Symmetries, Safety, and Self-Supervision

Barry M. Dillon[1], Gregor Kasieczka[2], Hans Olischlager[1], Tilman Plehn[1],
Peter Sorrenson[3], and Lorenz Vogel[1]

1 Institut für Theoretische Physik, Universität Heidelberg, Germany
2 Institut für Experimentalphysik, Universität Hamburg, Germany
3 Heidelberg Collaboratory for Image Processing, Universität Heidelberg, Germany

August 18, 2021

Abstract

Collider searches face the challenge of defining a representation of high-dimensional data such that physical symmetries are manifest, the discriminating features are retained, and the choice of representation is new-physics agnostic. We introduce JetCLR to solve the mapping from low-level data to optimized observables though self-supervised contrastive learning. As an example, we construct a data representation for top and QCD jets using a permutation-invariant transformer-encoder network and visualize its symmetry properties. We compare the JetCLR representation with alternative representations using linear classifier tests and find it to work quite well.

Content

---

1 Introduction

Symmetries [1] form the core of the fundamental description, phenomenological techniques, and experimental analyses in particle physics. LHC physics is defined by the symmetry structure of LHC data, from the detector geometry to the relativistic space-time symmetries and local gauge symmetries defining the underlying theory, and to new physics motivations like supersymmetry. Any new approach to LHC physics, including applications of machine learning, has to be seen in the context of symmetries eventually [2–5].

Modern machine learning (ML) has spurred the development of techniques which can, among other benefits, boost the development of high-level observables. We typically train a neural network to distinguish between different physical processes either based on low-level, high-dimensional data or on the corresponding Monte-Carlo simulations. The resulting classifier can be viewed as a high-level observable for a given analysis. If the dataset can be understood through first-principle simulations and is large enough to train the networks, this observable will be optimized for the respective task, but lack theoretical calculability. As long as the observable is calibrated and the systematic errors are understood, this lack of calculability is not a barrier for supervised classification.

This agnostic approach works well for supervised analyses, but it is not clear how it can be expanded to unsupervised analyses, like an anomaly search [6–10], a generalized side band analysis [11, 12], or a generalized model hypothesis [13]. For this purpose, we propose to replace a limited number of high-level observables by a high-dimensional representation, and replace full control over all possible physics processes with a structure driven by symmetries and fundamental theory.

The standard application driving ML-methods in LHC physics is jets, a fertile ground for supervised and unsupervised techniques. The most common jet representation is jet images [14–19], a high-dimensional representation defined in rapidity vs azimuthal angle, observables inspired by Lorentz transformations. Jet images typically include a pre-processing step exploiting their rotation symmetry. Alternative symmetry-inspired jet representations include permutation-invariant graphs [20–23], trees [24–27], the Lund plane [28,29], Lorentz-inspired networks [30–34], Deep-Sets networks [35], or Energy Flow Polynomials (EFPs) [36], a calculable basis with a notion of infrared and collinear safety.

Combining unsupervised learning and symmetries we define jet observables using Contrastive Learning of Representations (CLR) [37]. Our key idea is to frame the mapping between the jet constituents' phase space and a representation space as an optimization task with a contrastive loss function, *designed such that the representation space will be invariant to pre-defined symmetries and retains discriminative power.* The training is self-supervised in view of the network's discriminative power, because the optimization never uses truth labels for the jets. For the mapping of physics and representation spaces we employ a transformer-encoder network [38–40]. In addition to its built-in permutation symmetry we implement rotation and translation symmetries, as well as soft and collinear safety augmentations. To benchmark JetCLR we use a standard test in the ML community, the so-called linear classifier test (LCT). For this test a linear network is trained to classify between different processes, quantifying how well classes can be separated by a linear cut in representation space.

We start by introducing contrastive learning in Sec. 2. We then construct our JetCLR tool using a set of symmetries and augmentations in Sec. 3. In Sec. 4 we visualize the invariances of the JetCLR representation and study its performance using a linear classifier. Different such classifiers are discussed in the Appendix.

2 Contrastive learning

The goal of our network is to define a mapping between the jet constituents and a representation space,

$$f : \mathcal{J} \to \mathcal{R} , \tag{1}$$

which is, both,

1. invariant to symmetries and theory-driven augmentations, and
2. discriminative within the dataset it is optimized on.

We work with the top-tagging dataset [6,41,42], where the jets are simulated with Pythia8 [43] (default tune) using a center-of-mass energy of 14 TeV and ignoring pile-up and multi-parton interactions. Delphes [44] provides a fast detector simulation with the default ATLAS detector card, and the jets are defined through anti-k_T algorithm [45, 46] in FastJet [47] with a radius of $R = 0.8$. For each event we keep the leading jet, provided

$$p_T = 550 \;...\; 650 \text{ GeV} \qquad \text{and} \qquad |\eta| < 2 \; . \tag{2}$$

This narrow p_T-range induces the most distinctive feature in the jets, a finite geometric distance between the top decay products in the $\eta - \phi$ plane, whereas for QCD jets the average activity continuously drops away from the hardest constituent. The top jets are required to be matched to a parton-level top and all parton-level decay products to lay within the jet radius. The jet constituents are defined using the Delphes energy-flow algorithm, with the leading 200 constituents from each jet kept for the analysis. Particle-ID and tracking information are not included.

If we assume all jet constituents to be massless, each jet x_i is defined by its constituent coordinates,

$$x_i = \{(p_T, \eta, \phi)_k\} \qquad \text{with} \qquad k = 1 \;...\; n_C \; , \tag{3}$$

so the jet phase space $\mathcal{J}$ has dimension $3n_C$. For the training, we first sample a batch of jets $\{x_i\}$ from the dataset and apply a set of symmetry-inspired augmentations to each jet to generate an augmented batch $\{x'_i\}$. Pairs of original and augmented jets are defined as

$$\begin{aligned} \text{positive pairs:} \qquad & \{(x_i, x'_i)\} \\ \text{negative pairs:} \qquad & \{(x_i, x_j)\} \cup \{(x_i, x'_j)\} \quad \text{for} \quad i \neq j \; . \end{aligned} \tag{4}$$

The goal of the network training is to map positive pairs close together in the representation space $\mathcal{R}$ and negative pairs far apart. This way, positive pairs are used to impose invariances under symmetry transformations or theory augmentations of the jets in $\mathcal{R}$, while the negative pairs are used to ensure that the representation retains discriminative power within the dataset. Truth labels indicating if the jets are QCD or top are never used in the optimization.

