)$ is the HLN’s output given some input data $X_k$ and $f_A(I^k_j)$ is the AE’s output given a pixel $j$ in an image $I^k_j$ for the $k$th event. When using the HLN as an anomaly detector, we use the model’s output as the anomaly score. See App. [3] for the HLN training details.

IV. Results

In the previous section, we outlined two different architectures we could use to iteratively build neural networks whose decision functions would more closely resemble the AE’s decision function. Here, we provide the results of the iterative procedure and analyze the specific EFPs that are selected to mimic the anomaly detector. We will find that the EFPs selected are composite observables built out of only six prime EFP factors. We show that using only the prime components gives very similar results. Finally, we demonstrate that using the EFPs with a traditional anomaly detection technique, the isolation forest, gives very poor results. The failure of the isolation forest when provided with the same basic physics observables highlights the benefits of using our mimicker networks.

A. Background Decision Ordering

We start our iterative process by training both a PNN and HLN on jet mass and $p_T$ for QCD events in the training set and then compute the ADO for each model. Of the $\sim 5 \times 10^7$ pairs of events we use to compute the ADO, both the initial PNN and HLN correctly order $\sim 72\%$ of the events relative to the reconstruction error of the AE. Next, we take all of the pairs which are misordered and compute the ADO between all 102 EFPs and the AE. On this iteration, we find that the observable with the highest ADO between all of the pairs which are misordered and compute the reconstruction error of the AE. Next, we take all of the pairs of events we used to determine the next best observable to add; the shaded bands show the maximum and minimum ADO values obtained when recalculating the ADO an additional 50 times, using a different set of pairs of events each time. The $x$–axis denotes the iteration step of the iterative process. See Table I for the multi-graph and mathematical representations of the selected EFPs and the iteration step at which they were added. The blue ‘+’ (orange ‘×’) shows the ADO of a PNN (HLN) trained on only the 5 prime EFPs picked out by each method (see Eq. 12). The ADO of each model trained on $m, p_T$, and all of the $d \leq 5$ EFPs is the same to 3 significant digits, and is plotted as a single dashed line.

![FIG. 6. The ADOs for each PNN and HLN. The center line shows the ADO of the model that was used to select the EFPs. The shaded bands show the maximum and minimum ADO values obtained when recalculating the ADO an additional 50 times, using a different set of pairs of events each time. The $x$–axis denotes the iteration step of the iterative process. See Table I for the multi-graph and mathematical representations of the selected EFPs and the iteration step at which they were added. The blue ‘+’ (orange ‘×’) shows the ADO of a PNN (HLN) trained on only the 5 prime EFPs picked out by each method (see Eq. 12). The ADO of each model trained on $m, p_T$, and all of the $d \leq 5$ EFPs is the same to 3 significant digits, and is plotted as a single dashed line.](image)

for the pairs of events which are misordered by the respective networks.

Figure 2 shows the result of this iterative process. The solid lines show the ADO of the models we used to determine the next best observable to add; the shaded band shows the maximum and minimum value of the ADO for each model after recalculating the ADO an additional 50 times at each iteration using a different set of $\sim 5 \times 10^7$ pairs of events. We also created PNN and HLN models trained on $m, p_T$ and all $d \leq 5$ EFPs. The ADOs of these two models agree to 3 significant digits and thus is plotted as the single dashed line in the panel. Since they use all of the EFPs, this line gives a sense of the highest ADO each model is capable of achieving, given our set of observables. The blue ‘+’ and orange ‘×’ will be discussed in Sec. IV C. There are a few key takeaways from these plots. By the time the ADOs start to plateau, both the HLN and PNN are correctly ordering 83% of the pairs of events in the QCD sample relative to the AE. For the first two iterations, the model ADOs do not change. Looking at Table I, we see that the first two EFPs are EFP 2.

This observable is then added to the list of inputs. So in the next iteration the input for each event is given by $(m_j, p_T, \text{EFP 2})$. We then repeat this process 14 more times, recording both the ADO of each network, as well as which EFP has the largest ADO at Table I, we see that the first two EFPs are EFP 2.
and [EFP 2]², which are proportional to $m^2/p_T^2$ and $m^4/p_T^4$. Since the initial inputs to both the PNN and HLN are mass and $p_T$, these observables contain no new information, and thus it makes sense that the model ADO does not improve. This redundancy of information follows since the EFPs are a linear basis of substructure observables, whereas our neural networks can utilize nonlinear combinations of its inputs. Despite their underlying philosophical differences—the HLN are trying to match the AE’s anomaly score, while the PNN is trying to match the DO of the AE—both methods select the same set of 14 EFPs in the same order. In Table I, we list the multigraph and mathematical expression corresponding to each of these EFPs as well as the iteration step in which they were added. The agreement of the PNN and HLN approaches gives us confidence that these observables are important to detect jets which do not look like typical QCD jets. Also, since by the last iteration, the PNN and HLN have nearly reached the ADO of the dashed line, it suggests that the decision ordering of our mimickers has almost converged to what is possible with our set of EFPs.

B. Anomaly Detection

While both the HLN and PNN have demonstrated the ability to mimic the AE’s anomaly score on QCD events, it’s unclear if matching the decision ordering on in-distribution events will generalize to out-of-distribution events. In other words, having mimicked the AE on QCD background events with HLN and PNNs, we must test if this decision ordering transfers to boosted jet signals by comparing the AE, PNN, and HLN as anomaly detectors. To determine how well each network performs as an anomaly detector we use a popular metric, the AUC.

