Department of Physics and Astronomy
Heidelberg University

Bachelor Thesis in Physics
submitted by

Lorenz Vogel

born in Bad Frankenhausen (Germany)

2021

JetCLR: A Self-Supervised Machine Learning Framework for Contrastive Learning of Jet Representations

— Learning Better Observables for Jets —

This Bachelor Thesis has been carried out by

Lorenz Vogel

at the

Institute for Theoretical Physics
Heidelberg University

under the supervision of

Prof. Dr. Tilman Plehn

Abstract

The big challenge in phenomenological analysis is to find suitable high-level observables to represent the low-level, high-dimensional data collected in experiments. Contrastive learning of representations (CLR) recently showed great success in computer vision and audio processing. We therefore applied the CLR method to jet physics. In this thesis we will present JetCLR, a data-driven and self-supervised framework for contrastive learning of jet observables. JetCLR is developed upon the SimCLR framework proposed by the Google Brain team. By minimizing the contrastive loss function, JetCLR learns jet representations that are (approximately) invariant to physically-motivated symmetries and that also retain discriminative information contained in the dataset it is trained on. JetCLR is implemented using a simple fully-connected network (FCN) as well as a more sophisticated transformer-encoder network. Using the top-tagging reference dataset and a supervised linear classifier test (LCT), we will show that our JetCLR framework achieves significant improvement over the jet-image representation. Furthermore, the JetCLR representations even outperform the energy flow polynomials (EFPs).

Zusammenfassung

Die große Herausforderung in der phänomenologischen Analyse von experimentellen Daten besteht darin, geeignete (physikalische) Größen und Merkmale, sogenannte Observablen, zu finden, mit denen die meist hoch-dimensionalen Daten dargestellt werden können. Im Bereich der maschinellen Bild- und Tonverarbeitung haben kürzlich Anwendungen, die auf der CLR-Idee (*contrastive learning of representations*) beruhen, große Erfolge gezeigt – daher haben wir die CLR-Methode auf Teilchenjets angewendet. In dieser Arbeit werden wir, basierend auf dem von Google Brain entwickelten SimCLR-Framework, eine Methode vorstellen, die es ermöglicht, durch selbst-überwachtes maschinelles Lernen geeignete Observablen für Teilchenjets zu konstruieren. Wir nennen diese Methode *JetCLR* (*Contrastive Learning of Jet Representations*). Die mithilfe von JetCLR konstruierten Observablen sind dabei (annährend) invarant gegenüber physikalisch-motivierten Symmetrien, die wir den neuronalen Netzwerken durch sogennante *augmentations* präsentieren. Als Basis für unser JetCLR-Framework haben wir sowohl ein einfaches *fully-connected network (FCN)* als auch ein anspruchsvolleres *transformer-encoder network* verwendet. Mithilfe eines einfachen *linear classifier* Tests werden wir zeigen, dass die von unserer JetCLR-Methode konstruierten Observablen bei der Identifizierung von Top-Quarks (*top-tagging*) eine signifikante Verbesserung gegenüber der *jet-image* Darstellung liefern und sogar die Ergebnisse der *energy flow polynomials (EFPs)* übertreffen.

Contents

List of Abbreviations

ANN	artificial neural network
AUC	area under the ROC curve
BCE	binary cross-entropy
BSM	beyond the Standard Model
CERN	European Organization for Nuclear Research
CLR	contrastive learning of representations
CMS	center-of-mass system
CNN	convolutional neural network
DNN	deep neural network
EFP	energy flow polynomial
FCN	fully-connected network
HEP	high-energy physics
IRC-safe	infrared- and collinear-safe
LCT	linear classifier test
LHC	Large Hadron Collider
MHSA	multi-headed self-attention
ML	machine learning
NN	neural network
NT-Xent	normalized temperature-scaled cross-entropy
QCD	quantum chromodynamics
QFT	quantum field theory
ReLU	rectified linear unit
ResNet	residual neural network
ROC	receiver operating characteristic
SGD	stochastic gradient descent
SM	Standard Model
t-SNE	t-distributed stochastic neighbor embedding
VAE	variational autoencoder

List of Figures

1 Introduction

The Large Hadron Collider (LHC) at the European Organization for Nuclear Research (CERN) in Geneve, Switzerland, is a proton-proton (pp) collider that made it possible to study new kinematic regimes using center-of-mass energies of up to $\sqrt{s} = 14\,\text{TeV}$. Studying high-energy collisions at the LHC, for example, led to the discovery of the Higgs boson in 2012 [1, 2], which completed the Standard Model (SM) of particle physics.

However, experimental observations such as neutrino oscillations, implying that neutrinos must have non-zero mass [3], or dark matter and dark energy cannot be explained by the SM, which means that theories beyond the Standard Model (BSM) are necessary.

Many of these BSM theories allow the production of very heavy resonances, which in turn typically decay into top quarks. Top quarks produced in decays of new heavy resonances would have large transverse momenta (p_T). Hence, the identification of boosted, hadronically decaying top quarks is an important part of the direct BSM seach in high-energy physics (HEP) [4, 5].

The big challenge for phenomenological analysis is to represent low-level, high-dimensional data collected in experiments by high-level observables. In general, there are three conditions that high-level jet data representations should fulfill [6]:

1. Jets have physically-motivated symmetries [7] (for example due to detector geometry or because of relativistic space-time properties), such as rotations and translations in the η-ϕ plane or low-p_T noise, and we want our latent space to be invariant under these transformations.

2. We want our high-level jet observables to retain discriminative information contained within the low-level, high-dimensional dataset.

3. Since the search for new physics discoveries should be model-independent, we want our jet observables to be defined in a data-driven and new-physics-agnostic manner.

Instead of using hand-engineered features, we define jet observables in a self-supervised way by training a deep learning architecture on mapping the low-level raw input data to high-level latent space representations [6], which will hopefully contain more abstract and useful information to improve the downstream task performance [8].

Self-supervised models for the contrastive learning of representations (CLR), such as SimCLR [9], TCLR [10], SoundCLR [11], or MolCLR [12], recently demonstrated that they have the ability to learn powerful representations in the field of computer vision, audio processing, and chemistry.

In this thesis, we introduce and investigate a data-driven and self-supervised framework for contrastive learning of jet observables, which we call JetCLR (*JetCLR: Contrastive Learning of Jet Representations*). Our JetCLR model is developed upon the contrastive learning framework described in Ref. [9]. The idea behind JetCLR is to use the contrastive loss function to construct a mapping $\mathcal{J} \to \mathcal{R}$ of the jets at constituent level ($\mathcal{J}$) to a representation space ($\mathcal{R}$), which is invariant to certain transformations, but which also retains discriminative information.

JetCLR learns informative representations by optimizing the contrastive loss function, i.e. matching similar examples and pushing apart dissimilar examples. Each jet in latent space should be close to its augmented version and far from everything else. As symmetry augmentations for our jet data we try rotations and translations in the η-ϕ plane as well as low-p_T modifications. The mapping $\mathcal{J} \to \mathcal{R}$ is implemented by using a simple fully-connected network (FCN) and a more sophisticated transformer-encoder network. The transformer network enables the representation space $\mathcal{R}$ to be invariant to the jet constituents ordering (permutation invariance).

In order to assess the quality of the representations learned by JetCLR, we implement a supervised linear classifier test (LCT). Our studies and experiments were performed using the *Top-Tagging Reference Dataset*, a well-established and common benchmark dataset for top-tagging architectures [13, 14, 15]. We show that our proposed JetCLR method achieves significant improvement over the jet-image representation [16, 17, 18] and that the representations learned by JetCLR even outperform the theory-inspired energy flow polynomials (EFPs) [19] in the downstream LCT.

The presented results in this thesis were achieved in collaboration with other researchers and will soon be published [6].[1] The rest of this thesis is organized as follows: Chapter 2 goes over the key concepts of Standard Model (SM) physics and top-quark properties, followed by the basics of machine learning (ML) and neural networks (NNs) in Chapter 3. Important jet representations, such as the jet-image representation and the energy flow polynomials (EFPs), are introduced in Chapter 4. In Chapter 5, we briefly review the SimCLR framework proposed by the Google Brain team in Ref. [9] and then discuss the contrastive loss function in more detail. Furthermore, we introduce our JetCLR framework for contrastive learning of jet observables. Our experimental setup, including details on the top-tagging dataset and the network architectures, as well as the benchmark results are presented in Chapter 6. Finally, in Chapter 7, we summarize our findings and give an outlook on further applications of the learned representations in unsupervised anomaly detection tasks.

[1] Our preliminary results were already presented at the *ML4Jets2021* workshop. The presentation slides are available at https://indico.cern.ch/event/980214/contributions/4413523/. The JetCLR `Python` code will be available and maintained on GitHub: https://github.com/bmdillon/JetCLR.

2 Physics Background

In this chapter, we give a brief overview of kinematic variables used in high-energy physics (HEP). We also cover the basics of the Standard Model (SM) of particle physics. Since we are using the top-tagging reference dataset proposed in Refs. [13, 14, 15], we additionally give an introduction to the most important properties of the top quark.

2.1 Relativistic Kinematics and Kinematic Variables

In this thesis, unless otherwise noted, we will use natural units with $c = \hbar = 1$, as is customary in particle physics. By combining the relativistic energy $E = \gamma m$ and the three-momentum $\boldsymbol{p} = \gamma\beta m$ of a massive particle of rest mass m (β denotes the particle's velocity in units of c and $\gamma = 1/\sqrt{1-\beta^2}$ denotes the Lorentz factor), one gets the energy-momentum four-vector (short: four-momentum) of the particle:

$$p^\mu \coloneqq (E, \boldsymbol{p}) = (E, p_x, p_y, p_z) \qquad \text{with} \qquad E = \sqrt{|\boldsymbol{p}|^2 + m^2} \tag{2.1}$$

The expression $E^2 = |\boldsymbol{p}|^2 + m^2$ relates the total energy E to the three-momentum $\boldsymbol{p}$ and the rest mass m and is therefore called the energy-momentum relation (or relativistic dispersion relation). The energy-momentum relation can be obtained by calculating the Minkowski norm squared of the four-momentum:

$$p^2 \equiv \eta_{\mu\nu} p^\mu p^\nu = p_\mu p^\mu = E^2 - |\boldsymbol{p}|^2 = m^2 \quad , \tag{2.2}$$

where $\eta_{\mu\nu} = \text{diag}(+1, -1, -1, -1)$ denotes the metric tensor of special relativity, the so-called Minkowski metric. The particle's rest mass m is a Lorentz invariant quantity, which is why it is also called invariant mass.

Using spherical coordinates with the polar angle $\theta \in [0, \pi]$ and the azimuthal angle $\phi \in (-\pi, \pi]$, we can parameterize the three-momentum $\boldsymbol{p} = (p_x, p_y, p_z)$ as follows:

$$\boldsymbol{p} = |\boldsymbol{p}| \begin{pmatrix} \sin\theta\cos\phi \\ \sin\theta\sin\phi \\ \cos\theta \end{pmatrix} \tag{2.3}$$

The z-axis is defined as the beam direction and polar angle θ is the angle between the particle's trajectory, defined by the three-momentum $\boldsymbol{p}$, and the beam direction. Since the laboratory frame and the center-of-mass system (CMS) in high-energy collision experiments are conntected to each other by a Lorentz boost along the beam axis, we introduce some variables which have simple transformation properties under a Lorentz boost along this direction:

transverse momentum: $$p_\text{T} = \sqrt{p_x^2 + p_y^2} \tag{2.4a}$$

transverse mass: $$m_\text{T} = \sqrt{p_\text{T}^2 + m^2} \xrightarrow{m \to 0} p_\text{T} \tag{2.4b}$$

rapidity: $$y = \frac{1}{2} \ln\left(\frac{E + p_z}{E - p_z}\right) \tag{2.4c}$$

pseudo-rapidity: $$\eta = \frac{1}{2} \ln\left(\frac{|\boldsymbol{p}| + p_z}{|\boldsymbol{p}| - p_z}\right) = -\ln\left(\tan\frac{\theta}{2}\right) \xrightarrow{m \to 0} y \tag{2.4d}$$

While the transverse momentum p_T and the transverse mass m_T are invariant under Lorentz boosts along the beam axis, the rapidity y, however, is not invariant but additive under such a Lorentz transformation.

Since we are looking at ultra-relativistic particles (particles with velocities close to the speed of light, i.e. $|\boldsymbol{p}| \gg m$ and therefore $E \approx |\boldsymbol{p}|$), we can neglect the particle masses, such that the pseudo-rapidity η is equal to the rapidity y. In this case, the transverse momentum p_T (momentum component that is perpendicular to the beam axis), the pseudo-rapidity η and the azimuthal angle ϕ are useful quantities to describe the four-momentum of a particle:

$$p^\mu = \begin{pmatrix} m_\mathrm{T} \cosh y \\ p_\mathrm{T} \cos\phi \\ p_\mathrm{T} \sin\phi \\ m_\mathrm{T} \sinh y \end{pmatrix} \xrightarrow{m \to 0} p_\mathrm{T} \begin{pmatrix} \cosh\eta \\ \cos\phi \\ \sin\phi \\ \sinh\eta \end{pmatrix} \tag{2.5}$$

For a symmetric high-energy collision of two protons A and B (due to the high energies, the proton masses can be neglected), the four-momenta in the laboratory system are given by:

$$p_A^\mu = (E_\mathrm{B}, 0, 0, E_\mathrm{B}) \qquad \text{and} \qquad p_B^\mu = (E_\mathrm{B}, 0, 0, -E_\mathrm{B}) \tag{2.6}$$

Here, E_B denotes the beam energy. The square root of the Mandelstam variable s describes the center-of-mass energy E_CM, i.e. the total energy available in the CMS:

$$E_\mathrm{CM} = \sqrt{s} = \sqrt{(p_A^\mu + p_B^\mu)^2} = 2E_\mathrm{B} \tag{2.7}$$

As we can see, for a symmetric collision process the total energy is twice the beam energy.