Loss function

The mapping of Eq.(1) defines the network outputs z_i and z'_i, each of them vectors describing jets in $\mathbb{R}^{\dim(z)}$. The actual representation, however, is given by $f(x_i) = z_i/|z_i|$ and $f(x'_i) = z'_i/|z'_i|$, which means it is defined on a unit hypersphere

$$\mathcal{R} = S^{\dim(z)-1} \; . \tag{5}$$

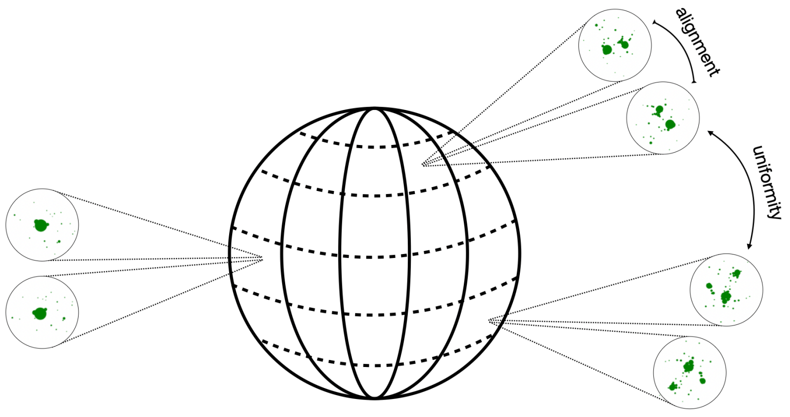

Figure 1: Illustration of the uniformity and alignment concepts behind the contrastive learning.

On this sphere we define the similarity between two jets as [37]

$$s(z_i, z_j) = \frac{z_i \cdot z_j}{|z_i||z_j|} = \cos\theta_{ij} \; , \tag{6}$$

with θ_{ij} being the angle between the jets in $\mathcal{R}$. The contrastive loss for a positive pair of jets is defined in terms of this distance as

$$\mathcal{L}_i = -\log \frac{e^{s(z_i, z_i')/\tau}}{\sum_{j\neq i \in \text{batch}} \left[e^{s(z_i, z_j)/\tau} + e^{s(z_i, z_j')/\tau} \right]} \; , \tag{7}$$

and the total loss is given by the sum over all positive pairs in the batch, $\mathcal{L} = \sum_i \mathcal{L}_i$. Because the positive pairs appear in the numerator, while the negative pairs contribute to the denominator, the loss decreases when the distance between positive pairs becomes smaller and when the distance between negative pairs becomes larger. The hyper-parameter τ is referred to as the temperature and controls the relative influence of positive pairs and negative pairs. The cosine similarity in Eq.(6) is not a proper distance metric, but we can define an angular distance as $d(z_i, z_j) = \theta_{ij}/\pi = 0 \; ... \; 1$, such that it satisfies the triangle inequality.

Uniformity vs alignment

The contrastive loss can be understood in terms of uniformity versus alignment on the unit hypersphere defining $\mathcal{R}$, illustrated in Fig. 1. The numerator of Eq.(7), describing the positive pairs, is minimal when all jets and their augmented counterparts are mapped to the same point, $s(z_i, z_i') = 1$. On a hypersphere, the negative pairs cannot be pushed infinitely far apart, as would be possible in $\mathbb{R}^{\dim(z)}$, so the corresponding loss is minimal when the jets are uniformly distributed on the hypersphere. We can measure uniformity and alignment through

$$\begin{aligned} \mathcal{L}_\text{align} &= \frac{1}{N_\text{batch}} \sum_{i\in\text{batch}} s(z_i, z_i') \\ \mathcal{L}_\text{uniform} &= \frac{1}{N_\text{batch}} \sum_{i\in\text{batch}} \log \sum_{j\neq i} \left[e^{-s(z_i, z_j)} + e^{-s(z_i, z_j')} \right] \; . \end{aligned} \tag{8}$$

While the alignment function has the trivial solution where all jets and all augmented jets are mapped to the same point, the uniformity function does not have such a solution. To map the jets to a uniform distribution in a high-dimensional space, the mapping must learn features of the jets to discriminate between them and map them to different points. Uniformity alone is a sufficient optimization task to obtain a representation with discriminative power. The additional alignment condition develops a mapping to $\mathcal{R}$, which focuses on the invariance with respect to augmentations and symmetries. The combined contrastive learning will not find representations which are perfectly aligned or perfectly uniform.

Symmetries and augmentations

The mapping to the representation space $\mathcal{R}$ is optimized to be approximately invariant to pre-defined symmetry transformations and data augmentations. Before applying symmetry transformations and augmentations in the contrastive learning method we center the jets such that the p_T-weighted centroid is at the origin in the $\eta - \phi$ plane.

Rotations around the jet axes turn out to be a very efficient symmetry we can impose on our representations. In the jet image representation this is included through preprocessing, where each jet is centered and then rotated such that its principal axis points at 12 o'clock. Energy flow polynomials are rotationally invariant by construction, since they are built from angular distances between the jet constituents. We apply rotations to a batch of jets by rotating each jet through angles sampled from $0\ ...\ 2\pi$. Such rotations in the $\eta - \phi$ plane are not Lorentz transformations and do not preserve the jet mass, but for narrow jets with $R \lesssim 1$ the corrections to the jet mass can be neglected.

As a second symmetry we implement translations in the $\eta - \phi$ plane. To do so, all constituents in a jet are shifted by the same random distance, where shifts in each direction are limited to between $-1\ ...\ 1$. This performs better than restricting to smaller shifts.

In addition to (approximate) symmetries, we also employ theory-inspired augmentations. The distinction between the two is much clearer in our physics application than it is in traditional machine learning. Quantum field theory tells us that soft gluon radiation is universal and factorizes from the hard physics in the jet splittings. To encode this invariance in $\mathcal{R}$ we augment our jets by smearing the positions of the soft constituents, *i.e.* by re-sampling the η and ϕ coordinates of each constituent from a Gaussian distribution centred on the original coordinates,

$$\eta' \sim \mathcal{N}\left(\eta, \frac{\Lambda_\text{soft}}{p_T}\right) \qquad \text{and} \qquad \phi' \sim \mathcal{N}\left(\phi, \frac{\Lambda_\text{soft}}{p_T}\right) \; , \tag{9}$$

with a p_T-suppression in the variance relative to $\Lambda_\text{soft} = 100$ MeV.

