jBOT: Semantic Jet Representation Clustering Emerges from Self-Distillation

Ho Fung Tsoi and Dylan Rankin

University of Pennsylvania, USA

{hftsoi,dsrankin}@sas.upenn.edu

Abstract

Self-supervised learning is a powerful pre-training method for learning feature representations without labels, which often capture generic underlying semantics from the data and can later be fine-tuned for downstream tasks. In this work, we introduce jBOT, a pre-training method based on self-distillation for jet data from the CERN Large Hadron Collider, which combines local particle-level distillation with global jet-level distillation to learn jet representations that support downstream tasks such as anomaly detection and classification. We observe that pre-training on unlabeled jets leads to emergent semantic class clustering in the representation space. The clustering in the frozen embedding, when pre-trained on background jets only, enables anomaly detection via simple distance-based metrics, and the learned embedding can be fine-tuned for classification with improved performance compared to supervised models trained from scratch.

Publication information to appear upon publication.

Received Date
Accepted Date
Published Date

Contents

1 Introduction **2**

2 Related Work **3**

3 jBOT **3**

4 Experiment **6**
- 4.1 Dataset 6
- 4.2 Implementation 6
- 4.3 Pre-training 7
- 4.4 Downstream classification 9
- 4.5 Downstream anomaly detection 10

5 Conclusion **13**

References **14**

---

1 Introduction

In high-energy physics (HEP) experiments such as ATLAS [1] and CMS [2] at the CERN Large Hadron Collider (LHC), identifying the originating particle of a jet from its substructure content (*jet tagging*) is one of the primary analysis tasks for precision Standard Model measurements and new physics discoveries. Unstable heavy particles are produced in high-energy collisions and can decay promptly in cascades until stable final states are reached and recorded by the detector. The resulting outgoing particles are Lorentz-boosted in the direction of the original energetic particle and appear as a collection of coherent particles confined within a narrow cone from the collision point, referred to as a jet. Jet tagging remains a challenging task because jet substructure is complex by nature, as a jet can contain $\mathcal{O}(100)$ or more nearby constituent particles and can be contaminated by background activity from other interactions. Many machine learning techniques have been explored to improve jet tagging performance, including different architectural designs under supervised learning [3–10].

In domains such as natural language modeling and computer vision, the paradigm has shifted predominately toward first pre-training on large amounts of generic unlabeled data using self-supervised learning (SSL) to learn a representation space that encodes underlying features, and then fine-tuning on domain-specific labeled data for downstream tasks. This two-stage approach seems to be more natural and has been shown to yield better performance than single-phase supervised learning. A representative example is the pretext task of masked language modeling (MLM) used by Bidirectional Encoder Representations from Transformers (BERT) [11], where the model is trained to predict masked words in sentences using large amounts of unlabeled text from diverse sources. The learned representations capture the contextual semantic meaning of each word in relation to the others in a sentence and serve as a foundational language knowledge, which improves performance when fine-tuned on labeled text such as sentiment classification.

The core of SSL is to *learn a generic feature representation through observation without supervision*. It aims to extract features from unlabeled data into an embedding space using self-supervised objectives such as contrastive learning [12, 13], where the model is trained to be invariant to augmentations of the same example (positive pairs) while distinguishing different examples (negative pairs), or self-distillation [14–18], where a student network is trained to match the representations encoded by a teacher, with architectural and training designs to prevent information collapse (i.e., learning trivial solutions such as a constant vector). This task-agnostic training encourages the model to encode features that preserve high-level semantics while being invariant to noise and low-level details, so the learned representations often exhibit meaningful properties, such as object segmentation in images or word semantics in text. For downstream tasks such as classification, a classifier head can be attached to the learned embeddings and fine-tuned with supervision instead of training from raw inputs. Because the embedding space already captures semantic structures from the data, fine-tuned models often outperform standalone models trained from scratch. This has inspired recent developments of SSL in HEP [19–26].

In this work, we present jBOT, a method adapted from the self-distillation pre-training framework iBOT [17] originally developed for computer vision, to learn jet representations that enable downstream tasks such as classification and anomaly detection. We observe that semantic clustering of jet classes emerges in the representation space when pre-trained on unlabeled jet data via self-distillation. The self-supervised features already exhibit decent class separation and enable, when pre-trained on background classes only, powerful anomaly detection using simple metrics such as distance-based score. When fine-tuned on downstream classification tasks with labeled data, jBOT consistently yields better performance than supervised models trained from scratch. The paper is structured as follows: Sec. 2 discusses some recent developments in the field, Sec. 3 describes the jBOT framework, Sec. 4 presents experimental setup and results, and Sec. 5 summarizes the work with outlook. Our code is available at https://github.com/hftsoi/jbot.

2 Related Work

Self-supervised visual learning. Early works such as MoCo [12] and SimCLR [13] are based on contrastive objectives which pull positive pairs together in representation space and push apart negative pairs. VICReg [27] does not require negative samples and prevents collapse by using regularizers to reduce redundancy in the representations. Recent developments have focused on the self-distillation paradigm inspired by knowledge distillation [28]. DINO [14–16] proposes a teacher-student architecture where collapse is prevented by applying stop-gradient to the teacher network whose weights are a slowly moving average of the student weights. The objective is to match predictions in a projected feature space between positive pairs, and research shows that features extracted by self-supervised Vision Transformer (ViT) [29] exhibit semantic properties such as object segmentation in images that may not emerge under supervised learning. Inspired by BERT's [11] masked word prediction, iBOT [17] extends the idea by masking image patches and additionally distilling the representations of the masked patches.

SSL applications in HEP. Historically, machine learning in HEP has largely focused on training supervised models from scratch on labeled data [4, 7]. Recent work has started exploring SSL via task-agnostic pre-training on unlabeled data and then fine-tuning on labeled data for downstream tasks. Contrastive methods such as SimCLR have been adapted in JetCLR [19] and DarkCLR [20], which use physics symmetries to construct jet augmentations, and in RS3L [24] which generates augmentations by re-simulation. Mask particle modeling (MPM) [22, 23] proposes a pre-training objective based on predicting representations of masked particles. J-JEPA [21] takes inspiration from join-embedding predictive architectures [18] for top tagging. MACK [25], adapted from VICReg [27], and RINO [26], adapted from DINO [14], propose using SSL to minimize performance difference when models trained on simulated labeled data are applied to real collision data, caused by mismodeling. These recent developments motivate exploring new applications in HEP and improving current methods, including this work.

3 jBOT

Our method largely follows iBOT [17], a self-supervised pre-training method for visual learning via self-distillation with an online tokenizer, and is adapted here to model jets with tokenized particles. The jBOT pre-training method is schematically illustrated in Fig. 1, and its components are described below.

Augmentations. The pre-training starts with data augmentation, which generates two views for each jet in a given batch, forming a positive pair as input to the pre-training architecture. Following Ref. [19], we consider three simple augmentations: (1) uniform rotation of particles around the jet axis, (2) Gaussian smearing of particle positions, and (3) collinear splitting of particles with conserved transverse momentum ($p_{\mathrm{T}}^{\mathrm{initial}} = p_{\mathrm{T}}^{\mathrm{aug},1} + p_{\mathrm{T}}^{\mathrm{aug},2}$).

