Published for SISSA by Springer

Received: May 26, 2021
Revised: July 8, 2021
Accepted: August 1, 2021
Published: August 17, 2021

Anomaly detection with convolutional Graph Neural Networks

Oliver Atkinson,[a] **Akanksha Bhardwaj,**[a] **Christoph Englert,**[a] **Vishal S. Ngairangbam**[b,c] **and Michael Spannowsky**[d,e]

[a] *School of Physics & Astronomy, University of Glasgow, Glasgow G12 8QQ, United Kingdom*
[b] *Theoretical Physics Division, Physical Research Laboratory, Shree Pannalal Patel Marg, Ahmedabad – 380009, Gujarat, India*
[c] *Discipline of Physics, Indian Institute of Technology, Palaj, Gandhinagar – 382424, Gujarat, India*
[d] *Institute for Particle Physics Phenomenology, Durham University, Durham DH1 3LE, United Kingdom*
[e] *Department of Physics, Durham University, Durham DH1 3LE, United Kingdom*

E-mail: o.atkinson.1@research.gla.ac.uk, akanksha.bhardwaj@glasgow.ac.uk, christoph.englert@glasgow.ac.uk, vishalng@prl.res.in, michael.spannowsky@durham.ac.uk

Abstract: We devise an autoencoder based strategy to facilitate anomaly detection for boosted jets, employing Graph Neural Networks (GNNs) to do so. To overcome known limitations of GNN autoencoders, we design a symmetric decoder capable of simultaneously reconstructing edge features and node features. Focusing on latent space based discriminators, we find that such setups provide a promising avenue to isolate new physics and competing SM signatures from sensitivity-limiting QCD jet contributions. We demonstrate the flexibility and broad applicability of this approach using examples of W bosons, top quarks, and exotic hadronically-decaying exotic scalar bosons.

Keywords: Jets

ArXiv ePrint: 2105.07988

https://doi.org/10.1007/JHEP08(2021)080

JHEP08(2021)080

Contents

JHEP08(2021)080

1 Introduction

The search for new physics beyond the SM (BSM), albeit unsuccessful so far at high energy colliders such as the Large Hadron Collider (LHC), remains a key pillar of the particle physics phenomenology programme. The mounting pressure of exclusion constraints for well-motivated BSM scenarios reported by the ATLAS and CMS experiments has shifted theoretical efforts more towards model-independent approaches for new physics discoveries. This is most prominently reflected in the recent resurgence of effective field theory interpretations of collider data (see e.g. ref. [1] for a recent review).

One method of searching for the presence of new BSM interactions is attempting to detect anomalies in otherwise well-understood and abundant collider data [2–10]. Hard jets produced primarily via QCD-mediated interactions at hadron colliders are a well-known laboratory for such strategies and their phenomenology has seen considerable development over the past decade [11]. QCD jets are produced with large rates at the LHC even when they are extremely hard, and therefore are the typical backgrounds that any search for new physics in hadronic final states needs to overcome. Turning this argument around, we can classify jets, possibly using motivated first-principle approaches [12, 13], and isolate more interesting generic BSM-type signatures by vetoing a "typical" QCD jet. Following this line of thought, anomaly-detection has become a primary application of unsupervised machine learning. This typically involves so-called autoencoders, which are artificial neural networks specifically tailored to reproduce the most common properties of a training data set via a reduction of the dimensionality of the input's features. When a jet behaves less like a common QCD jet (e.g. in case of a particular hadronic BSM decay) such a network

should perform poorly, i.e. the loss that parametrises the networks ability to reproduce the QCD signature can be used as a BSM-discriminating observable.

The evolution of a typical QCD event from high to low energies is well understood over a vast range of energy scales, as demonstrated by the successful application of QCD shower Monte Carlo programmes to the modelling of collider data (see e.g. [14]). This evolution also motivates the application of Graph Neural Networks (GNNs) [15, 16] to QCD phenomenology as recently done in ref. [17], and also to exploiting the Lund-plane representation of splittings [18, 19]. GNNs have also been studied in various scenarios [20–24] at the LHC. Moreover, they have also shown promising performances for use in real-time triggers [25]. In this work we consider a GNN-based autoencoder for anomaly detection in boosted QCD jets data. Convolutional autoencoders have been proposed and studied in [26–31] for distinguishing QCD jets from non-QCD jets using "jet-images" [32, 33] as the input space. However, convolutions on these images are expensive due to their extreme sparsity. Moreover, CNNs, in principle, are limited to the Euclidean domain. GNNs mitigate these two inadequacies, so studying their performance as anomaly finders is motivated. Supervised jet classification with GNNs has been studied in refs. [17, 34, 35], while unsupervised clustering of event-graphs with photonic quantum computers have been explored in ref. [36].A study of particle graph autoencoders for anomaly detection has been carried out in the LHC Olympics community challenge [37]. A typical obstacle of GNN-based autoencoders is achieving an appropriate reflection of all network features on the decoding side. Existing graph-autoencoders in the literature [38–42] are designed mostly for node-classification or link prediction, while we desire a network capable of classifying graphs. Moreover, jets provide us with multidimensional edge information, along with node features; classifying the entire graph thereby exploits the full kinematic information of the event. To solve this known difficulty of graph-autoencoders, we design a decoder network capable of simultaneously reconstructing multidimensional edge, and node features with the help of *Inner Product Layers*.

This paper is organised as follows: section 2 introduces our analysis setup. The graph neural network methodology that we use in this work is described in section 3, where we provide details on the network's architecture and its performance. Results are presented in section 4, and we conclude in section 5.

2 Elements of the simulation

For our proof-of-principle analysis,[1] we generate events using `MadGraph5` [43] at leading order (LO), followed by `Pythia8` [44] for showering and hadronization. The hadronic final states are then clustered into jets using the anti-k_t algorithm [45] with parameter $R = 1.5$ using `FastJet` [46]. Along with a requirement that the rapidity of jets is $|y| < 2.5$, the minimum transverse momentum of a jet is required to be $p_T > 1$ TeV for this "fat jet" cluster. Only the leading jet from each multi-jet event is used as an input to the graph network and we do not include detector effects to our analysis. The sample used for training

[1] Throughout this work, we will focus on 13 TeV LHC collisions.

JHEP08(2021)080

of the autoencoder (for details see below) is a QCD multi-jet background sample, consisting of 200k generated $pp \to jj$ events.

To test the autoencoder's anomaly detection performance we use three different signal samples, each consisting of 100k events generated with `MadGraph5`, using the same procedure described above. These samples consist of

(i) boosted hadronically-decaying W bosons as a benchmark for two-prong jet structure,

(ii) boosted hadronically-decaying top quarks, as a benchmark for a three-prong structure, and

(iii) a boosted scalar ϕ decaying as $\phi \to W^+W^- \to 4j$ to give a four-prong structure. The interaction is based on a simplified Lagrangian

$$\mathcal{L} \supset -\frac{c_1}{v}\phi W^{\mu\nu}W_{\mu\nu} - c_2(u\bar{u} + d\bar{d})\phi, \tag{2.1}$$

where c_1 and c_2 are dimensionless constants and v is the Higgs field's vacuum expectation value (vev). We choose $m_\phi = 700$ GeV for demonstration purposes, but note that our results are not too sensitive to the ϕ mass scale.

To map out an infrared safe input to the graph network, we first use the anti-k_T jet algorithm to re-cluster the fat jet constituents into microjets[2] with a finer resolution of $R = 0.1$ and minimum $p_T = 5$ GeV. We consider fat jets with at least three microjets for our neural network analysis.