Similar to soft splittings, also collinear splitting lead to divergences in perturbative quantum field theory. In practice. they are removed through the finite angular resolution of a detector, which will not be able to distinguish two constituents with $p_{T,a}$ and $p_{T,b}$ at vanishing separation $\Delta R_{ab} \ll 1$. We introduce collinear augmentations to encode this feature by selecting constituents and splitting them such that the total p_T in an infinitesimal region of the detector is unchanged,

$$\begin{aligned} p_{T,a} + p_{T,b} = p_T \qquad\qquad & \eta_a = \eta_b = \eta \\ & \phi_a = \phi_b = \phi \; . \end{aligned} \tag{10}$$

Our soft and collinear augmentations will enforce an approximate IRC-safety in the jet representation. Unlike for instance EFPs, we do not explicitly enforce it through a fixed set of angular correlations or p_T-scalings, but let the contrastive optimization determine the mapping to the representation space.

3 JetCLR

The symmetries discussed in the last section leave out one of the key symmetries in jet representations, namely permutation symmetry. We will include it through the transformer architecture, mapping jet phase space to representation space. The combination of contrastive loss and a permutation-invariant network architecture defines our JetCLR concept.

Attention

The key feature of transformer networks is attention [48,49], more specifically self-attention, which is an operation on a set of elements. Attention allows an element of the set to assign weights to other elements. These weights are then multiplied with some other piece of information from the elements, so that more 'attention' is placed on those elements which achieve a high weight. We use the scaled dot-product multi-headed self-attention of Ref. [50].

We illustrate the single-headed attention mechanism in Fig. 2. In this case, we apply it to a single jet constituent described by the phase space coordinate x_i, where we slightly abuse our notation such that the index now refers to constituents rather than jets and in the following we will just consider x_1 for one of the jets. The learned weight matrix W^Q transforms a single input x_1 to the corresponding query $q_1 = W^Q x_1$. The complete set of inputs x_1 ... x_C is transformed into keys k_1 ... k_C through a learned matrix W^K. Queries and keys are then combined through a scalar product, which is scaled by the dimension d of q_i, and normalized by a softmax function to a set of weights $a_i \in [0, 1]$. Finally, the complete set of inputs x_1 ... x_C is transformed into a set of values v_1 ... v_C through a third learned matrix W^V, and these v_i are weighted by the a_i to provide a network output

$$z_1 = \sum_i \text{softmax}_i(q_1, k_i)\, v_i = \sum_i \text{softmax}_i \left(\frac{(W^Q x_1) \cdot (W^K x_i)}{\sqrt{d}} \right) W^V x_i \; . \tag{11}$$

This can be thought of as a projection onto the basis v_i, where the coefficients are given by the softmax(q_1, k_i) between q_1 and the k_i. The equivalent operation is applied to all

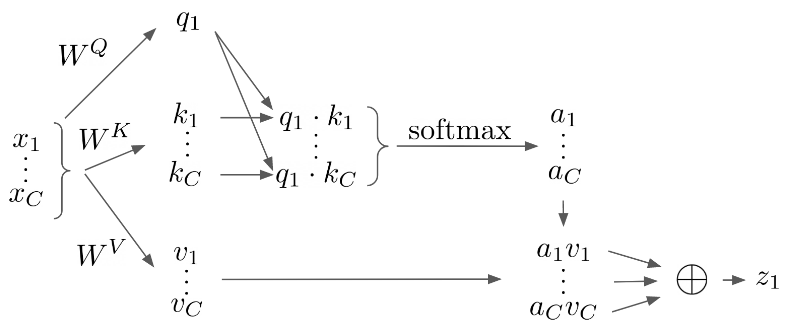

Figure 2: Illustration of single-headed self-attention. All elements are defined in the text.

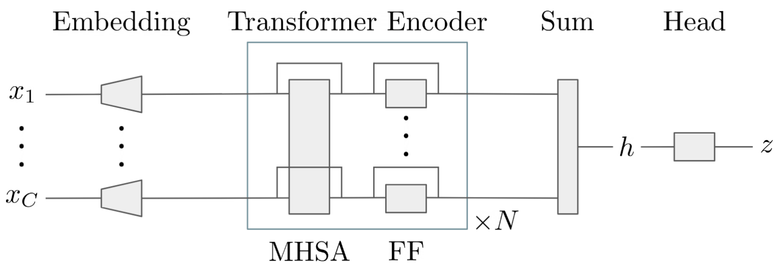

Figure 3: Illustration of the transformer network architecture. MHSA stands for multi-headed self-attention, and FF for a feed-forward block, as defined in the text.

x_j, leading to a set of outputs z_j. Due to the sum over set elements, each output z_i is invariant to the permutation of the other elements of the set, meaning that the entire self-attention operation is permutation equivariant.

A problem with the self-attention mechanism is that each element of the sequence tends to attend dominantly to itself [50]. This can be solved by extending the network to multiple heads, where we perform several self-attention operations in parallel, each with separate learned weight matrices, then concatenate the outputs before applying a final linear layer. In practice, the full calculation for all constituents, all attention heads, and for an entire batch is carried out in parallel with tensor operations.

Transformer encoder

In general, transformer networks include a complete encoder-decoder architecture. In our application, we are only interested in deriving a representation $\mathcal{J} \to \mathcal{R}$, so we use only the encoder part of Ref. [50]. It is a sequence-to-sequence operation, made up of N structurally identical, successive blocks.