Figure 7 shows how the HLN and PNN on their final iteration compare to the autoencoder on the same three signals as Fig. 6. The left panels show the normalized distributions of each network’s anomaly scores for the background and three of the signals. The right panel then shows all of the ROC curves for each model on each signal. We can see that both the HLN and PNN do a good job of mimicking the anomaly detector on events with higher anomaly scores. But the long tails in each of the background distributions indicates that HLN and PNN struggle to match the AE on less anomalous events, explaining their poorer background rejection at low signal efficiency.

Figure 8 shows how the mimickers perform on all eight signals described in Sec. II at each step of the iterative progress. The dashed black line in each panel shows the AUC when using the reconstruction error of the AE as the anomaly score. The blue and orange curves show the results of the PNN and HLN, respectively, as a function of the number of iterations for selecting extra observables. The solid center lines denote the AUC of the model used to select observables in the iterative process. The shaded bands show the maximum and minimum AUCs when retraining each network ten additional times, to give us a sense of how stable the training is. The bands are quite narrow, indicating that the results are robust to training uncertainties.

Like we saw with the ADOs in Fig. 6, the HLN and PNN perform similarly, despite their different approaches. For both the decision ordering and the AUCs, the results start to plateau around the fifth iteration. When the HLN and PNN AUC scores begin to plateau, we see that the value is similar to the AUC of the AE. This indicates that the HLN and PNN are performing comparably to the AE when all three are acting as anomaly detectors. It is surprising that mimicking the decision ordering on the in-distribution (QCD) events seems to also generalize to the relative differences between the signals and the background. Some of the mimicking networks even exceed the anomaly detection capability of the AE they are trying to mimic for certain signals.

For some signals—specifically the 20 GeV Higgs, 80 GeV $W$, 120 GeV $W$, and 174 GeV $W$—we see a drop in AUC around the 3rd iteration for both the PNN and HLN. While such dips are not ideal, they are not completely unexpected. Our iterative process is trying to pick out the observables that help to best order the background events, with no attention paid to how effective they may or may not be to picking out signal events. So, for those three signals, it appears that the EFP added at the iteration where the AUC dips improves the ADO relative to the AE, but at the same time makes it more difficult for the HLN and PNNs to distinguish those signal events from the background.

However, AUC is an inclusive figure of merit and, consequently, does not tell the whole story. As Fig. 7 highlights, networks with similar AUCs are not necessarily making the exact same decisions when used as anomaly detectors. Some more physically interpretable metrics are the background rejection ($1/\epsilon_B$) at fixed signal efficiency ($\epsilon_S$) and the signal efficiency at fixed background rejection. Table II shows the background rejection at two different fixed signal efficiencies—0.5 and 0.1—and the signal efficiency at two different fixed values of the background rejection—10 and 100—for all 8 signals and 5 different networks—the AE, HLN₀, PNN₀, HLN₁₄, and PNN₁₄.

There are a few key takeaways from this table. Looking at the signal efficiency at a fixed value of the background rejection, we can see that, in gen-
eral, our mimicker networks need to operate at lower signal efficiencies to achieve the same background rejection as the AE. The exceptions here are the final iteration of the mimicker networks when used as anomaly detectors for the 174 GeV Top and 80 GeV Higgs. These networks, when applied to these signals operate at comparable signal efficiencies to the AE lower fixed values of the background rejection. Shifting now to the background rejection at fixed signal efficiency, we see that our mimicker networks compare favorably to the AE at higher signal efficiencies across all of the anomalous signals we consider, but fall behind the AE at lower signal efficiencies. Again, the exception here are the mimicker networks applied to the 80 GeV Higgs. As was observed earlier in Fig. 7 as we make tighter cuts on our mimicker networks, forcing them to operate at lower signal efficiencies, they begin to deem the background as being more anomalous than the signal when compared to the autoencoder. While this type of behavior would be difficult to deal with in a real analysis, it is not unique to our mimicker networks and is a challenge with anomaly detection in general. The cuts that result in $\varepsilon_B > \varepsilon_S$ are highlighted in