Architecture. Similar to image data where fixed-sized patches are tokenized, given a jet with up to a fixed number of N_{p} constituent particles, each with d_{feat} features, we tokenize particles by embedding the d_{feat}-dimensional feature vectors into a d_{model}-dimensional space using a linear layer. We additionally prepend a special learnable [CLS] token, which allows

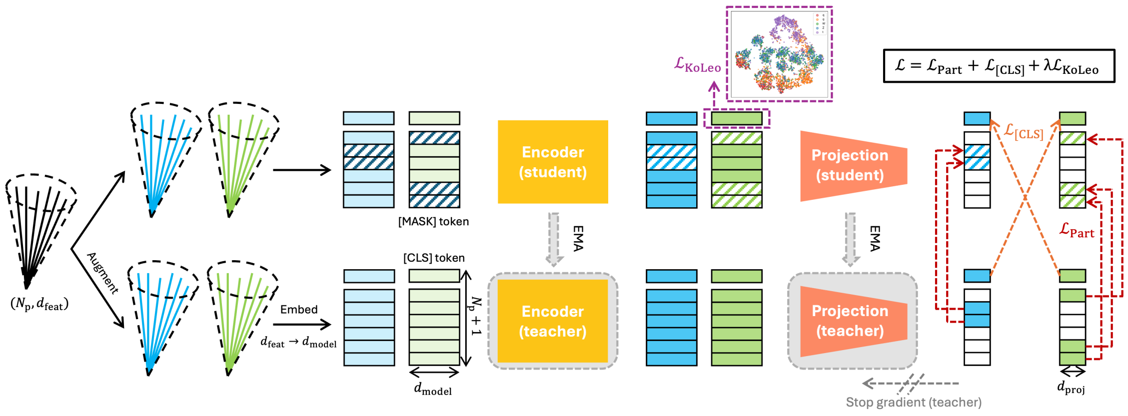

Figure 1: Schematic diagram of the jBOT pre-training method. A teacher-student architecture is used with a backbone encoder and a projection head, where stop-gradient is applied to the teacher network, whose weights are an EMA of the student network weights. Starting from an input jet, two augmented views are generated; in each view, each particle is embedded into a token space, and a [CLS] token is prepend. Both views are passed to the student and teacher networks. The student network processes distorted views where some of the particle tokens are masked and replaced by a learnable [MASK] token, while the teacher network processes the full views. Same-view and cross-view distillation losses are computed in the projection space, and the KoLeo loss is computed on the student [CLS] embedding from only one of the two views.

the network to encode global context from all other particle tokens through attention-based aggregation in the encoder, resulting in a total of $N_\text{p}+1$ tokens. When jet-level features are available from the dataset and are properly reweighted to eliminate dataset priors, they can be used as conditioning inputs and embedded into the [CLS] token (a robust jet tagger should infer from substructure content only); otherwise the [CLS] token is initialized without inputs. Each jet view therefore has an embedding shape of $(N_\text{p}+1, d_\text{model})$ and is processed by a ViT-style transformer encoder [29] to produce contextualized representations. For the self-supervised objectives, a projection head is used to map the encoded tokens into a d_proj-dimensional space where distillation losses are computed.

Self-distillation. The iBOT pre-training framework is formulated as knowledge distillation [28] with a teacher-student architecture for learning representations via self-distillation without labels. Both augmented views are processed by the teacher and student networks, and the student is trained to predict the teacher outputs both within the same view and across the two different views. Unlike standard knowledge distillation, where a large teacher is pre-trained and frozen to supervise a smaller student, the teacher and student here share the same architecture and initialization and are trained jointly from scratch for the distillation objective. To avoid collapse, an asymmetry between the teacher and student is enforced where only the student weights θ_s are back-propagated, while stop-gradient is applied to the teacher whose weights θ_t are updated via an exponential moving average (EMA) [12] of the student weights by $\theta_\text{t} \leftarrow \tau_\text{EMA}\theta_\text{t} + (1-\tau_\text{EMA})\theta_\text{s}$, where $\tau_\text{EMA} \in [0,1)$ is usually set close to 1 to smoothen the teacher targets. In the projection space, the teacher output, x, is re-centered to its batch mean as $x \leftarrow x - c$ with the center updated via EMA using $c \leftarrow \tau_\text{c} c + (1-\tau_\text{c})\bar{x}$, where τ_c is the centering momentum and $\bar{x}$ is the batch mean. Finally, the outputs are mapped by a temperature-scaled softmax into d_proj-dimensional probability distributions, so the teacher and student outputs can be naturally matched via, e.g., cross-entropy. There are local same-view and global cross-view distillations that are described in the following.

Particle-level objective (same-view distillation). In iBOT, each of the two image views passed to the student network has a fraction of patches masked by replacing the corresponding patch tokens with a learnable `[MASK]` token, so the student processes distorted views, while the teacher processes the complete views. For jets, similar to MPM [22, 23], masking is performed on a per-particle basis. Unlike images, where a grid is drawn and the image is divided into equally sized non-overlapping patches, particles within a jet can have widely varying positions and momenta. Therefore, instead of treating all particles equally by randomly masking a fixed number of particles, which can result in masking mostly the most energetic particles or mostly the least energetic particles, we use a simple momentum-aware scheme that masks particles such that the cumulative transverse momentum of the masked particles reaches a target ratio. For example, a 30% target masking ratio corresponds to randomly selecting a subset of particles to be masked such that approximately 30% of the jet transverse momentum is masked. For an input view u, the student network outputs $N_\mathrm{p}+1$ probability vectors in the projection space, $P_\mathrm{s}^{\mathrm{part},i=1,..,N_\mathrm{p}}(u) \in [0,1]^{d_\mathrm{proj}}$ and $P_\mathrm{s}^{\texttt{[CLS]}}(u) \in [0,1]^{d_\mathrm{proj}}$, and similarly for the teacher network outputs, denoted by P_t. For an augmented pair (u,v), the student is trained to predict the teacher outputs for the masked particle tokens within the same view using a cross-entropy loss:

$$\begin{aligned} \ell_\mathrm{Part}(u) &= \frac{1}{M(u)} \sum_{i=1}^{N_\mathrm{p}} m_i(u) \Big(-P_\mathrm{t}^{\mathrm{part},i}(u)^\mathrm{T} \cdot \log P_\mathrm{s}^{\mathrm{part},i}(u) \Big), \\ \mathcal{L}_\mathrm{Part} &= \frac{1}{2}\big(\ell_\mathrm{Part}(u) + \ell_\mathrm{Part}(v)\big), \end{aligned} \tag{1}$$

where $m_i(u) \in \{0,1\}$ indicates if the i-th particle is masked ($m_i = 1$) or not ($m_i = 0$), and $M(u) = \sum_{i=1}^{N_\mathrm{p}} m_i(u)$ is the number of masked particles in view u.

Jet-level objective (cross-view distillation). Complementary to the same-view distillation, which encourages the model to encode particle-level structure in the learned representations, the cross-view objective distills global representations by matching the teacher `[CLS]` output from view u to student `[CLS]` output from view v, and vice versa:

$$\begin{aligned} \ell_{\texttt{[CLS]}}(u,v) &= -P_\mathrm{t}^{\texttt{[CLS]}}(u)^\mathrm{T} \cdot \log P_\mathrm{s}^{\texttt{[CLS]}}(v), \\ \mathcal{L}_{\texttt{[CLS]}} &= \frac{1}{2}\big(\ell_{\texttt{[CLS]}}(u,v) + \ell_{\texttt{[CLS]}}(v,u)\big). \end{aligned} \tag{2}$$

Feature space diversification. In addition, similar to DINOv2 [15], we add the KoLeo regularizer [30] to encourage a diverse spread of different examples in the embedding space within a batch:

$$\mathcal{L}_\mathrm{KoLeo} = -\frac{1}{B} \sum_{i=1}^{B} \log\big(\min_{j\neq i} \|x_j - x_i\|\big) \tag{3}$$

where x_i is a vector in the embedding space after ℓ_2-normalization and the sum runs over a batch of size B. To simplify the computation, we apply the regularizer to the student `[CLS]` from only one of the two views.

To sum up, the pre-training loss is given by:

$$\mathcal{L} = \mathcal{L}_\mathrm{Part} + \mathcal{L}_{\texttt{[CLS]}} + \lambda \mathcal{L}_\mathrm{KoLeo} \tag{4}$$

where $\lambda > 0$ scales the KoLeo regularization. After pre-training, the projection head is removed, and test jets without augmentations and masking are processed by the student encoder to produce self-supervised features for downstream tasks, as illustrated in Fig. 2. For instance, one can attach a classifier head taking as input the encoded `[CLS]` token and fine-tune both the encoder and classifier for supervised classification, or directly probe the frozen representation for anomaly detection.