The starting point is a set of constituents x_i, which we embed into a higher-dimensional space by a single learned linear layer without activation. This increases the representational power of the network. A typical dimension of the embedding space is 1000. Working with the embedded jet constituents, each block contains the following operations: multi-headed self-attention is applied to the input constituent, and the result is added to the input, in a residual fashion. This output is normalized using layer normalization [51] and passed through a residual feed-forward network, which operates on each constituent individually. The transformer–encoder block is repeated N times. Finally, the output is normalized using layer normalization. The encoder outputs are summed over constituents to produce a fixed-size output h, which is passed through a final feed-forward head network to give the output z. In practice, our supervised linear classifier test will find that h is a better representation than z, consistent with typical practice in the self-supervised literature [37]. While the output of the transformer–encoder is permutation-equivariant, the sum makes the representation h permutation-invariant, similar to the Deep Sets approach [35,52]. Our network is implemented in PyTorch [53] with the TransformerEncoder module, we also make heavy use of NumPy [54].

Variable-length inputs

A general feature of jet constituents is that their number per jet is variable. As in all ML tools for jet analyses, we zero-pad jets with fewer constituents. This makes it easier

hyper-parameter	value
Model (embedding) dimension	1000
Feed-forward hidden dimension	same
Output dimension	same
# self-attention heads	4
# transformer layers (N)	4
# layers	2
Dropout rate	0.1

hyper-parameter	value
Optimizer	Adam($\beta_1 = 0.9, \beta_2 = 0.999$)
learning rate	5×10^{-5}
batch size	128
# epochs	500

Table 1: Default setup of the transformer encoder and the training, unless noted explicitly.

to convert a batch to a single tensor input for efficient computation and allows us to concatenate the batch elements with equal length. To ensure that this padding does not affect the network output, we implement masking in the transformer. To stop information flow from zero-valued constituents, we require the attention weights corresponding to those constituents to be zero, technically by adding negative infinity to the attention weight before the softmax normalization. In addition, we remove zero constituents from the sum over constituents to ensure that the transformer is completely invariant to zero padding.

This masking ensures that constituents with zero p_T have no effect on the output, but we can generalize this by defining the masking to be continuous in p_T. Instead of adding negative infinity to some pre-softmax attention weights, we add $\beta \log p_T$ ($\beta = 0.5$) to all pre-softmax attention weights. In addition, instead of setting some transformer outputs before summation to zero, we multiply all transformer outputs by the input p_T. This IR-safe attention mechanism renders the transformer network IR-safe by construction.

4 Pretty good results

After introducing all JetCLR elements, we have to investigate how its various symmetries and augmentations contribute to its performance and how our representation compares to alternative approaches.

Our transformer setup is given in Tab. 1. The temperature hyper-parameter τ determines the trade-off between the alignment and uniformity. It can strongly affect the performance of the representations in a LCT. We find that $\tau = 0.1$ ... 0.2 works best which, despite the very different applications, is in agreement with Refs [37,55]. We also see that more model dimensions result in better performance, although with the transformer network this performance gain seems to plateau around 1000 ... 1500 dimensions. Earlier tests using a fully connected network instead of a transformer indicated that this plateau happens at around 200 dimensions. We focus on the 1000-dimensional representations because this will eventually provide a fair comparison to the EFPs at $d \leq 7$, which is also a 1000-dimensional representation of the jets.

Our LCT is a linear neural network with a binary cross-entropy loss, optimized using stochastic gradient descent. The network is trained with 50k top and QCD jets each for 5000 epochs with a batch size of 2056. The exact set-up along with some alternative LCT setups are discussed in the Appendix, with a focus on their respective strengths and underlying assumptions.

Augmentation	$\epsilon^{-1}(\epsilon_s = 0.5)$	AUC
none	15	0.905
translations	19	0.916
rotations	21	0.930
soft+collinear	89	0.970
all combined (default)	181	0.980

S/B	$\epsilon^{-1}(\epsilon_s = 0.5)$	AUC
1.00	181	0.980
0.50	160	0.979
0.25	150	0.978
0.10	161	0.978
0.05	146	0.978
0.01	158	0.978

Table 2: Left: classification results for JetCLR trained with different symmetries and augmentations and $S/B = 1$. The default setup includes translation and rotation symmetries, combined with soft and collinear augmentations. Right: classification results for the combined (default) symmetries and augmentations, trained with different S/B.

JetCLR

From first principles, it is not clear which symmetries and augmentations work best for learning representations with JetCLR. In the left panel of Tab. 2 we summarize the results after applying rotational and translational symmetry transformations and soft+collinear augmentations. To get an idea, we quote the best of a number of runs for each case. Individually, the soft+collinear augmentation works best. Translations and rotations are less powerful individually, but the combination of all three provides by far the best representations. The results for the individual augmentations in Tab. 2 were obtained using regular masking in the transformer. When combining all symmetries and augmentations the IR-safe masking gives a slight boost, so our default in Tab. 2 includes IR-safe attention.

While our initial results are based on a dataset containing equal amounts of QCD and top jets, any application to anomaly detection requires our approach to work with much fewer top signal jets. In the right panel of Tab. 2 we show the performance of our default benchmark for a decreasing fraction of signal events in the training sample. For each signal model with $S/B < 1$ we only train one model, so we expect some noise in the results. The outcome indicates that the JetCLR performance in the LCT is hardly sensitive to the amount of signal jets in the training data, and that JetCLR can encode its fundamental structures based on QCD jets and the symmetries and augmentations alone. Due to the stochastic nature of jet data, no pattern is exclusive to top jets, so the QCD jet sample should indeed contain all relevant information. This result is very promising for future anomaly searches using JetCLR representations.

To test our JetCLR training we analyse the AUC and the mistagging rate of the LCT on the test data as a function of the training epoch. Given the impressive LCT scores we could just assume that optimizing the contrastive loss is a good auxiliary task for constructing good representations for classification. However, for anomaly detection we know that the auxiliary optimization task may appear to be converging to a good representation for classification initially, but then diverge at larger epochs. In Fig. 4 we show that the increasing performance of the JetCLR representations in the LCT is indeed aligned with the optimization of the contrastive loss function.