| EFP No. | EFP Multigraph | EFP Expression | PNN Iteration | HLN Iteration |
|---------|----------------|----------------|---------------|---------------|
| 1 | ![EFP Multigraph](image1) | $\sum_{a,b=1}^{N} z_a z_b \theta_{ab}$ | 5 | 5 |
| 2 | ![EFP Multigraph](image2) | $\sum_{a,b=1}^{N} z_a z_b \theta_{ab}^2$ | 1 | 1 |
| 54 | ![EFP Multigraph](image3) | $\sum_{a,b,c,d=1}^{N} z_a z_b z_c z_d \theta_{ab} \theta_{cd}$ | 6 | 6 |
| 57 | ![EFP Multigraph](image4) | $\sum_{a,b,c,d=1}^{N} z_a z_b z_c z_d \theta_{ab}^2 \theta_{cd}$ | 2 | 2 |
| 65 | ![EFP Multigraph](image5) | $\sum_{a,b,c,d,e,f=1}^{N} z_a z_b z_c z_d z_e z_f \theta_{ab} \theta_{cd} \theta_{ef}$ | 3 | 3 |
| 70 | ![EFP Multigraph](image6) | $\sum_{a,b,c,d,e,f=1}^{N} z_a z_b z_c z_d z_e z_f \theta_{ab}^2 \theta_{cd} \theta_{ef}^2$ | 4 | 4 |
| 85 | ![EFP Multigraph](image7) | $\sum_{a,b,c,d,e,f,g=1}^{N} z_a z_b z_c z_d z_e z_f z_g \theta_{ab} \theta_{ac} \theta_{de} \theta_{fg}$ | 13 | 13 |
| 94 | ![EFP Multigraph](image8) | $\sum_{a,b,c,d,e,f,g,h=1}^{N} z_a z_b z_c z_d z_e z_f z_g z_h \theta_{ab} \theta_{ac} \theta_{de} \theta_{fg} \theta_{gh}$ | 11 | 11 |
| 95 | ![EFP Multigraph](image9) | $\sum_{a,b,c,d,e,f,g,h=1}^{N} z_a z_b z_c z_d z_e z_f z_g z_h \theta_{ab} \theta_{ac} \theta_{de} \theta_{fg} \theta_{gh}$ | 8 | 8 |
| 96 | ![EFP Multigraph](image10) | $\sum_{a,b,c,d,e,f,g,h=1}^{N} z_a z_b z_c z_d z_e z_f z_g z_h \theta_{ab} \theta_{ac} \theta_{de} \theta_{fg} \theta_{gh}$ | 12 | 12 |
| 97 | ![EFP Multigraph](image11) | $\sum_{a,b,c,d,e,f,g,h=1}^{N} z_a z_b z_c z_d z_e z_f z_g z_h \theta_{ab} \theta_{ac} \theta_{de} \theta_{fg} \theta_{gh}$ | 14 | 14 |
| 98 | ![EFP Multigraph](image12) | $\sum_{a,b,c,d,e,f,g,h=1}^{N} z_a z_b z_c z_d z_e z_f z_g z_h \theta_{ab} \theta_{ac} \theta_{de} \theta_{fg} \theta_{gh}$ | 10 | 10 |
| 99 | ![EFP Multigraph](image13) | $\sum_{a,b,c,d,e,f,g,h,i=1}^{N} z_a z_b z_c z_d z_e z_f z_g z_h z_i \theta_{ab} \theta_{ac} \theta_{de} \theta_{fg} \theta_{gh} \theta_{ij}$ | 9 | 9 |

TABLE I. The EFP multigraphs and corresponding expressions for each of the EFPs selected by both the HLN and PNN. In the last two columns, we list the iteration step where the PNN or HLN selects the corresponding EFP.
FIG. 7. The performance of the AE, PNN\textsubscript{14}, and HLN\textsubscript{14} as anomaly detectors on the 80 GeV W, 174 GeV top, and 80 GeV Higgs. Note that the Higgs bosons are pair produced from the decay of a heavier Higgs, leading to potentially 4 prongs in the large-radius jet. The left panels show the normalized distribution of each method’s respective anomaly score for the background and each signal. The right panel shows the ROC curves for each signal, with the solid lines being the ROC curves for the AE, the dashed lines for HLN\textsubscript{14}, and the dashed-dot lines for PNN\textsubscript{14}.

TABLE I. Taken together, these indicate that most of the performance of our mimicker networks is coming at higher signal efficiencies, and the long tails in their anomaly scores for the background distribution holds them back from exactly matching the AE.

Finally, by the endpoint of the iterative process, we had found that the PNN and HLN agreed on ordering of background events at about 83% when compared to the AE. Here, we see that in terms of the AUC metric, 83% mimicking transferred quite well to the use of these mimickers as simpler anomaly detectors with comparable performance. We expect the tendency for the mimicker networks to tag the background as being more anomalous than the signal at low signal efficiencies to subside as the ADO of the mimickers approaches 1.

C. Using Only Prime EFPs

In examining the EFPs selected to improve the decision ordering, we note that even though we use up to 14 EFPs, they only depend on six prime EFP factors:

\[
\begin{align*}
\begin{array}{c}
\text{\includegraphics[width=1cm]{prong1.png}} \\
\text{\includegraphics[width=1cm]{prong2.png}} \\
\text{\includegraphics[width=1cm]{prong3.png}} \\
\text{\includegraphics[width=1cm]{prong4.png}} \\
\text{\includegraphics[width=1cm]{prong5.png}} \\
\text{\includegraphics[width=1cm]{prong6.png}}
\end{array}
\end{align*}
\]

Notably in these primes, the first and fifth prime factors are the energy correlation functions for two and three prong structures. It is also interesting to note that these prime factors are nonzero only for $\geq 2, 3$ prong structures. As the AE is learning to encode the predominantly 1-prong QCD events, it seems that it is losing information contained in these higher prong observables. With this loss of information, networks with direct access to these observables are able to explain the reconstruction error of the network.

The observation that the anomaly scores can be explained by composite operators which only have a few prime operators leads one to wonder if the prime EFPs are good enough. To test this, we trained both the PNN and the HLN using mass, $p_T$, and the six prime EFPs. The results are denoted in Fig. 6 and Fig. 7 by the blue ‘+’ and orange ‘\times’, respectively. Not only do these “prime-only” networks perform comparably to each other, which matches the behavior we saw from the networks trained on the composite EFPs, but the prime-only and composite networks also perform comparably across all of the signals. The results in Fig. 6 show the ADO of the prime-only networks computed on the same pairs of events as the center line for the composite models. The ADO of the prime-only models has a similar spread as the composite models, and thus the two do indeed perform comparably. Taken together, this seems to indicate that the prime EFPs alone contain all of the necessary information to construct simple anomaly detectors capable of matching much more complex ones. While each of the prime EFPs on their own would have been selected eventually, these results also suggest a more efficient iterative procedure for creating HLN and PNN mimickers, where one uses the redundancy in the full space of EFPs to their advantage and allows the algorithm to explore the full space of composite EFPs, but only selects...
D. Comparison with Isolation Forests

Through this iterative process, we've constructed two different types of dense neural networks that approximately match the AE not only in how their decision functions order background events, but also as anomaly detectors for classifying a variety of signals. It is clear then that the observables picked out by this procedure contain the information needed to match the AE on both fronts. One then wonders if an even simpler anomaly detector than the ones presented in Sec. [11] would give similar results. To investigate this possibility, we consider isolation forests as implemented by ISOLATION FOREST in SCIKIT-LEARN [80].