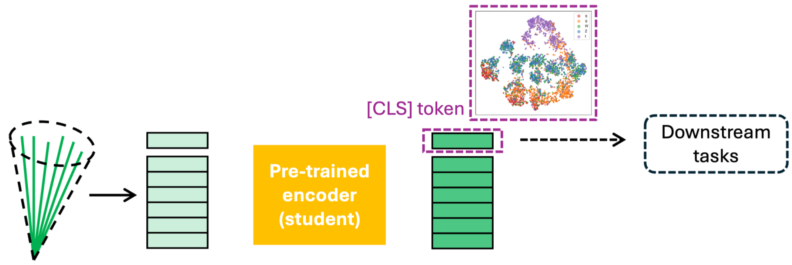

Figure 2: Downstream tasks are performed using the [CLS] embedding from the pre-trained student encoder.

4 Experiment

4.1 Dataset

We train and evaluate on the JetNet dataset [31,32]. The dataset consists of 880k simulated jets with transverse momentum (p_T) around 1 TeV, originating from light quarks (q), gluons (g), W bosons, Z bosons, and top quarks (t) produced in proton-proton collisions at a center-of-mass energy of 13 TeV. Jet clustering is performed using the anti-k_T algorithm [33] with a distance parameter of 0.8. Each jet stores up to 30 highest-p_T constituent particles, each with four features ($\eta_{\text{rel}}, \phi_{\text{rel}}, p_{\text{T,rel}}$, valid), where $\eta_{\text{rel}} = \eta - \eta_{\text{jet}}$ and $\phi_{\text{rel}} = \phi - \phi_{\text{jet}}$ are the pseudorapidity and azimuthal angle measured from the jet axis, $p_{\text{T,rel}} = p_T/p_{\text{T,jet}}$ is the p_T fraction relative to the jet, and "valid" is a boolean indicating whether the particle is padded when the jet contains fewer than 30 particles. The dataset also contains jet-level kinematic features such as jet p_T, η, and ϕ, but we do not consider them in the models, since their distributions differ between classes and these dataset priors, without proper re-weighting, may introduce bias into jet classification, which should be based on jet substructure information only. We note that in spite of this, the jBOT method is indeed capable of handling these global features.

4.2 Implementation

For the augmentations, the rotation angle is sampled uniformly from $-\pi$ to π per jet; following Ref. [19], each particle's η and ϕ are smeared independently by a Gaussian with a variance of Λ_{QCD}/p_T, where $\Lambda_{\text{QCD}} = 100$ MeV is the QCD scale; and collinear splitting is applied to jets with fewer than 30 valid particles. Masking is implemented by accumulating particles from a randomly reshuffled list until the cumulative p_T crosses the masking target; the subset whose cumulative p_T is closest to the target is masked, which avoids overshooting or undershooting the target masking ratio. Fig. 3 shows example augmented jets with masking applied.

The model and pre-training hyperparameters are summarized in Tab. 1; note that these parameters are not rigorously optimized but are reasonably chosen. We use a ViT-style transformer [29] as the backbone encoder, and consider two model sizes: small (jBOT-S) and base (jBOT-B). The projection head is a multilayer perceptron (MLP), and the weights are shared for projecting both the [CLS] and particle tokens. Dropout [34] with a rate of 20% is used in the transformer blocks. All hidden layers use Gaussian error linear unit (GELU) [35] activation. The model is implemented using Tensorflow [36] and Keras [37], and optimized using the AdamW optimizer [38].

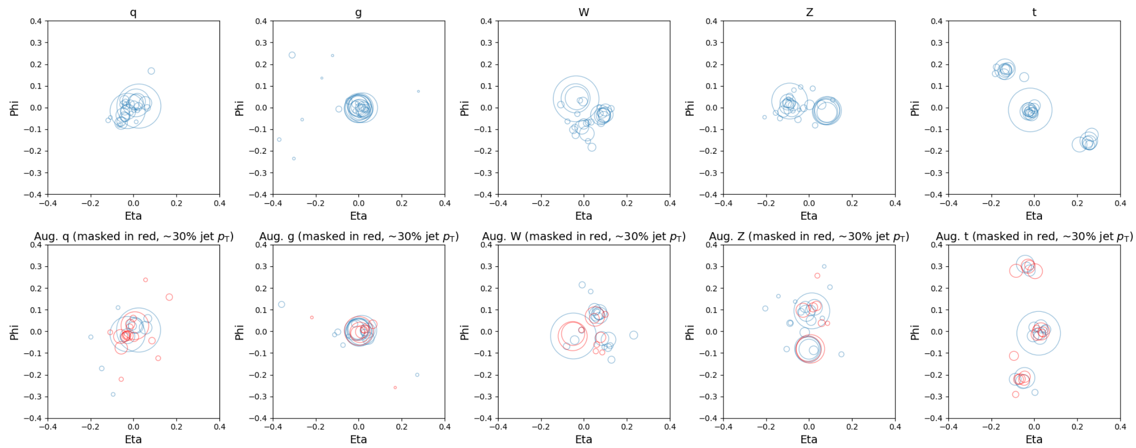

Figure 3: One example jet per class, where each circle represents a particle: the circle center is at the particle location and the radius is proportional to its p_T. Upper row: input jets. Lower row: augmented jets with masking shown in red (e.g., ∼30% of the jet p_T).

Table 1: Model and pre-training hyperparameters.

Model hyperparameters	
Embedding dimension d_model	32 (small), 64 (base)
# of transformer blocks in encoder	2 (small), 4 (base)
# of self-attention heads	4 (small), 6 (base)
Feedforward layer dim. in transformer block	$4d_\mathrm{model}$
Projection dimension d_proj	$d_\mathrm{model}/2$
Feedforward layer dim. in projection head	Two hidden layers: $8d_\mathrm{proj}$, d_proj
Feedforward layer dim. in classifier head	Two hidden layers: $2d_\mathrm{model}$, d_model
Pre-training hyperparameters	
Masking ratio (p_T)	Uniformly sampled from 0 to 50% per view
EMA momentum τ_EMA	Cosine schedule from 0.996 to 1 [39]
Centering momentum τ_c	0.9
Softmax temperature	0.04 (teacher), 0.1 (student)
KoLeo scale λ	0.01
Learning rate	$5 \times 10^{-4} \times$ (batch size/256), linear warm up for first 10 epochs
Batch size	1024
Weight decay	10^{-4}

4.3 Pre-training

We pre-train on three different training sets containing (1) all five jet classes, (2) the q, g, and t classes, and (3) the q and g classes, which are used downstream to probe five-class classification, top tagging, and anomaly detection, respectively. The training/validation/test split is around 80/10/10%, with balanced classes. We pre-train for 100 epochs on a single Nvidia A100 GPU, which takes, for example, around 11 min/epoch and in total 18 hours for the base model on the five-class training set with 700k examples. The learning curves are shown in Fig. 4, where losses, entropy ($-\sum p \log p$), and centering norm are monitored. Fig. 5 shows how the pre-trained encoder weights particles across attention heads by visualizing the attention weights from the [CLS] token as the query attending to the particle tokens in the last transformer layer. Fig. 6 shows the evolution of the [CLS] embedding during pre-training using 2D t-SNE projections, starting from random initialization and progressively developing class clustering, which shows that the embeddings learned without labels already exhibit discriminative structure before any supervised fine-tuning.