Encoded symmetries

Of the two basic JetCLR tasks, invariance and discriminative power, we first confirm that the network indeed encodes symmetries. To illustrate the encoded rotation symmetry we show how the representation is invariant to actual rotations of jets. We start with a

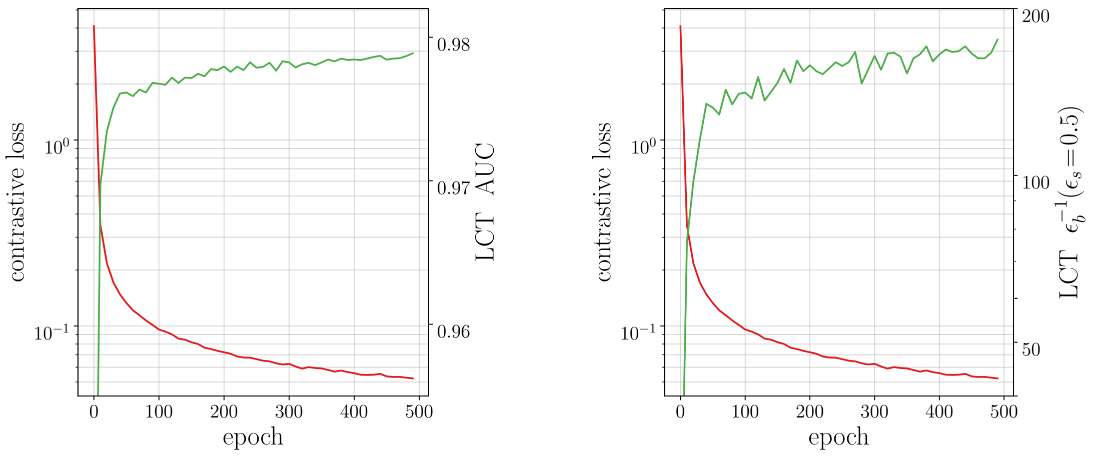

Figure 4: Loss and LCT performance on test data as a function of the training epochs. The LCT is performed every 10 epochs.

batch of 100 jets, and produce a set of rotated copies for each jet, with rotation angles evenly spaced in 0 ... 2π. We then pass each jet and its rotated copy through the network, and calculate their cosine similarity, Eq.(6), with the original jet. In the top panels of Fig. 5 we show the mean and standard deviation of the cosine similarity as a function of the rotation angle. First, from the scale of the radial axis $s(z, z')$ we see that the representations obtained by training JetCLR with rotations are much more similar to the original jets. Second, in the left panel the similarity varies between 0.5 and 1.0 as a function of the rotation angle, while in the right panel the JetCLR representation is indeed rotationally invariant.

Next, we create toy jets with $p_{T,j} = 600$ GeV, one with two constituents and one with three equally spaced constituents. The jet momentum is shared equally between the subjets. We then compare how rotationally invariant their JetCLR representations are in the lower panels of Fig. 5. The red lines represent the similarity functions for JetCLR representations of two-prong (left) and three-prong (right) jets, trained without rotational transformations. The maximum values of $s(z, z')$ reflect the degeneracies from the geometric symmetry of the toy jets. The green line represents the similarity function for the JetCLR representations trained with rotational transformations.

JetCLR performance

After confirming that the JetCLR indeed encodes symmetries, we turn to the second task, namely discriminative power. To put the results of Tab. 2 into context, we show ROC curves for JetCLR and various other representations in Fig. 6. For the constituents representation we take the 20 hardest constituents in each jet, flatten their (p_T, η, ϕ) components into a single vector, and feed them to the linear classifier. For the jet images representation we use the pre-processing of Ref. [6, 7], flattening the 40×40 image to a single 1600-dimensional vector and giving it to the linear classifier. Finally, the EFP representation is invariant to permutations and to rotations by construction, and its IR-safety guarantees independence from soft activity. In many ways, EFPs can be considered a theory-driven counterpart of our JetCLR tool. We use all coefficients up to degree seven, since it has been shown that adding higher powers does not improve the top-tagging [36], and choose $\beta = 0.5$ for the exponent of the p_T-weights. For the JetCLR representation we used the default setup, restricting the maximum number of constituents per jet to 50,

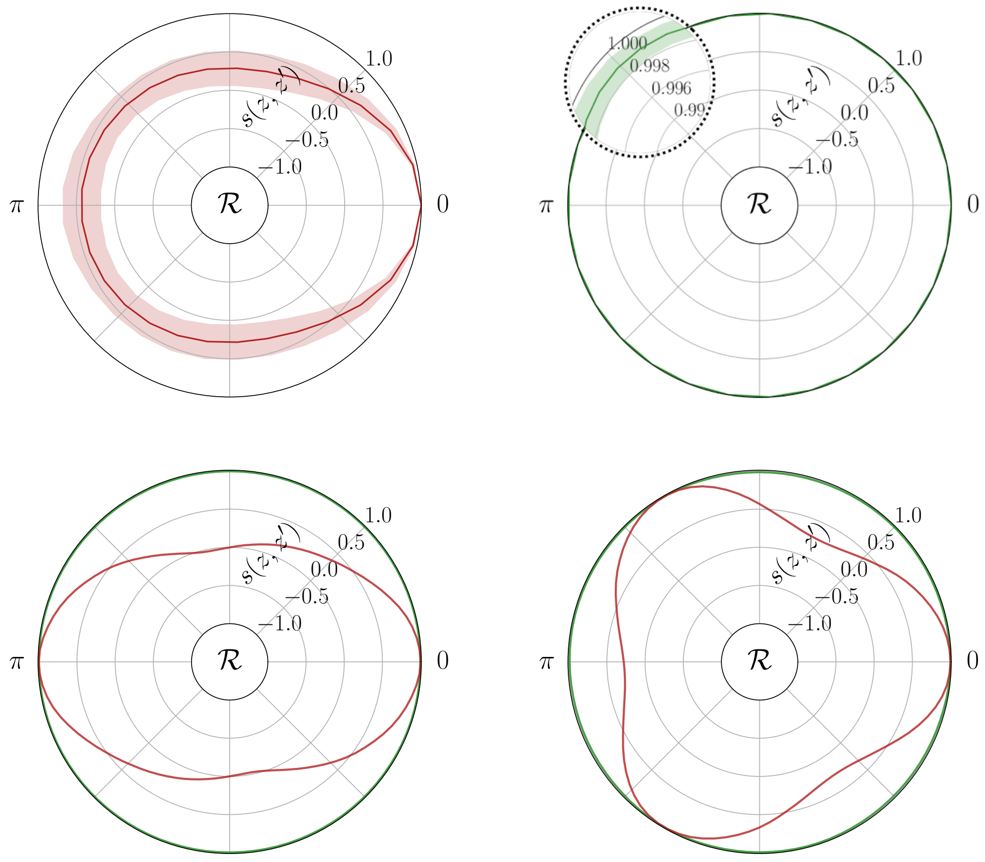

Figure 5: Visualization of the rotational invariance in representation space, keeping in mind that $s(z, z') = 1$ indicates identical representations. Top: JetCLR representation trained without (left) and with (right) rotational transformations. Note the different scales of the radial axes showing the jet similarity defined in Eq.(6). Bottom: JetCLR representation for 2-prong (left) and 3-prong (right) toy jets, trained without (red) and with (green) rotational transformations.

and masking jets with fewer constituents.

