

Department of Physics and Astronomy
Heidelberg University

Bachelor Thesis in Physics
submitted by

Hans Olischläger

born in Hanau (Germany)

2021

Contrastive Learning of Invariant Representations

This Bachelor Thesis has been carried out by

Hans Olschläger

at the

**Institute for Theoretical Physics
Heidelberg University**

under the supervision of

Prof. Dr. Tilman Plehn

Abstract

Physics searches at particle colliders rely on the analysis of high-dimensional measurement data. Information describing complex physical processes, like the formation and decay of particles in collision experiments, have to be represented in a way that exploits symmetries while retaining discriminative features. Inspired by recent advances in contrastive representation learning, we attempt to teach the task of finding such representations to a neural network in a self-supervised fashion. We show how physical symmetries can be enforced with a transformer architecture, contrastive loss and symmetry transformations. We train our method JetCLR to learn representations for a dataset of QCD and top jets and compare the representations with established jet substructure representations using a linear classifier test.

Zusammenfassung

Die Suche nach neuen physikalischen Phänomenen an Teilchenbeschleunigern beruht auf der Analyse hochdimensionaler Messdaten. Daten von komplexen physikalischen Prozessen, wie die Entstehung und der Zerfall von Teilchen in Kollisionsexperimenten, müssen so dargestellt werden, dass Symmetrien ausgenutzt werden und gleichzeitig Unterscheidungsmerkmale erhalten bleiben. Inspiriert von jüngsten Fortschritten im Lernen von Repräsentationen mittels Kontrasten, versuchen wir einem neuronalen Netzwerk die Aufgabe, solche Repräsentationen zu finden, mittels selbstüberwachtem Training beizubringen. Dafür zeigen wir, wie physikalische Symmetrien mit einer Transformatorarchitektur, einer kontrastmaximierenden Kostenfunktion und mittels der zugehörigen Transformationen durch ein neuronales Netzwerk realisiert werden können. Wir stellen den Ansatz JetCLR vor, mit dem wir eine Repräsentationsfunktion für QCD- und Top-Jets erlernen. Schließlich werden die erhaltenden Repräsentationen mit etablierten Jet-Substruktur-Repräsentationen anhand ihrer Eignung zur linearen Klassifikation von Top-Jets verglichen.

Contents

1	Introduction	1
2	Physics background	2
2.1	Setting	2
2.2	High energy regime	2
2.3	Destroying protons to create new particles	3
2.4	Detector coordinates	4
2.5	Parton to jet	4
2.6	IRC safety	5
2.7	Top quark production and decay	6
3	Machine learning background	7
3.1	Neural networks	7
3.2	Dimensionality reduction	8
4	Finding well suited observables	9
4.1	Jet substructure representations	9
5	Contrastive learning of jet representations	12
5.1	Contrastive loss	12
5.2	Symmetries and augmentations	14
5.3	Attention	15
5.4	Transformer	16
6	Experiments and results	18
6.1	Data and tools	18
6.2	FCN as testing ground	18
6.3	Building up from parts	20
7	Opening the box	24
7.1	Head	24
7.2	Sum	27
7.3	Transformer	28
8	Conclusions	30
	References	31
	Acknowledgements	34

1 Introduction

Contemporary physics research produces large amounts of measurement data. Particle physics in particular has experiments at the Large Hadron Collider (LHC) that record millions of events per day to allow the precise study of rare phenomena. General purpose detectors like ATLAS and CMS measure the energies and tracks of particles produced during collision events to capture the complex processes in detail.

The exponential increase of computational power during the last decades has been accompanied by new techniques to analyse and learn from large data sets [1]. Specialised computing hardware (such as GPUs) and powerful machine learning (ML) libraries are available to the public. Today, modern machine learning techniques are a rapidly evolving field of research and are applied to a broad range of commercial and scientific questions.

Bridging the gap between low-level single event measurements on the one hand and theoretical predictions on the other hand, physicists have always used statistical methods. Thus, in particle physics where large datasets are available, machine learning approaches naturally fit in. ML applications in particle physics include signal/background discrimination using supervised learning, generative modelling of probability densities, and unsupervised data mining to find anomalies and ultimately new physics [2].

When ML systems process physical data, the question naturally arises, how physical processes should be represented by data given the physics knowledge we have and given the analysis tools we want to use. Raw measurement data is usually preprocessed to allow the analysis methods to work, and there are universal characteristics of good input data that most analysis methods benefit from. Thus, in this work we ask:

How can we organise high-dimensional information about physical processes?

The challenge is to define a representation in which the underlying symmetries are exploited, that retains discriminative power, and that is new-physics agnostic.

We introduce JetCLR, in which the mapping of jet substructure measurements to representations is formulated as a self-supervised machine learning task. JetCLR is derived from SimCLR [3] and uses Contrastive Learning for Representations. The mapping function from measurements to representations is realised by a neural network. The method allows to define symmetry transformations that the mapping should be invariant to. The converged network will then be a function, by which subspaces of measurement space that are related by those transformations are collapsed (approximately) to single representation vectors.

First, we will go over the physics background and the basics of machine learning. Then we will expand on which kinds of observables make up useful representations and present existing representations for jet substructure. After that, we will explain the details of contrastive learning for representations and the transformer architecture. We will then discuss results of the application of JetCLR depending on different choices of hyperparameters, symmetry transformations and architectures. Finally, we also summarise the insights gained through ongoing efforts to analyse the inner workings of the network.

This thesis is based on the research done in close collaboration with Barry M. Dillon, Tilman Plehn, Peter Sorrenson and Lorenz Vogel that also resulted in the publication of Ref. [4]. The important results are in the paper; this thesis tries to give a more detailed account of the development and its context.

2 Physics background

2.1 Setting

Physics asks the question of what fundamentally constitutes the universe. The Standard Model (SM) describes three out of four fundamental forces in the universe and classifies all fundamental particles that we know of. It is believed to be mathematically self-consistent and has successfully predicted experimental discoveries like the top quark [5–7] and the Higgs boson [8–10]. Mathematically, it is formulated as a quantum field theory that incorporates special relativity, contains the unification of the weak interaction with quantum electrodynamics (QED), quantum chromodynamics (QCD) and the Higgs mechanism.

Despite these successes there are some phenomena the SM cannot explain and thus hint at physics beyond the Standard Model (BSM). For example the most successful theory of gravity, general relativity (GR), is not compatible with the SM. If dark matter – necessary to explain observed galaxy rotation curves within the Λ CDM-model – consists of particles, there might be a single hidden particle or even a whole hidden sector of particles [11] that remain unnoticed until now. Additionally, many BSM models rely on supersymmetry (SUSY), which also predicts the existence of additional particles.

Therefore, the Standard Model is treated as an effective field theory (EFT) of the underlying more fundamental theory. Testing the limitations of the Standard Model and searching for anomalies at large energy scales is the motivation for collider physics [12].

2.2 High energy regime

At collider accelerators, like the Large Hadron Collider (LHC) operated by CERN, two proton beams are accelerated as close to the speed of light as currently possible. Pushing the energy frontier is interesting for two main reasons. First, to analyse the inner structure of protons, unprecedented resolutions are required. Since de Broglie found the wavelike properties of matter, it is known that large resolutions are only possible with large frequencies and therefore large energies. Second, the most interesting phenomena, like the spontaneous production of very massive particles, only happen at energies several magnitudes above the normal energy levels of our macroscopic world.

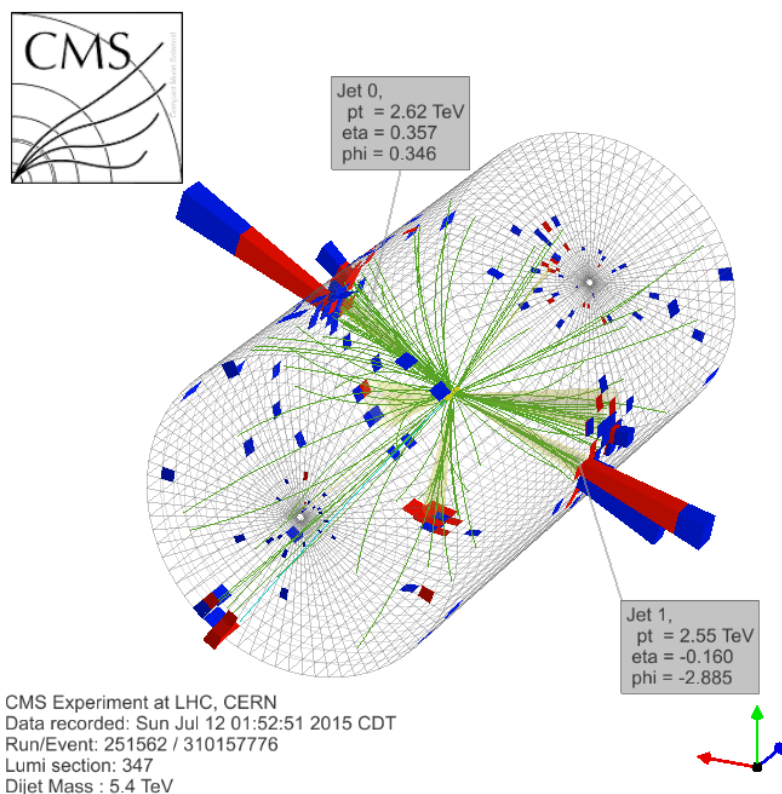


Figure 2.1: Collision event captured in CMS detector, cylinder is parametrised by η and ϕ , extrusion is p_T . Taken from [15].

2.3 Destroying protons to create new particles

Because protons are not among the fundamental building blocks of matter, but rather consist of a sea of quarks and gluons held together by the strong nuclear force, they perform complex scattering where the final state often contains hundreds of particles. At high energies, the constituents of protons become asymptotically free [13], in which case they are well described by the parton model [14]. It describes the protons that collide at the LHC as a collection of partons each carrying a fraction of the total proton's energy. During a deeply inelastic scattering, a substantial momentum is transferred between a highly energetic parton of each proton. The interaction of those two partons gives birth to a few elementary particles, like for example two new partons, a top-anti-top pair or new particles. Those in turn decay quickly and produce numerous hadrons that interact with the detector where they are recorded as measurements. It should be noted that this is a simplification. A parton is not a well-defined object, rather it is a part of a sea of partons, an excitation of the associated quantum field. This means, that there can be arbitrarily more complex parton interactions of higher order.

2.4 Detector coordinates

In order to not make our lives harder than necessary, we choose kinematic variables that fit the problem of particle collisions in a detector. As mentioned earlier, hard hadronic collisions are actually the interaction of two partons, one from each beam, each of them carrying an arbitrary fraction of the momentum of its proton. As the two partons usually do not carry the same fraction of momentum, the centre of mass of the interaction is boosted along the beam direction relative to the lab frame. Therefore, we want to use a coordinate system that is well-behaved with respect to longitudinal boosts of the system and rotations about the beam axis.

A constituent's four-vector (E, p_x, p_y, p_z) can be described by its mass m , the radius p_T and azimuthal angle ϕ in the transverse plane, and rapidity y .

$$p_T = \sqrt{p_x^2 + p_y^2} \quad y = \frac{1}{2} \log \left(\frac{E + p_z}{E - p_z} \right) \quad (2.1)$$

Rapidity differences are invariant to longitudinal boosts. Therefore, we can translate jets in the y - ϕ -plane without changing the underlying physics.

When final state constituents are fast and thus can be approximated as massless, the pseudo-rapidity η is used. It can be expressed as a function of the three-momentum \vec{p} and the polar angle θ .

$$\eta = \frac{1}{2} \log \left(\frac{|\vec{p}| + p_z}{|\vec{p}| - p_z} \right) = -\log \left(\tan \frac{\theta}{2} \right) \quad (2.2)$$

In the limit of massless constituents rapidity is the same as pseudo-rapidity. We thus have an energy like variable p_T and two variables with cylindrical geometry η and ϕ ; see Figure 2.1. The cylinder is also referred to as the η - ϕ -plane. On this plane we can define the angular separation

$$\Delta R = \sqrt{\Delta\eta^2 + \Delta\phi^2}. \quad (2.3)$$

2.5 Parton to jet

Many processes produce high energy quarks and gluons (partons). A parton can radiate from the incoming partons, some particles like the W , Z and Higgs boson can decay into quarks and the decay chain of new particles also might contain quarks and gluons [16].