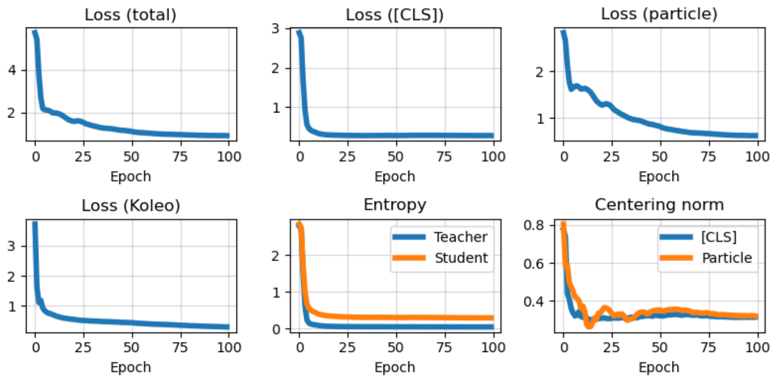

Figure 4: Learning curves for jBOT-B when pre-trained on five-class data. Upper left to right: total loss, $\mathcal{L}_{\texttt{[CLS]}}$, and $\mathcal{L}_{\text{Part}}$. Lower left to right: $\mathcal{L}_{\text{KoLeo}}$, entropy, and centering norm.

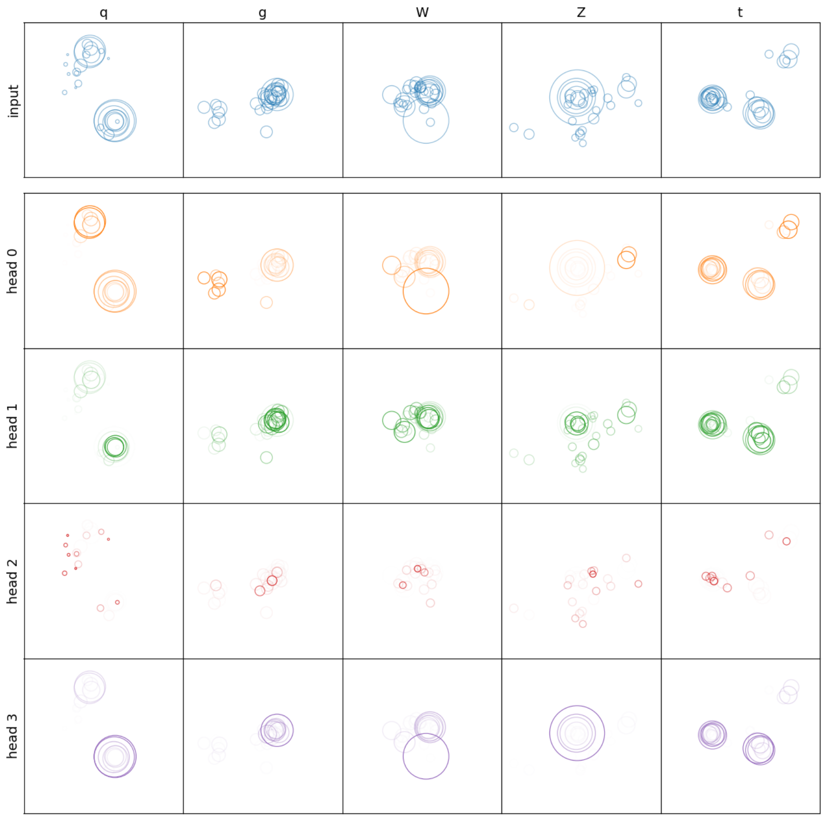

Figure 5: Attention weights from the last transformer block in jBOT-S pre-trained on all five classes, obtained using the [CLS] token as the query attending to the particle tokens. Top row: one example input jet per class (each circle represents a particle: the circle center is at the particle location, the radius is proportional to its p_{T}, and the edge alpha is uniform across all particles here). Other rows: attention weights per head for the same input jets, shown with the same drawing style as the input jets, but with the attention weight represented by the edge alpha (higher edge alpha indicates larger attention weight).

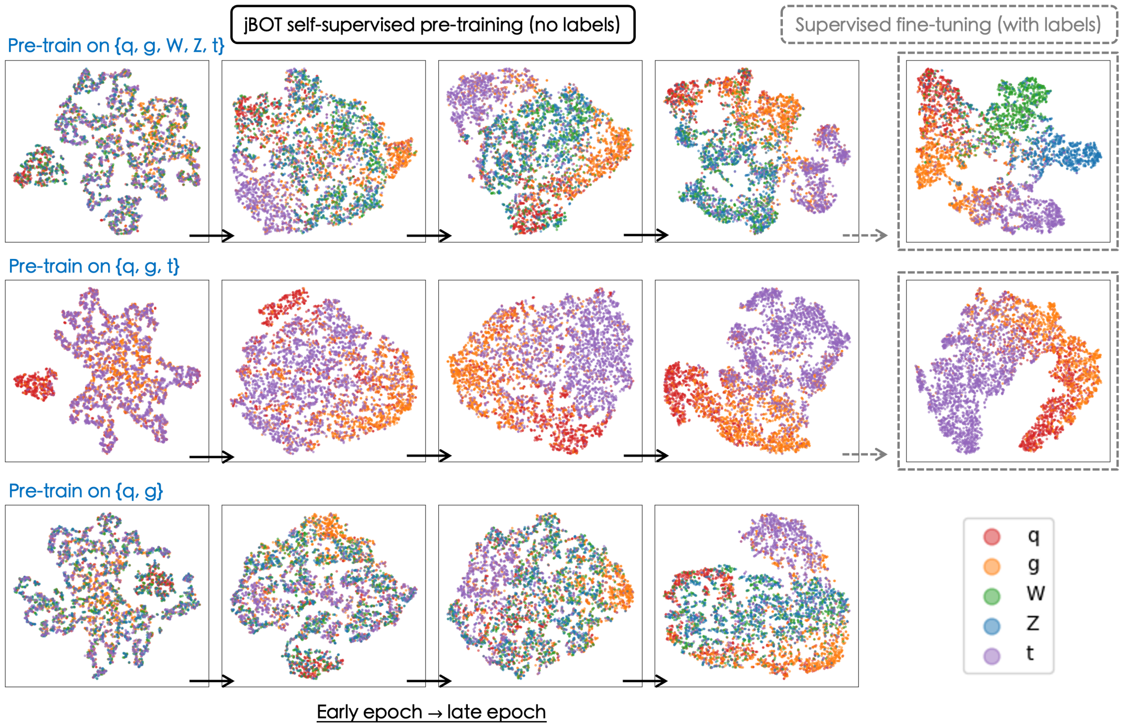

Figure 6: Example evolution of 2D t-SNE projections of the [CLS] token during pre-training (and after fine-tuning). Top row: pre-training on all five classes and then fine-tuning. Middle row: pre-training on the t, q, and g classes and then fine-tuning. Bottom row: pre-training on the q and g classes.

4.4 Downstream classification

We first evaluate the pre-trained encoder by probing classification on the frozen features from the [CLS] token. We probe five-class classification using the student encoder pre-trained on all five classes, and top tagging using the student encoder pre-trained on the t, q, and g classes. Following standard evaluation protocols [14, 17], we fit a k-nearest-neighbor (k-NN) classifier with $k = 30$ and a linear classifier on the frozen features. The performance is shown in Tab. 2 (five-class) and Tab. 3 (top tagging). For example, both k-NN and linear probe achieve accuracies of around 70% in the five-class set and 87% in the top tagging set.

We then construct a classifier by attaching a MLP head that takes as input the [CLS] token from the pre-trained encoder and perform supervised fine-tuning on the labeled training set. To preserve underlying semantic structures learned from pre-training and avoid degenerating into a supervised classifier trained from scratch, we use layer-wise learning rate decay (LLRD) [40, 41] for the fine-tuning: the classifier head receives the largest learning rate, and the learning rate decays by a multiplicative factor in each of the earlier blocks, so that the earliest layers in the encoder receive the smallest learning rate. We fine-tune separately using only 10% and the full training set, scanning decay factors $\{0.6, 0.65, 0.7, 0.75, 0.8\}$ and base learning rates $\{4\times10^{-5}, 8\times10^{-5}, 2\times10^{-4}, 4\times10^{-4}, 8\times10^{-4}, 2\times10^{-3}\}$, with a batch size of 1024 for 100 epochs, and then select the model with the best performance. The results are shown in Tab. 2 (five-class) and Tab. 3 (top tagging), and ROC curves are shown in Fig. 7. All fine-tuned models perform better than supervised models trained from scratch on the same dataset sizes, with the largest gains seen when the fine-tuning dataset is small (e.g., 10%), because the self-supervised models are fine-tuned from a feature representation that already encodes meaningful information.