For all representations we train the networks with 100k jets split evenly between top and QCD. For the alternative representations we run eight linear classifiers and use the mean over the mistag rates for the ROC curve. For the JetCLR representation an additional source of uncertainty arises from the training of the transformer-encoder network. We train two linear classifiers on four different representations from four different JetCLR runs and show the mean and standard deviation of the mistag rate vs the efficiency.

As expected, representations using more knowledge of the physical symmetries perform increasingly well. The top-performing EFP and JetCLR representations use the same latent dimension, and the self-supervised JetCLR method slightly outperforms the systematic EFP representation for a linear network with a binary cross-entropy loss, the LCT with the weakest assumptions about the data. As discussed in the Appendix, the EFP results improve for a linear discriminant analysis, where the underlying assumption on the data is not applicable to JetCLR. Obviously, any LCT can only serve as a proxy to estimate representation quality, the real test will be performance in an actual analysis.

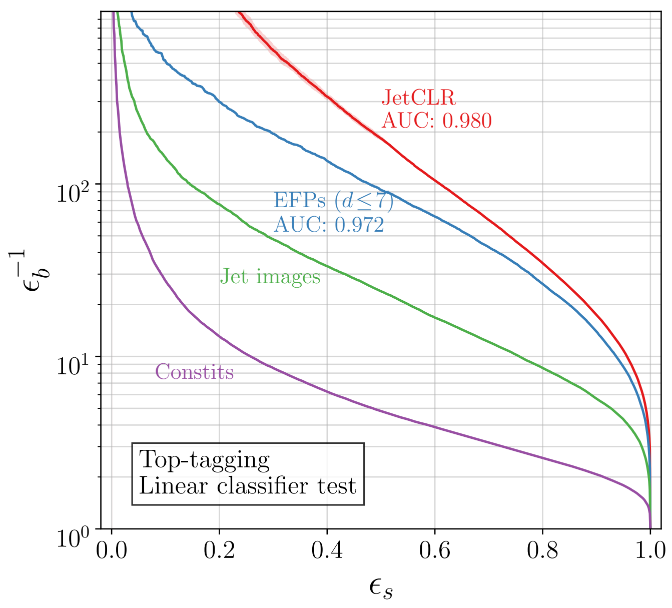

Figure 6: Comparison of JetCLR with other classification metrics.

5 Conclusions

We have introduced Contrastive Learning (CLR) to design observables which respect symmetries and data augmentations while retaining discrimination power within the dataset. We have applied this new method in jet physics, developing the JetCLR tool[*]. Guided by fundamental symmetries and principles of quantum field theory, we introduced a transformer-encoder network to encode rotation, translation, and permutation symmetries, as well as invariance under soft and collinear constituent augmentations.

After visualizing the symmetry-enhanced representation space, we evaluated the network performance using a linear classifier test, a simple supervised classifier trained on the representations to distinguish top jets from QCD jets. Due to the simplicity of the classifier, its performance can be interpreted as a quality measure for the representations. We find that self-supervised JetCLR outperforms simple jet images and is competitive with energy flow polynomials.

Regardless of our specific JetCLR application, our key point is that it is possible to incorporate symmetry principles and physics knowledge in self-supervised ML-tools and latent representations. This opens many avenues for future work with JetCLR and contrastive learning in general. Because of the way JetCLR incorporates symmetries from a single augmented data set, it is particularly well suited to enhance and control anomaly searches, one of the great ML-opportunities for future LHC runs.

Acknowledgments

We are, again, grateful to Ullrich Köthe for invaluable discussions. The research of TP is supported by the Deutsche Forschungsgemeinschaft under grant 396021762 – TRR 257 *Particle Physics Phenomenology after the Higgs Discovery*. PS is partly supported by the DFG Research Training Group GK-1940, *Particle Physics Beyond the Standard Model*.

[*]The JetCLR code will be maintained at https://github.com/bmdillon/JetCLR

BMD is supported by funding from BMBF. GK acknowledges the support of the Deutsche Forschungsgemeinschaft under Germany's Excellence Strategy – EXC 2121 *Quantum Universe* – 390833306.

A Linear classifier tests

Without a direct application to a specific task, comparing data representations is difficult. Downstream tasks can vary from anomaly detection to classification, or regression. One standard method for comparing representations is the linear classifier test (LCT), but even this test can be ambiguous. The idea behind the LCT is that the linearity of the classifier removes much of the expressive power from the classifier, so a linear classifier measures the expressive power of the representation. However, we find that in removing expressive power from the classifier, the results become much more dependent on the inductive biases incurred in the choice of loss function and optimization. We discuss a few different LCTs and explain the assumptions they make about the data they are optimized on. We provide a more complete comparison between the JetCLR and EFP representations using different LCTs in Tab. 3. All classifiers are trained using 10-fold cross validation to identify the best-performing hyperparameters. The reported performance is the average over the 10 folds.

Binary cross entropy loss

A linear classifier trained with binary cross-entropy loss, also known as logistic regression, makes an assumption about how the probability of each of the two classes changes in different parts of the space. If x denotes data and $y \in \{0, 1\}$ the two classes, the assumption is that

$$p(y = 1|x) = \text{sigmoid}(w^T x + c) = \frac{1}{1 + e^{-w^T x - c}} \,, \tag{12}$$

where w is some vector and c is a scalar bias. We find w and c by minimizing the KL-divergence between this model and the labeled data. In practice, this means minimizing the binary cross entropy

$$\mathcal{L} = \Big\langle - \log \text{sigmoid}(y(w^T x + c)) \Big\rangle_{x,y} + \lambda \|w\|^2, \tag{13}$$

where $\lambda \geq 0$ is a regularization parameter. Regularization is not strictly necessary but can improve performance. It can be turned off by setting $\lambda = 0$. We select the best $\lambda \in \{10^{-6}, 10^{-4}, 10^{-2}\}$ by 10-fold cross validation. This optimization problem is convex, so it should always give the same optimal w and c when using a standard algorithm such as gradient descent.