However, the high energy quarks and gluons are not directly observed by the detector. Rather, they undergo successive branching, producing a collimated shower of quarks and gluons. Below a certain energy threshold, hadronisation takes place. At these energies the QCD coupling constant α_s is high enough to confine every object that carries colour charge, so that quarks and gluons can only exist in a colourless (white) bound state. So when two partons get ripped apart in a collision, the interaction energy rises until it is big enough to produce a new quark-anti-quark pair. The ensemble of all those particles make up a *jet*. Thus, any high energy quark or gluon will be measured as a jet.

When faced with a set of measured final state particles, these have to be clustered together to reconstruct underlying jets. To recover the exact process and capture every particle that originated from the same parton is clearly desirable, but also impossible. To see how to solve the problem approximately, and to point out an important phenomenon of QCD, we will look at the cross section of gluon radiation.

$$d\sigma \propto \alpha_s \frac{dE d\theta}{E \theta} \quad (2.4)$$

The cross section diverges in the limit of gluons with $E \rightarrow 0$ which is called the *soft/infrared divergence*, and the limit of $\theta \rightarrow 0$ which is called the *collinear divergence*. The collinear divergence is in fact the reason why jets are collimated. This tendency to branch at low angles means that we can define an angular distance R for the upper limit of separation of particles clustered in the same jet. The soft/infrared divergence complicates the picture, as it means that unrelated soft particles are ubiquitous.

There are multiple jet algorithms addressing the clustering difficulties in different ways, with the *anti- k_T* algorithm [17] being the current default for the LHC. Here, a constituent metric for the distance between constituent i and j is defined by

$$d_{ij} = \min \left(\frac{1}{p_{T,i}}, \frac{1}{p_{T,j}} \right) \frac{\Delta R_{ij}^2}{R^2}, \quad d_{iB} = \frac{1}{p_{T,i}^2}. \quad (2.5)$$

The algorithm sequentially combines the constituents with the smallest distance d_{ij} and declares a constituent combination i a jet if d_{iB} turns out to be the smallest distance. This means that it starts combining close and high p_T constituents resulting in robust jets, and stops at a maximum jet radius R .

2.6 IRC safety

Generally, a cross-section $\sigma(v)$ with regard to a specific observable v is calculated perturbatively as an integral over the full phase space. Thus, all infrared and collinear (IRC) singularities have to cancel out at each perturbation order in order to get a finite result. Whenever a parton is split into two collinear partons, or a vanishingly soft parton is added, the observable must not change in order to achieve complete cancellation of the IRC singularities. This condition is called IRC safety. Suppose the generic observable v has measurement functions $V_n(k_1, \dots, k_n)$ of that observable in the n -body phase space. In order to achieve complete cancellation of the IRC singularities, observables must satisfy the following conditions called IRC safety [16]:

$$\text{IR safety: } V_{m+1}(\dots, k_{i-1}, k_i, k_{i+1}, \dots) \rightarrow V_m(\dots, k_{i-1}, k_{i+1}, \dots) \quad \text{for } k_i \rightarrow 0 \quad (2.6)$$

$$\text{C safety: } V_{m+1}(\dots, k_i, k_{i+1}, \dots) \rightarrow V_m(\dots, k_i + k_{i+1}, \dots) \quad \text{for } k_i \parallel k_{i+1} \quad (2.7)$$

While IRC safety is originally motivated by perturbative calculation, it is used to assess observables and algorithms also in an experimental context, because it can provide robustness in the context of the calorimeter measurements in particle detectors, where the pixels detectors have finite energy and space resolution. For example, the anti- k_T algorithm that defines jets satisfies IRC safety, because collinear constituents will be combined at the beginning of the sequence and soft constituents have a marginal impact on the jets that already contain hard constituents.

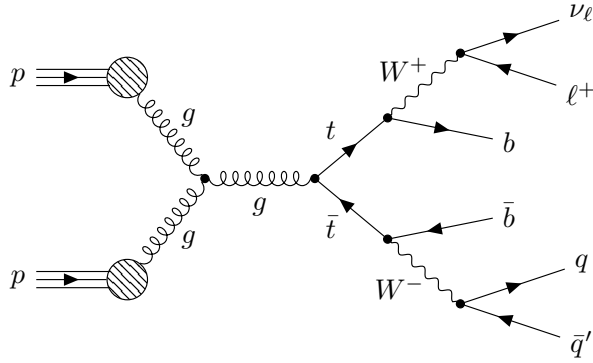


Figure 2.2: Example process for the production of a $t\bar{t}$ -pair. t decays leptonically, \bar{t} decays hadronically. Taken from [18].

2.7 Top quark production and decay

The top quark is the heaviest of the six quarks in the SM. Being the only fermion heavier than the W boson, it can decay semi-weakly into a W boson and a bottom quark. Therefore, it has a very short lifespan and decays before it can hadronise [19]. Additionally, the top quark induces a large quantum correction to the Higgs mass [12]. Because of this special role in SM and possibly BSM physics, the high-energy physics community has studied the top quark extensively [20, 21]. It is produced predominantly in top-antitop pairs through the processes $q\bar{q} \rightarrow t\bar{t}$ and $gg \rightarrow t\bar{t}$. Those decay predominantly into a bottom quark b and a W boson, which then in turn can decay either into a lepton and antilepton, or a quark and an antiquark.

In the case of a leptonic decay, the reconstruction of top quarks from detector measurements can be done with high sensitivity. However, the specificity regarding for example the distinction of top quarks from new particles decaying to top and missing energy is low, because energy carrying neutrinos of the leptonic decay cannot be measured, and thus are missing in the equation [12].

Top pairs decaying purely hadronically produce 6 quark jets. With a mass of (172.76 ± 0.30) GeV [22] top quarks are still well below the TeV scale of the LHC and thus will be produced boosted back-to-back with high transverse momentum p_T . Geometrically, this means that the decay products of each top quark are collimated into a *fat jet* consisting of three prongs.

For these reasons, the development of taggers to distinguish hadronic *top jets* from quark and gluon jets (*QCD jets*) has been an active field of research [23–29].

3 Machine learning background

3.1 Neural networks

Machine learning is the process in which a machine learns to do a task from data rather than being explicitly programmed for the task. To do this, a family of candidate functions \mathcal{F} with parameters β is chosen and the parameters are inferred using training data such that $f_\beta \in \mathcal{F}$ solves the desired task

$$y = f_\beta(X). \quad (3.1)$$

One approach to implement such systems is the paradigm of *artificial neural networks*. These are networks of artificial neurones whose connections are modelled by trainable weights. The most commonly used networks are feed forward neural networks, meaning neurons are organised in successive layers. First, data is passed into the first layer. Then, each following neuron in the hidden (middle) layers gets inputs from proceeding neurons, fires according to its activation function and then passes its activation along the connections to the next layer of neurons. Thus, information flows through the neurons according to their activation functions and weighted connections to finally result in the network output. Effectively, the network is a function defined by the neurons' activation functions, the connections, and their weights.

Universal approximation theorems have been proven for various function families for different conditions, including the most basic feed forward network with one hidden layer [30]. These theorems state that neural networks can approximate all functions in a function space (e.g. smooth functions) given that the networks are either wide, or deep enough. The ability to approximate all smooth functions in the limit of large network sizes implies the theoretical possibility to solve a wide range of tasks. However, the functions must be trained on data and that data usually has a finite size. The finite dataset size leads to a sampling error, because the data is just a sample from a true distribution. While infinite-size networks can approximate a desired function to arbitrary precision, the bigger the network is, the more training data is required to generalise beyond the training set.

Rather than memorising each data point and the expected outcome within the model, we want it to realise the general task. To keep track of the generalisation error, we thus have to use statistically independent data for training and for testing. In practice, this means the dataset is split up into a train and a test set.

Training

To learn from data and achieve the purpose of the network, the parameters, e.g. weights of the connections, are optimised to reduce a specifically designed loss function. The loss function evaluates if the network achieved the desired result. It can be thought of as defining a loss landscape for every possible configuration of the model's parameters. In this framing, training a model corresponds to finding a spot in the parameter landscape with minimal loss. Optimisers designed to train neural networks, typically estimate the gradient of the loss with respect to the current set of parameters [31] and determine slightly altered parameters.

For example, a commonly used optimiser is stochastic gradient descent (SDG), where the new parameters are obtained by estimating the gradient for every model parameter with the training batch and taking a step along the gradient vector scaled by a *learning rate*. Thus, the learning rate serves as a step size in the optimisation. Training with a constant learning rate can fail to find the global minimum in the loss landscape, either because a suboptimal local minimum cannot be left with a single step (learning rate too low) or because the global minimum is overstepped repeatedly (learning rate too high). Another approach to the problem is to start with a high learning rate and reduce it automatically when the loss stops decreasing.

Usually, the whole training dataset is used repeatedly for several epochs. In each epoch, the dataset is divided into batches and each batch is used to estimate the best possible optimisation step.

Activation functions

The most commonly used activation function is the rectified linear unit (ReLU). In this work, all neurons use ReLU activation.

$$\text{ReLU}(x) = \begin{cases} 0, & \text{if } x < 0 \\ x, & \text{otherwise} \end{cases} \quad (3.2)$$

The exception is the linear layer, which uses the identity instead.

We use the softmax function to normalise vectors $x \in \mathbb{R}^n$ such that sum over their entries is 1. It can be thought of as a normalisation into a probability distribution where each probability is proportional to the exponential of the input values.

$$\text{softmax}_i(x) = \frac{e^{x_i}}{\sum_{j=1}^n e^{x_j}} \quad (3.3)$$

3.2 Dimensionality reduction

t-SNE [32], i.e. t-distributed stochastic neighbour embedding, is a statistical method for visualising high-dimensional data by assigning each data point a location in 2D space. The embedding is done in a way such that closeness of data points is preserved as good as possible. t-SNE can reveal structure at different scales because it is non-linear and can perform different transformations on different regions in the high dimensional space. However, this flexibility comes with the cost of being easy to misread [33]. In particular, apparent volumes of clusters are arbitrarily distorted and thus not meaningful. Also, in some cases, different parameters give qualitatively different results. Therefore, the hyperparameters must be varied to check the robustness of a t-SNE mapping and the resulting images should be interpreted by their structure, rather than their volumes.

4 Finding well suited observables

We describe physical phenomena by measurable quantities. Defining useful observables is an essential part of physics. Fruitful observables are well-defined and close enough to fundamental reality, thus enabling us to use them in a mathematical structure, that in turn allows us to make further predictions.

Recently however, observables have also been used in a machine learning setting. Here, the challenge is to represent physical information in a way intelligible to a neural network. Neural networks have no trouble using many observables at the same time. Therefore, rather than choosing only the most informative features, we can set it up to incorporate all potentially interesting information, e.g. using low level (high dimensional) inputs. However, it is often infeasible to just use the raw experimental data directly. Usually, task-specific preprocessing of the data is necessary.

With a nice input data structure the necessary network size can be greatly reduced and training can become more stable. Conversely, if the input data is poorly structured, accuracy will suffer even with the best architectures.

Generally, in well structured data each dimension/observable adds important information rather than noise and follows a distribution of similar magnitude, i.e. is normalized. The latter is easily accomplished by rescaling the dimensions. The former however, might be very difficult when it is not an option to craft every single observable by hand. Additionally, the distance between instances of the dataset should reflect their true similarity to each other.

One well established test employed by the representation learning community is to check the linear separability of certain classes of instances in the dataset. If a hyperplane in the representation space is able to separate instances from the class (signal) from instances outside the class (background), linear separability would be perfect, which corresponds to the maximal possible discriminative power. In this spirit, we use the performance measures of the binary classifier as proxies of the discriminative power of representations. For binary classification, each data point is assigned a score in the range of 0 to 1 and a threshold is chosen above which data points are classified as signal. In the receiver operating characteristic (ROC) curve, the true positive rate, i.e. the efficiency ϵ_s to correctly tag signal as signal, is plotted against the false positive rate, i.e. the efficiency ϵ_b to wrongly tag background as signal. This captures the discriminative power over a the full range of threshold values. The area under the ROC curve (AUC) as well as the inverse background efficiency, $\epsilon_b^{-1}(\epsilon_s = 0.5)$, at a fixed signal efficiency of $\epsilon_s = 0.5$ condense the curve to two numerical values that can be referenced in tables. For the linear classifier tests (LCTs) in this work, we use a linear neural network without hidden layers and no activation function.