Table 2: Five-class classification performance (overall accuracy and per-class AUC) comparing jBOT with supervised models. Note that all models use particle features only and ignore jet-level features.

Model	Acc. [%]	AUC				
		q	g	W	Z	t
Frozen embedding (no labels)						
k-NN (jBOT-S)	0.7102 ± 0.0058	0.8869 ± 0.0058	0.8943 ± 0.0040	0.9272 ± 0.0021	0.9070 ± 0.0039	0.9209 ± 0.0052
Linear (jBOT-S)	0.6743 ± 0.0056	0.8702 ± 0.0049	0.8796 ± 0.0042	0.8904 ± 0.0017	0.8820 ± 0.0037	0.9166 ± 0.0044
k-NN (jBOT-B)	0.7053 ± 0.0057	0.8859 ± 0.0049	0.8883 ± 0.0051	0.9209 ± 0.0029	0.9057 ± 0.0040	0.9218 ± 0.0044
Linear (jBOT-B)	0.6942 ± 0.0046	0.8767 ± 0.0058	0.8820 ± 0.0051	0.9107 ± 0.0023	0.9019 ± 0.0034	0.9203 ± 0.0044
Fine-tuning (with labels)						
Sup.-S (10%)	0.7251 ± 0.0052	0.8908 ± 0.0057	0.8966 ± 0.0039	0.9463 ± 0.0020	0.9290 ± 0.0035	0.9335 ± 0.0040
jBOT-S (10%)	0.7425 ± 0.0056	0.9004 ± 0.0055	0.9061 ± 0.0035	0.9541 ± 0.0022	0.9359 ± 0.0031	0.9427 ± 0.0034
Sup.-S (100%)	0.7431 ± 0.0056	0.8987 ± 0.0056	0.9070 ± 0.0037	0.9535 ± 0.0023	0.9359 ± 0.0032	0.9427 ± 0.0033
jBOT-S (100%)	0.7526 ± 0.0064	0.9074 ± 0.0051	0.9177 ± 0.0035	0.9573 ± 0.0022	0.9387 ± 0.0030	0.9477 ± 0.0030
Sup.-B (10%)	0.7379 ± 0.0056	0.9007 ± 0.0051	0.9070 ± 0.0037	0.9517 ± 0.0023	0.9333 ± 0.0036	0.9450 ± 0.0038
jBOT-B (10%)	0.7499 ± 0.0053	0.9078 ± 0.0048	0.9174 ± 0.0038	0.9557 ± 0.0021	0.9372 ± 0.0027	0.9479 ± 0.0032
Sup.-B (100%)	0.7604 ± 0.0057	0.9135 ± 0.0052	0.9231 ± 0.0036	0.9596 ± 0.0018	0.9430 ± 0.0028	0.9518 ± 0.0028
jBOT-B (100%)	**0.7643 ± 0.0061**	**0.9155 ± 0.0048**	**0.9255 ± 0.0036**	**0.9605 ± 0.0017**	**0.9443 ± 0.0029**	**0.9538 ± 0.0024**

Table 3: Top tagging performance (accuracy, AUC, and signal efficiency ϵ_s at different background efficiencies) comparing jBOT with supervised models. Note that all models use particle features only and ignore jet-level features.

Model	Acc. [%]	AUC	$\epsilon_s(10^{-1})$	$\epsilon_s(10^{-2})$
Frozen embedding (no labels)				
k-NN (jBOT-S)	0.8777 ± 0.0037	0.9447 ± 0.0023	0.8352 ± 0.0125	0.3337 ± 0.0572
Linear (jBOT-S)	0.8709 ± 0.0039	0.9355 ± 0.0020	0.8115 ± 0.0138	0.2408 ± 0.0327
k-NN (jBOT-B)	0.8793 ± 0.0035	0.9475 ± 0.0019	0.8402 ± 0.0066	0.3494 ± 0.0699
Linear (jBOT-B)	0.8765 ± 0.0041	0.9438 ± 0.0028	0.8344 ± 0.0116	0.3892 ± 0.0315
Fine-tuning (with labels)				
Sup.-S (10%)	0.8723 ± 0.0040	0.9368 ± 0.0028	0.8172 ± 0.0158	0.2166 ± 0.0394
jBOT-S (10%)	0.8807 ± 0.0047	0.9499 ± 0.0019	0.8559 ± 0.0106	0.4306 ± 0.0317
Sup.-S (100%)	0.8852 ± 0.0047	0.9524 ± 0.0016	0.8659 ± 0.0080	0.4474 ± 0.0366
jBOT-S (100%)	0.8875 ± 0.0040	0.9554 ± 0.0015	0.8712 ± 0.0097	0.4814 ± 0.0328
Sup.-B (10%)	0.8784 ± 0.0051	0.9467 ± 0.0023	0.8429 ± 0.0132	0.4131 ± 0.0285
jBOT-B (10%)	0.8862 ± 0.0038	0.9542 ± 0.0017	0.8665 ± 0.0110	0.4843 ± 0.0306
Sup.-B (100%)	0.8899 ± 0.0035	0.9569 ± 0.0018	0.8756 ± 0.0072	0.5021 ± 0.0331
jBOT-B (100%)	**0.8911 ± 0.0029**	**0.9584 ± 0.0015**	**0.8771 ± 0.0079**	**0.5122 ± 0.0389**

4.5 Downstream anomaly detection

For downstream anomaly detection, we use the embedding pre-trained only on QCD jets (q, g) as the normal data, and test on W, Z, and t jets as anomalous signals. We freeze the backbone encoder and map all examples in the training set to vectors in the [CLS] embedding, which form an in-distribution reference set. We then define an anomaly score by computing a "distance" between each test example and the reference vectors in the embedding space, where an example far away from the reference bulk is more likely to be an anomalous signal.

We use four anomaly score metrics, namely k-NN, cosine similarity, Mahalanobis distance, and Gaussian mixture model (GMM). Let $z(x) \in \mathbb{R}^{d_{\text{model}}}$ denote the [CLS] embedding of a jet x, and is ℓ_2-normalized, and let $\mathcal{R} = \{z^{\text{ref}}_{(i)}\}_{i=1}^{M}$ be a reference set sampled from the training data. The k-NN score is defined as the average Euclidean distance from the test jet $z(x)$ to its k nearest neighbors $\{z^{\text{ref}}_{(i|x)}\}_{i=1}^{k}$ in $\mathcal{R}$:

$$s_{k\text{-NN}}(x;k) = \frac{1}{k}\sum_{i=1}^{k} \|z(x) - z^{\text{ref}}_{(i|x)}\|. \tag{5}$$

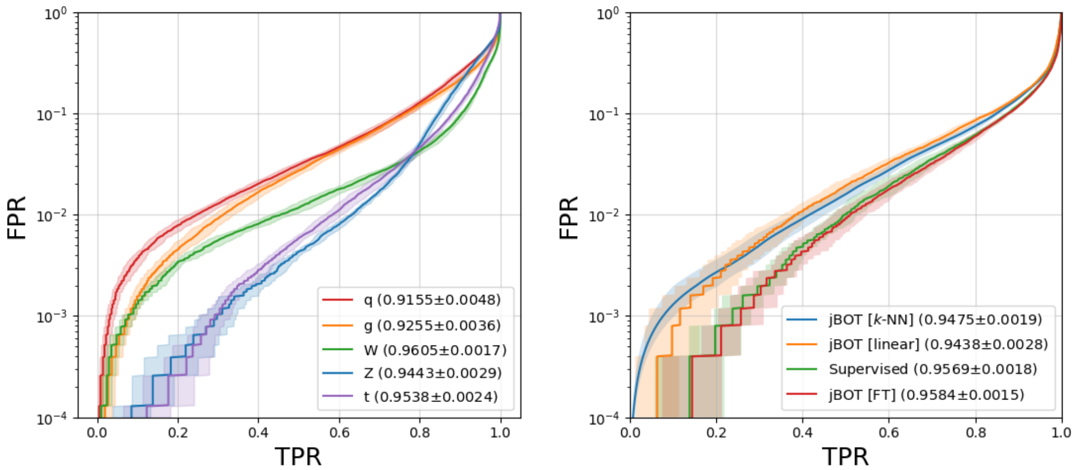

Figure 7: ROC curves for classification. Left: jBOT-B (fine-tuned) on five-class classification. Right: jBOT-B (using frozen features and fine-tuned) on top tagging.