2.2 The Standard Model of Particle Physics

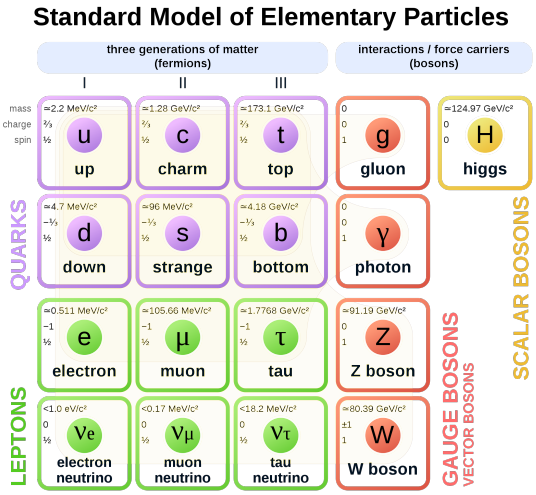


Figure 2.1: The Standard Model (SM) of particle physics, which describes all known elementary particles and three of the four known fundamental forces. Taken from [20].

The Standard Model (SM) of particle physics, formulated in the framework of quantum field theory (QFT), describes all known elementary particles and three of the four known fundamental forces (electromagnetism, the weak interaction and the strong force, but not gravity).

In the SM (see Figure 2.1), elementary particles are divided into fermions (spin-$\frac{1}{2}$ particles that make up matter, i.e. quarks and leptons) and bosons (spin-1 force carriers that mediate the fundamental forces, called gauge bosons). Photons mediate the electromagnetic interaction, gluons mediate the strong interaction, and $W^\pm$- and Z-bosons mediate the weak interaction.

Quarks, gluons and their strong interactions are described by quantum chromodynamics (QCD). Due to color confinement, quarks and antiquarks only occur in bound states, so-called hadrons. This is because QCD is asymptotically free. Half-integer spin hadrons are called baryons. Baryons are made up of three quarks, and each of these three quarks has a different color charge. Mesons are integer spin hadrons and consist of quark-antiquark pairs.

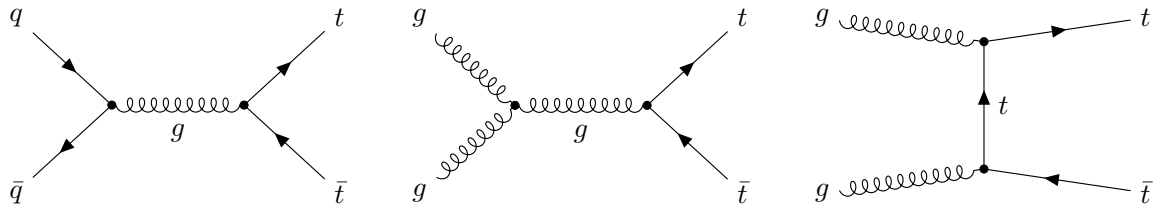
In 1995, the DØ and the CDF collaborations published their evidences for the discovery of the *top quark* [21, 22].They searched for the production of top-antitop ($t\bar{t}$) pairs in proton-antiproton ($p\bar{p}$) collisions at the Tevatron (Fermilab, USA) with a center-of-mass energy of $\sqrt{s} = 1.8\,\text{TeV}$. In 2009, both collaborations announced the observation of single top-quark production [23, 24].

There are several reasons why the top quark is of particular interest to experimental and theoretical physics: Due to its large mass, the top quark is the only quark which decays before hadronization [25]. This gives physicists direct access to the properties of a bare quark. Furthermore, the Yukawa coupling to the Higgs boson is of order unity [14, 25]. From a BSM physics point of view, the top quark could give insights into possible extensions of the SM [4, 26]. For example, many BSM physics theories predict particles and resonances that will decay to top quarks [4, 5].

The LHC at CERN enabled the possibility to study new energy regimes in proton-proton (pp) collisions and to produce such BSM particles. Starting with a center-of-mass energy of about $\sqrt{s} = 8\,\text{TeV}$ in Run-1 (2009–2013), the energy for LHC Run-2 (2015–2018) was already increased to $\sqrt{s} = 13\,\text{TeV}$. LHC Run-3 will start in the beginning of March 2022 and will feature a center-of-mass energy of $\sqrt{s} = 14\,\text{TeV}$.

2.2.1 Top-Quark Production and Decay

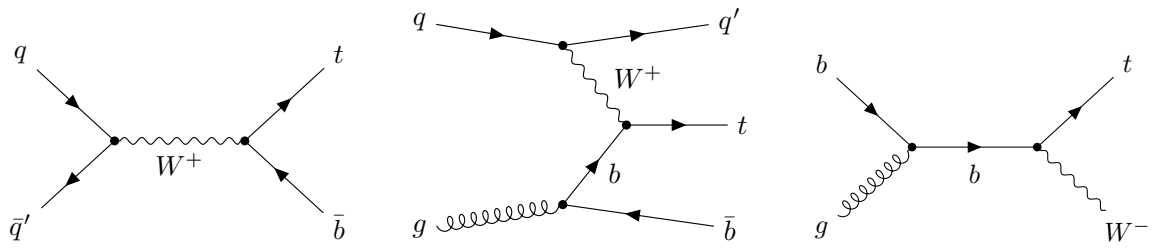
According to Ref. [25], in hadron collisions, top quarks are dominantly produced in $t\bar{t}$-pairs through quark-antiquark annihilation and gluon-gluon fusion via strong interactions. The corresponding Feynman diagrams can be seen in Figure 2.2. For example, at the LHC (proton-proton collisions at $\sqrt{s} = 14\,\text{TeV}$) about 90 % of the $t\bar{t}$-production cross section is from gluon-gluon fusion and 10 % is from quark-antiquark annihilation [25]. Feynman diagrams for the electroweak production of single top quarks are shown in Figure 2.3. However, in hadron collisions, these processes have much smaller cross sections [25].



(a) quark-antiquark annihilation (b) gluon-gluon fusion (s-channel) (c) gluon-gluon fusion (t-channel)

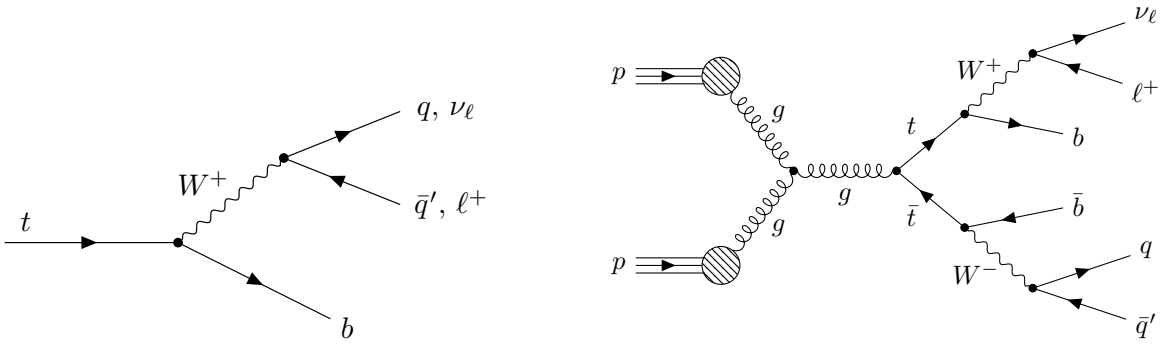
Figure 2.2: Leading order (LO) Feynman diagrams for the production of top-antitop pairs via strong interactions. Top-quark pair production is dominant in hadron collisions.

While all other quarks hadronize before they decay (i.e. they form bound states called baryons or mesons) [25, 27], the top quark decays before hadronization can occur [25]. This is because the top quark is the most massive elementary particle in the SM: For a top-quark mass of 173.3 GeV the decay rate is expected to be $\Gamma_t = 1.35\,\text{GeV}$ [25], which leads to a top-quark lifetime $\tau_t \approx 0.5 \times 10^{-24}\,\text{s}$ below the hadronization timescale. Hence, the top quark decays before it hadronizes, i.e. before top-flavored hadrons or $t\bar{t}$-quarkonium bound states can form [25].



(a) s-channel (b) t-channel (c) W-associated production (tW-channel)

Figure 2.3: Leading order (LO) Feynman diagrams for the electroweak production of single top quarks via $q\bar{q}' \to t\bar{b}$ and $qb \to q't$ (mediated by virtual s-channel and t-channel W-bosons), and through $gb \to tW^-$ in association with a W-boson [25].



(a) In the hadronic channel, two light-quark jets can be observed. In the leptonic decay, a lepton and a neutrino are produced.

(b) Production of a top-antitop ($t\bar{t}$) pair via gluon-gluon fusion after a proton-proton (pp) collision and semi-leptonic decay of the $t\bar{t}$-pair.

Figure 2.4: Leading order (LO) Feynman diagram for the electroweak decay of a single top quark via an on-shell W-boson (*left*) and example for the production and the semi-leptonic decay of a top-antitop pair (*right*). The light quarks in the final state radiate gluons and thus produce jets of hadrons [25].

Since the CKM[1] matrix magnitude $|V_{tb}|$, i.e. the probability for a top quark to decay into a bottom (b) quark, is estimated to be much larger than $|V_{ts}|$ and $|V_{td}|$, the dominant decay mode of the top quark is $t \to W^+b$. Top-quark decays into strange (s) or down (d) quarks are suppressed by the very small CKM matrix elements V_{ts} and V_{td} [25].

Depending on how the two W-bosons decay, the possible decay processes of top-antitop pairs are divided into three categories (with $\ell = e, \mu, \tau$, see Figure 2.4) [25]:

1. All-hadronic channel: Both W-bosons decay hadronically to light quarks.

$$t\bar{t} \longrightarrow (W^+b)(W^-\bar{b}) \longrightarrow (q\bar{q}'b)(q''\bar{q}'''b) \tag{2.8}$$

2. Semi-leptonic channel (lepton+jets): One of the two W-bosons decays leptonically to a charged lepton and a neutrino and the other one decays hadronically to two light quarks.

$$t\bar{t} \longrightarrow (W^+b)(W^-\bar{b}) \longrightarrow (q\bar{q}'b)(\ell^-\bar{\nu}_\ell\bar{b}) \tag{2.9a}$$
$$t\bar{t} \longrightarrow (W^+b)(W^-\bar{b}) \longrightarrow (\ell^+\nu_\ell b)(q\bar{q}'\bar{b}) \tag{2.9b}$$

[1]The Cabibbo-Kobayashi-Maskawa (CKM) matrix, or quark-mixing matrix, is a 3×3 unitary matrix describing the flavour-mixing between the three different families of quarks in the Standard Model (SM) of particle physics.

3. Dilepton channel: Both W-bosons decay leptonically to a charged lepton and a neutrino.

$$t\bar{t} \longrightarrow (W^+b)(W^-\bar{b}) \longrightarrow (\ell^+\nu_\ell b)(\ell'^-\bar{\nu}_{\ell'}\bar{b}) \tag{2.10}$$

Depending on the final states of the pair-production processes, there are different experimental signatures by which the $t\bar{t}$-decay can be identified: For example, two leptonically decaying top quarks can be identified through their decay leptons and missing transverse energy [25, 27]. Another way to identify top quarks is by looking for jets containing a bottom quark (b-tagging).

2.2.2 Tagging Boosted Hadronically Decaying Top Quarks

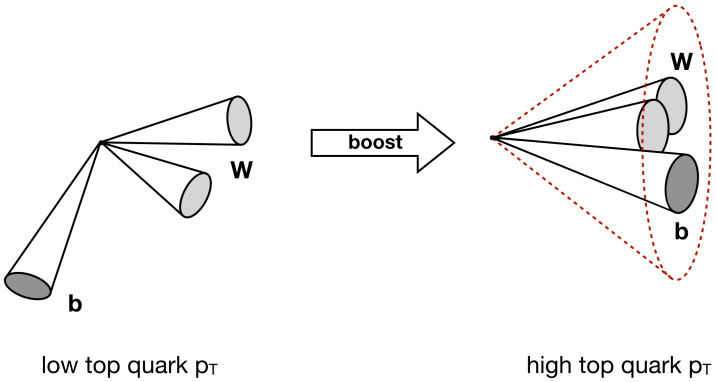


Figure 2.5: Visualization of a Lorentz-boosted, hadronically decaying top quark in the laboratory frame. For low-p_T top quarks (*left*), the decay products can be resolved as three distinct small-radius jets. For high-p_T top quarks (*right*), the particles are collimated in the top-quark flight direction and merge into one large-radius jet (fat jet) with a three-prong substructure [27, 28]. Taken from [29].

In order to define a jet from the energy deposition in a detector calorimeter, so-called jet clustering algorithms, like the Cambridge-Aachen (C/A) or the (anti-)k_T algorithms, are required. A detailed description of these jet algorithms can be found in Ref. [27].

At LHC energies, high-momentum particles can be produced, e.g. from the decay of heavy TeV-scale resonances [5]. As visualized in Figure 2.5, the decay products of Lorentz-boosted (high-p_T) top quarks are collimated and look like one single jet. Such jets with a large geometrical size are referred to as fat jets [27]. The difficulty in identifying boosted hadronically decaying particles, e.g. tagging high-p_T top quarks, is that the collimated decay products cannot be resolved as distinct jets by the calorimeter anymore [5, 26]. In order to identify boosted particle decays and to get information about the decay process, one has to study the internal structure of the fat jets. For example: Due to the top-quark decay process ($t \to W^+b \to q\bar{q}'b$), a top jet has three subjets, so-called prongs. This three-prong substructure can be used to distinguish top jets from light-quark and gluon jets [26]. Background events originating from light-quark and gluon jets are called *QCD background* because these jets can be fully described by the soft and collinear structure of quantum chromodynamics (QCD) [26, 27].