4.1 Jet substructure representations

In the following we will survey the existing representations for jet substructure and evaluate them according to the desirable properties described above. The representations are defined both by the raw data, and the function that maps the raw data into a representation space.

$$f : \mathcal{J} \rightarrow \mathcal{R} , \tag{4.1}$$

The raw data will consist of sets of final state constituents that together constitute a jet. Each constituent is defined by their transverse momentum p_T , pseudorapidity η and azimuthal angle

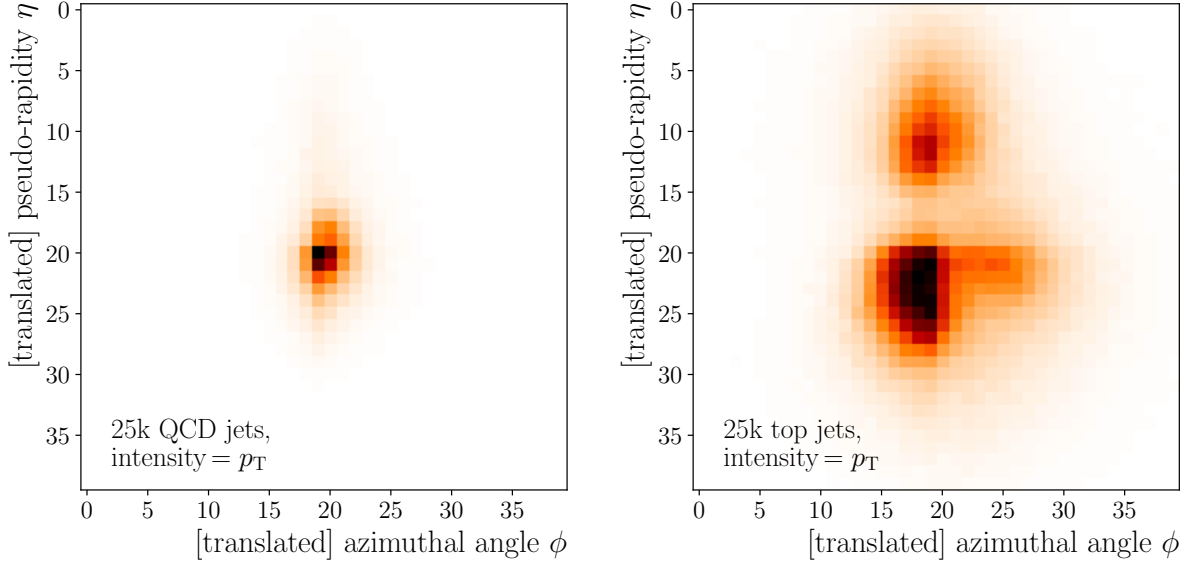


Figure 4.1: Jet image representation. Left: mean of QCD jets, Right: mean of top jets.

ϕ .

$$x_i = \{ (p_T, \eta, \phi) \mid (p_T, \eta, \phi) \text{ is constituent of jet } i \} \quad (4.2)$$

Rather than biasing the measurement towards a certain theoretic hypothesis, we want representations to only be informed by the most fundamental knowledge we have: symmetries. A good representation must incorporate invariances in order to have highly informative dimensions. For example, as we are mapping *sets* of constituents to the representation space, we should have permutation invariance. A jet representation should not change if the ordering of its constituents is shuffled. Failing to incorporate this symmetry means that the same underlying jet data will be mapped to different representations, thus introducing noise.

High-level observables like invariant mass and N-subjettiness [34] are specifically designed to be used to quantify jet phenomena, but using them for machine learning leaves the potential of high-dimensional analysis untapped.

Jet images

A well-established jet substructure representation is the jet image [35, 36]. Here, the constituents are binned into pixels of the η - ϕ -plane and the accumulated transverse momentum (or energy) for each pixel gives us a permutation invariant representation. For illustration, Figure 4.1 shows the mean of QCD and top jets in the fashion of jet images. While they are very intuitive to look at, they are far from perfect. If the jet image resolution is increased, the size of that representation scales quadratically and it gets sparse quickly, both of which are problems that the machine learning architecture will struggle with. Moreover, the representation is not invariant to all required symmetries and has to rely on preprocessing instead, e.g. rotations about the jet axis.

Energy flow polynomials

Another promising representation is the expansion of energy flow polynomials (EFPs) [37] to a fixed order. EFPs are a set of substructure observables constructed around the principal of IRC safety. They can be viewed as a set of C-correlators [38]

$$C_N^{f_N} = \sum_{i_1=1}^M \cdots \sum_{i_N=1}^M E_{i_1} \cdots E_{i_N} f_N(p_{i_1}^\mu, \dots, p_{i_N}^\mu) \quad (4.3)$$

with a specific choice for f_N .

EFPs provably span the whole space of IRC safe observables in a linear way, thus form a (over-)complete basis [37]. This property makes them particularly well suited for use as input data representation in machine learning. However, this complete basis is an expansion with infinitely many polynomials. Because any application can only use a finite number of inputs, the expansion must stop at a fixed order.

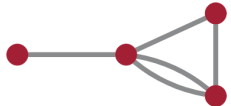
To visualise and calculate the correlators, there is a one-to-one correspondence of EFPs and loopless multigraphs. Those are graphs where two vertices can be connected by any number of edges, but no vertex can have a single-edge circle that connects to itself. A multigraph G with N vertices and edges $(k, l) \in G$ corresponds to

$$\text{EFP}_G = \sum_{i_1=1}^M \cdots \sum_{i_N=1}^M z_{i_1} \cdots z_{i_N} \prod_{(k,l) \in G} \Delta R_{i_k i_l}, \quad (4.4)$$

with

$$z_i = \frac{p_{T,i}}{p_{T,J}}, \quad p_{T,J} = \sum_{i=1}^M p_{T,i}. \quad (4.5)$$

Here, z_i is the transverse momentum fraction carried by particle i , $\Delta R_{i_k i_l}$ is the angular distance between the constituents i_k and i_l and there is a total of M constituents in the jet. Each edge (k, l) corresponds to a term $\Delta R_{i_k i_l}$ and each vertex corresponds to a factor of z_{i_j} , and a summation over i_j . Equation (4.6) shows an example of this.



$$= \sum_{i_1=1}^M \sum_{i_2=1}^M \sum_{i_3=1}^M \sum_{i_4=1}^M z_{i_1} z_{i_2} z_{i_3} z_{i_4} \Delta R_{i_1 i_2} \Delta R_{i_2 i_3} \Delta R_{i_3 i_4} \Delta R_{i_2 i_4}^2 \quad (4.6)$$

While this is impressive, it is also a rigid system, where it is difficult to incorporate additional particle level information like tracking and it is unclear whether the expansion to fixed order contains the most useful observables.

After having established the problem and existing approaches, in the next chapter we discuss how we can formulate the mapping instead as a self-supervised machine learning task.

5 Contrastive learning of jet representations

The function f mapping jet phase space \mathcal{J} into the representation space \mathcal{R} will be a neural network.

5.1 Contrastive loss

As usual for self-supervised learning, we set up a non-trivial task from the unlabelled data alone. To successfully accomplish this task the network will have to extract relevant features from the data. The task will be to map similar jets close to each other while pushing dissimilar jets apart. For these pairs of similar jets, positive pairs (x_i, x'_i) , are generated from the original jet data by applying random transformations to each jet. Negative pairs are defined as not matching, i.e. pairs of two distinct jets $\{(x_i, x_j)\} \cup \{(x_i, x'_j)\}$ for $i \neq j$. We will specify physics motivated augmentations in Section 5.2.

Each jet x_i and x'_i is passed through the network to obtain the network output z_i and z'_i .

$$\begin{aligned} z_i &= f(x_i) \\ z'_i &= f(x'_i) \end{aligned}$$

To measure similarity in the output space we define the cosine similarity s

$$s(z_i, z_j) := \frac{z_i \cdot z_j}{|z_i||z_j|} \equiv \cos \theta_{ij}, \quad (5.1)$$

where θ_{ij} is the angle between z_i and z_j .

By defining similarity in this way we have deliberately chosen that the extracted information will be organised on a hypersphere, because $s(z_i, z_j)$ is only a function of the angular distance of two vectors. The magnitude of the vectors is irrelevant for their similarity.

For every pair in the batch the contrastive loss is computed as

$$\mathcal{L}_i = -\log \frac{e^{s(z_i, z'_i)/\tau}}{\sum_{j \neq i \in \text{batch}} [e^{s(z_i, z_j)/\tau} + e^{s(z_i, z'_j)/\tau}]}, \quad (5.2)$$

with a temperature parameter τ .

The total loss is defined as the mean over all pairs, $\mathcal{L} = \langle \mathcal{L}_i \rangle$, and is backpropagated through the network to gradually adjust the network parameters. The loss is minimal if the similarity of positive pairs is high and the similarity of negative pairs is low, as they scale the numerator and denominator respectively and the fraction is the argument of the monotonically decreasing negative logarithm.

Desirable properties

There are reasons to use a sphere as a representation space beyond the fact that it naturally arises from this formulation of contrastive loss (5.2). First, for any machine learning task a fixed norm is helpful for training stability. Normalising onto a unit sphere by $\hat{z}_i = z_i/|z_i|$ thus will make downstream tasks that use the representation more stable.

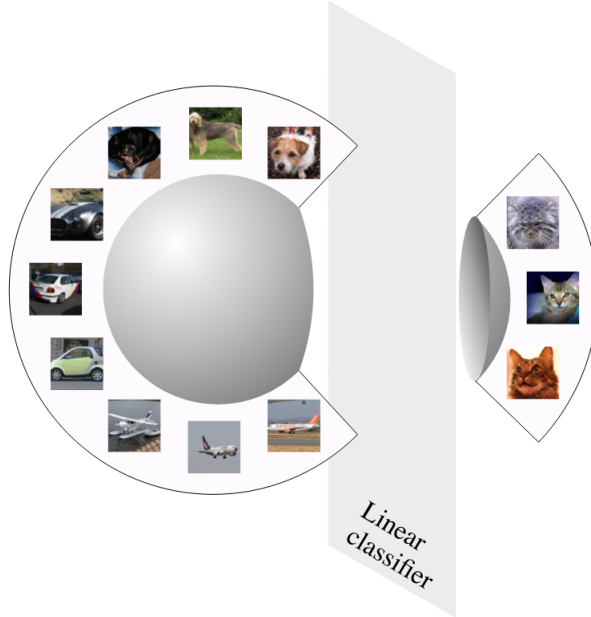


Figure 5.1: Illustration of linear separability of image representations on the unit hypersphere. Taken with permission from [39].

Second, if relevant classes of instances are sufficiently well clustered on a sphere, they are automatically linearly separable, see Figure 5.1. The same does not hold for Euclidean spaces. Obviously, the *discriminative power* is further increased the more of the sphere’s surface is used. Wang et al. [39] call this property *uniformity* and quantify it with the *uniformity loss*

$$\mathcal{L}_{\text{uniform}} = \frac{1}{N_{\text{batch}}} \sum_{i \in \text{batch}} \log \sum_{j \neq i} \left[e^{-s(z_i, z_j)} + e^{-s(z_i, z'_j)} \right]. \quad (5.3)$$

Indeed, we will later see that discriminative features are facilitated by uniformity when training a network where we do not apply any augmentations so that positive pairs just have two indistinguishable jets and the numerator does not play a role. Recently it was shown, that the contrastive loss is particularly good at optimizing uniformity because it is hardness-aware, i.e. the relative penalty for similarity of negative pairs is higher the closer they are [40]. Thus, the loss incentivizes to push the most difficult negative pairs apart and thereby to fix the local structure on the hypersphere.

But we do not stop at uniformity. By using augmentations inspired by symmetry transformations in Section 5.2, we can actually encode physics knowledge to reduce irrelevant noise. The desirable property that jets with the same underlying data are similar, thus point in the same direction, is called *alignment* in the framework of Wang et al. [39] and can be quantified with the *alignment loss*

$$\mathcal{L}_{\text{align}} = \frac{1}{N_{\text{batch}}} \sum_{i \in \text{batch}} s(z_i, z'_i). \quad (5.4)$$

Contrastive learning provably optimises for uniformity and alignment in the limit of large batch sizes [39]. The trade-off between the two can be controlled with the temperature τ . For very low temperatures, uniformity dominates over alignment. When the temperature increases, the latent distribution becomes less uniform and alignment increases.