Identifying each microjet as a node in the network, we construct a graph associated with each jet as follows:

- *Node feature vectors*: we associate five microjet observables as the node's feature vector $\mathbf{x}$. These are $\log p_t$, $\Delta\eta$, $\Delta\phi$, ΔR, and $\bar{m}$. Here, p_t is the transverse momentum of the microjet, $\Delta\eta, \Delta\phi$, and ΔR are differences in pseudorapidity, azimuthal angle, and angular distance between the microjet and the jet axis respectively. $\bar{m}$ is the mass of the microjet divided by 100 GeV, which, along with the log on p_t, reduces the disparity in the range with the other three angular variables.

- *Edge feature vectors*: after the nodes are defined, we define the graph as the complete graph with all possible edge connections. For each edge, we construct an associated edge-feature vector of three dimensions. Its components are the two distance parameters between the nodes as defined below, and one invariant mass parameter: $\mathbf{e}_{ij} \equiv (d_{ij}^{\mathrm{CA}}, \log d_{ij}^{k_t}, \log m_{ij})$. The metric d_{ij} is given by

$$d_{ij} = \min(p_{ti}^{2p}, p_{tj}^{2p})\frac{R_{ij}^2}{R^2},$$

where $p = 0$ for Cambridge-Aachen (CA) jets, $p = 1$ for k_t jets and R is radius parameter for the fat jet. The CA measure provides information about the geometric

[2] As shown in ref. [47], such objects are under good experimental control.

JHEP08(2021)080

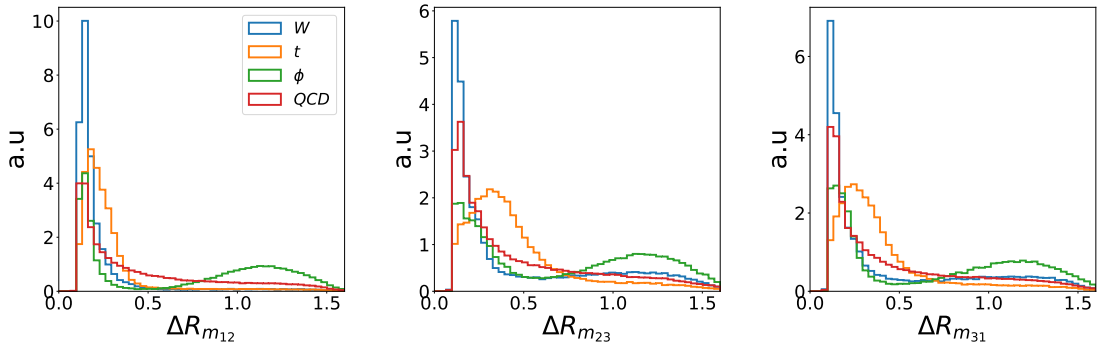


Figure 1. Normalised angular separation distribution between three leading microjets in the fat jet for the physics scenarios discussed in this work.

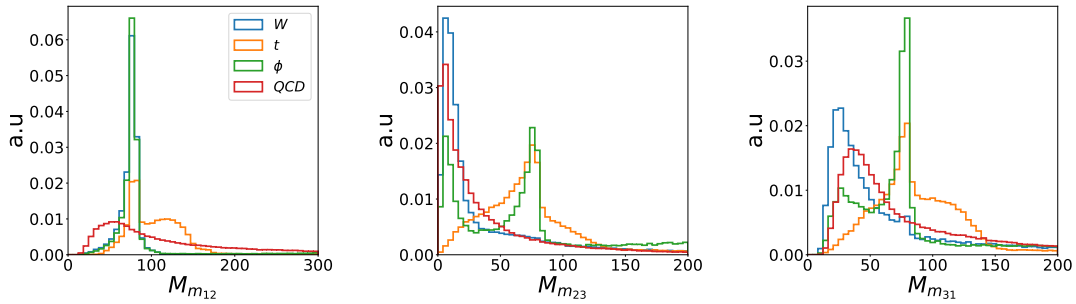


Figure 2. Similar to figure 1, but showing the normalised invariant mass distribution between three leading microjets in the fat jet.

distance between two microjets, whereas the k_t measure is motivated from QCD splittings [48, 49]. m_{ij} is the invariant mass of the two microjets. These three variables capture the essential physics between two nodes.

- *Adjacency Matrix*: we also construct the adjacency matrix for each edge feature to facilitate their reconstruction at the decoder side. It is defined as

$$A^a_{ij} = A^a_{ji} = \begin{cases} e^a_{ij} & \text{if } i \neq j \\ 1 & \text{otherwise} \end{cases} ,$$

where a is the vectorial index. Thus, for a jet-graph of N nodes, we have three $N \times N$ matrices. The network outputs the edge-features in this representation, and hence the edge-loss is defined as a function of these adjacency matrices.

The distribution of ΔR_{ij} and m_{ij} for the 3-leading microjets of each jet are shown in figures 1 and 2. The construction of the graphs and the network analysis are performed using the `Deep Graph Library` [50] with the `PyTorch` [51] backend.

JHEP08(2021)080

3 Graph Neural Networks

In this section, we describe the various components of our neural network analysis. We briefly detail the conceptual structure of GNNs before moving on to describe the ones we utilise in our analysis, along with the explicit form of the autoencoder's loss function. The network architecture and the process of training are described thereafter.

Graph Neural Networks are models that can extract features from graph-structured data. They generalise the inbuilt inductive biases in Convolutional Neural Networks (CNNs) like local connectivity and shared weights to variable length and possibly non-Euclidean data [52]. For supervised learning applications, this was formalised as Message Passing Neural Networks (MPNNs) in ref. [53]. We sketch the general paradigm and then describe in greater detail the two specific forms that are used in our work in the succeeding paragraphs. In the following, $\mathbf{h}_i^{(l)}$ is the i^{th} node's features at the l^{th} timestep (analogous to a layer in the usual ANNs). $\mathbf{e}_{ij}^{(l)}$ denotes the features of the edge connecting the nodes i and j, and $\mathcal{N}(i)$ is the set of nodes connected to the node i. For the input layer, we take $l = 0$, and $\mathbf{h}_i^{(0)} = \mathbf{x}_i$. MPNNs consist of a message passing phase and a graph readout layer. In the first phase, a message-passing function is defined for two nodes i and j

$$\mathbf{m}_{ij}^{(l)} = \mathbf{M}^{(l)}(\mathbf{h}_i^{(l)}, \mathbf{h}_j^{(l)}, \mathbf{e}_{ij}^{(l)}) \,, \tag{3.1}$$

which calculates the message $\mathbf{m}_{ij}$ for the edge connecting the nodes. The message function is usually a multilayer-perceptron (MLP) shared between all the edges, hence the term graph convolutions. For each timestep (or layer), the messages between all connected nodes are calculated, after which the features of each node are updated according to an aggregation function

$$\mathbf{h}_i^{(l+1)} = \square(\mathbf{h}_i^{(l)}, \{\mathbf{m}_{ij}^{(l)} | \, j \in \mathcal{N}(i)\}) \,, \tag{3.2}$$

which is a function of the node feature and all incoming[3] messages. The updated features $\{\mathbf{h}_i^{(l+1)}\}$, are used for the next timestep to perform another message passing and aggregation step. At the output of the final message-passing timestep, a graph readout layer performs a permutation-invariant[4] operation (for instance, max, sum, or mean) on the node features to give fixed-length vectors, regardless of the number of nodes. In supervised learning, this vector can be the final output which can be used either to minimise the loss function or fed into an MLP. However, in a graph-autoencoder, graph-level readouts are not applied to preserve the graph structure until the final output. Graph-autoencoders are typically designed for classifying nodes or edges, focusing on learning local features of a huge graph. However, as our goal is to classify small graphs, the network needs to learn global graph structures *and* local features. To overcome this, we design an edge-reconstruction network within the decoder, making our network capable of learning graph structures by reconstructing the graph in its entirety. The complete structure of our network is shown in figure 3, where the black boxes encase the encoder and the decoder. The edge-reconstruction network is shown bounded by the blue box. These are described in greater detail in the following passages.