Isolation forests work by randomly selecting a feature from a given set of inputs, and then randomly selecting a split value for that feature. This splitting process is repeated until each event the model is trained on has been isolated from the rest, resulting in a tree-like structure. We then build an ensemble, or “forest” of these classifiers. The anomaly score is the number of splittings needed to isolate each event, averaged over the entire ensemble. This kind of random partitioning tends to take fewer splittings to isolate anomalous events, so if the average number of splittings across a large ensemble is low, the event is likely to be anomalous. We wanted to see if the performance of the isolation forests saturate in the same way the HLNs and PNNs did, so we trained a series of them and added the new observable picked out by either the HLN or PNN each time. The details of our specific implementation is given in App. [3]. Since the HLNs and PNNs selected EFPs in a slightly different order, we trained 2 different sets of isolation forests. One set added observables in the order selected by the HLN, while the other added them in the order selected by the PNN.

FIG. 8. AUCs for the PNN and HLN at each iteration for each of the eight signals reserved for testing. Note that the Higgs bosons are pair produced from the decay of a heavier Higgs, leading to potentially 4 prongs in the large-radius jet. The solid center lines are the AUC of the model used in the iterative process, the shaded bands show the maximum and minimum AUCs from retraining each network an additional 10 times. The dashed black line corresponds to the AE's AUC. The dotted lines correspond to the isolation forest anomaly detectors and the blue ‘+’ (orange ‘×’) is the PNN(HLN) trained using mass, $p_T$, and the five prime factors in Eqn. [12].
**TABLE II.** The background rejection $(1/\varepsilon_B)$ at two different fixed signal efficiencies $(\varepsilon_S)$—0.5 and 0.1—and the signal efficiency at two different fixed values of the background rejection—10 and 100—for all 8 anomalous signals. We present these metrics for 5 different networks, the AE, PNN₀, HLN₀, PNN₁₄, and HLN₁₄. The values shown in red are those where $\varepsilon_B > \varepsilon_S$.

| 80 GeV Top | 174 GeV Top |
|------------|-------------|
| $\varepsilon_S(1/\varepsilon_B = 10)$ | $\varepsilon_S(1/\varepsilon_B = 10)$ |
| AE | HLN₁₄ | PNN₁₄ | HLN₀ | PNN₀ | AE | HLN₁₄ | PNN₁₄ | HLN₀ | PNN₀ |
| 0.252 | 0.012 | 0.114 | 0.071 | 0.071 | 0.470 | 0.357 | 0.428 | 0.146 | 0.148 |
| 0.022 | 0.007 | 0.008 | 0.007 | 0.008 | 0.088 | 0.016 | 0.022 | 0.013 | 0.013 |
| $1/\varepsilon_B(\varepsilon_S = 0.5)$ | $1/\varepsilon_B(\varepsilon_S = 0.5)$ |
| 4.24 | 4.03 | 3.95 | 2.29 | 2.29 | 8.93 | 6.94 | 8.26 | 5.96 | 6.00 |
| 26.5 | 12.0 | 11.3 | 7.33 | 7.39 | 87.6 | 28.6 | 38.0 | 12.8 | 12.9 |

| 20 GeV Higgs | 80 GeV Higgs |
|---------------|--------------|
| $\varepsilon_S(1/\varepsilon_B = 10)$ | $\varepsilon_S(1/\varepsilon_B = 10)$ |
| AE | HLN₁₄ | PNN₁₄ | HLN₀ | PNN₀ | AE | HLN₁₄ | PNN₁₄ | HLN₀ | PNN₀ |
| 0.240 | 0.027 | 0.086 | 0.032 | 0.033 | 0.446 | 0.549 | 0.565 | 0.030 | 0.031 |
| 0.025 | 0.001 | 0.001 | 0.001 | 0.001 | 0.036 | 0.022 | 0.020 | 0.002 | 0.002 |
| $1/\varepsilon_B(\varepsilon_S = 0.5)$ | $1/\varepsilon_B(\varepsilon_S = 0.5)$ |
| 4.06 | 3.39 | 4.14 | 4.87 | 4.91 | 8.58 | 11.3 | 11.9 | 4.67 | 4.70 |
| 25.7 | 6.82 | 9.67 | 6.68 | 6.72 | 42.4 | 46.1 | 50.1 | 6.41 | 6.44 |

| 59 GeV W | 80 GeV W |
|-----------|---------|
| $\varepsilon_S(1/\varepsilon_B = 10)$ | $\varepsilon_S(1/\varepsilon_B = 10)$ |
| AE | HLN₁₄ | PNN₁₄ | HLN₀ | PNN₀ | AE | HLN₁₄ | PNN₁₄ | HLN₀ | PNN₀ |
| 0.155 | 0.017 | 0.007 | 0.011 | 0.012 | 0.190 | 0.043 | 0.013 | 0.014 | 0.014 |
| 0.015 | 0.0003 | 0.0003 | 0.0007 | 0.0007 | 0.028 | 0.0005 | 0.0004 | 0.0009 | 0.0009 |
| $1/\varepsilon_B(\varepsilon_S = 0.5)$ | $1/\varepsilon_B(\varepsilon_S = 0.5)$ |
| 2.86 | 2.76 | 2.62 | 1.40 | 1.40 | 3.06 | 3.57 | 3.44 | 1.77 | 1.77 |
| 16.1 | 5.08 | 3.91 | 2.36 | 2.35 | 22.4 | 7.17 | 5.52 | 2.83 | 2.84 |