3 Machine Learning Background

In the following sections, we will introduce the most important terms and concepts of deep learning and neural network (NN) architectures. In particular, we use the same notation as in Ref. [30]. For further reading and detailed information on deep learning Ref. [31] is recommended.

In general, a machine learning (ML) procedure consists of a dataset, a model with trainable parameters, a loss function which defines the task, and an optimization algorithm that adapts the model parameters in order to minimize the loss function [31].

Typical machine learning tasks are classification problems, i.e. assigning the input samples to certain categories, and anomaly detection, i.e. finding unusual or atypical events in the dataset. Classification problems are usually associated with supervised learning algorithms: Based on the training samples and the corresponding ground truths (so-called targets or labels), the model learns to make predictions for the unseen test data samples [31]. In unsupervised learning algorithms, there are no labels. Instead, the model tries to learn useful properties or patterns in data (e.g. in order to recognize anomalies) [31].

3.1 Deep Learning and Neural Networks

The idea of neural networks is to take a linear model, called artificial neuron, and to connect many of these neurons in a directed graph (using non-linear activation functions), which we then call artificial neural network (ANN), or usually simply neural network (NN) [30]. The neurons are the nodes of this graph and are typically organized into layers.

Network parameters that are not adapted during the training algorithm, such as the number of hidden layers (depth of the network), the number of neurons per layer (width of each layer), or the choice of activation functions (connections of the layers), are called hyper-parameters and must be set before training the network [31].

3.1.1 Fully-Connected Feed-Forward Networks

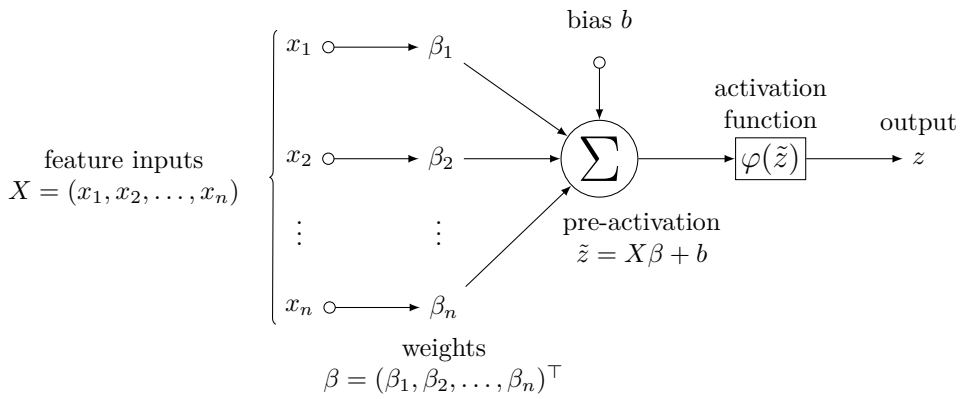


Figure 3.1: Basic structure of an artificial neuron. The neuron computes the weighted sum of the input features x_i and the corresponding weights β_i and then adds the bias b to this sum (pre-activation). Afterwards, the usually non-linear activation function φ is applied to the pre-activation. The output z is called activation.

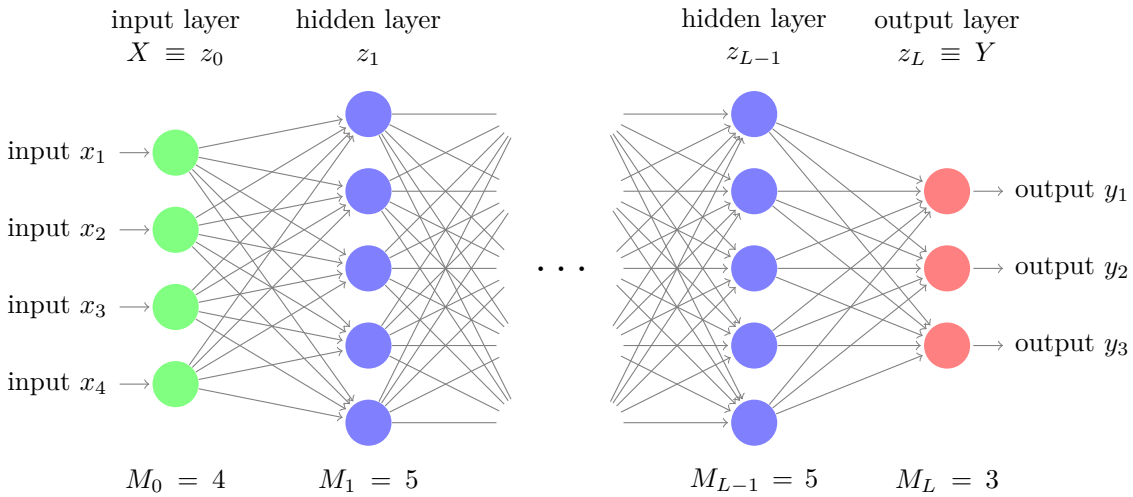


Figure 3.2: Schematic drawing of a feed-forward neural network composed of fully-connected layers (so-called dense layers). Layer $l = 0$ is the input layer (data $X \equiv z_0$), the final layer $l = L$ is called output layer (response $Y \equiv z_L$) and the layers in between $(1 \leq l \leq L-1)$ are called hidden layers.

Figure 3.1 illustrates the basic structure of an artificial neuron. Mathematically, the functionality can be expressed as follows:

$$z = \varphi(\tilde{z}) \qquad \text{with} \qquad \tilde{z} = X\beta + b = \sum_{i=1}^{n} x_i \beta_i + b \tag{3.1}$$

Here, the feature inputs $X = (x_1, x_2, \ldots, x_n)$ are understood as a row vector and the corresponding weights $\beta = (\beta_1, \beta_2, \ldots, \beta_n)^\top$ as a column vector. The affine transformation $\tilde{z} = X\beta + b$ of the feature inputs X is called pre-activation. After this linear operation, the activation function φ is applied to the pre-activation $\tilde{z}$. Activation functions will be discussed in more detail in the next section. Usually, the bias b is absorbed into the weight vector as $\beta_0 \equiv b$. $x_0 \equiv 1$ is then called the bias neuron.

An ANN which consists of many layers $l = 0, \ldots, L$ of neurons (typically $L \geq 3$) is called deep neural network (DNN). Neurons in a given layer work in parallel and neurons in subsequent layers work in series. Therefore, we call such networks feed-forward architectures. Figure 3.2 shows a schematic sketch of a fully-connected network (FCN). Fully-connected means that each neuron is connected to (i.e. receives input from) every neuron of the previous layer.

With $m = 0, \ldots, M_l$ we denote the indices of the neurons in layer l, i.e. layer l consists of M_l neurons ($m = 0$ is the bias neuron). The output vector (activation) of layer l is represented by $z_l \in \mathbb{R}^{1 \times M_l}$ (row vector). The action of layer $l \geq 1$ in matrix notation is given by:

$$z_l = \varphi_l(\tilde{z}_l) \qquad \text{with} \qquad \tilde{z}_l = [1, z_{l-1}] \cdot B_l \quad , \tag{3.2}$$

where $B_l \in \mathbb{R}^{(M_{l-1}+1) \times M_l}$ denotes the matrix of weights in layer l (the bias is absorbed into B_l via the bias neuron) and φ_l the activation function in layer l. Before applying the activation function, a weighted sum of the activations z_{l-1} from the previous layer is calculated (pre-activation). Then, the activation function φ_l is applied element-wise to the pre-activations $\tilde{z}_l$. Using this notation, the formula for a single neuron is given by $z_{lm} = \varphi_l([1, z_{l-1}] \cdot B_{lm})$, where B_{lm} is the weight vector of neuron m in layer l. For a feed-forward network, the response Y is defined recursively:

$$Y \equiv z_L = \varphi_L([1, \varphi_{L-1}(\ldots)] \cdot B_L) \tag{3.3}$$

3.1.2 Activation Functions

If the activation functions φ_l are a linear mapping then any network (with depth $L > 1$) is equivalent to a single-layer neural network, since the composition of linear functions is a linear function itself. Therefore, all layers of the neural network would collapse into one layer.

In deep learning, however, the purpose of activation functions is to introduce non-linearity to a neural network. Non-linear activations give the network the ability to learn complex patterns in the data and non-linear relations between the feature inputs and the desired output [31].

Historically important, for example, are the sigmoid and the hyperbolic tangent (tanh) functions. But due to their vanishing gradient for $x \to \pm\infty$, they are not suitable for learning: neurons can get stuck in an always-firing or an always-zero state [31].

Nowadays, the rectified linear unit (ReLU), and the existing variants of it like leaky ReLU or parametric ReLU, are the most common activation functions:

$$\mathrm{ReLU}(x) \coloneqq \max\{0, x\} = \begin{cases} x & \text{for } x > 0 \\ 0 & \text{for } x \leq 0 \end{cases} \tag{3.4}$$

Since the gradient of the ReLU function vanishes for negative x, dead neurons, i.e. neurons in the always-zero state, can still occur [31]. This issue can be avoided by using the leaky ReLU activation function:

$$\mathrm{LeakyReLU}(x) \coloneqq \begin{cases} x & \text{for } x > 0 \\ \alpha x & \text{for } x \leq 0 \end{cases} , \tag{3.5}$$

where $\alpha \ll 1$ is a hyper-parameter (not trainable). Parametric ReLU (PReLU) is basically the same as leaky ReLU, but here α is a learnable parameter, not a hyper-parameter [30].

The output activation function depends on the application of the neural network. The sigmoid function, for example, is often used for binary classification tasks (i.e. classification problems with just two classes):

$$\mathrm{sigmoid}(x) \equiv \sigma(x) \coloneqq \frac{1}{1 + \exp(-x)} \tag{3.6}$$

For classification problems with multiple classes, the softmax activation function can be used for the output neurons:

$$\mathrm{softmax}(\boldsymbol{x})_i \coloneqq \frac{\exp(x_i)}{\sum_{j=1}^{n} \exp(x_j)} , \tag{3.7}$$

with $i = 1, \ldots, n$ and $\boldsymbol{x} = (x_1, \ldots, x_n) \in \mathbb{R}^n$. The softmax function normalizes the n output values of the network in order to obtain a probability distribution over the output classes [31].

3.1.3 Training Deep Neural Networks

Due to memory constraints, large training datasets are usually divided into smaller subsets, called batches. The batch size describes the number of instances per batch. The term epoch refers to one loop over the entire dataset.

Loss Function. The loss function $\mathcal{L}$, or cost function, defines the target of an optimization problem and is therefore used to measure the performance of the model [31]. During training, the trainable model parameters, i.e. the network weights β (including the biases), are adjusted in order to minimize the loss function [31].

For our linear classifier test (top jets vs. QCD jets, binary classification problem), for example, we use the sigmoid activation on the output layer and the binary cross-entropy (BCE) loss as implemented in `PyTorch`. The BCE loss for a batch with N_{batch} instances is given by:

$$\mathcal{L}_{\text{BCE}} = \frac{1}{N_{\text{batch}}} \sum_{i=1}^{N_{\text{batch}}} -y_i \log(\hat{y}_i) - (1 - y_i) \log(1 - \hat{y}_i) \quad , \tag{3.8}$$

where $\hat{y}_i \in [0, 1]$ denotes the model prediction for the i-th sample and $y_i \in \{0, 1\}$ the corresponding ground truth label. The contrastive loss function will be discussed in Chapter 5.

Backpropagation and Optimization. As already mentioned, the training goal is to minimize the loss function $\mathcal{L}$ by adjusting the model parameters β. The gradient of the loss (with respect to the parameters), $\nabla_\beta \mathcal{L}$, can be computed efficiently using the back-propagation algorithm. An optimization algorithm then uses the gradient $\nabla_\beta \mathcal{L}$ to adjust the model parameters β towards the direction of steepest descent to minimize the loss function $\mathcal{L}$ [30, 31].

Most optimization algorithms are based on gradient descent:

$$\beta^{(t+1)} = \beta^{(t)} - \tau_t l^{(t)} \tag{3.9}$$

Here, τ_t is the learning rate in iteration t and $l^{(t)}$ denotes the corresponding loss gradient. The learning rate controls how much the weights are adjusted in one iteration. Stochastic gradient descent (SGD) computes the gradient $l^{(t)}$ from a random instance of the batch for every iteration, which is computationally efficient. It should be noted, however, that stochastic gradient methods require a decreasing learning rate in order to converge [30]. Examples for important optimizer that use adaptive gradient descent algorithms are `Adam` (derived from *adaptive moment estimation*) and `AdamW` (`Adam` algorithm with weight decay).

3.2 Self-Attention and Transformer-Encoder Networks

The transformer network is a sequence-to-sequence architecture, consisting of an encoder and a decoder, which entirely relies on an attention mechanism [32]. Originally, the transformer network was introduced for neural machine translation [33, 34].

Since we do not want to construct new sequences or reconstruct the inputs, we only use the encoder part of the transformer architecture described in Ref. [32] to obtain the representations. In addition, we do not use position encoding. The building blocks of the transformer-encoder network illustrated in Figure 3.4 are scaled dot-product multi-headed self-attention units and feed-forward layers [32].