Support vector machine

A (linear) support vector machine (SVM) separates two classes with as wide a margin as possible, aiming for robustness. When the two classes are not linearly separable, as in our application, the SVM minimizes how far on the wrong side of the decision boundary misclassified points are, but it does not consider points which are safely on the correct side of the boundary. This is in contrast to logistic regression, which pushes points to the

	EFPs ($d \leq 7$)		JetCLR	
	$\epsilon^{-1}(\epsilon_s = 0.5)$	AUC	$\epsilon^{-1}(\epsilon_s = 0.5)$	AUC
Binary cross-entropy (Fig. 6)	93	0.972	181	0.980
SVM (hinge loss)	88	0.971	130	0.977
SVM (squared hinge loss)	100	0.971	169	0.979
Linear discriminant analysis	165	0.979	133	0.977

Table 3: Comparison of JetCLR and energy flow polynomials as in Fig. 6, including different linear classifier tests.

correct side of the decision boundary no matter how far over it they already are. SVMs are expressible as a convex problem and make no assumptions about the distribution of the data. However, they involve tuning a hyperparameter which determines how strictly misclassifications are enforced. This will lead to different results depending on the choice of hyperparameter, which must be selected based on performance on some other metric, for instance a classification accuracy. We consider two variants of SVM, starting with the standard hinge loss

$$\mathcal{L} = \Big\langle \max(0, 1 - y(w^T x + c)) \Big\rangle_{x,y} + \lambda \|w\|^2 \; , \tag{14}$$

where $\lambda > 0$ is the regularization parameter. The variant with the squared hinge loss minimizes

$$\mathcal{L} = \Big\langle \max(0, 1 - y(w^T x + c))^2 \Big\rangle_{x,y} + \lambda \|w\|^2 \; . \tag{15}$$

The two differ in how strongly they penalize distance from the decision boundary. The squared variant enacts a weaker penalty for points which are only just on the correct side of the boundary, but a stronger penalty for points which are on the incorrect side. We select the best $\lambda \in \{10^{-6}, 10^{-4}, 10^{-2}\}$ by 10-fold cross validation.

Linear discriminant analysis

Linear discriminant analysis makes a stronger assumption, namely Gaussian distributed data. The data is modeled as a Gaussian mixture model with two equally likely classes, where the covariance matrix of the two classes is the same. The means $\mu_{0,1}$ and the covariance Σ are typically estimated from labeled data. The Bayes-optimal classifier is the log ratio of the two probabilities. Gaussian log probabilities are quadratic in x, so if the two covariance-terms cancel we obtain a linear equation in x. As with the previous classifiers, the classification score can be written as $w^T x + c$, where here $w = (\mu_1 - \mu_0)^T \Sigma^{-1}$ and $c = -w^T(\mu_0 + \mu_1)/2$. No cross validation is necessary to select hyperparameters, but the reported results are averages over 10-fold cross validation. The basic assumption of Gaussian data is not fulfilled by the JetCLR representations, which are optimized for uniformity on a unit hypersphere, so the linear discriminant analysis will not capture the structure of the JetCLR representation.

References

[1] E. Noether, *Invariante variationsprobleme*, Nachrichten von der Gesellschaft der Wissenschaften zu Göttingen, Mathematisch-Physikalische Klasse **1918** (1918) 235.

[2] S. Krippendorf and M. Syvaeri, *Detecting Symmetries with Neural Networks*, arXiv:2003.13679 [physics.comp-ph].

[3] G. Barenboim, J. Hirn, and V. Sanz, *Symmetry meets AI*, arXiv:2103.06115 [cs.LG].

[4] A. Maiti, K. Stoner, and J. Halverson, *Symmetry-via-Duality: Invariant Neural Network Densities from Parameter-Space Correlators*, arXiv:2106.00694 [cs.LG].

[5] S. Krippendorf, R. Kroepsch, and M. Syvaeri, *Revealing systematics in phenomenologically viable flux vacua with reinforcement learning*, arXiv:2107.04039 [hep-th].

[6] T. Heimel, G. Kasieczka, T. Plehn, and J. M. Thompson, *QCD or What?*, SciPost Phys. **6** (2019) 3, 030, arXiv:1808.08979 [hep-ph].

[7] M. Farina, Y. Nakai, and D. Shih, *Searching for New Physics with Deep Autoencoders*, Phys. Rev. D **101** (2020) 7, 075021, arXiv:1808.08992 [hep-ph].

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

[9] B. Bortolato, B. M. Dillon, J. F. Kamenik, and A. Smolkovič, *Bump Hunting in Latent Space*, arXiv:2103.06595 [hep-ph].

[10] B. M. Dillon, T. Plehn, C. Sauer, and P. Sorrenson, *Better Latent Spaces for Better Autoencoders*, arXiv:2104.08291 [hep-ph].

[11] E. M. Metodiev, B. Nachman, and J. Thaler, *Classification without labels: Learning from mixed samples in high energy physics*, JHEP **10** (2017) 174, arXiv:1708.02949 [hep-ph].

[12] G. Kasieczka, B. Nachman, M. D. Schwartz, and D. Shih, *Automating the ABCD method with machine learning*, Phys. Rev. D **103** (2021) 3, 035021, arXiv:2007.14400 [hep-ph].

[13] J. Barron, D. Curtin, G. Kasieczka, T. Plehn, and A. Spourdalakis, *Unsupervised Hadronic SUEP at the LHC*, arXiv:2107.12379 [hep-ph].

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

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

[16] G. Kasieczka, T. Plehn, M. Russell, and T. Schell, *Deep-learning Top Taggers or The End of QCD?*, JHEP **05** (2017) 006, arXiv:1701.08784 [hep-ph].

[17] J. Lin, M. Freytsis, I. Moult, and B. Nachman, *Boosting $H \to b\bar{b}$ with Machine Learning*, JHEP **10** (2018) 101, arXiv:1807.10768 [hep-ph].