| 120 GeV W | 174 GeV W |
|-----------|---------|
| $\varepsilon_S(1/\varepsilon_B = 10)$ | $\varepsilon_S(1/\varepsilon_B = 10)$ |
| AE | HLN₁₄ | PNN₁₄ | HLN₀ | PNN₀ | AE | HLN₁₄ | PNN₁₄ | HLN₀ | PNN₀ |
| 0.244 | 0.070 | 0.089 | 0.021 | 0.022 | 0.289 | 0.124 | 0.190 | 0.064 | 0.064 |
| 0.040 | 0.001 | 0.001 | 0.001 | 0.001 | 0.052 | 0.003 | 0.003 | 0.003 | 0.003 |
| $1/\varepsilon_B(\varepsilon_S = 0.5)$ | $1/\varepsilon_B(\varepsilon_S = 0.5)$ |
| 3.71 | 4.01 | 4.76 | 2.97 | 2.97 | 4.40 | 4.21 | 5.53 | 6.05 | 6.10 |
| 32.9 | 8.52 | 9.58 | 4.30 | 4.31 | 42.4 | 11.4 | 14.7 | 8.61 | 8.57 |

Thus, we suspect it is the mimicking aspect of our procedure which allows for good anomaly detection with the simple set of observables.

**V. Conclusion**

In this paper, we have extended the results of Ref. [65] to build simpler, more interpretable anomaly detectors. Starting with a convolutional autoencoder, we iteratively built a network that mimics the autoencoder’s ordering of background events, where the network’s inputs are high-level variables taken from a set of Energy Flow Polynomials. We presented two network architectures for the mimickers, the High-Level Network and the Paired Neural Network. The High-Level Network aims to reproduce the reconstruction error of the autoencoder, while the Paired Neural Network takes in two events and is trained to order them like the autoen-
encoder. Note that both the PNN and HLN are trained to order anomalous events from the physics observables, which is an inherently different task than the autoencoder, which was only trained to compress and decompress background data. This highlights the difference with Ref. [65], in which the black-box network and mimicking network have the same task of binary classification. Given this fundamental difference between our AE and mimicking networks, it is not obvious that employing the same strategy will work when trying to mimic the autoencoder’s ordering. However, we find that these two complementary approaches give similar performance, ~83% agreement, when ordering background events and also pick out the same list of EFPs, suggesting the commonality of the information that is needed to order events like the autoencoder.

After mimicking the autoencoder on ordering of background events, we take these networks and apply them as anomaly detectors on eight different signals. Even though the mimickers and autoencoder have never seen these events, we find that the similarity in ordering transfers to these events, making the mimickers as good (or better) than the autoencoder as an anomaly detector for seven of the eight signals. It is worth emphasizing how such results were not guaranteed to occur. The autoencoder, having been trained only on background events, has no concept of what is anomalous. So it is not obvious that mimicking the ordering of events for the background will generalize to anomalous events, especially given a large set of signal classes.

Since the high-level observables picked out by these mimickers rely only on six prime Energy Flow Polynomials, it indicates that the information required to order events like the autoencoder is reasonably small. However, since the isolation forests based on these high-level inputs did not perform as well, it shows that mimicking the autoencoder’s background ordering is crucial in creating a simpler anomaly detector.

In terms of future directions, it would be interesting to extend the list of Energy Flow Polynomials to check that one can saturate the decision ordering of the autoencoder and to determine what prime Energy Flow Polynomials are needed for that. Applying this technique to other anomaly detection methods on the same dataset would help uncover what high-level variables are being used by these methods and could help in designing more powerful anomaly detectors. Finally, it would be interesting to see if one can extend this technique to cases where there is no known high-level variable basis (like the Energy Flow Polynomials) and to see to what extent decision ordering transfers to different signals. For instance, the methods which performed best on the Dark Machines anomaly score challenge [25, 51, 54] used variational autoencoder structures which only aimed to make a Gaussian latent space and did not try to reconstruct events. It would be very interesting to see what physics these methods are using, but there is no obvious basis of observables to use.

Acknowledgements

The work of LB and SC was supported in part by the U.S. Department of Energy under Grant Number DE-SC0011640. BO is supported by the National Science Foundation under Cooperative Agreement PHY-2019786 (The NSF AI Institute for Artificial Intelligence and Fundamental Interactions, http://inif.org/). This work also benefited from access to the University of Oregon high performance computing cluster, Talapas.

A. Simulation Details

In this appendix, we provide further details of the simulated public datasets we use in this work [38, 69, 70]. All of the QCD dijet, W, top, and Higgs samples are subject to the same selection criteria, showering, and detection simulation parameters. The background and anomalous events are generated using MadGraph [87] and Pythia8 [88], with detector effects being simulated by Delphes [89]. The jets are then clustered using FastJet [90, 91] using the anti-kT algorithm [92] with a cone size of R = 1.0. All events are required to have two hard jets, with the leading jet having pT > 450 GeV and the sub-leading jet having pT > 200 GeV. We then take only the leading jet in each event.