[3] The message passing function M need not be symmetric in $\mathbf{h}_i$ and $\mathbf{h}_j$.

[4] A permutation invariant function makes the output invariant to graph isomorphisms.

JHEP08(2021)080

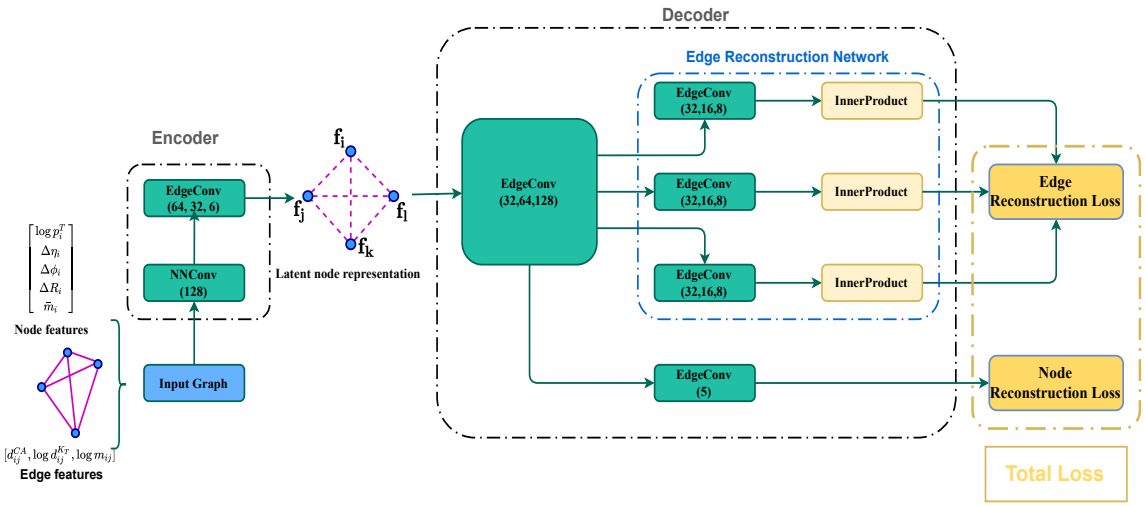


Figure 3. A schematic representation of a graph-autoencoder network. The network contains the (a) Encoder and the (b) Decoder. We employ an edge reconstruction network in the decoder to reconstruct the multidimensional edge information.

3.1 Autoencoder

Autoencoders are neural networks that map an input space to a bottleneck dimension (the latent dimension) and then back again to a space identical to the input. We use the graph-convolutions proposed in ref. [53] to incorporate the multi-dimensional edge information along with the input node features. Our network, therefore, learns the physics information that is encoded into our 3-dimensional edge feature. The timesteps until we reach the latent space employ edge-convolution [54], which has proved excellent performance in supervised learning scenarios [17, 35]. We refer to these two layers as *NNConv*, and *EdgeConv* respectively, according to the python class name implemented in the `Deep Graph Library`. The encoder block outputs a graph with the same edge connections as that of the input with updated latent features $\mathbf{f}_i$ for each node. The decoder reconstructs the node and edge features from this latent node representation. As shown in figure 3, the decoder has a shared block of edge convolutions, after which the output feeds into four different blocks of edge convolutions: a single layer for the node reconstruction, and three edge reconstruction blocks. These three blocks are identical in structure and reconstruct each edge feature independently from the propagated information from the shared block. We use an *Inner Product Layer* [38] to reconstruct the edge information in the form of three adjacency matrices. These three components and the composition of the loss function are explained in the subsequent paragraphs.

NNConv: the first layer takes the node and edge features as input and performs a weighted graph convolution by making use of an MLP, referred to as edge function F_w. This takes the edge features as input and maps it to a dimension of $m \times n$, where m is the input node's dimensions (5 in the present case), and n is the dimension of the updated node features. The message passing function performs a broadcasted element-wise multiplication

JHEP08(2021)080

of the form

$$ {}^{ab}m_{ij}^{(1)} = {}^{ab}F_e(\mathbf{e}_{ij}) \times {}^{ab}\tilde{h}_j^{(0)} \,, \tag{3.3} $$

where a and b are the indices of the matrix, and ${}^{ab}\tilde{h}_j^{(0)}$ is formed by repeating $\mathbf{h}_j^{(0)}$, the input node features, n times. The aggregation step takes the mean of ${}^{ab}m_{ij}^{(1)}$ over all neighbouring nodes j, and then sums over the a index of the matrix:

$$ {}^{b}h_i^{(1)} = \sum_a \text{mean}_{j\in\mathcal{N}(i)} \left(\left\{ {}^{ab}m_{ij}^{(1)} \right\} \right) \,, \tag{3.4} $$

to give updated n dimensional node features $\mathbf{h}_i^{(1)}$.

EdgeConv: the backbone of our architecture is the edge convolution operation [54]. This involves two linear layers: Θ_w and Φ_w, with identical input and output dimensions, which determine the dimensions of original and updated node features respectively. The message passing function is defined as

$$ \mathbf{m}_{ij}^{(l)} = \Theta_w(\mathbf{h}_j^{(l)} - \mathbf{h}_i^{(l)}) + \Phi_w(\mathbf{h}_i^{(l)}) \,, \tag{3.5} $$

while the aggregation step involves taking the maximum value

$$ {}^{a}h_i^{(l+1)} = \max_{j\in\mathcal{N}(i)} \{ {}^{a}m_{ij}^{(l)} \} \,, \tag{3.6} $$

in each component a of the incoming message vectors to give the updated node features $\mathbf{h}_i^{(l+1)}$.

Inner Product Layer: the edge-reconstruction network uses an Inner Product Layer to reconstruct the edge features from the node features of the final edge convolution output. The inner product makes the correspondence to the two-node indices for each edge. Since our graphs are undirected, the layer constructs a symmetric $N \times N$ matrix, N being the number of nodes in the graph. Its components are therefore

$$ \hat{A}_{ij} = \mathbf{h}_i \,.\, \mathbf{h}_j \,, \tag{3.7} $$

where $\mathbf{h}_i$ and $\mathbf{h}_j$ are node-feature vectors.