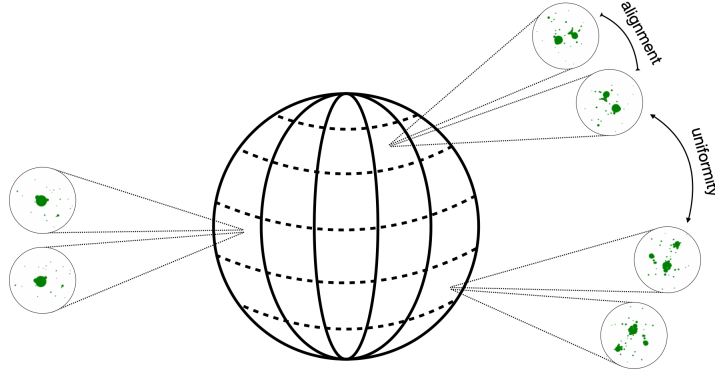


Figure 5.2: Illustration of contrastive learning goals: alignment of positive pairs and uniformity of negative pairs. Taken from [4].

The contrastive loss, being unsupervised, treats every negative sample equally. However, with strong augmentations and the ability of neural networks to find abstract features, we can hope for a latent structure that somewhat resembles the true physical relationships of the data.

5.2 Symmetries and augmentations

By telling the network which kinds of jets are positive pairs and therefore should be similar in \mathcal{R} , we can flexibly learn (approximate) invariances to symmetry transformations (and data augmentations).

The symmetry transformations we consider are rotations around the jet axis, reflections and translations in the η - ϕ -plane. In the jet image representation, rotation, translation and flip symmetries must be done in the preprocessing.

Additionally, we also apply data augmentations which are not strictly symmetry inspired, but express prior knowledge about jet structure. Due to the infrared divergence soft activity is noisy. The soft radiation noise is independent of the hard process at the core of the event. This is encoded with an augmentation that smears soft constituents. We chose to sample distortions out of a Gaussian distribution $\mathcal{N}(\mu, \sigma)$ with a standard deviation σ of $\frac{\Lambda_{\text{soft}}}{p_T}$, $\Lambda_{\text{soft}} = 100 \text{ MeV}$.

$$\begin{pmatrix} p_T \\ \eta \\ \phi \end{pmatrix} \rightarrow \begin{pmatrix} p_T \\ \mathcal{N}\left(\eta, \frac{\Lambda_{\text{soft}}}{p_T}\right) \\ \mathcal{N}\left(\phi, \frac{\Lambda_{\text{soft}}}{p_T}\right) \end{pmatrix} \quad (5.5)$$

A collinear splitting augmentation finds motivation in different viewpoints. Collinear divergence is the theoretic motivation. The fact that particle detectors bin measurements into pixels, effectively treating multiple collinear jet constituents as one, is the experimental motivation. Teaching the network different constituent dimension categories, i.e. η and ϕ are to be treated as space coordinates, p_T as an additive quantity, is the technical machine learning motivation. The augmentation chooses jet constituents and splits them into two softer constituents such that the combined p_T measured by the (simulated) detector does not change because both constituents have matching η and ϕ .

As the number of constituents in a jet varies, but jets have to be presented by a fixed length vector in the dataset, any remaining empty constituents are zero-padded, i.e. filled with zeros. There are two reasons why this is important. First, if nothing is done about it, the resulting jet representations encode the number of jet constituents via the number of zero-pads. Aiming at highly informative representations, this does not have to concern us, however we know that this information is not IRC safe and very noisy.

Second, relying on zero-pads is suboptimal, because it means having two distinct classes of constituents, nonzero and zero-pads, thus making the constituent space discontinuous, which complicates the task the network has to fulfil. One way to address this is to apply an augmentation that fills all zero-pads with soft noise, that is centred at the jet axis with a standard deviation equal to the jet radius R . In the implementation used here soft constituents were sampled as

$$p_T = |\mathcal{N}(0, 0.1)|, \quad \eta = \mathcal{N}(0, R), \quad \phi = \mathcal{N}(0, R). \quad (5.6)$$

We can incorporate permutation invariance of jet constituents explicitly by architecture [41]. At the core, we use that the sum over constituents is permutation invariant. Before summing up, we process constituents in parallel, where information flow between constituents is mediated by an attention mechanism in an equivariant way. The inner workings of the attention mechanism are detailed in the next section.

5.3 Attention

The origin of attention in machine learning is in natural language processing. Here, sequence information like sentences need to be processed such that the context of each word is paid attention to. It works from the principal that embeddings of set elements can be contextualized by performing a weighted sum that also includes every other element in the set. Thereby, each embedding is altered according to its context; the higher the weight of an element, the higher the attention to it. For JetCLR, we employ the scaled dot-product multi-headed self-attention of Ref. [42].

Dot-product reweighting

Suppose we have constituent embeddings x_i . If we calculate the weight w_{ij} of attention x_j pays to x_i with the dot product, $w_{ij} = x_i \cdot x_j$, aligned embeddings will be attended to, while orthogonal ones are irrelevant. Applying a softmax ensures that the weights will sum to 1. Thus, the reweighted embeddings would be calculated by

$$x'_i = \sum_j \text{softmax}_j(x_i \cdot x_j) x_j. \quad (5.7)$$

Single-head self-attention

To flexibly attend to more complex patterns, we can introduce learnable weight matrices W^Q , W^K and W^V , respectively for queries, keys and values. With d being the dimension of keys, the dot-product single-headed self-attention is

$$x'_i = \sum_j \text{softmax}_j \left(\frac{(W^Q x_i) \cdot (W^K x_j)}{\sqrt{d}} \right) W^V x_j. \quad (5.8)$$

Multi-head self-attention

In a single attention mechanism, embeddings tend to only attend to themselves. Thus, multiple matrices W_i^Q , W_i^K and W_i^V linearly project queries, keys and values to a smaller dimension to calculate the attention multiple times. The resulting reweighted value vectors are concatenated and passed to a final linear layer, see Figure 5.3. This allows the network to attend to different embedding subspaces of different constituents simultaneously.

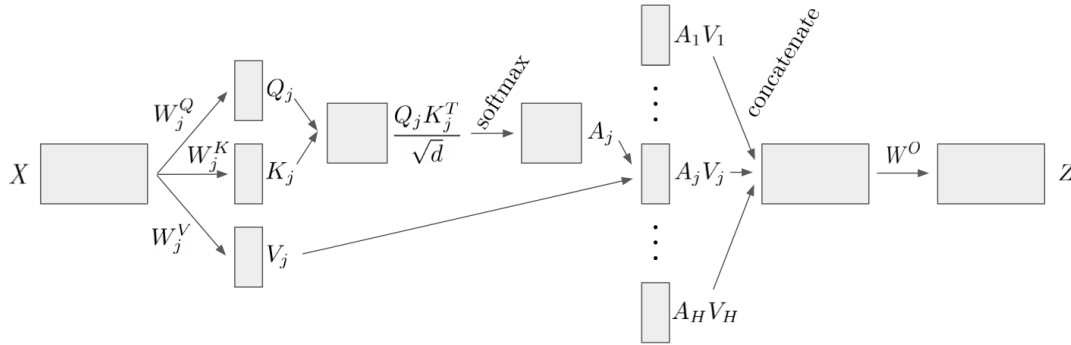


Figure 5.3: Full information flow of multi-head self-attention. Reweighted embeddings of multiple attention heads in parallel are concatenated and passed to a final linear layer.

5.4 Transformer

A typical transformer, e.g. for language translation, consists of an encoder and a decoder. For our representation learning purposes, we only use an encoder. It is made up of N identical blocks, each featuring multi-head self-attention with a skip connection and layer normalisation, followed by a feed forward network also with a skip connection and layer normalisation.

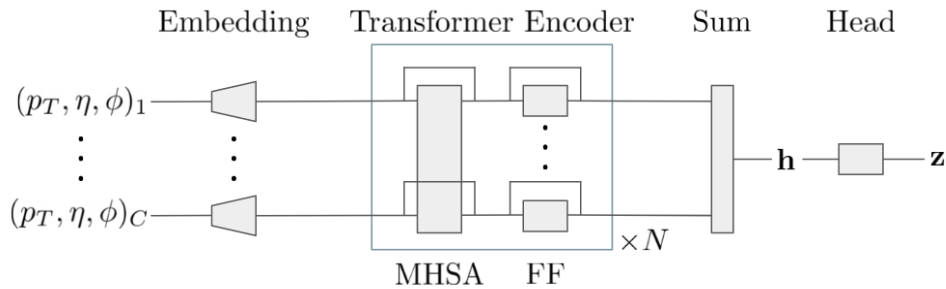


Figure 5.4: Network architecture consists of embedding layer, transformer encoder layers with multi-head self-attention and feed forward network, sum over constituents and head layers. Taken from [4].

Figure 5.4 summarises the full architecture. Each constituent, at first only represented by its kinematic variables (p_T, η, ϕ) is embedded in a d_{model} -dimensional space by a learnable linear layer with the identity as activation function. The higher dimension increases the expressiveness of constituent embeddings and thus jet representations. These embeddings are accumulating context information by successive transformer encoder layers.

After the transformer encoder, all constituent embeddings are summed to achieve a permutation invariant jet representation.

We found in accordance to Ref. [3] that it is beneficial for representation quality to separate the layer from which jet representations are taken from the last layer, where the contrastive loss is applied, by a non-linear function. This is realised by a small fully connected head network. In practice it is not necessary to settle on a specific head layer from which to choose representations at training time. Therefore, the head network is implemented in a way that allows to evaluate the representation quality before, within and after the head network flexibly.

6 Experiments and results

6.1 Data and tools

We work with the top tagging dataset [26], where collision events are simulated with *Pythia8* [43] (default tune) with a centre of mass energy of 14 TeV. Pile-up and multi parton interactions are neglected. These simulated events are then converted into detector measurements with *Delphes* [44], using the default ATLAS detector card. The simulated calorimeter measurements are used to reconstruct jets defined by the *anti- k_T* -algorithm [17] in *FastJet* [45] with a jet radius $R = 0.8$. Finally, we keep the leading jet if its transverse momentum is between 550 and 650 GeV. Labels indicating whether a jet is a top jet are included. For this, top jets are defined as jets that can be matched to a parton-level top quark and contain all parton-level decay products within the jet radius.

Neglecting the vanishing masses of jet constituents, we calculate the kinematic variables p_T , η and ϕ as detailed in Section 2.4. Then we centre the p_T weighted centroid of each jet to be in the origin of the η - ϕ -plane and order jet constituents by their p_T .

The networks were implemented in *Pytorch* [46]. For the evaluation and analysis of learned representation we used *scikit-learn* [47]. Finally, *NumPy* [48] allowed the vectorisation of on-the-fly augmentations.

6.2 FCN as testing ground

The approach contains many hyper-parameters that we need to choose, such that the learnable network parameters converge to a good model. As these optimal hyper-parameters are not known to us, we have to systematically search the parameter space.

We start by simply using a small fully connected network (FCN), no head and few augmentations. Compared to a large and complex transformer network, the FCN allows fast training and easier troubleshooting. Furthermore, we initially crop the number of constituents in a jet to $n_C = 20$, thus dropping much of the difficult soft noise. The fully connected network consists of an input layer of size $3n_C$, a single hidden layer of size 80 and an output layer of size 200 where both the representations are taken from and the loss is applied to. We train for 750 epochs with a batch size of 1000 on a dataset with signal to background ratio of 0.05. For the early tests we only use the rotation symmetry transformation and the low p_T distortion augmentation. Using the fully connected network instead of the transformer means sacrificing permutation invariance. In this setting, ordering the jets constituents by their transverse momentum as a preprocessing step helps the fully connected network to still make out radiation patterns.

During training, we periodically evaluate not only the contrastive loss, but also uniformity, alignment and representation quality. We find that all of these metrics are indeed correlated. Training is done with the Adam optimizer and can take up to 500 epochs until convergence.

The temperature parameter τ of the contrastive loss function was shown to be crucial for contrastive learning of representation to work [39, 40]. Figure 6.1 shows the results of a systematic temperature scan on the FCN. Despite fluctuation, we see that temperatures of 0.1 to 0.2 perform best.