Scaled Dot-Product Self-Attention. Figure 3.3 illustrates the scaled dot-product self-attention mechanism (single-headed) applied to a single constituent [6]. The attention weights a_i in Figure 3.3 basically describe how the first constituent $\boldsymbol{x}_1$ in the sequence (represented by the query $\boldsymbol{q}_1$) is connected to all the other constituents in the sequence (represented by the keys $\boldsymbol{k}_i$). Such an attention mechanism allows it to place more attention on sequence elements with high weights [6]. In practice, the output for a set of queries is computed simultaneously [32]:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d}}\right)V \quad , \tag{3.10}$$

where the matrix Q contains the queries, K the keys and V the values.

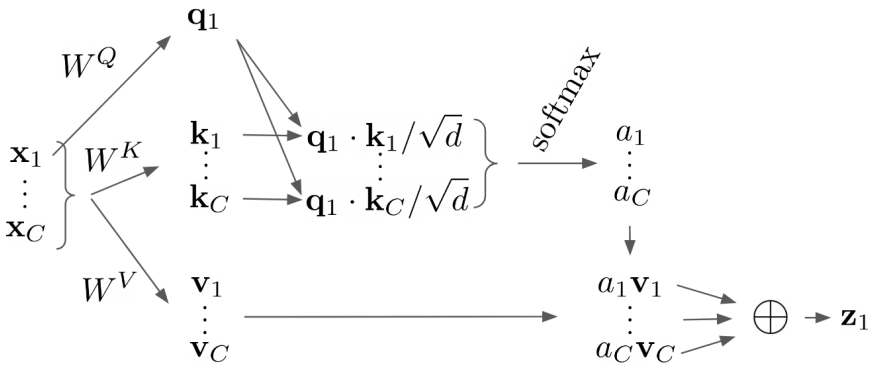


Figure 3.3: Illustration of the scaled dot-product self-attention mechanism (single-headed) applied to the single constituent embedding $\boldsymbol{x}_1$. For each constituent $i = 1, \ldots, C$ the input embedding $\boldsymbol{x}_i$ is multiplied with each of the three learnable weight matrices W^Q, W^K and W^V to obtain the query vector $\boldsymbol{q}_i = \boldsymbol{x}_i W^Q$, the key vector $\boldsymbol{k}_i = \boldsymbol{x}_i W^K$, and the value vector $\boldsymbol{v}_i = \boldsymbol{x}_i W^V$. The attention weights a_i for constituent $\boldsymbol{x}_1$ are calculated using the dot product between the query $\boldsymbol{q}_1$ and the keys $\boldsymbol{k}_i$ divided by $\sqrt{d}$ (for stability reasons), where d is the dimension of $\boldsymbol{q}_1$ and $\boldsymbol{k}_i$. The softmax operation normalizes the weights. Finally, the output $\boldsymbol{z}_1$ is computed as the sum of the value vectors $\boldsymbol{v}_i$ weighted with the corresponding attention weights a_i. Taken from [6].

Multi-Headed Self-Attention (MHSA). The idea of multi-headed attention is to perform several of the single-headed operations in parallel [6, 32]. The d_v-dimensional outputs of the h heads are then concatenated and projected to the output using a linear layer $W^O \in \mathbb{R}^{hd_v \times d_{\text{model}}}$:

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \ldots, \text{head}_h)\, W^O \tag{3.11a}$$

$$\text{with} \qquad \text{head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V) \quad , \tag{3.11b}$$

where $W_i^Q, W_i^K \in \mathbb{R}^{d_{\text{model}} \times d}$ and $W_i^V \in \mathbb{R}^{d_{\text{model}} \times d_v}$ are the learnable weight matrices in head i.

Transformer-Encoder Architecture. The architecture of a transformer-encoder network is illustrated in Figure 3.4. By summing the transformer-encoder output along the constituent dimension, the permutation equivariant model (i.e. the encoder operation commutes with permutations) becomes invariant to permutations of the constituents [6]. The head network is added to provide additional representational power [6].

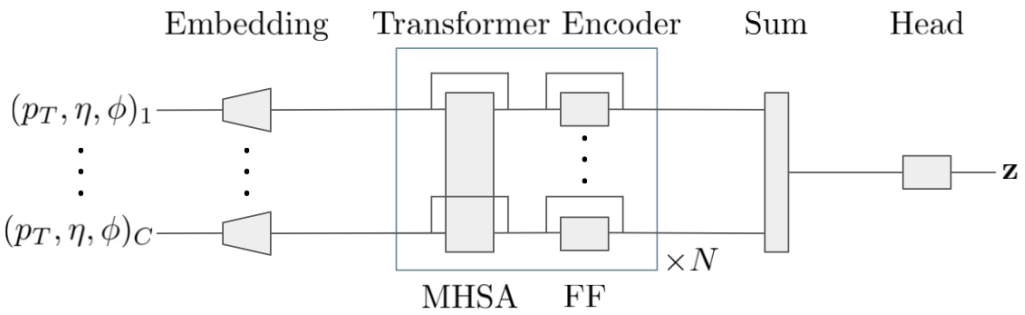


Figure 3.4: Illustration of the transformer-encoder network architecture. Using a learned linear embedding network, the input constituents are first mapped into a higher-dimensional space (to increase the representational power of the network). The transformer-encoder part consists of N blocks of multi-headed self-attention (MHSA) mechanisms, followed by layer normalization, and feed-forward (FF) layers. The outputs of the transformer-encoder part are summed along the constituent dimension to produce a fixed size output, which is finally passed through a feed-forward network (head) to obtain the representation $\boldsymbol{z}$ [6]. Taken from [6].

3.3 Performance Measure for Binary Classification Problems

For binary classification problems, the one-dimensional model output, the so-called classification score, is compared to a threshold value to get a class label prediction. Receiver operating characteristic (ROC) curves are used to measure the classification performance of such a binary classifier at different thresholds. To plot a ROC curve, one needs the following two parameters:

- True-positive rate (TPR): the proportion of signal events that were correctly identified as signals by the classifier (signal efficiency ϵ_s).
- False-positive rate (FPR): the proportion of background events that were incorrectly identified as signals by the classifier (background mistag rate ϵ_b).

In a standard ROC curve (ϵ_b vs. ϵ_s), the true-positive rate (signal efficiency ϵ_s) is plotted against the false-positive rate (background mistag rate ϵ_b) for various classification thresholds. In an inverse ROC curve (ϵ_b^{-1} vs. ϵ_s), the inverse background mistag rate (ϵ_b^{-1}) is plotted against the signal efficiency (ϵ_s). The AUC value characterizes the area under the ROC curve and is invariant to the classification threshold. An AUC value of 0.5 indicates that the model randomly assigns class labels to the samples. Generally, the higher the AUC, the better the classification performance.

4 Jet Representations and Observables

As previously mentioned, one has to examine the internal structure of a fat jet in order to study and identify the decay processes of boosted particles [19]. The difficulty is to find observables, also called representations, that are suitable to describe the jet substructure information. There are many jet substructure observables that allow to study the radiation patterns and particle distributions inside fat jets. However, not every jet representation is suitable for every tagging problem, i.e. the choice of observables is not model-independent.

In this chapter, we give a general overview of the most important jet representations, namely jet images [16, 17, 18] and energy flow polynomials (EFPs) [19]. Other jet representations are, for example, N-subjettiness [35], graphs [36], trees or point clouds, but we will not go into detail on them.

4.1 Image-Based Approach: Calorimeter Images

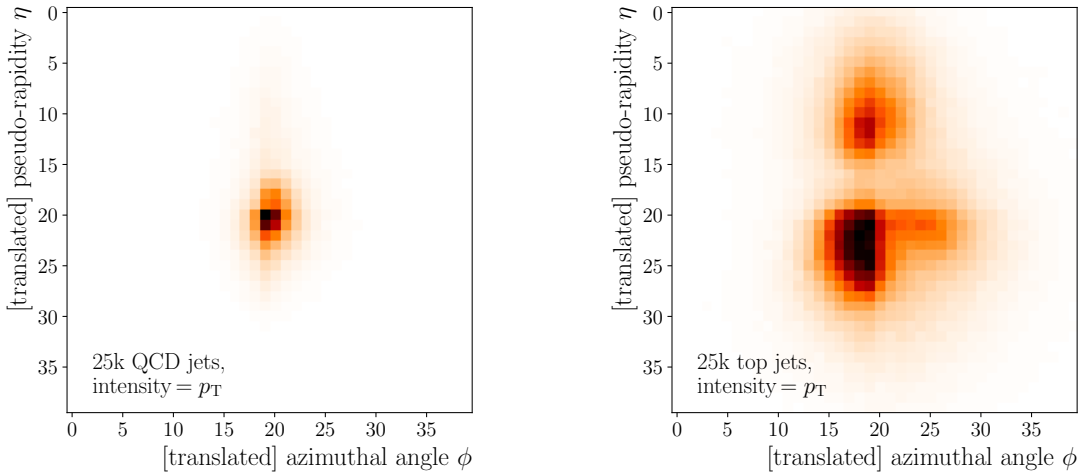
A calorimeter image, or jet image, is a two-dimensional representation of the energy distribution in the calorimeter detector [16, 14]. The image pixels represent the calorimeter towers (position in the η-ϕ plane) and the intensity typically describes the transverse momenta p_T of the particles.

After computing the $(p_\mathrm{T}, \eta, \phi)$ values for each jet constituent from the corresponding four-momenta (taking into account the periodicity in the azimuthal angle direction), the following pre-processing steps must be performed to generate a standardized and consistent jet-image representation [16, 37, 38]:

1. Centering: First, the jet constituents are translated such that the p_T-weighted centroid of the jet is at the origin in the η-ϕ plane. It should be noted that shifting the constituents in η corresponds to a Lorentz-boost along the z-axis.
2. Rotation: The jet is rotated such that the principle axis points in the direction of increasing pseudo-rapidity η, i.e. the principle axis is vertically aligned.
3. Flipping: Next, the jets are flipped along both axes such that the maximum sum of p_T is in the upper right quadrant.
4. Pixelization, cropping and normalization: The jets are pixelized in the η-ϕ plane and the images are cropped to the desired size (usually 40×40 pixels). Finally, the intensity of each pixel is divided by the total intensity.

The jet-image representation is not rotationally invariant, but centering, rotating and flipping the jets reduces the impact of not having rotational invariance [6]. Pixelizing the jets in the η-ϕ plane avoids an implicit ordering of the jet constituents. However, the exact pre-processing steps may differ depending on the classes of jets that are studied, which makes the jet-image representation model-dependent.

Figure 4.1 shows the averaged signal (top jets) and background (QCD jets) images from 25 000 individual images after pre-processing. Well-established machine learning tools and techniques from the field of computer vision, such as convolutional neural networks (CNNs), can be applied to jet images [16, 17, 18]. Just as facial recognition models try to learn the expected distribution of pixel intensities in order to classifiy a face in an image [16], image-based taggers use jet images to learn the expected p_T-distribution in the η-ϕ plane. Image-based taggers, that use calorimeter information in the form of jet images to classify jets, have already achieved good results [14].



(a) Averaged QCD jet images

(b) Averaged top jet images

Figure 4.1: Background QCD jet and signal top jet images averaged over 25 000 individual images. The pixel position denotes the location in the η-ϕ plane and the color corresponds to the p_T-magnitude. The averaged QCD jet image (*left*) shows a uniform distribution of particles inside the jets. In the averaged top jet image (*right*), the three-prong substructure of the fat top jet is visible (one subjet for each decay product) [37].

4.2 Theory-Inspired Approach: Energy Flow Polynomials

Based on the fact that any infrared- and collinear-safe (IRC-safe)[1] observable can be written as a linear combination of C-correlators

$$\mathcal{C}_N^{f_N} = \sum_{i_1=1}^{M} \cdots \sum_{i_N=1}^{M} E_{i_1} \cdots E_{i_N} \, f_N(\hat{p}_{i_1}^{\mu}, \ldots, \hat{p}_{i_N}^{\mu}) \quad , \tag{4.1}$$

one can expand the angular weighting function f_N (which is only a function of the particle directions and not of their energies E_i) in terms of a discrete set of polynomials in pairwise angular distances [19]. This expansion leads to the so-called energy flow polynomials (EFPs), a (over-)complete discrete linear basis for IRC-safe jet substructure observables [19]. The set of EFPs is called the energy flow basis. According to Ref. [19], the corresponding EFP for a multigraph G with N vertices and edges $(k, l) \in G$ takes the form:

$$\mathrm{EFP}_G = \sum_{i_1=1}^{M} \cdots \sum_{i_N=1}^{M} z_{i_1} \cdots z_{i_N} \prod_{(k,l)\in G} \theta_{i_k i_l} \tag{4.2}$$

Here, M denotes the number of particles in the jet, $z_i = E_i/E_J$ (with $E_J \equiv \sum_{i=1}^{M} E_i$) is the energy fraction carried by particle i, and θ_{ij} is the angular distance between particles i and j. By normalizing the particle energies, the EFPs become independent on the overall jet kinematics and can therefore be used for jet substructure studies [19].