Similarly, the cosine-similarity score measures angular distance to its k nearest neighbors:

$$s_{\cos}(x;k) = -\tau \log\left(\frac{1}{k}\sum_{i=1}^{k} \exp\left(\frac{z(x)^{\mathrm{T}} \cdot z^{\mathrm{ref}}_{(i|x)}}{\tau}\right)\right), \tag{6}$$

where $\tau = 0.05$ is a temperature parameter. The Mahalanobis distance [42] fits class-conditional Gaussians with a tied covariance to the reference set and takes the minimum distance as the anomaly score:

$$s_{\mathrm{Maha}}(x) = \min_{c\in\{\mathrm{q,g}\}} \left(z(x)-\mu_c\right)^{\mathrm{T}} \Sigma^{-1}\left(z(x)-\mu_c\right), \tag{7}$$

where μ_c is the class mean and Σ is the shared covariance. The GMM score fits a mixture of weighted Gaussians to the reference set and uses the negative log-likelihood as the anomaly score:

$$s_{\mathrm{GMM}}(x;K) = -\log\left(\sum_{i=1}^{K} w_i \mathcal{N}(z(x)|\mu_i,\Sigma_i)\right), \tag{8}$$

where K is the number of mixture components and $\{w_i,\mu_i,\Sigma_i\}_{i=1}^{K}$ are obtained by fitting the mixture model to the reference set. We note that these metrics depend on the detailed clustering structure of the learned embedding, which can vary with randomness from the pre-training, so we pre-train ten jBOT-S models with identical configuration and report results from the best model. We set $k = 30$ for the k-NN and cosine similarity scores, and $K = 4$ for the GMM score, as these values yield the highest performance on the combined signal for most models.

Fig. 8 shows the anomaly score distributions, and Fig. 9 shows the ROC curves, where most anomalous signals are reasonably separated from the QCD background. Tab. 4 lists the AUCs for the individual signals and for the combined signal. For a baseline comparison, we also quote results from Ref. [43], which uses reconstruction-based autoencoders for this task with different architectures, including a convolutional neural network (CNNAE), a graph neural network (GNNAE), and a Lorentz group equivariant network (LGAE). Our method performance is broadly comparable to the reconstruction-based models and can perform better for some signals. For example, our method using k-NN, cosine similarity, and GMM yields AUCs above 0.8 for the W and Z signals, while the reconstruction-based models yield AUCs below 0.8; for the t signal, our method using Mahalanobis distance yields the highest AUC among our metrics at around 0.86, comparing to around 0.89 for CNNAE and GNNAE; for the combined

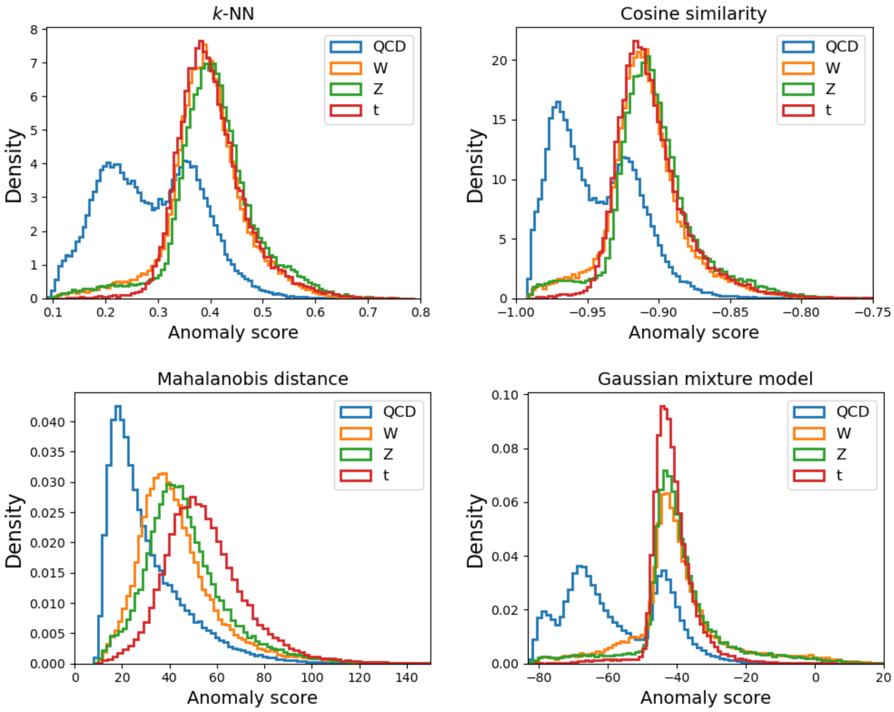

Figure 8: Anomaly score distributions: k-NN distance (upper left), cosine similarity (upper right), Mahalanobis distance (lower left), and GMM (lower right).

Table 4: Anomaly detection performance comparing jBOT (particle features only) and reconstruction-based autoencoder models from Ref. [43].

Model	AUC			
	W	Z	t	Combined
CNNAE [43]	0.6886	0.7247	**0.8962**	0.7700
GNNAE [43]	0.7558	0.7805	**0.8917**	0.8195
LGAE [43]	0.7489	0.7909	0.8669	**0.8313**
jBOT-S (k-NN)	**0.8072 ± 0.0028**	**0.8355 ± 0.0025**	0.8356 ± 0.0029	0.8261 ± 0.0028
jBOT-S (Cosine)	**0.8064 ± 0.0028**	**0.8355 ± 0.0027**	0.8388 ± 0.0028	**0.8269 ± 0.0027**
jBOT-S (Maha.)	0.7431 ± 0.0030	0.7821 ± 0.0029	0.8620 ± 0.0026	0.7957 ± 0.0037
jBOT-S (GMM)	0.8062 ± 0.0040	0.8204 ± 0.0040	0.8197 ± 0.0049	0.8155 ± 0.0038

signal, our method using k-NN, cosine similarity, and GMM yields AUCs of around 0.82, which is close to the highest AUC of 0.83 for LGAE.

It can also be seen that the performance varies with the choice of anomaly score. This is expected because there is a high degree of freedom in defining the metric and hyperparameters for comparing a test example to a large reference set in a high-dimensional embedding space, and different choices probe different aspects of the structure: e.g., local structure for k-NN vs. global structure for GMM. This flexibility introduces a large space to explore, and optimizations such as hyperparameter scans are required to select the best performing models. Nonetheless, we show that measuring similarity between the test examples and the nominal examples in a self-supervised embedding pre-trained only on nominal data provides a viable anomaly detection strategy and can yield competitive or even better performance than common reconstruction-based methods.