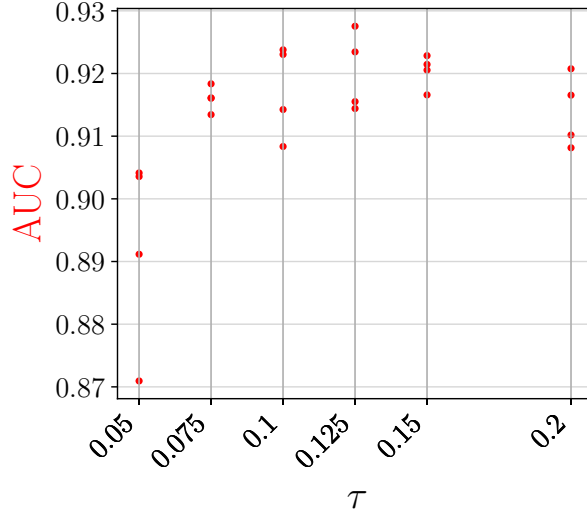


Figure 6.1: Probing different temperatures τ in a systematic hyper-parameter scan. Representation quality is measured by proxy of Area Under the receiver operator Curve (AUC).

When the number of constituents is increased such that jets include more soft radiation, constituents included in the representations become less relevant to the hard physics. We show in Figure 6.2 that the low p_T distortion augmentation counters the deterioration of representation quality. With this augmentation more constituents add valuable information.

Chen et al. have stressed that large batch sizes are beneficial for contrastive learning [3]. Indeed, tests with lower batch size proved to perform worse. Increasing the batch size further is not possible due to memory constraints on the GPU, because the gradients for all parameters must be stored for each jet instance in the batch to compute the next optimisation step.

Misalignment

A final lesson from the FCN tests, that later proved to generalise to the transformer architecture, was that the separation of representation and loss layer really is useful. The introduction of the head makes it possible to take jet representations on a different layer than the last one. Empirically we observed, that the output layer, (even though) optimised for contrastive loss, consistently resulted in representations far worse than earlier head layers. At first this might be unexpected as the contrastive loss seems to be well aligned with the interests of representation learning. After all, direct optimisation of alignment introduces physics knowledge, uniformity forces to retain discriminative power and the hypersphere favours linear separability.

On the other hand, it is a known phenomenon that encodings in the middle of the network are more general than at the end, both in contrastive learning [3] and elsewhere [49]. Rather speculatively, this strategy to encode abstract representations a few layers before the last one might be favoured because individual layers tend to do as little work as possible. Put differently, learning well separated abstract features inside the model allows it to perform difficult mapping tasks efficiently. It is also well-known from Computer Vision that successive layers encode features which correspond to more and more abstract concepts (lines, curves, textures, objects, ..., geographic regions) [50]. Towards the end, the features become less general and more relevant to the specific loss function [51].

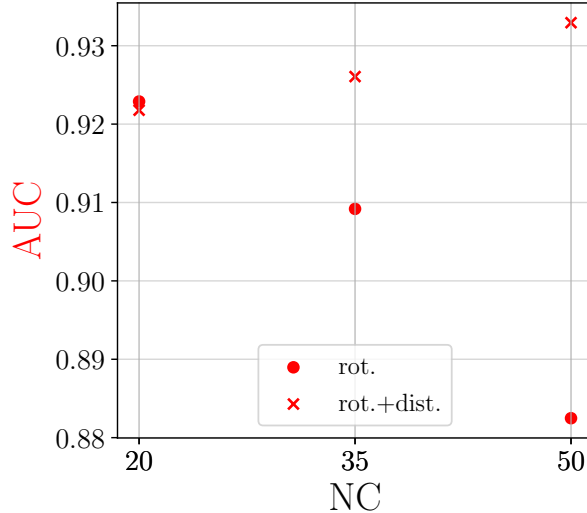


Figure 6.2: Low p_T distortion augmentation makes larger number of constituents ($n_C \equiv \text{NC}$) possible. Only applying rotation symmetry transformations (circles) results in deteriorating representation quality with larger number of constituents. This is reversed if the low p_T distortion augmentation is added (crosses).

In our case, alignment and uniformity on a hypersphere might be just difficult enough to require abstract feature formation, but already too constrained by the specific task to give an optimal data representation in the last layer.

6.3 Building up from parts

With a working setup and after finding appropriate hyper-parameters for training with contrastive loss, we can now introduce the transformer architecture. We use a model dimension of 1000 and four attention heads. After early tests, four successive transformer encoder layers seemed promising. We started with rotational symmetry transformation and low p_T distortion augmentation, like with the FCN tests, and trained on batch sizes of 128 with a signal to background ratio of 1.

Because many tests were done and there are many important aspects of a well performing contrastive learning transformer model, we will build up the performance from parts.

The truly first step is the freshly initialised neural network before any training is done. The network at that stage is a random function that maps sets of 50 constituents to \mathbb{R}^{1000} . Because the architecture contains summing up constituent embeddings, this mapping is already permutation invariant. By passing jet data through this network and evaluating the linear separability of top jets from QCD jets we quantify the representation quality. And we find it to be already higher (AUC ~ 0.87) than the quality of the raw constituent data (AUC ~ 0.80).

Next, we train the network while not applying any augmentation. Thus, positive pairs are identical jets and will invariably be mapped to the same representation. As the loss cannot be reduced by optimising alignment, the network only learns to map jet data uniformly onto a hypersphere. To reduce the problems of the noisy soft regime, the jets were cropped to only include the hardest $n_C = 20$ constituents. This network converges to a LCT performances of AUC ~ 0.93 .

Again just like with the FCN, the transformer network also benefits from a nonlinear projection head between the layer where the representations are taken from and the layer where the loss is applied. This benefit seems to plateau at 2 ... 3 head layers. We also tested the effect of layer normalisation on the head layers and found no significant deviation.

It is not obvious which augmentations work best. However, many different augmentations were implemented to shape what is learnt by the network and to overcome specific issues. One challenge was to be able to encode a high number of jet constituents n_C without introducing more noise than useful information. Figure 6.2 already established the efficacy of low p_T distortion augmentations. This addressed the distracting noisiness of η and ϕ in soft constituents.

The soft fill augmentation addresses another issue that arises when going to a higher number of constituents. For technical reasons, jets have to be stored in a fixed length vector; remaining entries of jets that have fewer than n_C constituents are padded with zeros. We know from the soft and collinear divergences of QCD, that the number of constituents is very noisy. If no countermeasures are taken, constituent embeddings of zero-pads strongly differ from embeddings of non-zero constituent. Thereby, the noisiness of the number of constituents has a high impact on the jet representations. In our dataset, the number of constituents is a distribution with a median ~ 50 , thus the magnitude of variation from this increases strongly when n_C is increased from 20 to 50. The soft fill augmentation tackles this by removing all zero-pads and replacing them with random soft noise of order 100 MeV. Because the soft noise will differ between each jet of a positive pair, this augmentation forces the representations to be irrelevant to any soft activity below 100 MeV.

An orthogonal approach to address the noise problems of soft constituents is to prevent that zero-pads are processed by the network in the first place. Of course the easy way to do this is to crop off soft activity entirely and use a low fixed n_C . However, this is not a solution we are seeking, because it also disregards possibly relevant information. Rather, this naive approach is what we are trying to circumvent. We prepare a boolean mask for separating non-zero constituents from zero-pads. The application of this mask will be to selectively ignore zero-pads in the attention mechanism. First, the mask decides which constituents are summed over and thus enter the jet representation. Second, the mask must be applied to suppress attention to any zero-pads. We do this by modifying the attention mechanism. Recall, that the attention mechanism, Equation (5.8), uses a softmax to determine the attention weights for every constituent. To effectively ignore constituents, we add negative infinity to the argument of that softmax function. Thus, regardless of the product of key and query, the weight of a zero-pad constituent will be 0. This binary mask makes sure that constituents with exactly $p_T = 0$ get no attention. We also tested a continuous version, where we added a function of p_T that diverges to negative infinity at $p_T \rightarrow 0$ to the softmax argument. For this, the binary function

$$\text{binary masking} \quad a(p_T) = \begin{cases} -\infty, & \text{if } p_T = 0 \\ 0, & \text{otherwise} \end{cases} \quad (6.1)$$

was modified to be

$$\text{IR masking} \quad a(p_T) = \frac{1}{2} \log(p_T) \quad . \quad (6.2)$$

Continuous masking deserves the name IR masking because it ensures *infrared safety of attention*.

Binary masking on the other hand, does not suppress soft activity and does not even suppress the number of constituents. While it stops zero-pads from entering explicitly, the jet representation will still encode the number of constituents. This is the case, because the embedding network has bias nodes and dimensions with a non-zero expectation value. When summed up, those dimensions will therefore invariably depend on the number of summands.

We find the effect of masking does not separate from other moving parts during the exploratory testing phase. That is, binary masking as well as IR masking improves the discriminative features of representations for some settings but also sometimes contributes to further worsen results. We find however, that IR masking performs better than binary masking.

The arguments in favour of incorporating more symmetries into the representations are clear: avoid pre-processing and reduce noise. To explore how this can be pushed further beyond our setting, we try to achieve translation invariance. The dataset we use throughout this work only contains jets that are already centred such that the p_T weighted centroid is in the origin of the η - ϕ -plane.

Because the data degeneracy corresponding to translations is already removed, there is no strong reason that incorporating translation invariance would benefit the quality of representations obtained from this dataset.

However, ultimately we are interested in developing the method and not the specific jet representations. Therefore, we explore if the method can be generalized to make another preprocessing step obsolete. To simulate a setting where the dataset is not centred, we augment the dataset with the translation symmetry transformation before evaluation. Translation invariance turned out to be a difficult challenge. Experiments were done both with the FCN and the transformer, with different amounts of translation and treatment of the circular structure of the η - ϕ -plane at $\phi = 2\pi$ as well as increased FCN hidden layer size; all without success. The metrics for representation quality were never comparably good as without translation invariance. Highlighting the interdependence of the augmentations we found later that translations are only beneficial if accompanied by a collinear splitting augmentation.

The collinear splitting of constituents to fill up zero-pads helped to push the performance of our representations beyond the mark of background rejection $\epsilon_b^{-1}(\epsilon_s = 0.5) = 120$.

The model that finally produced the best performing representations used rotation, low p_T distortion, translations, and collinear splitting to fill zero-pads. The hyper-parameters that define the architecture and the learning of the network are given in Table 6.1. In Figure 6.3, the corresponding linear classification performance is compared to the benchmark representations. That comparison shows JetCLR can outcompete the established jet substructure representations.

hyper-parameter	value
model (embedding) dimension	1000
feed-forward hidden dimension	1000
output dimension	1000
# self-attention heads	4
# transformer layers	4
# layers	2
dropout rate	0.1
optimizer	Adam($\beta_1 = 0.9, \beta_2 = 0.999$)
learning rate	5×10^{-5}
batch size	128
# epochs	500
continuous masking	✓
rotation + translation	✓
soft p_T + collinear	✓

Table 6.1: Hyper-parameters defining the transformer architecture and its training. Values are the most successful combination.

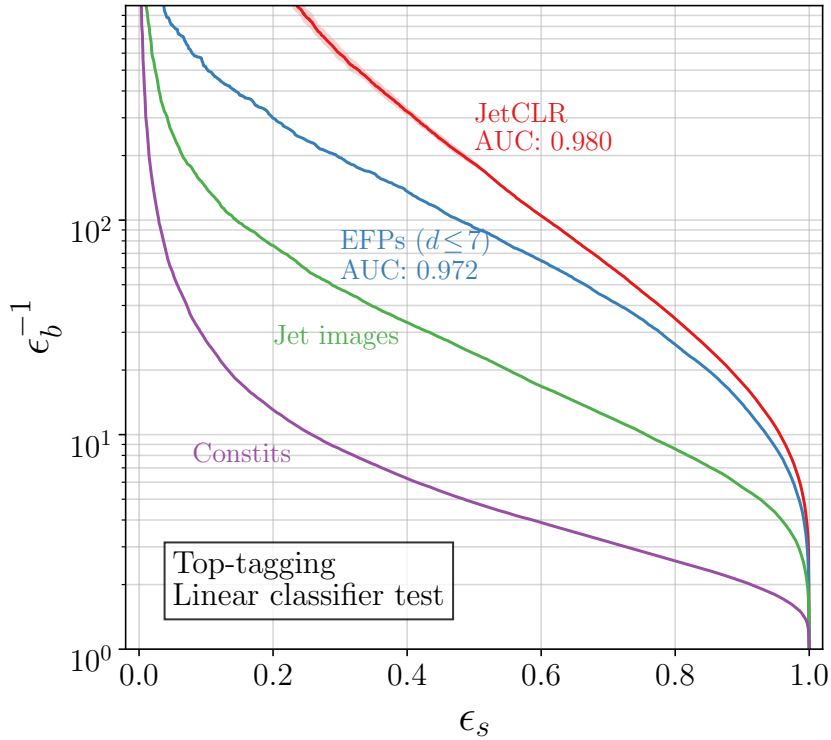


Figure 6.3: Top-tagging performance of different jet representations. Taken from [4]

7 Opening the box

Our networks contain 26 million learnable parameters. The deconstruction of the inner workings of neural networks is important both for understanding by which algorithms networks determine their output and for understanding shortcomings and side effects [52] in order to overcome them. Additionally, we might be able to learn the essential parts of the networks mapping and apply them for the creation of techniques that do not need to be trained at all.