The energy and angular measures (E_i and θ_{ij}) are collider-dependent: For electron-positron ($e^- e^+$) collisions, energy and spherical (θ, ϕ) coordinates are used, but for hadronic collisions we

[1] Collinear safety means that the observable is insensitive to the splitting of a hard particle into two collinear particles and infrared safety means that the observable is insensitive to low-energy modifications, such as the emission of a soft gluon.

use the particle's transverse momenta p_T and rapidity-azimuth (y, ϕ) coordinates:

$$z_i = \frac{p_{\mathrm{T},i}}{p_{\mathrm{T},J}} \qquad \text{with} \qquad p_{\mathrm{T},J} \equiv \sum_{i=1}^{M} p_{\mathrm{T},i} \tag{4.3a}$$

$$\theta_{ij} = \left(\Delta y_{ij}^2 + \Delta\phi_{ij}^2\right)^{\beta/2} \quad , \tag{4.3b}$$

where $\Delta y_{ij} \equiv y_i - y_j$ and $\Delta\phi_{ij} \equiv \phi_i - \phi_j$ are determined by the rapidity y_i and the azimuthal angle ϕ_i of particle i [19]. The above choice of measure for hadronic collisions is rotationally-symmetric in the y-ϕ plane and respects Lorentz-boosts along the beam axis [19]. The choice of the exponent β impacts the convergence of the EFP expansion [19].

In the space of jet substructure, the EFPs form a (over-)complete discrete linear basis for all IRC-safe observables [19], i.e. any IRC-safe observable $\mathcal{S}$ can be linearly approximated by EFPs:

$$\mathcal{S} \simeq \sum_{G \in \mathcal{G}} s_G \, \mathrm{EFP}_G \quad , \tag{4.4}$$

where $\mathcal{G}$ is some finite set of multigraphs and s_G are some real coefficients. For example, using an appropriate choice of measure, the jet mass and the energy correlation functions can be written as finite linear combinations of EFPs [19].

Instead of calculating the EFPs with the naive complexity of $\mathcal{O}(M^N)$ (N nested sums over M particles, see Equation (4.2)), one can use their graph-theoretic representation, i.e. their correspondence to non-isomorphic multigraphs, to reduce the computational complexity [19].

The EFP framework allows to study jet substructures using linear methods: Komiske et al. show that linear classification with EFPs on three representative jet-tagging problems (quark/gluon discrimination, boosted W-tagging, and boosted top-tagging) performs comparably to well-established machine learning techniques, such as jet images with CNNs [19].

5 Contrastive Learning of Representations

In this chapter, we first introduce the basics of contrastive representation learning by looking at the SimCLR framework (*a simple framework for contrastive learning of visual representations*) presented by the Google Brain team in Ref. [9]. Self-supervised tools for contrastive learning of representations (CLR), such as SimCLR [9], TCLR [10], SoundCLR [11], or MolCLR [12], have shown great success in learning representations by outperforming previous work using rather simple network architectures [9, 39, 40].

The basic structure of the SimCLR framework is shown in Figure 5.1 and described in Algorithm 1. For each image sample $\boldsymbol{x}_i$, two randomly augmented versions $\tilde{\boldsymbol{x}}_i$ and $\tilde{\boldsymbol{x}}_j$ are created (through cropping/resizing, random colour distortions, and Gaussian blur) [9]. A pair $\tilde{\boldsymbol{x}}_i$ and $\tilde{\boldsymbol{x}}_j$ of similar examples created by different transformations (augmentations) of the same data instance $\boldsymbol{x}$ is called a positive pair. A positive pair provides two correlated views of the same example [9]. Views generated from different instances are denoted as negative pairs.

The two augmented versions $\tilde{\boldsymbol{x}}_i$ and $\tilde{\boldsymbol{x}}_j$ are passed through a residual neural network (ResNet) encoder, denoted by f, to get vector representations $\boldsymbol{h}_i$ and $\boldsymbol{h}_j$ (which are used for classification). $\boldsymbol{h}_i$ and $\boldsymbol{h}_j$ are then passed through another smaller network g (a FCN with one hidden layer and ReLU activations), called projection head, in order to obtain $\boldsymbol{z}_i$ and $\boldsymbol{z}_j$. Finally, the contrastive loss function (see next section) is applied to $\boldsymbol{z}_i$ and $\boldsymbol{z}_j$.

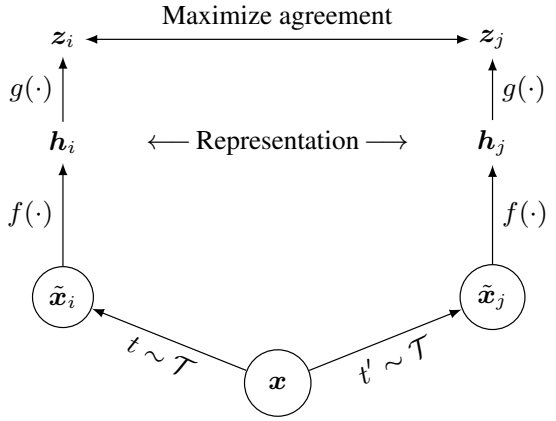


Figure 5.1: Illustration of the self-supervised SimCLR contrastive learning framework for visual representations. SimCLR learns representations from unlabeled data by maximizing the similarities of randomly augmented views $\boldsymbol{z}_i$ and $\boldsymbol{z}_j$ of the same instance $\boldsymbol{x}$ (pull together similar inputs, positive pair) and minimizing the similarities of all different instances (push apart dissimilar inputs, negative pairs) via a contrastive loss in the latent space [9, 40]. Taken from [9].

The idea of contrastive loss is to construct informative representations, which are invariant to the applied transformations, by contrasting feature vectors from positive pairs (similar samples) against feature vectors from negative pairs (dissimilar samples). This means that for positive pairs, the learned feature representations should be close together in latent space, while pushing apart the representations of negative pairs [9, 39, 40].

According to Ref. [9], the use of a non-linear projection head improves downstream task performance (due to a loss of information induced by the contrastive loss) [9]. Therefore, the contrastive loss is defined on $\boldsymbol{z}_i$ and $\boldsymbol{z}_j$, but $\boldsymbol{h}_i$ and $\boldsymbol{h}_j$ are used for downstream tasks.

Algorithm 1: SimCLR's main learning algorithm

Input: batch size N, constant τ, structure of f, g, $\mathcal{T}$
for sampled mini-batch $\{\boldsymbol{x}_k\}_{k=1}^{N}$ **do**
  forall $k \in \{1, \dots, N\}$ **do**
    draw two augmentation functions $t \sim \mathcal{T}$, $t' \sim \mathcal{T}$
    # the first augmentation
    $\tilde{\boldsymbol{x}}_{2k-1} = t(\boldsymbol{x}_k)$
    $\boldsymbol{h}_{2k-1} = f(\tilde{\boldsymbol{x}}_{2k-1})$ # representation
    $\boldsymbol{z}_{2k-1} = g(\boldsymbol{h}_{2k-1})$ # projection
    # the second augmentation
    $\tilde{\boldsymbol{x}}_{2k} = t'(\boldsymbol{x}_k)$
    $\boldsymbol{h}_{2k} = f(\tilde{\boldsymbol{x}}_{2k})$ # representation
    $\boldsymbol{z}_{2k} = g(\boldsymbol{h}_{2k})$ # projection
  end
  forall $i \in \{1, \dots, 2N\}$ and $j \in \{1, \dots, 2N\}$ **do**
    $s_{i,j} = \boldsymbol{z}_i^\top \boldsymbol{z}_j / (\|\boldsymbol{z}_i\| \|\boldsymbol{z}_j\|)$ # pairwise similarity
  end
  define $\ell(i, j)$ **as** $\ell(i, j) = -\log \frac{\exp(s_{i,j}/\tau)}{\sum_{k=1}^{2N} \mathbb{1}_{[k \neq i]} \exp(s_{i,j}/\tau)}$
  $\mathcal{L} = \frac{1}{2N} \sum_{k=1}^{N} [\ell(2k-1, 2k) + \ell(2k, 2k-1)]$
  update networks f and g to minimize $\mathcal{L}$
end
return encoder network $f(\cdot)$, and throw away $g(\cdot)$

5.1 Contrastive Loss Function

The contrastive learning task in the SimCLR framework is defined by the normalized temperature-scaled cross-entropy (NT-Xent) loss, simply called contrastive loss, which is for a single positive pair (i, j) given by

$$\ell(i, j) = -\log \left\{ \frac{\exp[\text{sim}(\boldsymbol{z}_i, \boldsymbol{z}_j)/\tau]}{\sum_{k=1}^{2N} \mathbb{1}_{[k \neq i]} \exp[\text{sim}(\boldsymbol{z}_i, \boldsymbol{z}_k)/\tau]} \right\} \quad , \tag{5.1}$$

where $\boldsymbol{z}_i$ and $\boldsymbol{z}_j$ denote latent vectors extracted from a positive pair, N is the batch size, and $\tau > 0$ is a scalar hyper-parameter, called temperature. $\mathbb{1}_{[k \neq i]} \in \{0, 1\}$ is an indicator function that is defined as:

$$\mathbb{1}_{[k \neq i]} = \begin{cases} 1 & \text{for } k \neq i \\ 0 & \text{otherwise} \end{cases} \tag{5.2}$$

The similarity between two vectors $\boldsymbol{z}_i$ and $\boldsymbol{z}_j$ is measured by the cosine-similarity:

$$\text{sim}(\boldsymbol{z}_i, \boldsymbol{z}_j) \coloneqq \frac{\boldsymbol{z}_i \cdot \boldsymbol{z}_j}{\|\boldsymbol{z}_i\| \|\boldsymbol{z}_j\|} \tag{5.3}$$

The total loss $\mathcal{L}$ is computed across all positive pairs (i, j) and (j, i) in the batch:

$$\mathcal{L} = \frac{1}{2N} \sum_{k=1}^{N} [\ell(2k-1, 2k) + \ell(2k, 2k-1)] \tag{5.4}$$

Obviously, maximizing the similarity between similar samples (numerator) and minimizing the

similarity of dissimilar samples (denominator) reduces the contrastive loss in Equation (5.1). By forcing small distances between positive pairs, the representations become (approximately) invariant to the applied augmentations, and by forcing large distances between negative pairs, the model learns to extract non-trivial discriminative features.

Representation Learning on the Unit Hypersphere. Many representation learning approaches use the unit hypersphere as the feature space by normalizing the feature vectors using the ℓ_2 norm. Hyperspherical latent spaces are known to provide better representations than the Euclidean space [39]. Using the unit hypersphere not only ensures better computation stability [39] but also has manifold mapping reasons: Well-clustered classes on the hypersphere can be separated linearly, but this does not apply to Euclidean spaces [39]. Furthermore, without normalization, the contrastive loss can be made arbitrarily small by simply scaling the feature vectors.

Temperature, Alignment and Uniformity. In this subsection, we discuss the alignment and uniformity properties of the contrastive loss function to get an understanding of what exactly the contrastive learning framework does and how contrastive representation learning relates to hypersphere geometry. As proposed in Ref. [39], we introduce theoretically-motivated and quantifiable metrics for both properties, which can be used to measure the representation quality. It can be shown that there is a strong connection between the alignment and uniformity metrics and the downstream task performance [39, 40].

The representations obtained using CLR should have the following properties [39, 40]:

- Alignment means that the latent vectors of a positive pair should be close to each other in latent space (similar samples should have similar features), making the representations invariant to the augmentations.
- Uniformity means that feature vectors should be uniformly distributed on the unit hypersphere to preserve maximal information of the data and to learn separable features.

As described in Ref. [40], the temperature τ in Equation (5.1) is a crucial hyper-parameter in order to compromise between learning separable features (uniformly distributed features) and invariance to the augmentations (alignment). This is called the uniformity-tolerance dilemma.

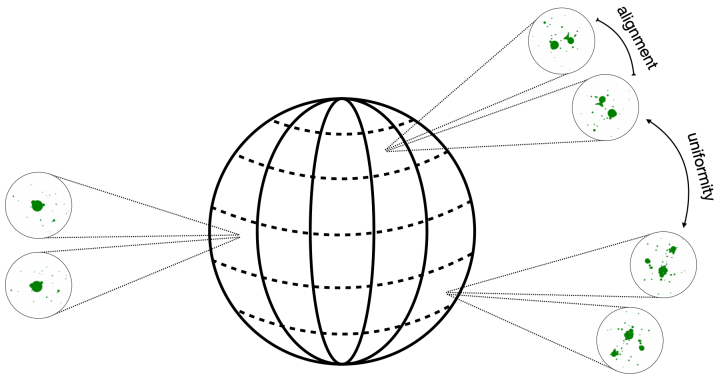


Figure 5.2: Visualization of alignment and uniformity in the latent space (unit hypersphere). Alignment describes the closeness of features from positive pairs and uniformity describes how uniform the distribution of the (normalized) features on the unit hypersphere is [39, 40]. Taken from [6].

In the following, p_{data} denotes the data distribution over $\mathbb{R}^n$, p_{pos} the distribution of positive pairs over $\mathbb{R}^n \times \mathbb{R}^n$, and f the network mapping. The unit hypersphere is denoted by $\mathcal{S}$. The alignment loss is then defined by the expected distance between positive pairs:

$$\mathcal{L}_{\mathsf{align}}(f;\alpha) \coloneqq \mathbb{E}_{(x,y)\sim p_{\mathsf{pos}}}\left[\|f(x)-f(y)\|_2^\alpha\right] \ , \quad \alpha > 0 \tag{5.5}$$

Using the Gaussian potential kernel

$$G_t \colon \mathcal{S}^d \times \mathcal{S}^d \to \mathbb{R}_+ \qquad \text{with} \qquad G_t(u,v) \coloneqq \mathrm{e}^{-t\|u-v\|_2^2} \ , \quad t > 0 \tag{5.6}$$

one can define the uniformity loss as the logarithm of average pairwise feature potentials:

$$\mathcal{L}_{\mathsf{uniform}}(f;t) \coloneqq \log \mathbb{E}_{x,y \overset{\text{i.i.d.}}{\sim} p_{\mathsf{data}}}\left[G_t(x,y)\right] \tag{5.7a}$$

$$= \log \mathbb{E}_{x,y \overset{\text{i.i.d.}}{\sim} p_{\mathsf{data}}}\left[\mathrm{e}^{-t\|f(x)-f(y)\|_2^2}\right] \ , \quad t > 0 \tag{5.7b}$$

It can be shown that the uniform distribution is the unique distribution that minimize the expected pairwise Gaussian potential [39]. Obviously, pulling together similar samples reduces the alignment loss and pushing apart dissimilar samples results in a more uniform distribution and therefore reduces the uniformity loss.