Loss Function: we use root-mean squared error (RMSE) for the node as well as the edge reconstruction losses. For the node feature this is

$$ L_{\text{node}} = \sqrt{\sum_{ia} \frac{(\hat{x}_i^a - x_i^a)^2}{N \times 5}} \,, \tag{3.8} $$

where a is the node-feature index, i is the node index, $\hat{x}_i^a$ and x_i^a are the reconstructed and input node features, respectively. We define the edge reconstruction loss as the sum of three individual RMSEs for each edge feature

$$ L_{\text{edge}} = \sum_a \sqrt{\sum_{ij} \frac{(\hat{A}_{ij}^a - A_{ij}^a)^2}{N \times N}} \quad , \tag{3.9} $$

JHEP08(2021)080

where a is the edge-feature index, i and j are node indices. $\hat{A}^a_{ij}$ and A^a_{ij} are the reconstructed and input adjacency matrices respectively. The total loss is the weighted sum of the individual losses,

$$L_{\text{auto}} = \lambda_{\text{node}}\, L_{\text{node}} + \lambda_{\text{edge}}\, L_{\text{edge}} \tag{3.10}$$

We choose $\lambda_{\text{node}} = 0.3$ and $\lambda_{\text{edge}} = 1$, so that the combined node features get the same weight as each individual edge feature, which carry more relevant physics information. Note that the loss function is invariant to node permutations of the input graph since, mean is a permutation invariant function, and the architecture respects permutation invariance: any change in the node ordering changes the output of each layer(via the graph readout) in conjunction with the adjacency matrix. Our network however, does not reconstruct an arbitrarily permuted graph for a given input, which is not strictly necessary since we concentrate on the reconstruction error of a single graph and not of an equivalence class of graphs.

3.2 Network architecture and training

Neural networks require a careful optimal choice of hyperparameters. As this is a proof-of-principle analysis, we do not perform an extensive hyperparameter scan. However, we scan over the latent dimension, which is critical for any autoencoder. For the first layer of the graph-encoder (NNConv), we use an MLP of hidden dimensions: 256, 128, 64, and 32 as the edge function to map the 3-dimensional edge features to a 5×128 dimensional output. The hidden layers have ReLU activations, whereas the final layer has a sigmoid activation. The limited range of the sigmoid activation helps in giving the addition operation in aggregation (as defined in eq. (3.4)) an interpretation of a weighted sum over messages in an additional dimension without the dynamics being entirely dominated by the outputs of the edge function. Each hidden layer has a dropout layer with fraction 0.2 of disconnected nodes between layers to avoid overfitting and achieve better generalisation. After the aggregation, we get a 128-dimensional output that feeds into a series of edge-convolution layers with linear layers as Θ_w and Φ_w. The output dimensions of the linear layers are 64 and 32 and outputs a 6 dimensional latent node encoding. This value is chosen after a scan over different latent dimensions which we elaborate on in the next section. The shared block of the decoder uses the encoder's reversed dimensions: 32, 64, and 128. With the 128-dimensional vector as input, the node reconstruction layer performs an edge-convolution to give the reconstructed node vectors $\hat{\mathbf{x}}$. Similarly, each edge reconstruction network has three successive edge convolutions of output dimensions 32, 16, and 8. We calculate the inner products on the 8-dimensional vector space to give the reconstructed adjacency matrices $\hat{A}^a_{ij}$.

We train the network with the Adam optimiser [55] initialised with a 0.001 learning rate on mini-batches of 64 samples. The learning rate is decayed with a reduce-on-plateau condition with decay factor 0.5, and a patience of five epochs with an additional five epochs of cool-down. We use 85k jets to train the network. After each epoch, we calculate the loss of an independent validation dataset containing 28k QCD jets. We stop the training once the learning rate goes below 10^{-8}. The epoch with minimum validation loss is used for further inference.

JHEP08(2021)080

4 Results and discussion

In order to test the performance of the graph-autoencoder for the different non-QCD signals described in section 2, we evaluate the discrimination power of the total loss function as defined in eq. (3.10). We use an independent testing data set of 28k for QCD jets and a similar number for the signal samples. We first scan the latent dimension from 2 to 12 in multiples of two, keeping all other hyperparameters fixed. The Area-Under-(the)Curve (AUC) between the signal acceptance and the background rejection for each latent dimension is shown in figure 4(a). In figure 4(b), we show the mean loss for each class as a function of the latent dimension. We can see that although the mean loss is relatively stable for QCD jets after 4-dimensions, the AUC of the different signal vs. QCD scenarios varies significantly. The variation is due to the unsupervised nature of the algorithm; the network has no information about the signal classes.

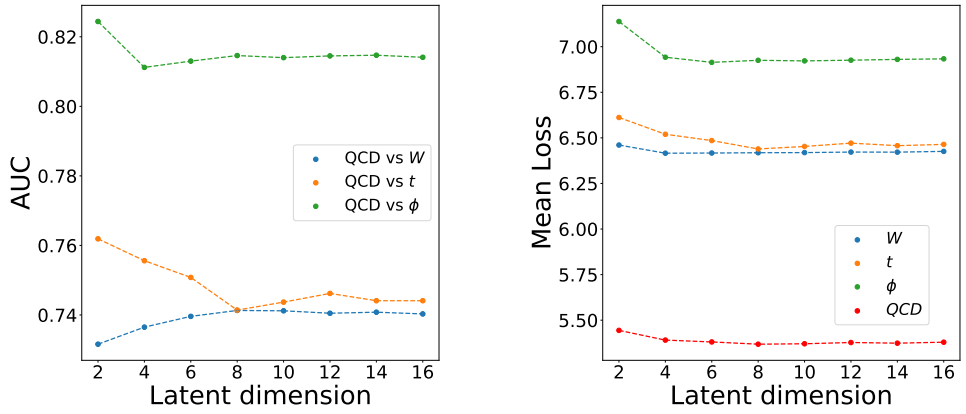
On the other hand, from the different nature of the AUC curves, we can understand the information passed for differing latent dimensions. The increasing AUCs for the W classification hints that the network sees them as similar to QCD jets when the information passed in the bottleneck is smaller, but the features of a typical QCD jet are not fully modelled for low dimensions, thus making this discrimination not reliable. Increasing the bottleneck dimension makes the network learn QCD features, which then leads to robust anomaly detection for top quarks and W bosons. The ϕ jets, which have the most noticeably different structure from QCD jets, reach a stable AUC much faster than tops and W bosons. We infer that latent dimension of ~ 6 shows a stable performance for all three classes (in particular for QCD jets) and has reached the plateau in the mean loss. Since we cannot optimise the network to each class in anomaly detection, we fix six as the latent dimension parameter. The normalised distribution of the loss function for all classes is shown in figure 5(a). As the network is trained using QCD jets, the autoencoder reconstructs them with lower loss, while all signal classes have a relatively higher loss. By vetoing the QCD jets with lower losses, we tag (new physics) signal jets (anomalous class); the Receiver-Operator-Characteristic (ROC) curve between the signal acceptance and background rejection is shown in figure 5(b).[5] The performance increases as the prong structure becomes richer for the signal classes. We discuss the correlation of the loss with some important jet-level observables in appendix B.