Until now, we only know what is learnt in terms of what is optimised at the last layer, i.e. the contrastive loss, and in terms of all possible ways information can be processed in the network, i.e. the architecture.

7.1 Head

First of all, we measure the properties that were optimised through the minimisation of contrastive loss: alignment and uniformity. Strictly, the properties are only optimised in the representations of the last layer. However, as both properties are desirable for representations in general, we are interested also in the other representation layers, namely before the head and between the two head layers.

To evaluate the alignment of positive pairs in a controlled fashion we create rotated versions of the same jet for equidistant angles between 0 and 2π . Moving around the full circle and recording the similarity of the rotated jets to the original jet, we can directly measure how a representation depends on its rotation angle. As can be seen in Figure 7.1, training a network with symmetry transformations indeed leads to approximately invariant representations. Recall, that ideal similarity corresponds to $s(z, z') = 1$, as the chosen measure is the cosine of the angular distance on the hypersphere (5.2). The green line is the similarity curve of a network trained with rotated positive pairs. Its difference to 1 is several orders of magnitude lower than the red line, which is the similarity curve of a network trained *without* rotated positive pairs. This proves the effectiveness of the contrastive learning approach to align positive pairs.

Because rotating clockwise or anticlockwise cannot make a difference and rotating by an angle of $\alpha = 0$ leaves the jet unchanged, the similarity is minimal at $\alpha = \pi$. We observed that, for the network that was not trained with rotated positive pairs, this minimal similarity is ~ 0.5 .

As all jets are clusters of constituents centred at zero, they are, at first approximation, themselves already rotationally invariant. The more a jet resembles a cluster with equal density of constituents at any angle, the more rotationally invariant it will be even before a pass through our representation network. This effect becomes visible, when we trace the similarity curves of toy jets consisting of two prongs and three prongs, because they are less like circular clusters. The similarity curves of the representations nicely show the periodicity. We also observe that the minimal similarity is now ~ 0 . This is below the minimum of the similarity curve observed for jets with more constituents, showing that indeed inherent rotational invariance of raw jets data was the reason for their elevated similarity.

To complete the analysis of the network's outputs, we also have to consider negative pairs. In principle, their similarities will be contained in the interval $[-1, 1]$, but the distribution of similarities will depend on the trade-off between alignment and uniformity realised by the model.

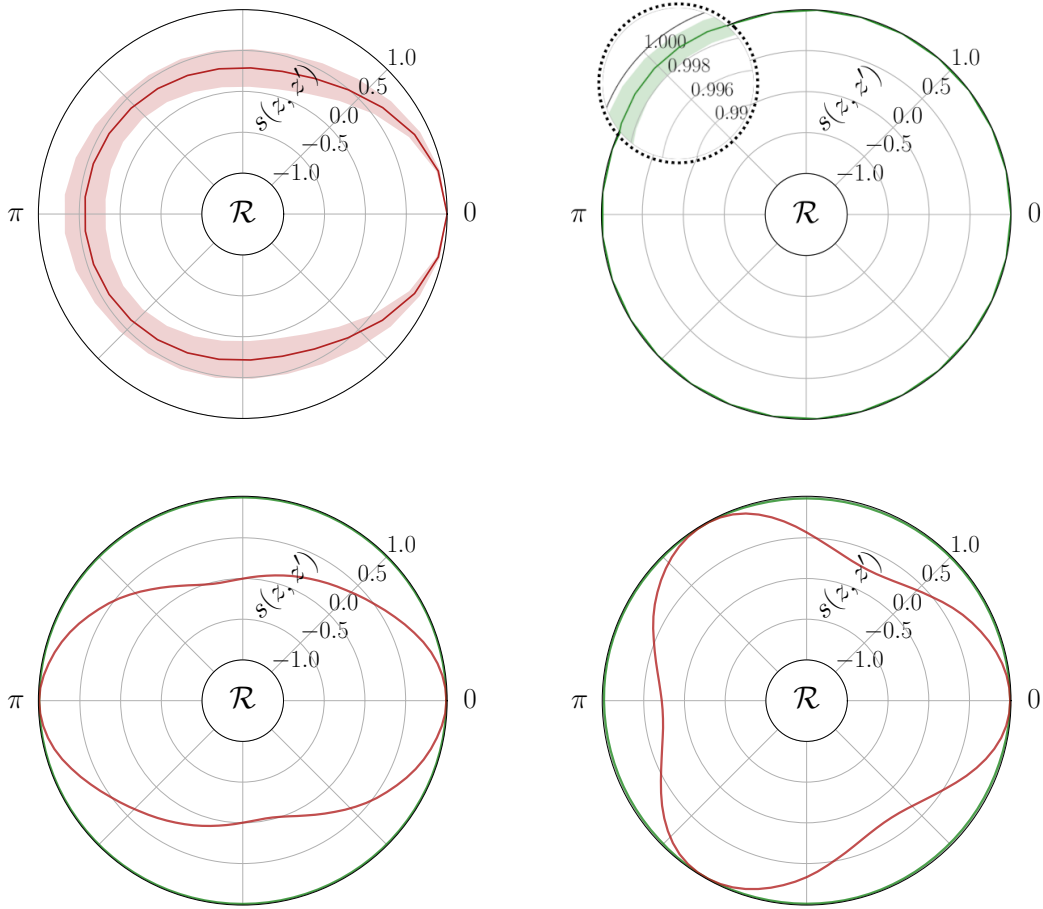


Figure 7.1: Visualisation of the rotational invariance in representation space. $s(z, z') = 1$ indicates identical representation. Top: representation trained without (left) and with (right) rotational transformations. Bottom: toy jets consisting of two and three prongs, similarity curves are green for training with rotational transformations, red for training without rotational transformations

Empirically, we observed that the *negative pair similarity* varies significantly across the different positions in the head. While the last representation and the intermediate representation have a similarity distribution centred at zero, the representation before the head network has a higher expected similarity, i.e. strong negative pair alignment.

This hints at an additional¹ explanation for the virtue of the head network:

- It is costly to model uniformity just with transformer encoder layers.
- It is advantageous to outsource the fulfilment of this important aspect of the loss function to the head network.

Figure 7.2 shows pairwise similarity matrices of a list of 100 QCD jets and 100 top jets for different representations. The key insight from this figure is that representations before the head network (upper right) have positive expected similarity (largely red upper triangle) and presentations within and after the head network (bottom) have vanishing expected similarity (largely grey upper triangle).

¹The first explanation was the misalignment of the contrastive task with the LCT test, Section 6.2.

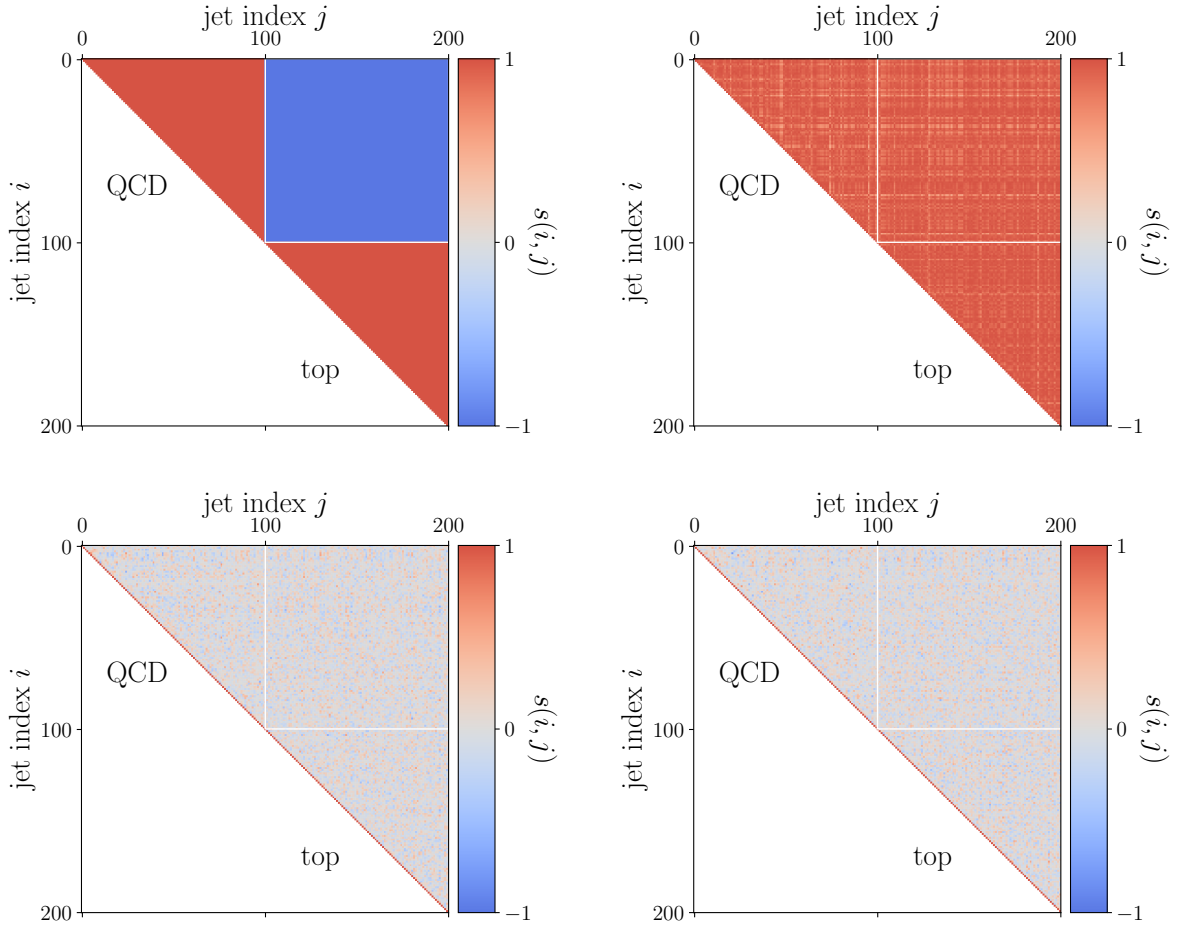


Figure 7.2: Pairwise similarity matrices of jet list. First 100 jets are QCD jets, last 100 are top jets. Diagonal line is similarity of each jet with itself. White lines were drawn to separate QCD-QCD, QCD-top, and top-top jet pairs. Four different representations are shown. **Upper left:** theoretic jet representation that only encodes the class label as opposing vectors on the hypersphere. **Upper right:** jet representation taken before the head network. Note the *high similarities between negative pairs*. **Bottom:** jet representation taken after the first, respectively the second head layer.

A rather trivial property of the representations becomes apparent when any of the three learned representations is compared with the toy representation in the upper left of Figure 7.2. This toy representation contains only the class label, such that there is perfect in-class alignment. This manifests in the plot as a high in-class pairwise similarity (red) and low out-of-class pairwise similarity (blue). However, as expected, in the actual representations the class label does not dominate the similarities. During training the class label is not shown to the network and the 1000 observables predictably encode many different properties of substructure than just the class label. Measuring the similarity compares *all* jet observables at once. To obtain the class label, a linear classifier must be trained to classify top and QCD jets from the representations that weight the observables appropriately.