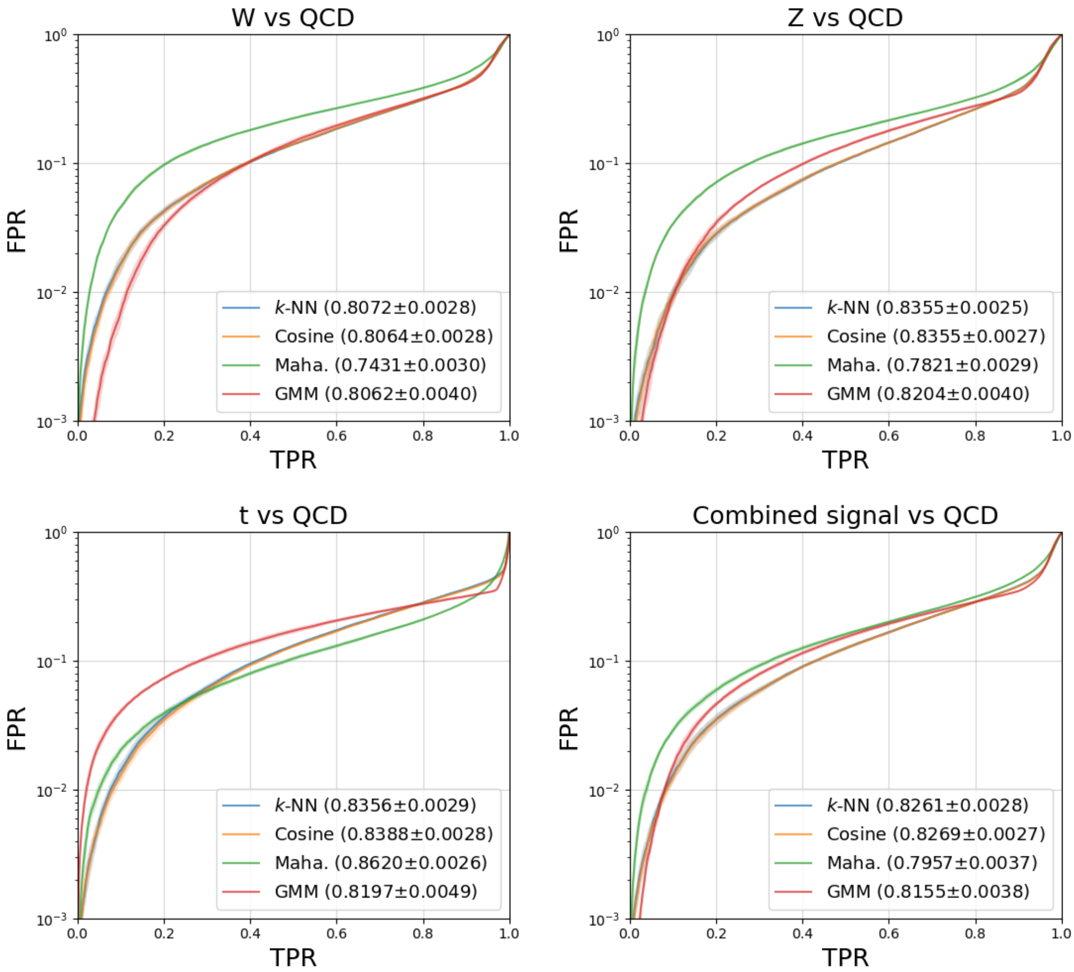

Figure 9: ROC curves for anomaly detection: W vs. QCD (upper left), Z vs. QCD (upper right), t vs. QCD (lower left), and the combined signal vs. QCD (lower right).

5 Conclusion

In this work, we have introduced jBOT, a self-supervised pre-training method based on the iBOT framework from computer vision, and applied it to jet data in HEP experiments at the CERN LHC. We have shown that semantic clustering of different jet classes emerges in the jet representations learned via self-distillation objectives without supervision, and that probing the frozen embedding with k-NN or a linear classifier already achieves a five-class accuracy of around 70%, compared to 76% for a supervised model. When the pre-trained model is fine-tuned on labeled data, it generally yields better performance than a supervised model trained from scratch, especially when the labeled dataset is small. We have also shown that the frozen embedding from a self-supervised model trained only on unlabeled background jets can be used for anomaly detection, with the flexibility to define various metrics that can be competitive or better than common reconstruction-based autoencoder architectures. We hope this work, as a new pre-training method based on self-distillation applicable to jets or similar physics data, contributes to ongoing developments in self-supervised learning for the HEP domain. Future directions include scaling up pre-training on much larger unlabeled datasets, more robust anomaly detection strategy with potentially dedicated augmentations, and fine-tuning on more diverse labeled data.

Acknowledgements

DR is supported by the U.S. Department of Energy (DOE), Office of Science, Office of High Energy Physics Early Career Research program under Award No. DE-SC0025324. This work used resources available through the National Research Platform (NRP) at the University of California, San Diego [44]. NRP has been developed, and is supported in part, by funding from National Science Foundation, from awards 1730158, 1540112, 1541349, 1826967, 2112167, 2100237, and 2120019, as well as additional funding from community partners.

References

[1] G. Aad *et al.*, *The ATLAS Experiment at the CERN Large Hadron Collider*, JINST **3**, S08003 (2008), doi:10.1088/1748-0221/3/08/S08003.

[2] S. Chatrchyan *et al.*, *The CMS Experiment at the CERN LHC*, JINST **3**, S08004 (2008), doi:10.1088/1748-0221/3/08/S08004.

[3] P. T. Komiske, E. M. Metodiev and J. Thaler, *Energy Flow Networks: Deep Sets for Particle Jets*, JHEP **01**, 121 (2019), doi:10.1007/JHEP01(2019)121, 1810.05165.

[4] H. Qu and L. Gouskos, *ParticleNet: Jet Tagging via Particle Clouds*, Phys. Rev. D **101**(5), 056019 (2020), doi:10.1103/PhysRevD.101.056019, 1902.08570.

[5] E. A. Moreno, O. Cerri, J. M. Duarte, H. B. Newman, T. Q. Nguyen, A. Periwal, M. Pierini, A. Serikova, M. Spiropulu and J.-R. Vlimant, *JEDI-net: a jet identification algorithm based on interaction networks*, Eur. Phys. J. C **80**(1), 58 (2020), doi:10.1140/epjc/s10052-020-7608-4, 1908.05318.

[6] J. Shlomi, P. Battaglia and J.-R. Vlimant, *Graph neural networks in particle physics*, Machine Learning: Science and Technology **2**(2), 021001 (2020), doi:10.1088/2632-2153/abbf9a.

[7] H. Qu, C. Li and S. Qian, *Particle Transformer for jet tagging*, In *Proceedings of the 39th International Conference on Machine Learning*, pp. 18281–18292 (2022), 2202.03772.

[8] A. Bogatskiy, B. Anderson, J. Offermann, M. Roussi, D. Miller and R. Kondor, *Lorentz group equivariant neural network for particle physics*, In *Proceedings of the 37th International Conference on Machine Learning*, vol. 119 of *Proceedings of Machine Learning Research*, pp. 992–1002. PMLR (2020).

[9] S. Gong, Q. Meng, J. Zhang, H. Qu, C. Li, S. Qian, W. Du, Z.-M. Ma and T.-Y. Liu, *An efficient Lorentz equivariant graph neural network for jet tagging*, JHEP **07**, 030 (2022), doi:10.1007/JHEP07(2022)030, 2201.08187.

[10] C. Li, H. Qu, S. Qian, Q. Meng, S. Gong, J. Zhang, T.-Y. Liu and Q. Li, *Does Lorentz-symmetric design boost network performance in jet physics?*, Phys. Rev. D **109**(5), 056003 (2024), doi:10.1103/PhysRevD.109.056003, 2208.07814.

[11] J. Devlin, M.-W. Chang, K. Lee and K. Toutanova, *BERT: Pre-training of deep bidirectional transformers for language understanding*, In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 4171–4186, doi:10.18653/v1/N19-1423 (2019).

[12] K. He, H. Fan, Y. Wu, S. Xie and R. Girshick, *Momentum Contrast for Unsupervised Visual Representation Learning*, In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 9726–9735, doi:10.1109/CVPR42600.2020.00975 (2020).