Theoretically, the contrastive loss optimizes for alignment and uniformity asymptotically, i.e. in the limit of infinite negative samples [39]. But in practice we can only have a finite amount of negative samples. Therefore, direct optimization of a linear combination of the alignment loss ($\mathcal{L}_{\mathsf{align}}$) and the uniformity loss ($\mathcal{L}_{\mathsf{uniform}}$) outperforms CLR models that optimize only the NT-Xent loss (implying that directly optimizing $\mathcal{L}_{\mathsf{align}}$ and $\mathcal{L}_{\mathsf{uniform}}$ leads to comparable or even better representations) [39].

5.2 Contrastive Representation Learning for Jet Physics

Our goal is to use a self-supervised contrastive learning of representations (CLR) framework to define a mapping $f : \mathcal{J} \to \mathcal{R}$ between the low-level jet constituents space ($\mathcal{J}$) and some high-level representation space ($\mathcal{R}$). Since our low-level dataset uses the transverse momentum p_{T}, the pseudo-rapidity η and the azimuth ϕ to describe the (approximately) massless constituents, $\mathcal{J}$ has a dimensionality of $\dim(\mathcal{J}) = 3n_C$ where n_C is the number of constituents in the jet.

The training algorithm of our JetCLR method is as follows:

1. First, we randomly sample a batch $\{\boldsymbol{x}_i\}$ of N jets from the training dataset.
2. We then create an augmented version of each jet by applying stochastic data transformations (augmentations) such that we have two different but correlated views of the same jet. Hence, for each jet in the original batch $\{\boldsymbol{x}_i\}$ we get an augmented version in the augmented batch $\{\boldsymbol{x}'_i\}$. Similar examples $\{(\boldsymbol{x}_i, \boldsymbol{x}'_i)\}$ are called positive pairs and dissimilar examples $\{(\boldsymbol{x}_i, \boldsymbol{x}_j)\} \cup \{(\boldsymbol{x}_i, \boldsymbol{x}'_j)\}$ with $i \neq j$ are called negative pairs.
3. All $2N$ jets, i.e. the original batch $\{\boldsymbol{x}_i\}$ and the augmented batch $\{\boldsymbol{x}'_i\}$, are passed through the network to extract the representations. The jets $\boldsymbol{x}_i$ and their augmented versions $\boldsymbol{x}'_i$ are mapped to latent space vectors $\boldsymbol{z}_i$ and $\boldsymbol{z}'_i$ in $\mathbb{R}^{\dim(\boldsymbol{z})}$. By normalizing the latent space vectors, the representation space $\mathcal{R}$ is then defined as the unit hypersphere $\mathcal{S}^{\dim(\boldsymbol{z})-1}$ embedded in $\mathbb{R}^{\dim(\boldsymbol{z})}$.

The contrastive loss function is defined on positive pairs $(\boldsymbol{z}_i, \boldsymbol{z}_i')$ of augmented jets in the latent space. For a single positive pair i, the contrastive loss is defined as [9]:

$$\ell_i = -\log\left\{\frac{\exp[\mathrm{sim}(\boldsymbol{z}_i, \boldsymbol{z}_i')/\tau]}{\sum_{i\neq j\in\mathrm{batch}}\left(\exp[\mathrm{sim}(\boldsymbol{z}_i, \boldsymbol{z}_j)/\tau] + \exp[\mathrm{sim}(\boldsymbol{z}_i, \boldsymbol{z}_j')/\tau]\right)}\right\} \tag{5.8}$$

The total loss $\mathcal{L}$ is computed across all positive pairs in the batch. As proposed in Ref. [9], we use the cosine-similarity to measure the similarity between the jets in the representation space $\mathcal{R}$:

$$\mathrm{sim}(\boldsymbol{z}_i, \boldsymbol{z}_j) \coloneqq \frac{\boldsymbol{z}_i \cdot \boldsymbol{z}_j}{\|\boldsymbol{z}_i\|\|\boldsymbol{z}_j\|} = \cos\theta_{ij} \tag{5.9}$$

In order to compare different jets in the representation space $\mathcal{R}$, we have to define a proper distance metric. However, the cosine-similarity does not fulfill the metric properties. Instead, we can define an angular distance $d(\boldsymbol{z}_i, \boldsymbol{z}_j) = \theta_{ij}/\pi$ using the angle θ_{ij} between the jets in $\mathcal{R}$.

Symmetry Augmentations. There are two ways to learn (approximate) symmetries using CLR: On the one hand through model constraints (for example by using a permutation invariant transformer-encoder network) and on the other hand through symmetry transformations, so-called augmentations [6]. Since the NT-Xent loss in Equation (5.8) forces the network to map similar samples close together in latent space while pushing apart dissimilar samples [8], the representations will be (under perfect convergence) invariant to the jet augmentations we use to create the positive pairs [6].

6 Experiments and Results

In the following sections, we provide detailed information on the top-tagging dataset and on the networks we use to parameterize the mapping between the low-level jet constituents and the high-level observables. We describe our training and optimization details including the applied symmetry augmentations. All models were implemented using the standard deep learning framework `PyTorch`.

6.1 Data Simulation and Pre-Processing Steps

6.1.1 Top-Tagging Reference Dataset

The top-tagging reference dataset[1] we use to train and test our JetCLR framework was initially produced to evaluate and compare different top-tagging architectures [13, 14, 15]. It is therefore a well-established dataset to benchmark classification algorithms that distinguish signal top jets from background QCD jets.

The dataset consists of Monte-Carlo simulated hadronically decaying top quarks and QCD dijet samples. The signal top jets and the light-quark and gluon background jets were generated with Pythia [41] using the default center-of-mass energy of $\sqrt{s} = 14\,\text{TeV}$. Multiple parton interactions (MPIs) and pile-up were ignored.

Delphes [42] was used with its standard ATLAS detector card to simulate the calorimeter detector. The particle-flow entries were clustered into fat jets using the anti-k_T algorithm [43] implemented in FastJet [44] with a radius of $R = 0.8$. The transverse momenta p_T of the jet constituents are restricted to $p_\text{T} \in [550, 650]$. Besides, the jets are required to have $|\eta_\text{jet}| < 2$. Additionally, all top jets are matched to a parton-level top within $\Delta R = 0.8$ and to all top decay partons within 0.8.

The four-momenta of the leading 200 constituents of each jet are stored. The constituents are ordered by decreasing transverse momentum p_T. Jets with fewer than 200 constituents are zero-padded. Moreover, each jet is flagged with a signal or background class label. Additional details such as particle or tracking information are not included [14].

6.1.2 Pre-Processing and Symmetry Augmentations

We use a minimally pre-processed set of low-level jet data: For each jet constituent, we calculate the transverse momentum p_T, the pseudo-rapidity η, and the azimuthal angle ϕ. Each jet is then represented as a flattened, p_T-ordered list of its constituents with (p_T, η, ϕ) information, in the approximation of massless constituents. The only pre-processing step we make is to shift the constituents such that the jet axis, defined by the p_T-weighted centroid of the jet constituents

$$(\eta_0, \phi_0) = \left(\sum_{i \in \text{ jet}} \eta_i \, p_{\text{T},i} \, , \sum_{i \in \text{ jet}} \phi_i \, p_{\text{T},i} \right) \quad , \tag{6.1}$$

is aligned at $(\eta_0, \phi_0) = (0, 0)$. The augmentations applied to the jet constituents are crucial to which symmetries the representations will be (approximately) invariant to. In the following we present our symmetry augmentations and network constraints.

[1] More information is available at https://docs.google.com/document/d/1Hcuc6LBxZNX16zjEGeq16DAzsp-kDC4nDTyjMp1bWHRo/edit. The public dataset can be downloaded from https://desycloud.desy.de/index.php/s/llbX3zpLhazgPJ6.

Rotations in the η-ϕ Plane. To achieve rotational symmetry around the jet axis we create augmented jets by rotating each jet by a different angle randomly sampled from the interval $[0, 2\pi]$. It should be noted that rotations in the η-ϕ plane do not preserve the jet mass because they are no Lorentz transformations, but for narrow jets, i.e. $R \lesssim 1$, the relative jet mass corrections are very small and can therefore be neglected.

Translations in the η-ϕ Plane. In addition to rotations in the η-ϕ plane, translations in the η-ϕ plane would also be an obvious choice for augmentations. We tried using linear transformations in the η-ϕ plane as augmentations, but that did not produce good results due to technical reasons: Jets have a variable number of constituents and since fully-connected networks (FCNs), for example, cannot handle a variable number of input dimensions, we need to zero-pad the inputs. When creating translated versions of the jets, we only shift the constituents with non-zero p_T, which means that zero-padding is an issue when trying to implement translational invariance. Without a fixed reference point, i.e. the jet axis, the calculation of distances becomes more complex for the network: Either the network learns that the jet axis has been shifted, or the network has to calculate the pair-wise distances between the constituents. For rotations this is not an issue because the zero-padded entries are located in the center of the jet and rotations around the jet axis do not change these constituents.

Low-p_T Modifications. Since the distribution of low-p_T constituents in jets, that were produced by the same process with the same kinematics, can vary, we introduce distortion of the low-p_T constituents as an augmentation. The low-p_T modificated jets are produced by applying noise to the positions of the low-p_T constituents:

$$\eta' \sim \mathcal{N}\left(\eta, \frac{\Lambda_\mathrm{soft}}{p_\mathrm{T}} R\right) \qquad \text{and} \qquad \phi' \sim \mathcal{N}\left(\phi, \frac{\Lambda_\mathrm{soft}}{p_\mathrm{T}} R\right) \tag{6.2}$$

Here, $\mathcal{N}(\mu, \sigma)$ denotes a Gaussian distribution with p_T-dependent standard deviation. For our studies, we choose $\Lambda_\mathrm{soft} = 100\,\mathrm{MeV}$.

Permutation Invariance of the Constituents. Permutation invariance is implemented by using appropriate networks, such as a transformer-encoder network. The representations obtained with the FCN are not invariant to the constituent ordering because flattening the input implies an implicit choice in the ordering of the jet constituents [6].

6.2 Network Architectures and Implementation Details

6.2.1 Training Details

In the case where we have a signal-to-background ratio of $t/Q = 1.0$ we take 100k QCD jets and 100k top jets for the analysis. When we have a class imbalance we always take 100k jets for the dominant class and select the number of jets for the other class based on the t/Q ratio. Before training, we shuffle the dataset and split it up into a test and a training dataset (for the test dataset, we randomly select 10 % of the jets from the entire dataset). For performance evaluation, the test datasat is used. For a signal-to-background ratio of $t/Q = 0.05$, we have 94 500 jets for training and 10 500 jets for testing. The temperature hyper-parameter is set to $\tau = 0.1$. Other hyper-parameters, such as the number of epochs and the learning rate, are network dependent and described below.

Fully-Connected Network (FCN). Unless otherwise specified, our default setup for the FCN is as follows: The FCN, which consists of one hidden layer with 80 neurons, is trained with a batch size of 1000 for 750 epochs. The `AdamW` optimizer with weight decay and a learning rate of 0.001 is used to adjust the network parameters in order to minimize the NT-Xent loss. We only use the $n_C = 50$ hardest constituents such that the input size of the FCN is given by $3n_C = 150$; for the output dimension (dimension of the latent space) we choose a value of 200. The signal-to-background ratio is set to $t/Q = 0.05$.

Transformer-Encoder Network. The transformer-encoder architecture is implemented in `PyTorch` using the `TransformerEncoder` module. For our studies we employ $h = 4$ parallel attention layers, so-called heads. The feed-forward network consists of 3 layers and the final head of 2 layers. The model dimension, the output dimension and the feed-forward dimension are set to 1000. Setting the output dimension to a value of 1000 means that we get 1000-dimensional representations, which is exactly the same dimension that we get with EFPs of degree $d \leq 7$. The network parameters are adjusted using the `Adam` optimizer with a learning rate of 5×10^{-5}.

It should be noted: Most of the calculations in a transformer-based architectures can be parallelized using tensor operations. However, transformer networks are still computationally expensive. In our case, the computation time scales with $\mathcal{O}(n_C^2)$.

6.2.2 Supervised Linear Classifier Test

Since the Monte-Carlo simulated top-tagging dataset is labeled, i.e. for each sample we know whether it is a top jet or a light-quark or gluon jet, we can perform a supervised linear classifier test (LCT) to evaluate the performance of JetCLR. The binary classifier is basically a FCN with no hidden layers and a linear output (obtained by the sigmoid activation). The binary classifier is trained to map the representations learned by JetCLR to a binary label. Binary cross-entropy (BCE) loss is implemented for the classification task. The linear classifier is trained for 500 epochs using the `Adam` optimizer with a learning rate of 0.001 and batch size of 1000.

6.3 Downstream Task Performance and Comparisons

In this section, we present our studies of how various aspects of our JetCLR framework affect the performance of the learned representations under a supervised LCT. Except for the results in Figure 6.5, all results were obtained with the FCN. The results for the transformer-encoder network will soon be published [6].