We also investigate the latent representation learned by the graph-autoencoder to explore compressed representations for QCD jets. Latent representations have also been investigated in similar, and indeed different, physical scenarios recently in refs. [56–58]. Even though we do not perform graph readouts during the training, the graph-autoencoder learns the graph structure via the edge reconstruction network. We use a graph readout that takes the mean in each dimension of the latent node features to obtain a fixed-dimensional latent graph representation. More precisely, we consider

$$\tilde{f}^a = \frac{1}{N} \sum_{i \in G} f_i^a \,,$$

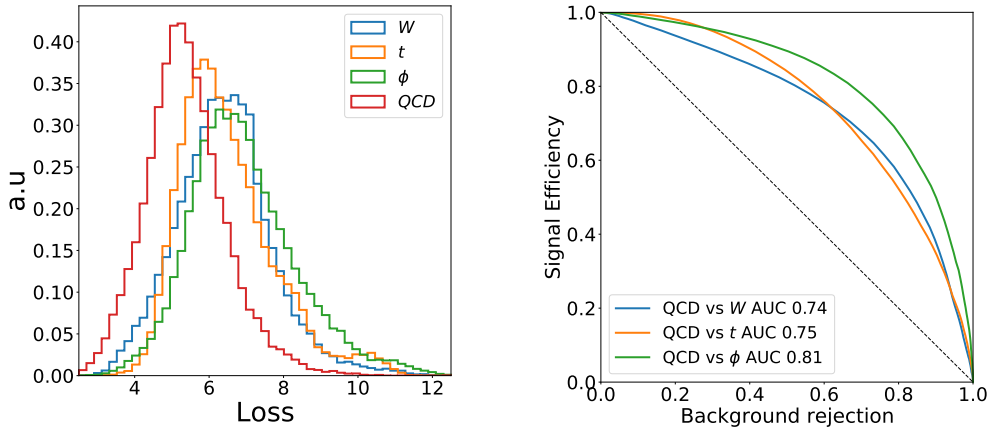
[5] We compare the results obtained for our dataset with particle graph autoencoders used in ref. [37] in appendix A.

JHEP08(2021)080



(a) (b)

Figure 4. The AUC and mean loss for the three signal classes as a function of latent dimension from 2 to 12 for the given architecture.



(a) (b)

Figure 5. The loss of the graph-autoencoder (a) and ROC curves (b) for a network trained only on QCD jets.

where a is the vector-index, i is the node index and G is the set of all nodes of the graph. The normalised distribution of the four classes for each latent dimension is shown in figure 6, while the corresponding AUC for each signal vs. QCD discrimination is shown in figure 7. We find that $\tilde{f}^2$ performs best for top and ϕ jets, while $\tilde{f}^5$ gives the maximum AUC for W jets. The AUC for top quarks and scalar ϕ from $\tilde{f}^2$ is 0.74 and 0.84 respectively. Thus, we find a significant improvement for ϕ from the value obtained with the loss function, which is also the case for the W jets whose AUC is 0.78 from $\tilde{f}^5$. The latent distributions are prone to training uncertainties since they do not have any regularising terms in the loss function.

JHEP08(2021)080

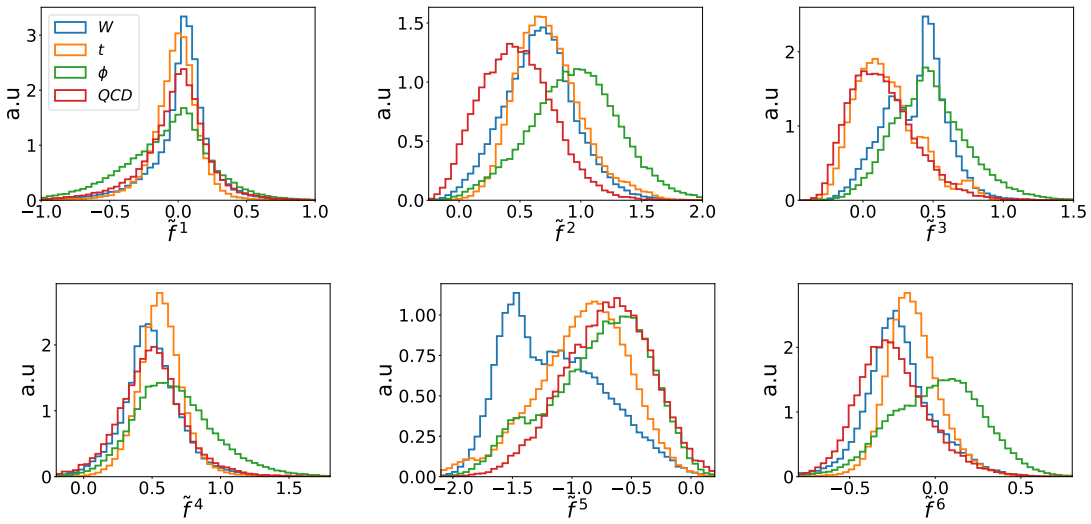


Figure 6. The distribution of six dimensional latent space after the training is performed only on QCD jets.

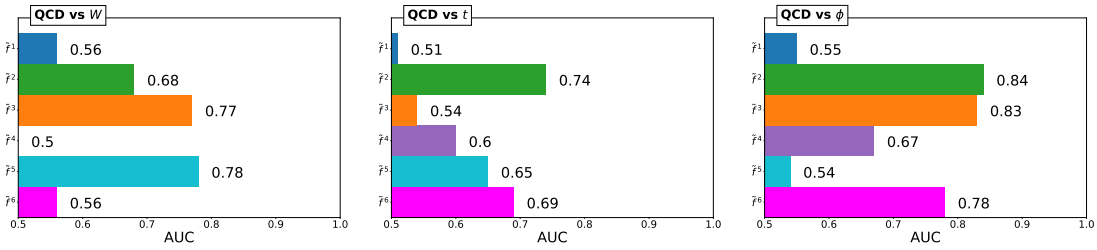


Figure 7. The AUC for all three signal classes corresponding to each latent dimension in the network.

More precisely, the shapes and location of these distributions will vary significantly for different training instances even when they give very similar distributions of the loss function. There are available remedies [59–61] for the training class, but controlling the signal distributions during unsupervised training is not possible by design. However, it may be possible to control them using physically motivated priors, which is beyond the scope of our present work. Nevertheless, once we have a single training instance, latent dimension-based anomaly finders can be used by trimming the encoder network after training to contain only these two outputs. Control samples can be used to quantify the latent space distributions and could therefore find applications in trigger optimisation.

5 Conclusions

In this work, we have introduced a graph neural network-based autoencoder for unsupervised anomaly detection in QCD boosted jet data. We design a novel edge-reconstruction

JHEP08(2021)080

network for the graph-decoder, which allows us to reconstruct multidimensional edge information. This gives the graph-autoencoder the capacity to classify entire graphs, unlike previously existing graph-autoencoders. We use NNConv to incorporate the multidimensional edge and node features as inputs to a graph-autoencoder while utilising edge convolutions to learn inductive latent space representations of QCD jets' graph-structured data.

The anomaly finder based on the reconstruction loss shows good performance for the non-QCD scenarios that we consider. We further explore the possibility of exploiting latent space variables as discriminants for anomalous jets and find that latent variables can indeed lead to improved anomaly detection by accessing the compressed information of the QCD data. While GNNs are known to be good candidates for trigger-level implementations, we study latent dimension-based anomaly finders with graph-autoencoders. Using latent dimensions instead of the loss has the additional appeal of halving the number of layers, thus resulting in a shallower network. Studying the latent dimension representation of QCD jets therefore provides a compressed arena for new physics discovery by using these observables directly.