[18] P. T. Komiske, E. M. Metodiev, and M. D. Schwartz, *Deep learning in color: towards automated quark/gluon jet discrimination*, JHEP **01** (2017) 110, arXiv:1612.01551 [hep-ph].

[19] 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].

[20] I. Henrion, K. Cranmer, J. Bruna, K. Cho, J. Brehmer, G. Louppe, and G. Rochette, *Neural Message Passing for Jet Physics*, Proceedings of the Deep Learning for Physical Sciences Workshop at NIPS (2017) (2017) . 2017.

[21] S. R. Qasim, J. Kieseler, Y. Iiyama, and M. Pierini, *Learning representations of irregular particle-detector geometry with distance-weighted graph networks*, Eur. Phys. J. C **79** (2019) 7, 608, arXiv:1902.07987 [physics.data-an].

[22] A. Chakraborty, S. H. Lim, and M. M. Nojiri, *Interpretable deep learning for two-prong jet classification with jet spectra*, JHEP **07** (2019) 135, arXiv:1904.02092 [hep-ph].

[23] J. Shlomi, P. Battaglia, and J.-R. Vlimant, *Graph Neural Networks in Particle Physics*, arXiv:2007.13681 [hep-ex].

[24] G. Louppe, K. Cho, C. Becot, and K. Cranmer, *QCD-Aware Recursive Neural Networks for Jet Physics*, arXiv:1702.00748 [hep-ph].

[25] A. Andreassen, I. Feige, C. Frye, and M. D. Schwartz, *JUNIPR: a Framework for Unsupervised Machine Learning in Particle Physics*, arXiv:1804.09720 [hep-ph].

[26] B. M. Dillon, D. A. Faroughy, and J. F. Kamenik, *Uncovering latent jet substructure*, Phys. Rev. **D100** (2019) 5, 056002, arXiv:1904.04200 [hep-ph].

[27] 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].

[28] F. A. Dreyer, G. P. Salam, and G. Soyez, *The Lund Jet Plane*, JHEP **12** (2018) 064, arXiv:1807.04758 [hep-ph].

[29] S. Carrazza and F. A. Dreyer, *Lund jet images from generative and cycle-consistent adversarial networks*, Eur. Phys. J. **C79** (2019) 11, 979, arXiv:1909.01359 [hep-ph].

[30] A. Butter, G. Kasieczka, T. Plehn, and M. Russell, *Deep-learned Top Tagging with a Lorentz Layer*, SciPost Phys. **5** (2018) 3, 028, arXiv:1707.08966 [hep-ph].

[31] M. Erdmann, E. Geiser, Y. Rath, and M. Rieger, *Lorentz Boost Networks: Autonomous Physics-Inspired Feature Engineering*, JINST **14** (2019) 06, P06006, arXiv:1812.09722 [hep-ex].

[32] A. Bogatskiy, B. Anderson, J. T. Offermann, M. Roussi, D. W. Miller, and R. Kondor, *Lorentz Group Equivariant Neural Network for Particle Physics*, arXiv:2006.04780 [hep-ph].

[33] X. Ju and B. Nachman, *Supervised Jet Clustering with Graph Neural Networks for Lorentz Boosted Bosons*, arXiv:2008.06064 [hep-ph].

[34] C. Shimmin, *Particle Convolution for High Energy Physics*, 7, 2021. arXiv:2107.02908 [hep-ph].

[35] P. T. Komiske, E. M. Metodiev, and J. Thaler, *Energy Flow Networks: Deep Sets for Particle Jets*, JHEP **01** (2019) 121, arXiv:1810.05165 [hep-ph].

[36] 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].

[37] T. Chen, S. Kornblith, M. Norouzi, and G. E. Hinton, *A simple framework for contrastive learning of visual representations*, 2020.

[38] V. Mikuni and F. Canelli, *ABCNet: An attention-based method for particle tagging*, Eur. Phys. J. Plus **135** (2020) 6, 463, arXiv:2001.05311 [physics.data-an].

[39] V. Mikuni and F. Canelli, *Point cloud transformers applied to collider physics*, Mach. Learn. Sci. Tech. **2** (2021) 3, 035027, arXiv:2102.05073 [physics.data-an].

[40] A. Shmakov, M. J. Fenton, T.-W. Ho, S.-C. Hsu, D. Whiteson, and P. Baldi, *SPANet: Generalized Permutationless Set Assignment for Particle Physics using Symmetry Preserving Attention*, arXiv:2106.03898 [hep-ex].

[41] A. Butter *et al.*, *The Machine Learning Landscape of Top Taggers*, SciPost Phys. **7** (2019) 014, arXiv:1902.09914 [hep-ph].

[42] L. Benato, E. Buhmann, M. Erdmann, P. Fackeldey, J. Glombitza, N. Hartmann, G. Kasieczka, W. Korcari, T. Kuhr, J. Steinheimer, H. Stöcker, T. Plehn, and K. Zhou, *Shared data and algorithms for deep learning in fundamental physics*, arXiv:2107.00656 [cs.LG].

[43] 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, arXiv:1410.3012 [hep-ph].

[44] DELPHES 3, 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].

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

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

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

[48] D. Bahdanau, K. Cho, and Y. Bengio, *Neural machine translation by jointly learning to align and translate*, arXiv:1409.0473 [cs.CL].

[49] M.-T. Luong, H. Pham, and C. D. Manning, *Effective approaches to attention-based neural machine translation*, arXiv:1508.04025 [cs.CL].

[50] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, *Attention is all you need*, arXiv:1706.03762 [cs.CL].

[51] J. L. Ba, J. R. Kiros, and G. E. Hinton, *Layer normalization*, arXiv:1607.06450 [stat.ML].

[52] M. Zaheer, S. Kottur, S. Ravanbakhsh, B. Poczos, R. Salakhutdinov, and A. Smola, *Deep sets*, 2018.

[53] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, *Pytorch: An imperative style, high-performance deep learning library*, 2019.

[54] C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, R. Kern, M. Picus, S. Hoyer, M. H. van Kerkwijk, M. Brett, A. Haldane, J. Fernández del Río, M. Wiebe, P. Peterson, P. Gérard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke, and T. E. Oliphant, *Array programming with NumPy*, Nature **585** (2020) 357–362.

[55] F. Wang and H. Liu, *Understanding the behaviour of contrastive loss*, arXiv:2012.09740 [cs.LG].