OmniLearn: A Method to Simultaneously Facilitate All Jet Physics Tasks

Vinicius Mikuni[1, *] and Benjamin Nachman[2, 3, †]

[1] *National Energy Research Scientific Computing Center, Berkeley Lab, Berkeley, CA 94720, USA*
[2] *Physics Division, Lawrence Berkeley National Laboratory, Berkeley, CA 94720, USA*
[3] *Berkeley Institute for Data Science, University of California, Berkeley, CA 94720, USA*

Machine learning has become an essential tool in jet physics. Due to their complex, high-dimensional nature, jets can be explored holistically by neural networks in ways that are not possible manually. However, innovations in all areas of jet physics are proceeding in parallel. We show that specially constructed machine learning models trained for a specific jet classification task can improve the accuracy, precision, or speed of all other jet physics tasks. This is demonstrated by training on a particular multiclass classification task and then using the learned representation for different classification tasks, for datasets with a different (full) detector simulation, for jets from a different collision system (*pp* versus *ep*), for generative models, for likelihood ratio estimation, and for anomaly detection. Our OmniLearn approach is thus a foundation model and is made publicly available for use in any area where state-of-the-art precision is required for analyses involving jets and their substructure.

CONTENTS

* vmikuni@lbl.gov
† bpnachman@lbl.gov

I. INTRODUCTION

The study of high-energy hadronic final states – jet physics – is seeing a paradigm shift from modern machine learning (ML). Jets are composed of many particles, each with properties of their own. This means that jets are represented in high-dimensional spaces and are thus difficult to analyze manually. There has been incredible progress over the last two decades in classical jet physics to develop observables and other techniques using direct physics reasoning [1–8], but the deep learning revolution of the last few years has shown that automation and indirect physics reasoning (e.g. through simulations and general physics considerations) can significantly improve performance in many tasks. For example, classifying a jet as originating from a top quark or a generic quark/gluon jet is over 20 times more effective with the latest deep learning solutions compared with classical methods [9–11]. Deep learning has also enabled new studies that were unimaginable before, like unbinned differential cross sections in tens or hundreds of dimensions simultaneously [12–21]. However, one feature these innovations have in common is that they are all advancing in parallel. Our question is simple: *can we make progress on all jet physics tasks at the same time?*

One answer to this question has been inspired by recent progress in large language modeling (LLMs). Tools like ChatGPT (`https://chat.openai.com`) and others [22] are called *foundation models* because they are able to approach many downstream tasks either with little or no finetuning to the specific problem. They have many millions (or billions/trillions) parameters and are trained using many millions (or much more) of examples. Recently, LLMs have also been adapted for particle physics-specific queries [23]. Such tools complement our approach: they target the process of particle physics research while we focus on processing particle physics data.

Foundation models are usually trained using a form of self-supervision (e.g. mask data and learn to fill in the blanks) in order to learn to represent the structure

of data. This sort of representation learning has been studied recently in particle physics using a number of interesting approaches [24–28]. As desired, the learned representations are also demonstrated to improve downstream tasks. However, the way these models are trained is not aligned with any actual analysis goal (we do not actually want to fill in blanks). Our intuition is that foundation models are useful because they increase the effective size of the training dataset for a downstream task. The closer the foundation model training is to the downstream task, the larger the increase in the effective size for a fixed sample size used to train the foundation model. A key difference between foundation models in (particle) physics and foundation models in society at large is the existence of *ab initio* simulations. These simulations provide large datasets that can be used to target specific tasks. Since we are in a privileged situation in which machine learning models can be constructed for dedicated problems, *can such a ML model dedicated to a specific jet physics task act as a foundation model?*

The process of (pre)training an ML model for one task and then applying it elsewhere, usually with some fine-tuning to the downstream task, is called *transfer learning*. This strategy has been shown to be effective in a number of applications across particle physics [29–32]. In all of these cases, there was only one downstream task. Furthermore, either the pretraining was very far from the target task (e.g. pretraining with generic non-physics images) or very close to the target task (e.g. classify jet of type A and transfer to type B). These papers were not trying to build foundation models, but we hypothesize that a significantly scaled up version of the transfer learning task can form the basis for a foundation model in particle physics. As such, our goal is to see if we can build a foundation model for jet physics by using a supervised instead of self-supervised learning. Our neural network will have many millions of parameters and will be trained on 100 million jets. We will achieve success if the implicit representation learned by our model improves (in training speed, accuracy, or both) essentially all downstream tasks in jet physics.

An approach with a similar philosophy to ours is OmniJet-α [33]. The authors of OmniJet-α use a generative model trained on a specific jet physics task in order to build a foundation model. A key difference is that OmniJet-α is based on a language model and so jet constituents are discretized (into 'tokens') and the generation is autoregressive (like generating a sentence from left to right). On the application side, OmniJet-α was trained on one generative task and applied on one classification task using the same dataset, whereas we try to explore most areas of jet physics across a variety of datasets with our model.