Acknowledgments

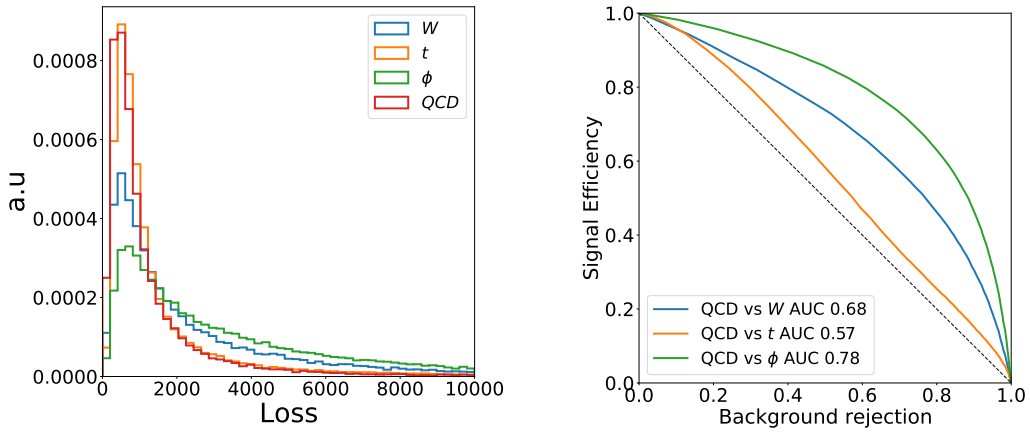
O.A. is supported by the U.K. Science and Technology Facilities Council (STFC) under grant ST/V506692/1. A.B. and C.E. are supported by the STFC under grant ST/T000945/1. C.E is also supported by the IPPP Associateship Scheme. M.S. is supported by the STFC under grant ST/P001246/1. The neural network implementations of this work was performed using the HPC resources (Vikram-100 HPC) and TDP project at PRL.

A Comparison with particle graph autoencoder

We compare the performance of our network with the particle graph autoencoders (PGAE) proposed in ref. [37][6] with our dataset. This study focussed on identifying anomalous events with dijet signatures (large-radius jets and high p_T) and used the two leading jets in the event to learn latent event representations. In contrast, our present focus is jet-level classification. For the input, we consider the four-vector of each microjet as the node feature and use a complete graph with all possible connections. The network is a graph-autoencoder that takes the vectors as input with a single edge convolution to map it to a two-dimensional latent node representation and maps it back with another single edge convolution. We use the mean squared error as the loss function and train with a batch size of 32. For more details of the architecture, we refer the reader to section 3.7 of ref. [37]. We show the distribution of the loss function and the corresponding ROC curve in figure 8. The first thing that we notice is that the location of the peaks is identical for all four classes, with the only difference coming in the tail of the distribution. The value of the AUCs is significantly reduced for W and top jets compared to our work, while for ϕ-jets, the reduction is not that drastic. Out of the three signal classes, ϕ jets are

[6] We use the code available in https://github.com/stsan9/AnomalyDetection4Jets/tree/rnd_cuts.

JHEP08(2021)080



(a) (b)

Figure 8. Distribution of the loss function of the PGAE (a) and the corresponding ROC curves (b) for the different signal classes for a network trained only on QCD jets.

the least QCD-like, and hence, the networks find it easier to distinguish them with less information. At the same time, the edge-reconstruction employed in our architecture helps identify the W and top jets more efficiently. Hence, we infer that the edge-reconstruction and the multidimensional edge feature representation is crucial for a graph-autoencoder as these complex and physically relevant features are not learned by the network even though they are, in principle, constructed from the node features. Moreover, using only the node features, the graph autoencoder is insensitive to the n-prong structure of the signals as the AUCs do not follow the usual QCD intuition. The addition of the edge features and their reconstruction enables the graph-autoencoder to learn the signal jets' n-prong topology in an unsupervised manner.

B Correlation of the loss function with jet variables

The correlation of the loss function with different jet variables is essential in determining the trained network's biases. Although perfectly decorrelated discriminants to the jet's physical variables like transverse momentum (p_T), mass (M), or the number of constituents are highly coveted, it is not possible in practice — known methods to decorrelate them, like adversarial training, diminish the power of the discriminant. We discuss the correlation of our network's loss function with the quantities mentioned earlier in this section. The class-wise correlation of the four quantities is shown in figure 9. We see that the loss function and the p_T are uncorrelated with small positive values (the highest being 0.27 for ϕ), indicating that the loss function tends to increase with an increase in transverse momentum of the jet slightly, although the increase is minimal for the background QCD jets (~ 0.10). Jet mass is an important variable that helps in discriminating different classes of jets. However, a discriminant (the loss function) needs to be decorrelated entirely with jet mass as putting a cut on a correlated variable will lead to artificial bumps in the jet-mass distribution of

JHEP08(2021)080

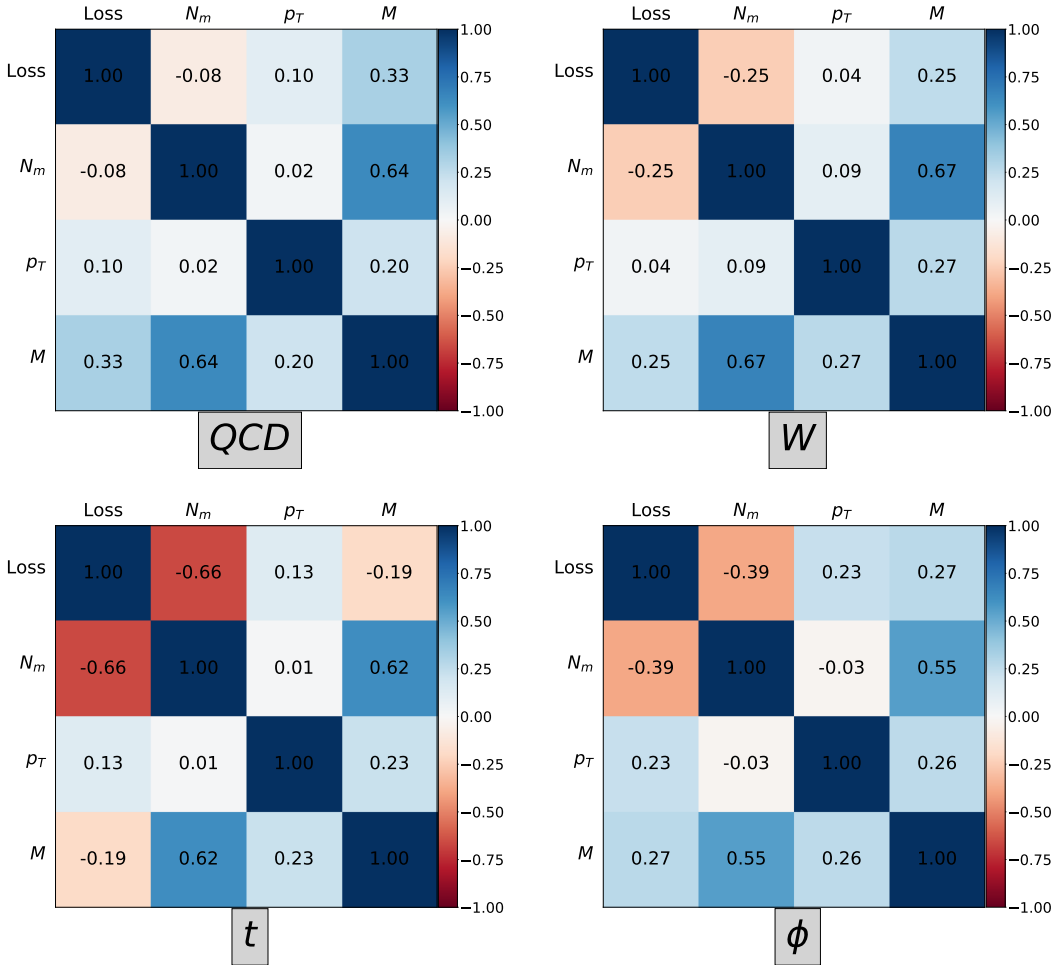


Figure 9. Linear correlation coeeficient between the loss, number of microjets(N_m), transverse momentum (p_T), mass(M) of the jet for the four jet classes: QCD(top left), W-boson (top right), top-quark (bottom left) and ϕ(bottom right).

the selected events. As can be seen, from figure 9, the loss function is reasonably correlated with the jet mass even for the QCD jets. Decorrelating the jet mass from the loss can be done via an adversarial network [27].