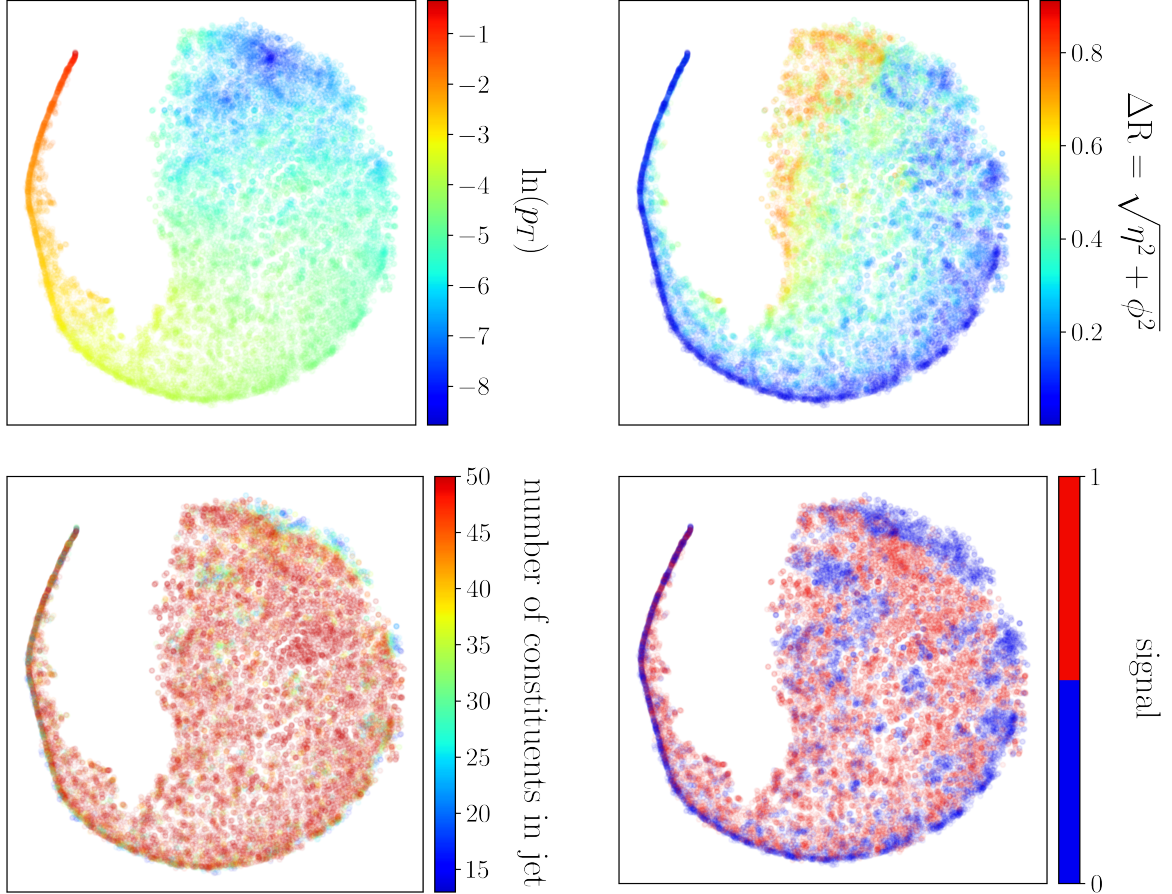


Figure 7.3: Manifold of constituent embeddings visualised with t-SNE and different colour filters. Top: transverse momentum and distance from centroid of constituent, Bottom: properties of the constituents’ parent jet: n_C and if it is a top jet (signal=1).

7.2 Sum

To learn more about how the transformer processes constituent data, we will investigate the constituent embeddings. An appropriate network position to take them from is before they are summed to form a jet-level representation, because by then they will have accumulated context information through the attention mechanisms. The summation constrains how constituent level information can enter jet observables. The dimensions of constituent embedding vectors should be viewed as additive contributions to the respective jet observable. Put differently, the value $x_i^{(m)}$ of constituent embedding vector x_i at index m is the additive contribution to jet observable m of constituent i given the context of its jet.

The 1000 dimensional constituent embeddings are visualised via dimensionality reduction with t-SNE; see Section 3.2. The specific t-SNE plots shown in Figure 7.3 use a perplexity of 130 and are trained until convergence with a learning rate of 200 for 1000 iterations. As metric for t-SNE we use the default Euclidean distance. The specific parameter choice, however, had no qualitative effect on the embedding. The constituents of 200 QCD and 200 top jets are prepared as a dataset. Corresponding labels are prepared for transverse momentum p_T and distance from centre $\Delta R = \sqrt{\eta^2 + \phi^2}$, as well as the number of constituents n_C of their jets and whether their jet was a top (signal) or a QCD (background) jet. Using those labels as colour maps, allows us to explore the embedding manifold.

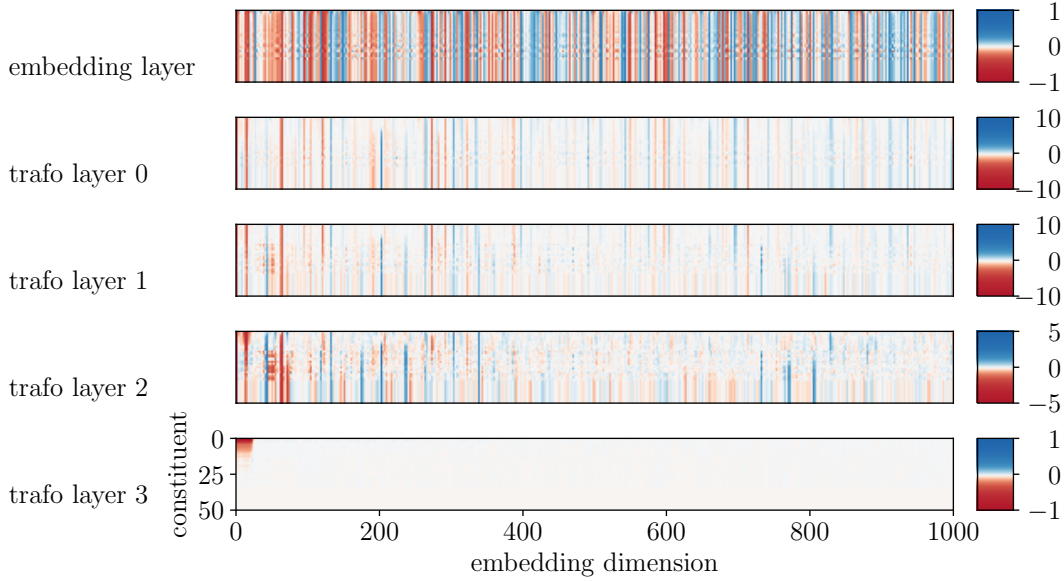


Figure 7.4: Jet data going through the transformer. For each transformer layer, 50 constituent embeddings with 1000 dimensions are shown as a matrix, with axes ordered by p_T and standard deviation of "trafo layer 3" embedding respectively. Note the *first 24 embedding dimensions* on the left in the last trafo layer.

From the t-SNE visualisation, we first observe that the constituents are mapped to a connected space, rather than segregated in multiple distinct clusters. This space is ordered both by transverse momentum and distance to the jet centre, as can be seen by the strong colour separation in Figure 7.3 (top). We also observe that top jet constituents and QCD jet constituents form separate spots (bottom right). Even though we masked-out zero-pads and thus do not show them here, it was tested that they form a separate cluster, which fits their qualitatively different nature. Note, it makes intuitive sense that the embedding manifold is predominantly ordered by transverse momentum and distance to centre, as this is the basic information that defines constituents. The context information acquired through the attention mechanism is more subtle.

From the connectedness of the constituent space, we can infer the connectedness of the jet representation space. Specifically, the representation used throughout this work was taken from before the head network, thus the jet representations are exactly the sums of constituent embeddings from Figure 7.3.

7.3 Transformer

We will now follow the processing of all constituents of a single jet through the network. This is not done to learn something specific about this jet or its constituents, but rather to get a first grasp of how constituent embeddings are created and to see if something unexpected awaits discovery.

Each of the 50 constituents of the same jet, initially defined by p_T , η and ϕ , are embedded to a 1000 dimensional vector each. Thus, we process a matrix of the shape 50×1000 , which we can illustrate as an image where each pixel is a matrix entry. Within the matrix, constituents are p_T -ordered. Since also the embedding dimensions can be reordered to simplify our understanding, we order them by the standard deviation of the embedding dimensions after the last transformer encoder layer.

We find that a small subset of embedding dimensions differ strongly from the rest. Interestingly, the constituent embeddings after the transformer have 24 dimensions with a standard deviation several orders of magnitude higher than the rest of the embedding dimensions. Moreover, these dimensions only show negative entries that scale with a monotonous function of transverse momentum. These 24 constituent-level jet-context-aware observables, and their jet-level counterparts should be studied more in detail at a later stage. In Figure 7.4 their values in a specific jet can be viewed. The key insight from this figure is that the last transformer layer outputs constituent embeddings, that have just a few dimensions with large magnitude, and that those dimensions also vanish, when the transverse momentum vanishes. Apart from that, other layers show dimensions that are not understood and are not the focus here. For this work, it must suffice to say that the network appears to have learnt to scale the largest constituent observables with transverse momentum.

It is an interesting question, why all of those observables show negative values. That a single observable only realises same-sign values, makes sense insofar that it avoids the cancellation of multiple constituents in the constituent sum. This tendency to avoid constituent cancellations would also explain the strong alignment of negative pairs that was observed in Figure 7.2. When contributions of constituents mostly have the same sign for any dimension, the expected value will be different from zero. This in turn leads a positive expected value of pairwise similarity (5.1).

This exploration should be closed by some remarks about the uncertainties and open questions. First, it is very well possible that other representation dimensions than the 24 special ones encode important information, just at a different scale. The existence of two types of representation dimensions, separated by their magnitude, nevertheless implicates that the set of 24 observables found in this analysis plays an important role. Second, the analysis only applies to the model presented in Figure 6.3. While earlier models were evaluated similarly and some of the described features are present in all of the analysed models, the detailed evaluation presented here is not yet done fully for a separate model. And third, what context information exactly is accumulated inside the transformer encoder layers, each 6 million parameters in size, is still unclear. More explainability research [53, 54] has to be done if one wants to harness the representation capability of this machine learning model. In principle, this could be a research direction for developing new analytical jet substructure observables.

8 Conclusions

Modern physics produces large datasets of high dimensional measurement data. Analysing this data for data driven research faces the challenge of how to represent high-dimensional information about physical processes. We have explained why good representations must exploit fundamental and approximate symmetries, retain discriminative features and be new-physics agnostic. We have discussed existing high dimensional jet substructure representations, namely jet images and energy flow polynomials (EFPs). The jet image must rely on preprocessing to account for invariances and its size scales quadratically with resolution. EFPs include important symmetries, but are inflexible compared to machine learning methods.

Proposing a method for learning representation, we have shown how symmetries can be incorporated into representations with contrastive learning and transformers, and have presented the JetCLR tool¹ to solve this task for jet substructure.

The evaluation of the discriminative features was done by training a linear classifier on the representations. A good performance of said classifier indicates that the representation is powerful, because the linear classifier itself is not expressive enough to untangle features from suboptimal representations. We have shown that JetCLR can outcompete jet images and EFPs in these tests.

Systematically, we have identified which hyperparameters, symmetry transformations and data augmentations lead to the best representations. Because augmentations can be added flexibly just by defining a transformation on the input data space and sampling augmented positive pairs from it, they also became a handle to overcome obstacles by shaping the importance of different features in the representation.

We found that a nonlinear projection head network separating the representation layer from the output layer is crucial for contrastive learning to work in our setting. The benefit of it is twofold: First, the contrastive loss is misaligned with the linear classifier test on the output layer, i.e. optimising the contrastive loss directly on a representation is not optimal for LCT performance. By separating the layers with a head, this can be remedied. Second, converged models are avoiding the cancellation of constituent embedding dimensions in the summation after the transformer, effectively leading to aligned jets representations before the head. This is only possible when the summation is followed by a head network, because the loss punishes any alignment of negative pairs.

Finally, first steps were taken to look at the processing of jet data within the neural network.