6.3.1 Number of Constituents

We first investigate how the number of constituents alter the performance. Here, we use the FCN to construct the jet representations. Figure 6.1 visualizes the representation quality, characterized by the AUC value and the inverse background mistag rate ϵ_b^{-1} at a signal efficency of $\epsilon_s = 0.5$, for different number of constituents. If we only use rotations as augmentations, the performance in the LCT decreases as the number of constituents increases, which is because the low-p_T constituents contain noise. By introducing low-p_T modifications in addition to the rotations, the FCN learns to ignore noise in the low-p_T constituents, which significantly improves performance.

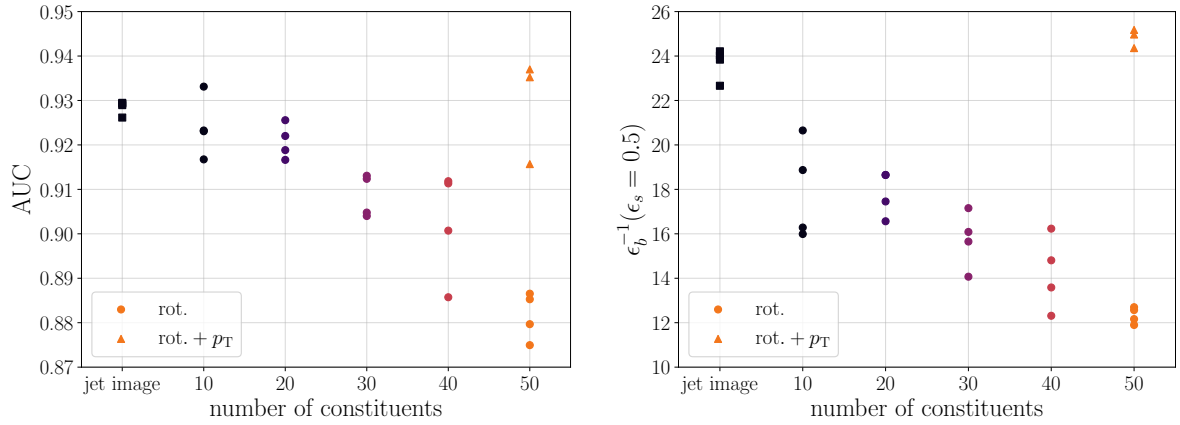


Figure 6.1: Downstream task performance for different numbers of constituents. By introducing low-p_T distortion in addition to the rotations, the FCN learns to ignore noise in the low-p_T constituents, which significantly improves performance.

6.3.2 Temperature, Alignment and Uniformity

Again, we use the FCN to construct the jet representations. As augmentations we use rotations and low-p_T distortion. We implemented the alignment loss ($\mathcal{L}_{\text{align}}$) and the uniformity loss ($\mathcal{L}_{\text{uniform}}$) as described in Ref. [39] using $\alpha = 2$ and $t = 2$. During training, we measured the downstream task performance as well as the $\mathcal{L}_{\text{align}}$ and $\mathcal{L}_{\text{uniform}}$ metrics. Figure 6.2 visualizes the trends between the alignment and uniformity loss and the representation quality at the end of the training procedure for different temperature values τ. We observe that alignment and uniformity strongly agree with the performance in the linear classifier test: The best-performing networks are the ones with low $\mathcal{L}_{\text{align}}$ and $\mathcal{L}_{\text{uniform}}$. We can also see that the choice of the temperature hyper-parameter τ impacts the downstream task performance of our contrastive learning model. The performance drops for very low and very high temperatures.

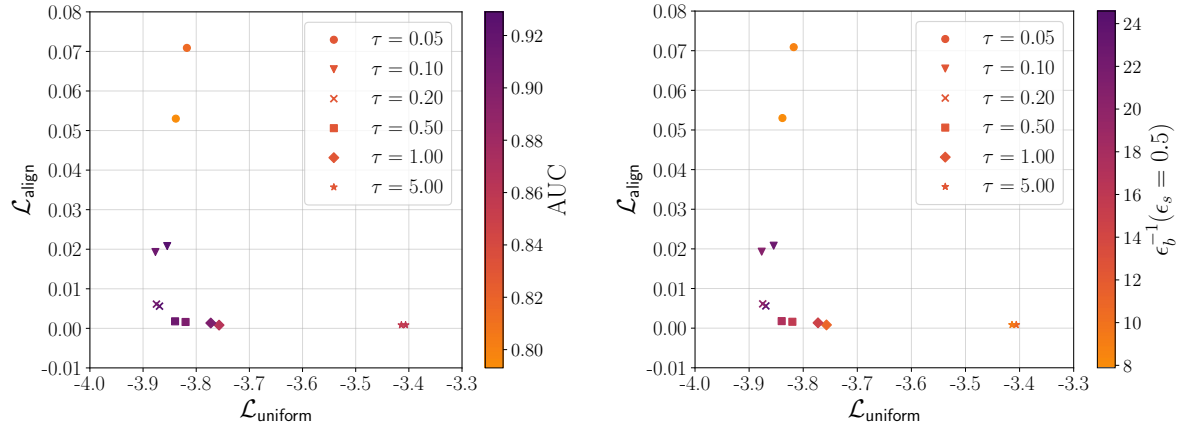


Figure 6.2: Visualization of the alignment and uniformity metrics for different temperatures τ. Alignment and uniformity strongly agree with the downstream task performance: The best-performing networks are the ones with low $\mathcal{L}_{\text{align}}$ and $\mathcal{L}_{\text{uniform}}$, i.e. the ones in the lower left corner.

As proposed in Ref. [39], we studied the effect of directly optimizing only alignment and uniformity. The results are shown in Figure 6.3. Due to the U-shaped distributions that peak at around $\lambda = 0.5$ we can conclude that both alignment and uniformity are indeed desirable properties for representations and necessary for good downstream task performance.

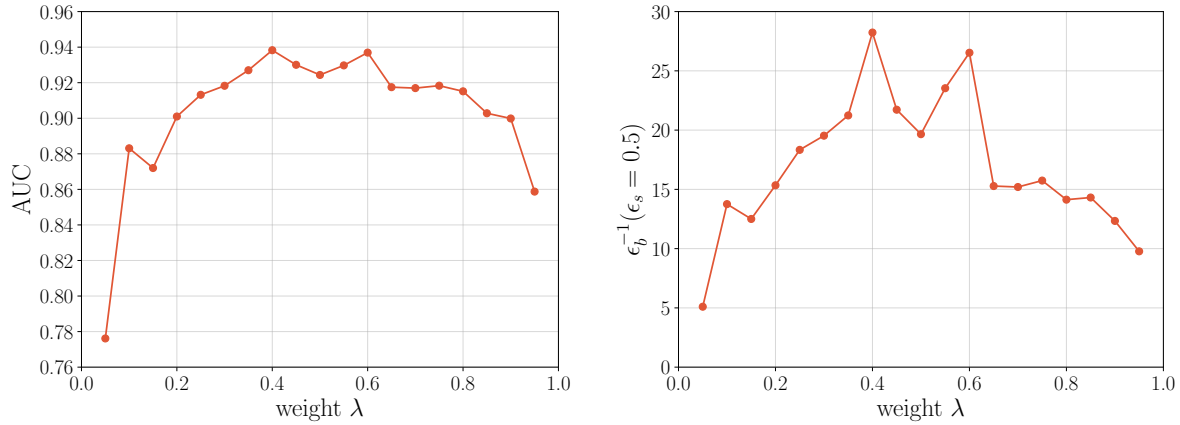


Figure 6.3: Effect of directly optimizing different weighted combinations $(1-\lambda)\mathcal{L}_{\text{align}} + \lambda\mathcal{L}_{\text{uniform}}$ of the alignment and uniformity loss. Both alignment and uniformity are necessary for good jet representations.

6.3.3 Representation Dimension

The representation dimension, i.e. the output dimension of the FCN and the transformer-encoder network, determines how much information we can embed in the representation space $\mathcal{R}$ [6]. Intuitively, one might expect that the larger the latent space dimension, the more (separable) information we can encode and the better the downstream task performance. Figure 6.4 shows, however, that this is true for an increase from 50 to 100 dimensions, but that the performance is almost the same for 100-, 150- and 200-dimensional latent spaces.

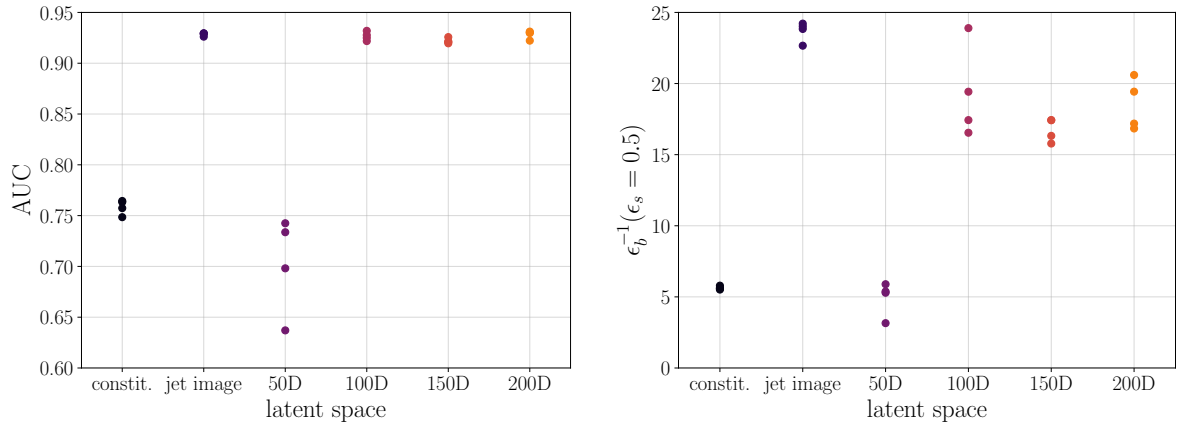


Figure 6.4: Downstream task performance for different representation dimensions.

6.3.4 Comparison with other Representations

Finally, we compare our JetCLR representations to some of the more widely used high-level jet observables. To keep it simple we just use rotations and low-p_T distortion as augmentations. We now use the transformer-encoder network to learn the 1000-dimensional jet representations. The results are shown in Figure 6.5. For the EFP top-tagging baseline, we used the raw constituent data to compute all $d \leq 7$ EFPs with an angular exponent of $\beta = 0.5$ and the default hadronic measure (treating all particles as massless). We do not need any additional pre-processing for the EFP calculation. Implementations of the EFPs are available in the `EnergyFlow` package.

We can see that our JetCLR framework achieves significant improvement over the jet-image representation. The representations learned by JetCLR even outperform the theory-inspired energy flow polynomials (EFPs). The raw constituents are not invariant to any of the symmetries associated with a jet and there is no guarantee that the raw (p_T, η, ϕ) information provides discriminative power [6].

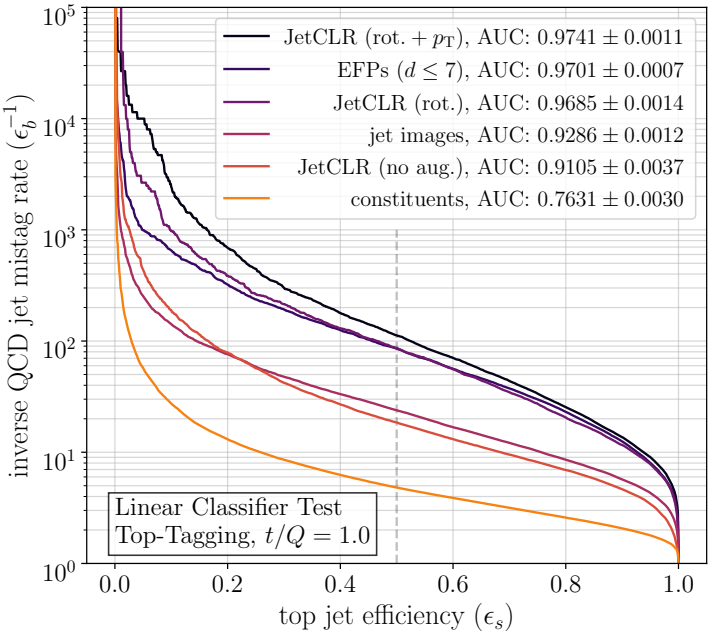


Figure 6.5: Comparison of the JetCLR representations (using the transformer-encoder network) to other high-level observables.

7 Conclusion and Outlook

Using both a fully-connected network (FCN) and a transformer-encoder network, we implemented a self-supervised learning framework that maps the raw constituent data of jets to some high-level observables by optimizing the contrastive loss function. We call this framework JetCLR (*Contrastive Learning of Jet Representations*). Through the use of suitable augmentation strategies and because of the alignment and uniformity properties of the contrastive loss function, the representations learned by JetCLR are (approximately) invariant to physically-motivated symmetries and retain the discriminative information contained in the dataset.

Since top quarks are common decay products in many beyond the Standard Model (BSM) theories [15], we trained our JetCLR method on a dataset that contains signal top jets and background QCD dijets. The representations learned by JetCLR are then used in a supervised linear classifier test (LCT) to identify boosted, hadronically decaying top quarks. The benchmark experiments show that our JetCLR framework achieves state-of-the-art performance on the top-tagging reference dataset when compared to jet images and energy flow polynomials (EFPs). This indicates that JetCLR is capable of learning informative representations.

However, the question arises whether we can get some physical interpretation on what these representations are. The t-SNE visualization (t-distributed stochastic neighbor embedding), for example, could be helpful for this. It would also be interesting to visualize the attention vectors of the transformer-encoder architecture.

There are several possible improvements and next steps that can be explored: We are currently working on adding masking to the transformer-encoder network in order to handle variable length inputs. It is also known that larger batch sizes and more training steps lead to better downstream task performance in contrastive learning tasks [9, 12], indicating that contrastive learning of representations (CLR) benefits from a larger number of negative samples. In addition, CLR benefits from deeper and wider networks [9]. However, our batch and network sizes were limited by hardware memory constraints.