The reconstruction efficiency of convolutional autoencoders has been shown to decrease with an increase in the number of non-zero pixels [30], which leads to the possibility of missing out on potential signals with lower complexities than QCD jets. We find that our network behaves in the opposite way: the reconstruction error reduces with an increase in the number of microjets. More importantly, this reduction is minimal for the QCD jets, suggesting that the network learns a uniform feature of the jet graph regardless of the number of microjets. We can understand this independence via the structure of a graph neural network. A graph convolution layer essentially learns a set of weights shared for all the nodes and edges and hence learns the underlying feature regardless of the number of nodes/edges in the graphs. However, there is a strong negative correlation of the loss of

JHEP08(2021)080

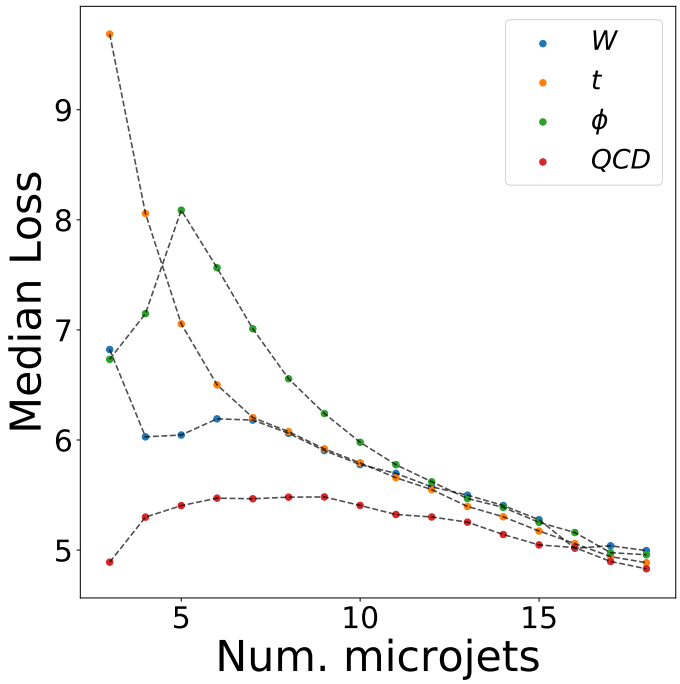


Figure 10. The median loss of events (from the test dataset) with fixed number of microjets for the various types of jets.

the different signal classes with the number of microjets which can be understood via the fact that any extra radiation other than the said multiplicities essentially arise from QCD splittings. To further understand this behavior, we plot the median loss of the events with a fixed number of microjets for the four classes in figure 10. The initial increase in the median loss from three to five for the four-pronged ϕ jets further solidifies the preceding argument regarding the decrease of the loss function with an increase in the microjet multiplicity. Such a peak is absent for the lower multiplicity signal classes.

References

[1] I. Brivio and M. Trott, *The Standard Model as an Effective Field Theory*, *Phys. Rept.* **793** (2019) 1 [arXiv:1706.08945] [INSPIRE].

[2] J.H. Collins, P. Martín-Ramiro, B. Nachman and D. Shih, *Comparing weak- and unsupervised methods for resonant anomaly detection*, *Eur. Phys. J. C* **81** (2021) 617 [arXiv:2104.02092] [INSPIRE].

[3] CMS collaboration, *MUSiC: a model-unspecific search for new physics in proton-proton collisions at* $\sqrt{s} = 13$ *TeV*, *Eur. Phys. J. C* **81** (2021) 629 [arXiv:2010.02984] [INSPIRE].

JHEP08(2021)080

[4] ATLAS collaboration, *A strategy for a general search for new phenomena using data-derived signal regions and its application within the ATLAS experiment*, *Eur. Phys. J. C* **79** (2019) 120 [arXiv:1807.07447] [INSPIRE].

[5] J.H. Collins, K. Howe and B. Nachman, *Anomaly Detection for Resonant New Physics with Machine Learning*, *Phys. Rev. Lett.* **121** (2018) 241803 [arXiv:1805.02664] [INSPIRE].

[6] A. Blance, M. Spannowsky and P. Waite, *Adversarially-trained autoencoders for robust unsupervised new physics searches*, *JHEP* **10** (2019) 047 [arXiv:1905.10384] [INSPIRE].

[7] J. Hajer, Y.-Y. Li, T. Liu and H. Wang, *Novelty Detection Meets Collider Physics*, *Phys. Rev. D* **101** (2020) 076015 [arXiv:1807.10261] [INSPIRE].

[8] A. De Simone and T. Jacques, *Guiding New Physics Searches with Unsupervised Learning*, *Eur. Phys. J. C* **79** (2019) 289 [arXiv:1807.06038] [INSPIRE].

[9] B. Nachman and D. Shih, *Anomaly Detection with Density Estimation*, *Phys. Rev. D* **101** (2020) 075042 [arXiv:2001.04990] [INSPIRE].

[10] B. Nachman, *Anomaly Detection for Physics Analysis and Less than Supervised Learning*, arXiv:2010.14554 [INSPIRE].

[11] S. Marzani, G. Soyez and M. Spannowsky, *Looking inside jets: an introduction to jet substructure and boosted-object phenomenology*, vol. 958, Springer (2019), [DOI] [arXiv:1901.10342] [INSPIRE].

[12] D.E. Soper and M. Spannowsky, *Finding physics signals with shower deconstruction*, *Phys. Rev. D* **84** (2011) 074002 [arXiv:1102.3480] [INSPIRE].

[13] D.E. Soper and M. Spannowsky, *Finding physics signals with event deconstruction*, *Phys. Rev. D* **89** (2014) 094005 [arXiv:1402.1189] [INSPIRE].

[14] ATLAS collaboration, *Measurements of the W production cross sections in association with jets with the ATLAS detector*, *Eur. Phys. J. C* **75** (2015) 82 [arXiv:1409.8639] [INSPIRE].

[15] J. Zhou et al., *Graph neural networks: A review of methods and applications*, (2018) [arXiv:1812.08434].

[16] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang and P.S. Yu, *A comprehensive survey on graph neural networks*, *IEEE Trans. Neural Networks and Learning Systems* **32** (2021) 4 [arXiv:1901.00596].

[17] F.A. Dreyer and H. Qu, *Jet tagging in the Lund plane with graph networks*, *JHEP* **03** (2021) 052 [arXiv:2012.08526] [INSPIRE].

[18] B. Andersson, G. Gustafson, L. Lönnblad and U. Pettersson, *Coherence Effects in Deep Inelastic Scattering*, *Z. Phys. C* **43** (1989) 625 [INSPIRE].

[19] A. Lifson, G.P. Salam and G. Soyez, *Calculating the primary Lund Jet Plane density*, *JHEP* **10** (2020) 170 [arXiv:2007.06578] [INSPIRE].

[20] V. Mikuni and F. Canelli, *ABCNet: An attention-based method for particle tagging*, *Eur. Phys. J. Plus* **135** (2020) 463 [arXiv:2001.05311] [INSPIRE].

[21] O. Knapp, O. Cerri, G. Dissertori, T.Q. Nguyen, M. Pierini and J.-R. Vlimant, *Adversarially Learned Anomaly Detection on CMS Open Data: re-discovering the top quark*, *Eur. Phys. J. Plus* **136** (2021) 236 [arXiv:2005.01598] [INSPIRE].