The QCD jets are created via pp → jj. The W jets are created using pp → W′ → W (→ jj) Z (→ νν) with mW′ = 1.2 TeV. The top jets are produced via pp → Z′ → t̅t with mZ′ = 1.3 TeV. Finally, the Higgs jets are produced with pp → HH, H → hh, h → jj with mH = 174 GeV. For each of these signals, we only consider jets with pT ∈ [550, 650] GeV. This same pT cut is applied to the background training and testing sets.

B. Network Training Hyperparameters

Here, we provide the details of the training hyperparameters of the AE, PNN, HLN, and isolation forests. For all three deep neural network architectures, we use the ReduceLROnPlateau and EarlyStopping callbacks from Keras to dynamically reduce the learning rate and stop training early,
respectively. All three neural networks are trained with the Adam optimizer \[93\].

For the AE, our training hyperparameters are:

- Train for 100 epochs with EARLYSTOPPING on the validation loss with a patience of 10 epochs.
- Initial learning rate of $10^{-3}$ with REDUCELRONPLATEAU on the validation loss with a patience of 5 epochs.
- Batch size of 256.

For the HLN and PNN, our training hyperparameters are:

- Train for 200 epochs with EARLYSTOPPING on the validation loss with a patience of 10 epochs.
- Initial learning rate of $10^{-3}$ with REDUCELRONPLATEAU on the validation loss with a patience of 5 epochs.
- Batch size of 256.

With the early stopping conditions, the AE trains in $\sim$30 epochs, the PNN trains in $\sim$50 epochs, and the HLN trains in $\sim$60 epochs.

For the isolation forests, our training hyperparameters are:

- 250 estimators in the ensemble.
- The max_features used to train each estimator is set to the number of inputs for each event.
- contamination is set to ‘auto’ since there is no way to determine what fraction of events can reliably be called outliers a priori.
- bootstrap is set to ‘False’, so individual trees are trained on random subsets of the data without replacement.

---

[1] B. Nachman, “Anomaly Detection for Physics Analysis and Loss less than Supervised Learning,” arXiv:2010.14654 [hep-ph]

[2] G. Kasieczka et al., “The LHC Olympics 2020 a community challenge for anomaly detection in high energy physics,” Rept. Prog. Phys. 84 (2021) no. 12, 124201 [arXiv:2101.08320 [hep-ph]]

[3] J. H. Collins, K. Howe, and B. Nachman, “Anomaly Detection for Resonant New Physics with Machine Learning,” Phys. Rev. Lett. 121 (2018) no. 24, 241803 [arXiv:1805.02664 [hep-ph]]

[4] R. T. D’Agnolo and A. Wulzer, “Learning New Physics from a Machine,” Phys. Rev. D 99 (2019) no. 1, 015014 [arXiv:1806.02350 [hep-ph]]

[5] A. De Simone and T. Jacques, “Guiding New Physics Searches with Unsupervised Learning,” Eur. Phys. J. C 79 (2019) no. 4, 289. arXiv:1807.06038 [hep-ph]

[6] A. Casas and G. Menardi, “Nonparametric semisupervised classification for signal detection in high energy physics,” arXiv:1809.02977 [stat.AP]

[7] B. M. Dillon, D. A. Faroughy, and J. F. Kamenik, “Uncovering latent jet substructure,” Phys. Rev. D 100 (2019) no. 5, 056002 [arXiv:1904.04200 [hep-ph]]

[8] A. Mullin, S. Nicholls, H. Pacey, M. Parker, M. White, and S. Williams, “Does SUSY have friends? A new approach for LHC event analysis,” JHEP 02 (2021) 160 [arXiv:1912.10625 [hep-ph]]

[9] R. T. D’Agnolo, G. Grosso, M. Pierini, A. Wulzer, and M. Zanetti, “Learning multivariate new physics,” Eur. Phys. J. C 81 (2021) no. 1, 89 [arXiv:1912.12155 [hep-ph]]

[10] B. Nachman and D. Shih, “Anomaly Detection with Density Estimation,” Phys. Rev. D 101 (2020) 075012 [arXiv:2001.04990 [hep-ph]]

[11] A. Andreassen, B. Nachman, and D. Shih, “Simulation Assisted Likelihood-free Anomaly Detection,” Phys. Rev. D 101 (2020) no. 9, 095004 [arXiv:2001.05001 [hep-ph]]

[12] ATLAS Collaboration, G. Aad et al., “Dijet resonance search with weak supervision using $\sqrt{s} = 13$ TeV $pp$ collisions in the ATLAS detector,” Phys. Rev. Lett. 125 (2020) no. 13, 131801 [arXiv:2005.02983 [hep-ex]]

[13] B. M. Dillon, D. A. Faroughy, J. F. Kamenik, and M. Szewc, “Learning the latent structure of collider events,” JHEP 10 (2020) 206 [arXiv:2005.12319 [hep-ph]]

[14] K. Benkendorfer, L. L. Pottier, and B. Nachman, “Simulation-assisted decorrelation for resonant anomaly detection,” Phys. Rev. D 104 (2021) no. 3, 035003 [arXiv:2009.02205 [hep-ph]]

[15] V. Mikuni and F. Canelli, “Unsupervised clustering for collider physics,” Phys. Rev. D 103 (2021) no. 9, 092007 [arXiv:2010.07106 [physics.data-an]]

[16] G. Stein, U. Seljak, and B. Dai, “Unsupervised in-distribution anomaly detection of new physics through conditional density estimation,” in 34th Conference on Neural Information Processing Systems. 12, 2020. arXiv:2012.11638 [cs.LG]