One could also implement the JetCLR method using other permutation invariant architectures such as graph networks. Other augmentations could also be implemented: By filling the zero-padded elements with randomly generated, very low-p_T points and splitting a single constituent into two constituents at the same position (with the p_T-sum being equal to the original p_T value), the network could learn to construct IRC-safe representations. This would be interesting since IRC-safe observables have proven to be more robust for experimental tasks [19]. Furthermore, JetCLR could be useful for unsupervised anomaly detection, by training, for example, a variational autoencoder (VAE) [45, 46, 47] on the representations learned by JetCLR. For the application of JetCLR in anomaly detection it would be interesting to investigate how a smaller signal-to-background ratio affects the quality of the learned JetCLR representations. We are currently working on some of the above topics and the results will soon be available in our paper [6].

References

[1] CMS Collaboration. "Observation of a new boson at a mass of 125 GeV with the CMS experiment at the LHC". *Physics Letters B* 716.1 (Sept. 17, 2012), pp. 30–61. DOI: 10.1016/j.physletb.2012.08.021.

[2] 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), pp. 1–29. DOI: 10.1016/j.physletb.2012.08.020.

[3] Super-Kamiokande Collaboration. "Evidence for oscillation of atmospheric neutrinos". *Phys.Rev.Lett.* (July 3, 1998). DOI: 10.1103/PhysRevLett.81.1562. arXiv: hep-ex/9807003 [hep-ex].

[4] Tilman Plehn and Michael Spannowsky. "Top Tagging" (Dec. 19, 2011). DOI: 10.1088/0954-3899/39/8/083001. arXiv: 1112.4441 [hep-ph].

[5] Sebastian Schätzel and Michael Spannowsky. "Tagging highly boosted top quarks". *Phys. Rev. D 89, 014007 (2014)* (Aug. 2, 2013). DOI: 10.1103/PhysRevD.89.014007. arXiv: 1308.0540 [hep-ph].

[6] Barry M. Dillon et al. "Contrastive Learning of Jet Representations". *Manuscript in preparation* (2021).

[7] Emmy Noether. "Invariante Variationsprobleme". *Nachrichten von der Gesellschaft der Wissenschaften zu Göttingen, Mathematisch-Physikalische Klasse 1918* (1918), pp. 235–257. URL: https://eudml.org/doc/59024.

[8] Phuc H. Le-Khac, Graham Healy, and Alan F. Smeaton. "Contrastive Representation Learning: A Framework and Review". *IEEE Access* 8 (2020), pp. 193907–193934. DOI: 10.1109/access.2020.3031549.

[9] Ting Chen et al. "A Simple Framework for Contrastive Learning of Visual Representations" (Feb. 13, 2020). arXiv: 2002.05709 [cs.LG].

[10] Ishan Dave et al. "TCLR: Temporal Contrastive Learning for Video Representation" (Jan. 20, 2021). arXiv: 2101.07974 [cs.CV].

[11] Alireza Nasiri and Jianjun Hu. "SoundCLR: Contrastive Learning of Representations For Improved Environmental Sound Classification" (Mar. 2, 2021). arXiv: 2103.01929 [cs.SD].

[12] Yuyang Wang et al. "MolCLR: Molecular Contrastive Learning of Representations via Graph Neural Networks" (Feb. 19, 2021). arXiv: 2102.10056 [cs.LG].

[13] Anja Butter et al. "Deep-learned Top Tagging with a Lorentz Layer". *SciPost Phys. 5, 028 (2018)* (July 27, 2017). DOI: 10.21468/SciPostPhys.5.3.028. arXiv: 1707.08966 [hep-ph].

[14] G. Kasieczka, T. Plehn, A. Butter, et al. "The Machine Learning Landscape of Top Taggers". *SciPost Phys. 7, 014 (2019)* (Feb. 26, 2019). DOI: 10.21468/SciPostPhys.7.1.014. arXiv: 1902.09914 [hep-ph].

[15] Lisa Benato et al. "Shared Data and Algorithms for Deep Learning in Fundamental Physics" (July 1, 2021). arXiv: 2107.00656 [cs.LG].

[16] Josh Cogan et al. "Jet-Images: Computer Vision Inspired Techniques for Jet Tagging" (July 21, 2014). DOI: 10.1007/JHEP02(2015)118. arXiv: 1407.5675 [hep-ph].

[17] Michael Kagan et al. "Boosted Jet Tagging with Jet-Images and Deep Neural Networks". *EPJ Web of Conferences* 127 (2016). Ed. by R. Frühwirth et al., p. 00009. DOI: 10.1051/epjconf/201612700009.

[18] Luke de Oliveira et al. "Jet-Images – Deep Learning Edition". *JHEP 07 (2016) 069* (Nov. 16, 2015). DOI: 10.1007/JHEP07(2016)069. arXiv: 1511.05190 [hep-ph].

[19] Patrick T. Komiske, Eric M. Metodiev, and Jesse Thaler. "Energy flow polynomials: A complete linear basis for jet substructure". *JHEP 04 (2018) 013* (Dec. 19, 2017). DOI: 10.1007/JHEP04(2018)013. arXiv: 1712.07124 [hep-ph].

[20] Wikimedia Commons. *Standard Model of Elementary Particles*. 2019. URL: https://commons.wikimedia.org/wiki/File:Standard_Model_of_Elementary_Particles.svg (visited on 07/02/2021).

[21] DØ Collaboration. "Observation of the Top Quark". *Phys.Rev.Lett.* (Mar. 3, 1995). DOI: 10.1103/PhysRevLett.74.2632. arXiv: hep-ex/9503003 [hep-ex].

[22] CDF Collaboration. "Observation of Top Quark Production in $p\bar{p}$ Collisions". *Phys.Rev.Lett.* (Mar. 2, 1995). DOI: 10.1103/PhysRevLett.74.2626. arXiv: hep-ex/9503002 [hep-ex].

[23] DØ Collaboration. "Observation of Single Top-Quark Production". *Phys.Rev.Lett.* (Mar. 4, 2009). DOI: 10.1103/PhysRevLett.103.092001. arXiv: 0903.0850 [hep-ex].

[24] CDF Collaboration. "First Observation of Electroweak Single Top Quark Production". *Phys.Rev.Lett.103:092002,2009* (Mar. 5, 2009). DOI: 10.1103/PhysRevLett.103.092002. arXiv: 0903.0885 [hep-ex].

[25] Piotr A. Zyla et al., Particle Data Group. "Review of Particle Physics". *Progress of Theoretical and Experimental Physics* (Aug. 2020). DOI: 10.1093/ptep/ptaa104.

[26] David E. Kaplan et al. "Top-tagging: A Method for Identifying Boosted Hadronic Tops". *Phys.Rev.Lett.101:142001,2008* (June 5, 2008). DOI: 10.1103/PhysRevLett.101.142001. arXiv: 0806.0848 [hep-ph].

[27] Tilman Plehn. *Lectures on LHC Physics*. Springer International Publishing, 2015. DOI: 10.1007/978-3-319-05942-6.

[28] Sebastian Schätzel. "Boosted top quarks and jet structure". *The European Physical Journal C* 75.9 (Sept. 2015). DOI: 10.1140/epjc/s10052-015-3636-x.

[29] Björn Penning. "The Pursuit of Dark Matter at Colliders – An Overview" (Dec. 4, 2017). DOI: 10.1088/1361-6471/aabea7. arXiv: 1712.01391 [hep-ex].

[30] Ullrich Köthe. "Lecture Notes on Advanced Machine Learning". Heidelberg University, 2021. URL: https://hci.iwr.uni-heidelberg.de/teaching/advanced_machine_learning_2021 (visited on 07/04/2021).

[31] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. http://www.deeplearningbook.org. MIT Press, 2016.

[32] Ashish Vaswani et al. "Attention Is All You Need" (June 12, 2017). arXiv: 1706.03762 [cs.CL].

[33] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. "Neural Machine Translation by Jointly Learning to Align and Translate" (Sept. 1, 2014). arXiv: 1409.0473 [cs.CL].

[34] Minh-Thang Luong, Hieu Pham, and Christopher D. Manning. "Effective Approaches to Attention-based Neural Machine Translation" (Aug. 17, 2015). arXiv: 1508.04025 [cs.CL].

[35] Jesse Thaler and Ken Van Tilburg. "Identifying Boosted Objects with N-subjettiness". *JHEP 1103:015,2011* (Nov. 10, 2010). DOI: 10.1007/JHEP03(2011)015. arXiv: 1011.2268 [hep-ph].

[36] Jonathan Shlomi, Peter Battaglia, and Jean-Roch Vlimant. "Graph Neural Networks in Particle Physics" (July 27, 2020). DOI: 10.1088/2632-2153/abbf9a. arXiv: 2007.13681 [hep-ex].

[37] Gregor Kasieczka et al. "Deep-learning Top Taggers or The End of QCD?" *JHEP 05 (2017) 006* (Jan. 30, 2017). DOI: 10.1007/JHEP05(2017)006. arXiv: 1701.08784 [hep-ph].

[38] Ivan Oleksiyuk. "Unsupervised learning for tagging anomalous jets at the LHC". Bachelor thesis. RWTH Aachen University, July 2020.

[39] Tongzhou Wang and Phillip Isola. "Understanding Contrastive Representation Learning through Alignment and Uniformity on the Hypersphere" (May 20, 2020). arXiv: 2005.10242 [cs.LG].

[40] Feng Wang and Huaping Liu. "Understanding the Behaviour of Contrastive Loss" (Dec. 15, 2020). arXiv: 2012.09740 [cs.LG].

[41] Torbjörn Sjöstrand et al. "An Introduction to PYTHIA 8.2" (Oct. 11, 2014). DOI: 10.1016/j.cpc.2015.01.024. arXiv: 1410.3012 [hep-ph].

[42] J. de Favereau et al. "DELPHES 3, A modular framework for fast simulation of a generic collider experiment" (July 24, 2013). DOI: 10.1007/JHEP02(2014)057. arXiv: 1307.6346 [hep-ex].

[43] Matteo Cacciari, Gavin P. Salam, and Gregory Soyez. "The anti-k_t jet clustering algorithm". *JHEP 0804:063,2008* (Feb. 8, 2008). DOI: 10.1088/1126-6708/2008/04/063. arXiv: 0802.1189 [hep-ph].

[44] Matteo Cacciari, Gavin P. Salam, and Gregory Soyez. "FastJet user manual" (Nov. 25, 2011). DOI: 10.1140/epjc/s10052-012-1896-2. arXiv: 1111.6097 [hep-ph].

[45] Taoli Cheng et al. "Variational Autoencoders for Anomalous Jet Tagging" (July 3, 2020). arXiv: 2007.01850 [hep-ph].

[46] Thorben Finke et al. "Autoencoders for unsupervised anomaly detection in high energy physics" (Apr. 19, 2021). arXiv: 2104.09051 [physics.data-an].

[47] Theo Heimel et al. "QCD or What?" *SciPost Phys. 6, 030 (2019)* (Aug. 27, 2018). DOI: 10.21468/SciPostPhys.6.3.030. arXiv: 1808.08979 [hep-ph].

Acknowledgements

First and foremost, I am extremely grateful to my research supervisors, Prof. Dr. Tilman Plehn and Dr. Barry M. Dillon, for their continuous support and for all their help and advice throughout the project and with this bachelor thesis. Thank you very much, Tilman, for giving me the opportunity to be part of this interesting project and to gain first experiences in research. Words cannot express how grateful I am to you, Barry. Without your guidance, comments and feedback throughout this project, this thesis would not have been possible.

Thanks also goes to Prof. Dr. Jan Martin Pawlowski, who kindly agreed to be the second examiner for this thesis, and to Jun.-Prof. Dr. Gregor Kasiezcka from Universität Hamburg, for giving us access to the DESY GPU cluster. I am also grateful to Dr. Elmar Bittner for always answering my questions and helping me with problems regarding the GPU farm of the ITP.

Furthermore, I would like to thank Hans Olischläger for the extremely helpful discussions about physics topics and machine learning. I really appreciated our time at the institute. I am also grateful to Peter Sorrenson and Thorsten Buss for being there to discuss machine learning and programming problems (and for being my *Advanced Machine Learning* tutor and enduring my exercises... thank you, Peter). It has been a great pleasure working with you guys. I would like to extend my sincere thanks to all members of the *LHC Physics and New Particles* group here in Heidelberg for the interesting conversations and the great time during our lunch-time walks.

I especially enjoyed the three days of the hybrid *ML4Jets2021* workshop. Thank you, Tilman, for organizing the workshop and thank you to all participants. It was a lot of fun watching football with you guys (and of course listening to the talks...).

A special thanks goes to Robert Schmier, not only for proof-reading this thesis, but especially for all your advice and support throughout the first two semesters of my physics studies. Thank you, Robert. I would also like to thank my proof-readers Wiebke Nissen, Anna Katharina Färber and Lukas Kalvoda. I am very grateful for your feedback.

Getting through physics studies required more than academic and scientific support. Therefore, I would like to say thank you to my friends Sara Ditsch, Sebastian Willenberg, Teresa Förster, Sarah Hoffmann and Simon Groß-Bölting. A very special thanks goes to Lukas Kalvoda for being such a great friend and lab course partner!

Finally, I would like to thank my mother as well as my siblings and their partners for their love and support throughout my life – they helped me to become who I am today. And I would like to thank my father, who couldn't experience this adventure with me, but who has been always in my mind.

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, 16 July 2021 Lorenz Vogel

Lorenz Vogel

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