[22] V. Mikuni and F. Canelli, *Unsupervised clustering for collider physics*, *Phys. Rev. D* **103** (2021) 092007 [arXiv:2010.07106] [INSPIRE].

JHEP08(2021)080

[23] G. Dezoort et al., *Charged particle tracking via edge-classifying interaction networks*, arXiv:2103.16701 [INSPIRE].

[24] M. Abdughani, J. Ren, L. Wu and J.M. Yang, *Probing stop pair production at the LHC with graph neural networks*, *JHEP* **08** (2019) 055 [arXiv:1807.09088] [INSPIRE].

[25] Y. Iiyama et al., *Distance-Weighted Graph Neural Networks on FPGAs for Real-Time Particle Reconstruction in High Energy Physics*, *Front. Big Data* **3** (2020) 598927 [arXiv:2008.03601] [INSPIRE].

[26] M. Farina, Y. Nakai and D. Shih, *Searching for New Physics with Deep Autoencoders*, *Phys. Rev. D* **101** (2020) 075021 [arXiv:1808.08992] [INSPIRE].

[27] T. Heimel, G. Kasieczka, T. Plehn and J.M. Thompson, *QCD or What?*, *SciPost Phys.* **6** (2019) 030 [arXiv:1808.08979] [INSPIRE].

[28] T.S. Roy and A.H. Vijay, *A robust anomaly finder based on autoencoders*, arXiv:1903.02032 [INSPIRE].

[29] C.K. Khosa and V. Sanz, *Anomaly Awareness*, arXiv:2007.14462 [INSPIRE].

[30] T. Finke, M. Krämer, A. Morandini, A. Mück and I. Oleksiyuk, *Autoencoders for unsupervised anomaly detection in high energy physics*, *JHEP* **06** (2021) 161 [arXiv:2104.09051] [INSPIRE].

[31] T. Cheng, J.-F. Arguin, J. Leissner-Martin, J. Pilette and T. Golling, *Variational Autoencoders for Anomalous Jet Tagging*, arXiv:2007.01850 [INSPIRE].

[32] J. Cogan, M. Kagan, E. Strauss and A. Schwartzman, *Jet-Images: Computer Vision Inspired Techniques for Jet Tagging*, *JHEP* **02** (2015) 118 [arXiv:1407.5675] [INSPIRE].

[33] L. de Oliveira, M. Kagan, L. Mackey, B. Nachman and A. Schwartzman, *Jet-images — deep learning edition*, *JHEP* **07** (2016) 069 [arXiv:1511.05190] [INSPIRE].

[34] X. Ju and B. Nachman, *Supervised Jet Clustering with Graph Neural Networks for Lorentz Boosted Bosons*, *Phys. Rev. D* **102** (2020) 075014 [arXiv:2008.06064] [INSPIRE].

[35] H. Qu and L. Gouskos, *ParticleNet: Jet Tagging via Particle Clouds*, *Phys. Rev. D* **101** (2020) 056019 [arXiv:1902.08570] [INSPIRE].

[36] A. Blance and M. Spannowsky, *Unsupervised Event Classification with Graphs on Classical and Photonic Quantum Computers*, arXiv:2103.03897 [INSPIRE].

[37] G. Kasieczka et al., *The LHC Olympics 2020: A Community Challenge for Anomaly Detection in High Energy Physics*, arXiv:2101.08320 [INSPIRE].

[38] T.N. Kipf and M. Welling, *Variational graph auto-encoders*, (2016) [arXiv:1611.07308].

[39] P.V. Tran, *Learning to make predictions on graphs with autoencoders*, in *2018 IEEE 5th international conference on data science and advanced analytics (DSAA)*, IEEE, (2018) [arXiv:1802.08352].

[40] G. Salha, R. Hennequin and M. Vazirgiannis, *Simple and effective graph autoencoders with one-hop linear models*, (2020) [arXiv:2001.07614].

[41] S. Pan, R. Hu, G. Long, J. Jiang, L. Yao and C. Zhang, *Adversarially regularized graph autoencoder for graph embedding*, (2018) [arXiv:1802.04407].

[42] J. Park, M. Lee, H.J. Chang, K. Lee and J.Y. Choi, *Symmetric graph convolutional autoencoder for unsupervised graph representation learning*, in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, (2019) [arXiv:1908.02441].

JHEP08(2021)080

[43] J. Alwall et al., *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] [INSPIRE].

[44] T. Sjöstrand, S. Mrenna and P.Z. Skands, *PYTHIA 6.4 Physics and Manual*, *JHEP* **05** (2006) 026 [hep-ph/0603175] [INSPIRE].

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

[46] M. Cacciari, G.P. Salam and G. Soyez, *FastJet User Manual*, *Eur. Phys. J. C* **72** (2012) 1896 [arXiv:1111.6097] [INSPIRE].

[47] *Performance of shower deconstruction in ATLAS*, CERN, Geneva (Feb, 2014) ATLAS-CONF-2014-003.

[48] S. Catani, Y.L. Dokshitzer, M. Olsson, G. Turnock and B.R. Webber, *New clustering algorithm for multi - jet cross-sections in e^+e^- annihilation*, *Phys. Lett. B* **269** (1991) 432 [INSPIRE].

[49] S.D. Ellis and D.E. Soper, *Successive combination jet algorithm for hadron collisions*, *Phys. Rev. D* **48** (1993) 3160 [hep-ph/9305266] [INSPIRE].

[50] M. Wang et al., *Deep graph library: A graph-centric, highly-performant package for graph neural networks*, (2019) [arXiv:1909.01315].

[51] A. Paszke et al., *Pytorch: An imperative style, high-performance deep learning library*, (2019) [arXiv:1912.01703].

[52] Y. Lecun, L. Bottou, Y. Bengio and P. Haffner, *Gradient-based learning applied to document recognition*, *Proceedings of the IEEE* **86** (1998) 2278.

[53] J. Gilmer, S.S. Schoenholz, P.F. Riley, O. Vinyals and G.E. Dahl, *Neural message passing for quantum chemistry*, in *International Conference on Machine Learning*, PMLR, (2017) [arXiv:1704.01212].

[54] Y. Wang, Y. Sun, Z. Liu, S.E. Sarma, M.M. Bronstein and J.M. Solomon, *Dynamic Graph CNN for Learning on Point Clouds*, *Acm Transactions On Graphics (tog)* **38** (2019) 1 arXiv:1801.07829 [INSPIRE].

[55] D.P. Kingma and J. Ba, *Adam: A Method for Stochastic Optimization*, arXiv:1412.6980 [INSPIRE].

[56] 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] [INSPIRE].

[57] B. Bortolato, B.M. Dillon, J.F. Kamenik and A. Smolkovič, *Bump Hunting in Latent Space*, arXiv:2103.06595 [INSPIRE].

[58] B.M. Dillon, T. Plehn, C. Sauer and P. Sorrenson, *Better Latent Spaces for Better Autoencoders*, arXiv:2104.08291 [INSPIRE].

[59] D.P. Kingma and M. Welling, *Auto-Encoding Variational Bayes*, arXiv:1312.6114 [INSPIRE].

[60] A. Makhzani, J. Shlens, N. Jaitly, I. Goodfellow and B. Frey, *Adversarial autoencoders*, (2015) [arXiv:1511.05644].

[61] G. Patrini et al., *Sinkhorn autoencoders*, (2018) [arXiv:1810.01118].

JHEP08(2021)080