[17] J. Batson, C. G. Haaf, Y. Kahn, and D. A. Roberts, “Topological Obstructions to Autoencoding,” JHEP 04 (2021) 280 [arXiv:2102.08380 [hep-ph]]

[18] A. Blance and M. Spannowsky, “Unsupervised event classification with graphs on classical and photonic quantum computers,” JHEP 21 (2020)
J. Gou, B. Yu, S. J. Maybank, and D. Tao, “Autoencoders for unsupervised anomaly detection in high energy physics," *JHEP* **06** (2021) 161. [arXiv:2104.09051 [hep-ph]]

O. Atkinson, A. Bhadrakaj, C. Englert, V. S. Ngairangbam, and M. Spannowsky, “Anomaly detection with convolutional Graph Neural Networks,” *JHEP* **08** (2021) 080. [arXiv:2105.07988 [hep-ph]]

B. M. Dillon, T. Pichin, C. Sauer, and P. Sorrenson, “Better Latent Spaces for Better Autoencoders,” *SciPost Phys.* **11** (2021) 061. [arXiv:2104.08291 [hep-ph]]

A. Kahn, J. Gonski, I. Ochoa, D. Williams, and B. M. Dillon, T. Plehn, C. Sauer, and O. Atkinson, A. Bhardwaj, C. Englert, V. S. Ngairangbam, and M. Spannowsky, “Anomaly detection with convolutional Graph Neural Networks,” *JHEP* **08** (2021) 080. [arXiv:2105.07988 [hep-ph]]

B. Ostdiek, “Deep Set Auto Encoders for Anomaly Detection,” *SciPost Phys.* **11** (2021) 061. [arXiv:2104.08291 [hep-ph]]

E. Govorkova et al., “Autoencoders on FPGAs for real-time, unsupervised new physics detection at 40 MHz at the Large Hadron Collider,” [arXiv:2106.10164 [hep-ph]]

J. Gonski, J. Lai, B. Nachman, and I. Ochoa, “High-dimensional Anomaly Detection with Radiative Return in $e^+ e^-$ Collisions,” [arXiv:2108.03986 [physics.ins-det]]

J. Gonski, J. Lai, B. Nachman, and I. Ochoa, “High-dimensional Anomaly Detection with Radiative Return in $e^+ e^-$ Collisions,” [arXiv:2108.03986 [physics.ins-det]]

B. Ostdiek, “Deep Set Auto Encoders for Anomaly Detection in Particle Physics,” *SciPost Phys.* **12** (2022) 045. [arXiv:2109.01695 [hep-ph]]

K. Fraser, S. Homiller, R. K. Mishra, B. Ostdiek, and M. D. Schwartz, “Challenges for Unsupervised Anomaly Detection in Particle Physics,” [arXiv:2110.06948 [hep-ph]]

P. Baldi, P. Sadowski, and D. Whiteson, “Searching for Exotic Particles in High-Energy Physics with Deep Learning,” *Nature Commun.* **5** (2014) 4308. [arXiv:1402.4735 [hep-ph]]

A. A. Alemi, I. Fischer, J. V. Dillon, and K. Murphy, “Deep variational information bottleneck,” *CoRR* abs/1612.00410 (2016), 1612.00410. [http://arxiv.org/abs/1612.00410](http://arxiv.org/abs/1612.00410)

S. Chang, T. Cohen, and B. Ostdiek, “What is the Machine Learning?” *Phys. Rev. D* **97** (2018) no. 5. [arXiv:1709.10106 [hep-ph]]

S. Wunsch, R. Fries, R. Wolf, and G. Quast, “Identifying the relevant dependencies of the neural network response on characteristics of the input space,” *Comput. Softw. Big Sci.* **2** (2018) no. 1, 5. [arXiv:1803.08782 [physics.data-an]]

T. Roxlo and M. Reece, “Opening the black box of neural nets: case studies in stop/top discrimination,” [arXiv:1804.09278 [hep-ph]]

J. Gou, B. Yu, S. J. Maybank, and D. Tao, “Knowledge distillation: A survey,” *CoRR* abs/2006.05525 (2020). [arXiv:2006.05525 [hep-ph]]

G. Agarwal, L. Hay, I. Iashvili, B. Mannix, C. McLean, M. Morris, S. Rappoccio, and U. Schubert, “Explainable AI for ML jet taggers using expert variables and layerwise relevance propagation.” *JHEP* **05** (2021) 208. [arXiv:2011.13466 [physics.data-an]]

F. Mokhtar, R. Kansal, D. Diaz, J. Duarte, J. Pata, M. Pierini, and J.-R. Vlimant, “Explaining machine-learned particle-flow reconstruction,” in 35th Conference on Neural Information Processing Systems. 11, 2021. [arXiv:2111.12840 [physics.data-an]]

J. Craven, V. Jejijala, and A. Kar, “Disentangling a deep learned volume formula,” *JHEP* **06** (2021) 040. [arXiv:2012.03955 [hep-th]]

T. Fauchet, J. Thaler, and D. Whiteson, “Mapping Machine-Learned Physics into a Human-Readable Space,” *Phys. Rev. D* **103** (2021) no. 3, 036020. [arXiv:2010.11998 [hep-ph]]

J. Collado, J. N. Howard, T. Fauchet, T. Tong, P. Baldi, and D. Whiteson, “Learning to identify electrons,” *Phys. Rev. D* **105** (2021) no. 11, 116025. [arXiv:2111.01984 [physics.data-an]]

J. Collado, K. Bauer, E. Witkowski, T. Fauchet, D. Whiteson, and P. Baldi, “Learning to isolate muons,” *JHEP* **21** (2020) 206. [arXiv:2102.02278 [physics.data-an]]