¹The core code for JetCLR is maintained at <https://github.com/bmdillon/JetCLR>

References

- [1] P. Mehta et al., *A high-bias, low-variance introduction to machine learning for physicists*, *Physics Reports* **810** (2019) 1-124, [10.1016/j.physrep.2019.03.001](https://doi.org/10.1016/j.physrep.2019.03.001) (2018), [arXiv:1803.08823](https://arxiv.org/abs/1803.08823) [[physics.comp-ph](#)].
- [2] M. Feickert and B. Nachman, *A Living Review of Machine Learning for Particle Physics*, (2021), [arXiv:2102.02770](https://arxiv.org/abs/2102.02770) [[hep-ph](#)].
- [3] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, *A Simple Framework for Contrastive Learning of Visual Representations*, *Proceedings of the 37th International Conference on Machine Learning*, *PMLR* **119**, 1597 (2020), [arXiv:2002.05709](https://arxiv.org/abs/2002.05709) [[cs.LG](#)].
- [4] B. M. Dillon, G. Kasieczka, H. Olschlager, T. Plehn, P. Sorrenson, and L. Vogel, *Symmetries, safety, and self-supervision*, (2021), [arXiv:2108.04253](https://arxiv.org/abs/2108.04253) [[hep-ph](#)].
- [5] M. Kobayashi and T. Maskawa, *CP-violation in the renormalizable theory of weak interaction*, *Progress of Theoretical Physics* **49**, 652 (1973).
- [6] CDF Collaboration, *Observation of Top Quark Production in $\bar{p}p$ Collisions with the Collider Detector at Fermilab*, *Physical Review Letters* **74**, 2626 (1995), [arXiv:hep-ex/9503002](https://arxiv.org/abs/hep-ex/9503002) [[hep-ex](#)].
- [7] DØ Collaboration, *Observation of the Top Quark*, *Physical Review Letters* **74**, 2632 (1995), [arXiv:hep-ex/9503003](https://arxiv.org/abs/hep-ex/9503003) [[hep-ex](#)].
- [8] P. W. Higgs, *Broken Symmetries and the Masses of Gauge Bosons*, *Physical Review Letters* **13**, 508 (1964).
- [9] ATLAS Collaboration, *Observation of a new particle in the search for the Standard Model Higgs boson with the ATLAS detector at the LHC*, *Physics Letters B* **716**, 1 (2012), [arXiv:1207.7214](https://arxiv.org/abs/1207.7214) [[hep-ex](#)].
- [10] CMS Collaboration, *Observation of a new boson at a mass of 125 GeV with the CMS experiment at the LHC*, *Physics Letters B* **716**, 30 (2012), [arXiv:1207.7235](https://arxiv.org/abs/1207.7235) [[hep-ex](#)].
- [11] M. J. Strassler and K. M. Zurek, *Echoes of a hidden valley at hadron colliders*, *Physics Letters B* **651**, 374 (2007).
- [12] D. E. Morrissey, T. Plehn, and T. M. Tait, *Physics searches at the LHC*, *Physics Reports* **515**, 1 (2012).
- [13] D. J. Gross and F. Wilczek, *Ultraviolet behavior of non-abelian gauge theories*, *Physical Review Letters* **30**, 1343 (1973).
- [14] R. P. Feynman, *High energy collisions : Third International Conference held at State University of New York* (Gordon & Breach, New York London, 1969), pages 237–249.
- [15] *Search for Narrow Resonances using the Dijet Mass Spectrum*, technical report (CERN, Geneva, 2015), <https://cds.cern.ch/record/2048099>.
- [16] S. Marzani, G. Soyez, and M. Spannowsky, *Looking inside jets* (Springer International Publishing, 2019).
- [17] M. Cacciari, G. P. Salam, and G. Soyez, *The anti- k_t jet clustering algorithm*, *Journal of High Energy Physics*, *JHEP* **2008**, 063 (2008), [arXiv:0802.1189](https://arxiv.org/abs/0802.1189) [[hep-ph](#)].
- [18] L. Vogel, *JetCLR: A Self-Supervised Machine Learning Framework for Contrastive Learning of Jet Representations*, Bachelor thesis (Heidelberg University, 2021).
- [19] I. Bigi, Y. Dokshitzer, V. Khoze, J. Kühn, and P. Zerwas, *Production and decay properties of ultra-heavy quarks*, *Physics Letters B* **181**, 157 (1986).

-
- [20] C. T. Hill and S. J. Parke, *Top quark production: Sensitivity to new physics*, *Physical Review D* **49**, 4454 (1994).
- [21] M. Beneke et al., *Top quark physics*, (2000), [arXiv:hep-ph/0003033 \[hep-ph\]](#).
- [22] P. A. Zyla et al., Particle Data Group, *Review of Particle Physics*, *Progress of Theoretical and Experimental Physics* **2020**, 083C01 (2020).
- [23] D. E. Kaplan, K. Rehermann, M. D. Schwartz, and B. Tweedie, *Top Tagging: A Method for Identifying Boosted Hadronically Decaying Top Quarks*, *Physical Review Letters* **101**, 142001 (2008), [arXiv:0806.0848 \[hep-ph\]](#).
- [24] T. Plehn and M. Spannowsky, *Top Tagging*, *Journal of Physics G: Nuclear and Particle Physics* **39**, 083001 (2012), [arXiv:1112.4441 \[hep-ph\]](#).
- [25] G. Kasieczka, T. Plehn, M. Russell, and T. Schell, *Deep-learning Top Taggers or The End of QCD?*, *Journal of High Energy Physics*, *JHEP* **2017**, 6 (2017), [arXiv:1701.08784 \[hep-ph\]](#).
- [26] A. Butter, G. Kasieczka, T. Plehn, and M. Russell, *Deep-learned Top Tagging with a Lorentz Layer*, *SciPost Physics* **5**, 28 (2018), [arXiv:1707.08966 \[hep-ph\]](#).
- [27] S. Macaluso and D. Shih, *Pulling Out All the Tops with Computer Vision and Deep Learning*, *Journal of High Energy Physics*, *JHEP* **2018**, 121 (2018), [arXiv:1803.00107 \[hep-ph\]](#).
- [28] G. Kasieczka et al., *The Machine Learning Landscape of Top Taggers*, *SciPost Physics* **7**, 14 (2019), [arXiv:1902.09914 \[hep-ph\]](#).
- [29] T. Heimel, G. Kasieczka, T. Plehn, and J. M. Thompson, *QCD or What?*, *SciPost Physics* **6**, 30 (2019), [arXiv:1808.08979 \[hep-ph\]](#).
- [30] K. Hornik, M. Stinchcombe, and H. White, *Multilayer feedforward networks are universal approximators*, *Neural Networks* **2**, 359 (1989).
- [31] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, *Learning representations by back-propagating errors*, *Nature* **323**, 533 (1986).
- [32] L. Van der Maaten and G. Hinton, *Visualizing data using t-SNE*. *Journal of machine learning research* **9** (2008).
- [33] M. Wattenberg, F. Viégas, and I. Johnson, *How to use t-SNE effectively*, *Distill* **1**, e2 (2016).
- [34] J. Thaler and K. V. Tilburg, *Identifying Boosted Objects with N -Subjettiness*, *Journal of High Energy Physics*, *JHEP* **2011**, 15 (2011), [arXiv:1011.2268 \[hep-ph\]](#).
- [35] J. Cogan, M. Kagan, E. Strauss, and A. Schwartzman, *Jet-Images: Computer Vision Inspired Techniques for Jet Tagging*, *Journal of High Energy Physics*, *JHEP* **2015**, 118 (2015), [arXiv:1407.5675 \[hep-ph\]](#).
- [36] L. de Oliveira, M. Kagan, L. Mackey, B. Nachman, and A. Schwartzman, *Jet-Images – Deep Learning Edition*, *Journal of High Energy Physics*, *JHEP* **2016**, 69 (2016), [arXiv:1511.05190 \[hep-ph\]](#).
- [37] P. T. Komiske, E. M. Metodiev, and J. Thaler, *Energy flow polynomials: A complete linear basis for jet substructure*, *Journal of High Energy Physics*, *JHEP* **2018**, 13 (2018), [arXiv:1712.07124 \[hep-ph\]](#).
- [38] F. V. Tkachov, *Measuring multijet structure of hadronic energy flow or what is a jet?*, *Int.J.Mod.Phys. A12* (1997) 5411-5529, [10.1142/S0217751X97002899](#) (1996), [arXiv:hep-ph/9601308 \[hep-ph\]](#).

-
- [39] T. Wang and P. Isola, *Understanding Contrastive Representation Learning through Alignment and Uniformity on the Hypersphere*, Proceedings of the 37th International Conference on Machine Learning, PMLR **119**, 9929 (2020), [arXiv:2005.10242 \[cs.LG\]](#).
- [40] F. Wang and H. Liu, *Understanding the Behaviour of Contrastive Loss*, arXiv e-print (2020), [arXiv:2012.09740 \[cs.LG\]](#).
- [41] J. Lee, Y. Lee, J. Kim, A. R. Kosiorek, S. Choi, and Y. W. Teh, *Set transformer: a framework for attention-based permutation-invariant neural networks*, (2018), [arXiv:1810.00825 \[cs.LG\]](#).
- [42] A. Vaswani et al., *Attention Is All You Need*, Advances in Neural Information Processing Systems, NIPS **30** (2017), [arXiv:1706.03762 \[cs.CL\]](#).
- [43] T. Sjöstrand et al., *An Introduction to PYTHIA 8.2*, Computer Physics Communications **191**, 159 (2015), [arXiv:1410.3012 \[hep-ph\]](#).
- [44] J. de Favereau et al., *DELPHES 3: a modular framework for fast simulation of a generic collider experiment*, Journal of High Energy Physics, JHEP **2014**, 57 (2014), [arXiv:1307.6346 \[hep-ex\]](#).
- [45] M. Cacciari, G. P. Salam, and G. Soyez, *FastJet user manual*, The European Physical Journal C **72**, 1896 (2012), [arXiv:1111.6097 \[hep-ph\]](#).
- [46] A. Paszke et al., *PyTorch: An Imperative Style, High-Performance Deep Learning Library*, arXiv e-print (2019), [arXiv:1912.01703 \[cs.LG\]](#).
- [47] F. Pedregosa et al., *Scikit-learn: machine learning in Python*, Journal of Machine Learning Research **12**, 2825 (2011).
- [48] C. R. Harris et al., *Array programming with NumPy*, Nature **585**, 357 (2020).
- [49] M. Noroozi and P. Favaro, *Unsupervised Learning of Visual Representations by Solving Jigsaw Puzzles*, (2016), [arXiv:1603.09246 \[cs.CV\]](#).
- [50] C. Olah, A. Mordvintsev, and L. Schubert, *Feature visualization*, Distill **2**, 10.23915/distill.00007 (2017).
- [51] M. Chen et al., *Generative pretraining from pixels*, in Proceedings of the 37th international conference on machine learning, Vol. 119, edited by H. D. III and A. Singh, Proceedings of Machine Learning Research (2020), pages 1691–1703, <http://proceedings.mlr.press/v119/chen20s.html>.
- [52] S. McGregor, *Preventing Repeated Real World AI Failures by Cataloging Incidents: The AI Incident Database*, (2020), [arXiv:2011.08512 \[cs.CY\]](#).
- [53] K. Clark, U. Khandelwal, O. Levy, and C. D. Manning, *What Does BERT Look At? An Analysis of BERT’s Attention*, (2019), [arXiv:1906.04341 \[cs.CL\]](#).
- [54] H. Chefer, S. Gur, and L. Wolf, *Generic Attention-model Explainability for Interpreting Bi-Modal and Encoder-Decoder Transformers*, (2021), [arXiv:2103.15679 \[cs.CV\]](#).

Acknowledgements

I would like to thank Tilman Plehn and Barry Dillon for creating an open atmosphere, ensuring that the pandemic did not hinder communication and for letting me work in such close cooperation on this interesting project. Special thanks goes to Lorenz Vogel for being a great collaborator. Furthermore, I would like to thank Peter Sorrensen for answering my machine learning questions with fascinating insights about general features in neural networks. I also want to thank Maximilian Herzog for all his comments to this thesis. Finally, I want to thank the whole phenomenology group for the nice environment they provide.

Declaration

I herewith formally declare that I have composed the present thesis myself and without use of any other than the cited sources and aids.*

Heidelberg, 17 August 2021


.....
Hans Olischläger

*Erklärung: Hiermit versichere ich, dass ich diese Arbeit selbstständig verfasst habe und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.