[13] T. Chen, S. Kornblith, M. Norouzi and G. Hinton, *A simple framework for contrastive learning of visual representations*, In *Proceedings of the 37th International Conference on Machine Learning* (2020).

[14] M. Caron, H. Touvron, I. Misra, H. Jégou, J. Mairal, P. Bojanowski and A. Joulin, *Emerging properties in self-supervised vision transformers*, In *Proceedings of the International Conference on Computer Vision (ICCV)* (2021).

[15] M. Oquab, T. Darcet, T. Moutakanni, H. V. Vo, M. Szafraniec, V. Khalidov, P. Fernandez, D. HAZIZA, F. Massa, A. El-Nouby, M. Assran, N. Ballas *et al.*, *DINOv2: Learning robust visual features without supervision*, Transactions on Machine Learning Research (2024).

[16] O. Siméoni, H. V. Vo, M. Seitzer, F. Baldassarre, M. Oquab, C. Jose, V. Khalidov, M. Szafraniec, S. Yi, M. Ramamonjisoa, F. Massa, D. Haziza *et al.*, *DINOv3* (2025), 2508.10104.

[17] J. Zhou, C. Wei, H. Wang, W. Shen, C. Xie, A. Yuille and T. Kong, *ibot: Image bert pre-training with online tokenizer*, International Conference on Learning Representations (ICLR) (2022).

[18] M. Assran, Q. Duval, I. Misra, P. Bojanowski, P. Vincent, M. Rabbat, Y. LeCun and N. Ballas, *Self-supervised learning from images with a joint-embedding predictive architecture*, In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 15619–15629 (2023).

[19] B. M. Dillon, G. Kasieczka, H. Olischlager, T. Plehn, P. Sorrenson and L. Vogel, *Symmetries, safety, and self-supervision*, SciPost Phys. **12**, 188 (2022), doi:10.21468/SciPostPhys.12.6.188.

[20] L. Favaro, M. Krämer, T. Modak, T. Plehn and J. Rüschkamp, *Semi-visible jets, energy-based models, and self-supervision*, SciPost Phys. **18**(2), 042 (2025), doi:10.21468/SciPostPhys.18.2.042, 2312.03067.

[21] S. Katel, H. Li, Z. Zhao, F. Mokhtar, J. Duarte and R. Kansal, *Learning Symmetry-Independent Jet Representations via Jet-Based Joint Embedding Predictive Architecture*, In *Machine Learning and the Physical Sciences: Workshop at NeurIPS 2024* (2024), 2412.05333.

[22] T. Golling, L. Heinrich, M. Kagan, S. Klein, M. Leigh, M. Osadchy and J. A. Raine, *Masked particle modeling on sets: towards self-supervised high energy physics foundation models*, Mach. Learn. Sci. Tech. **5**(3), 035074 (2024), doi:10.1088/2632-2153/ad64a8, 2401.13537.

[23] M. Leigh, S. Klein, F. Charton, T. Golling, L. Heinrich, M. Kagan, I. Ochoa and M. Osadchy, *Is tokenization needed for masked particle modeling?*, Mach. Learn. Sci. Tech. **6**(2), 025075 (2025), doi:10.1088/2632-2153/addb98.

[24] P. Harris, J. Krupa, M. Kagan, B. Maier and N. Woodward, *Resimulation-based self-supervised learning for pretraining physics foundation models*, Phys. Rev. D **111**(3), 032010 (2025), doi:10.1103/PhysRevD.111.032010, 2403.07066.

[25] L. R. Sheldon, D. S. Rankin and P. Harris, *MACK: Mismodeling addressed with contrastive knowledge*, SciPost Phys. **18**(5), 150 (2025), doi:10.21468/SciPostPhys.18.5.150, 2410.13947.

[26] Z. Hao, R. Kansal, A. Gandrakota, C. Sun, J. Ngadiuba, J. Duarte and M. Spiropulu, *RINO: Renormalization Group Invariance with No Labels* (2025), 2509.07486.

[27] A. Bardes, J. Ponce and Y. LeCun, *Vicreg: Variance-invariance-covariance regularization for self-supervised learning*, In *ICLR* (2022).

[28] G. Hinton, O. Vinyals and J. Dean, *Distilling the knowledge in a neural network*, arXiv preprint arXiv:1503.02531 (2015).

[29] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit and N. Houlsby, *An image is worth 16x16 words: Transformers for image recognition at scale*, ICLR (2021).

[30] A. Sablayrolles, M. Douze, C. Schmid and H. Jégou, *Spreading vectors for similarity search*, In *International Conference on Learning Representations* (2019).

[31] R. Kansal, J. Duarte, H. Su, B. Orzari, T. Tomei, M. Pierini, M. Touranakou, J.-R. Vlimant and D. Gunopulos, *Particle Cloud Generation with Message Passing Generative Adversarial Networks*, In *35th Conference on Neural Information Processing Systems* (2021), 2106.11535.

[32] R. Kansal, J. Duarte, H. Su, B. Orzari, T. Tomei, M. Pierini, M. Touranakou, J.-R. Vlimant and D. Gunopulos, *Jetnet*, doi:10.5281/zenodo.6975118 (2022).

[33] M. Cacciari, G. P. Salam and G. Soyez, *The anti-k_t jet clustering algorithm*, JHEP **04**, 063 (2008), doi:10.1088/1126-6708/2008/04/063, 0802.1189.

[34] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever and R. Salakhutdinov, *Dropout: A simple way to prevent neural networks from overfitting*, Journal of Machine Learning Research **15**(56), 1929 (2014).

[35] D. Hendrycks and K. Gimpel, *Gaussian error linear units (gelus)*, arXiv preprint arXiv:1606.08415 (2016).

[36] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow *et al.*, *TensorFlow: Large-scale machine learning on heterogeneous systems*, Software available from tensorflow.org (2015).

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

[38] I. Loshchilov and F. Hutter, *Decoupled weight decay regularization*, In *International Conference on Learning Representations* (2019).

[39] J.-B. Grill, F. Strub, F. Altché, C. Tallec, P. Richemond, E. Buchatskaya, C. Doersch, B. Avila Pires, Z. Guo, M. Gheshlaghi Azar, B. Piot, k. kavukcuoglu *et al.*, *Bootstrap your own latent - a new approach to self-supervised learning*, In *Advances in Neural Information Processing Systems*, vol. 33, pp. 21271–21284 (2020).

[40] H. Bao, L. Dong, S. Piao and F. Wei, *BEit: BERT pre-training of image transformers*, In *International Conference on Learning Representations* (2022).

[41] X. Dong, J. Bao, T. Zhang, D. Chen, G. Shuyang, W. Zhang, L. Yuan, D. Chen, F. Wen and N. Yu, *Clip itself is a strong fine-tuner: Achieving 85.7% and 88.0% top-1 accuracy with vit-b and vit-l on imagenet*, arXiv preprint arXiv:2212.06138 (2022).

[42] K. Lee, K. Lee, H. Lee and J. Shin, *A simple unified framework for detecting out-of-distribution samples and adversarial attacks*, In *Advances in Neural Information Processing Systems*, vol. 31 (2018).

[43] Z. Hao, R. Kansal, J. Duarte and N. Chernyavskaya, *Lorentz group equivariant autoencoders*, Eur. Phys. J. C **83**(6), 485 (2023), doi:10.1140/epjc/s10052-023-11633-5, 2212.07347.

[44] D. Weitzel, A. Graves, S. Albin, H. Zhu, F. Wuerthwein, M. Tatineni, D. Mishin, E. Khoda, M. Sada, L. Smarr, T. DeFanti and J. Graham, *The national research platform: Stretched, multi-tenant, scientific kubernetes cluster*, In *Practice and Experience in Advanced Research Computing 2025: The Power of Collaboration*, PEARC '25. Association for Computing Machinery, New York, NY, USA, ISBN 9798400713989, doi:10.1145/3708035.3736060 (2025).