P. T. Komiske, E. M. Metodiev, and J. Thaler, “Energy flow polynomials: A complete linear basis for jet substructure,” *JHEP* **04** (2018) 013. [arXiv:1712.07124 [hep-ph]]

J. Leissner-Martin, T. Cheng, and J.-F. Arguin, “QCD Jet Samples with Particle Flow Constituents,” July, 2020. [https://doi.org/10.5281/zenodo.4641460](https://doi.org/10.5281/zenodo.4641460)

T. Cheng, “Test Sets for Jet Anomaly Detection at the LHC,” Mar., 2021. [https://doi.org/10.5281/zenodo.4641656](https://doi.org/10.5281/zenodo.4641656)

V. Mikuni, B. Nachman, and D. Shih, “Online-compatible Unsupervised Non-resonant Anomaly Detection,” [arXiv:2111.06417 [cs.LG]]

S. Macaluso and D. Shih, “Pulling Out All the Tops with Computer Vision and Deep Learning,” *JHEP* **10** (2018) 121. [arXiv:1803.00107 [hep-ph]]

[https://energyflow.network](https://energyflow.network)

D.-A. Clevert, T. Unterthiner, and S. Hochreiter, “Fast and accurate deep network learning by exponential linear units (elus),” 2016.

F. Chollet et al., “Keras.” [https://keras.io](https://keras.io) 2015.

M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng,
“TensorFlow: Large-scale machine learning on heterogeneous systems,” 2015.  
https://www.tensorflow.org/ Software available from tensorflow.org.

[77] A. J. Larkoski, G. P. Salam, and J. Thaler, “Energy Correlation Functions for Jet Substructure,” *JHEP* 06 (2013) 108  
[arXiv:1305.0007 [hep-ph]]

[78] A. J. Larkoski, I. Moult, and D. Neill, “Power Counting to Better Jet Observables,” *JHEP* 12 (2014) 009  
[arXiv:1409.6298 [hep-ph]]

[79] A. Banfi, G. P. Salam, and G. Zanderighi, “Principles of general final-state resummation and automated implementation,” *JHEP* 03 (2005) 073  
[arXiv:hep-ph/0407286]

[80] G. Gur-Ari, M. Papucci, and G. Perez, “Classification of Energy Flow Observables in Narrow Jets,” [arXiv:1101.2905 [hep-ph]]

[81] M. Jankowiak and A. J. Larkoski, “Jet Substructure Without Trees,” *JHEP* 06 (2011) 057  
[arXiv:1104.1646 [hep-ph]]

[82] I. Moult, L. Necib, and J. Thaler, “New Angles on Energy Correlation Functions,” *JHEP* 12 (2016) 153  
[arXiv:1609.07483 [hep-ph]]

[83] CMS Collaboration, S. Chatrchyan et al., “Search for a Higgs boson in the decay channel H to ZZ(*) to q qbar ℓ−ℓ+ in pp collisions at √s = 7 TeV,” *JHEP* 04 (2012) 036  
[arXiv:1202.1416 [hep-ex]]

[84] A. J. Larkoski, J. Thaler, and W. J. Waalewijn, “Gaining (Mutual) Information about Quark/Gluon Discrimination,” *JHEP* 11 (2014) 129  
[arXiv:1408.3122 [hep-ph]]

[85] P. Gras, S. Höche, D. Kar, A. Larkoski, L. Lönnblad, S. Plätzer, A. Siódmok, P. Skands, G. Soyez, and J. Thaler, “Systematics of quark/gluon tagging,” *JHEP* 07 (2017) 091  
[arXiv:1704.03878 [hep-ph]]

[86] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, “Scikit-learn: Machine learning in Python,” *Journal of Machine Learning Research* 12 (2011) 2825–2830.

[87] J. Alwall, R. Frederix, S. Frixione, V. Hirschi, F. Maltoni, O. Mattelaer, H. S. Shao, T. Stelzer, P. Torrielli, and M. Zaro, “The automated computation of tree-level and next-to-leading order differential cross sections, and their matching to parton shower simulations,” *JHEP* 07 (2014) 079  
[arXiv:1405.0301 [hep-ph]]

[88] T. Sjöstrand, S. Ask, J. R. Christiansen, R. Corke, N. Desai, P. Ilten, S. Mrenna, S. Prestel, C. O. Rasmussen, and P. Z. Skands, “An introduction to PYTHIA 8.2,” *Comput. Phys. Commun.* 191 (2015) 159–177  
[arXiv:1410.3012 [hep-ph]]

[89] DELPHES 3 Collaboration, 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,” *JHEP* 02 (2014) 057  
[arXiv:1307.6346 [hep-ex]]

[90] M. Cacciari, G. P. Salam, and G. Soyez, “FastJet User Manual,” *Eur. Phys. J. C* 72 (2012) 1896  
[arXiv:1111.6097 [hep-ph]]

[91] M. Cacciari and G. P. Salam, “Dispelling the N^3 myth for the k_t jet-finder,” *Phys. Lett. B* 641 (2006) 57–61  
[arXiv:hep-ph/0512210]

[92] M. Cacciari, G. P. Salam, and G. Soyez, “The anti-k_t jet clustering algorithm,” *JHEP* 04 (2008) 063  
[arXiv:0802.1189 [hep-ph]]

[93] D. P. Kingma and J. Ba, “Adam: A Method for Stochastic Optimization,” *arXiv:1412.6980 [cs.LG]*