Our paper is organized as follows. Section II describes how jets can be represented as point clouds and the neural network architecture at the center of our foundation model. Just how OmniFold [12] can unfold all dimensions simultaneously, our approach – called OmniLearn – can learn useful representations for all jet physics tasks simultaneously. This is demonstrated in subsequent sections by showing that our model can enhance and/or accelerate tasks other than the one it was trained on (multi-class jets from fast simulation) to binary classification on a different dataset (Sec. III), to binary classification using a full detector simulation (Sec. IV), to jets originating from a different collision system (Sec. V), to generative models (Sec. VI), to likelihood ratio estimation (Sec. VII), and to anomaly detection (Sec. VIII). The paper ends with conclusions and outlook in Sec. IX.

II. POINT CLOUDS FOR JET PHYSICS

The interpretation of jets as point clouds motivated the development of new neural network architectures to naturally address the data structure represented by an unordered set of particles with varying number of constituents. Successful neural network models using sets [34, 35], graph neural networks (GNNs) [36–38], and more recently transformers [39–41], were all able to successfully improve upon previous deep learning approaches in collider physics. The first transformer model in high energy physics [39] used graph-attention networks [42], combining the advantages of both GNNs and transformers. The rise in popularity of transformer models is also attributed to their strong scaling properties [43], making it the backbone of almost all modern foundational models and the choice of neural network architecture used to build OmniLearn.

For collider physics applications, we expect that in the presence of large datasets and suitable tasks, a foundational model for jets will be able to leverage the information to learn a general representation of jets, thus providing a stronger inductive bias that will be transferable to different datasets and tasks. We also notice that the overwhelming majority of machine learning applications for jet physics can be summarized as a classification or generation tasks. These include the applications in jet tagging and jet generation as well as complex tasks such as event reweighting, unfolding, and anomaly detection. Motivated by this observation, we conjecture that a flexible model, capable of learning both to generate and classify jets, will also learn a useful and general data representation that can be employed to quickly adapt to new downstream datasets and applications.

While multiple generative models have been proposed for point cloud generation in particle physics based on generative adversarial networks [44, 45], variational autoencoders [46], normalizing flows [47, 48], and autoregressive transformers [49, 50], we choose to focus on diffusion generative models [51]. Diffusion generative models use time-dependent perturbations applied to the data to learn an approximation of the score function, or gradients of the logarithm of the probability density of the data. This choice also aligns with the classification task. Since the perturbation process is designed such that no

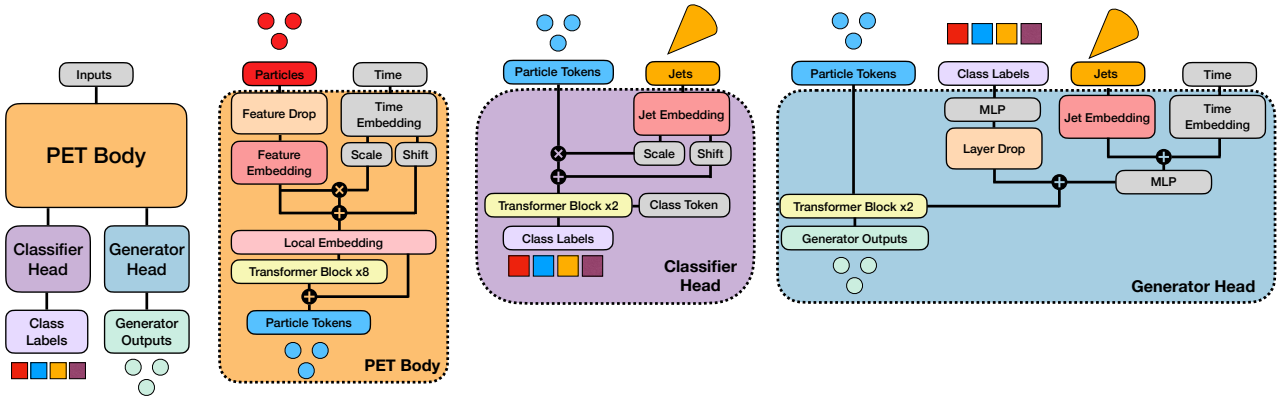

FIG. 1. Neural network architecture used to train OmniLearn. The main neural network blocks of the architecture are shown in the further left with detailed architecture design shown for each block in the right. See the text for more details.

perturbation is applied at the initial time $t = 0$, conditioning the model over the time parameter ensures the network is able to accommodate both perturbed and unperturbed data simultaneously. To build OmniLearn, we design the architecture with a shared representation whose outputs are then fed to task-specific neural networks. This approach enables flexibility and efficient design, since downstream task applications only need to load the shared representation and relevant task-specific network, reducing the overall model. The main building blocks of the network are summarized in Fig. 1. In the following subsections we will provide a detailed description of the model and the core design choices.

A. Point-Edge Transformer

The shared representation of the network takes as inputs the particles clustered inside the jets and is conditioned on the diffusion time parameter. The time information, following previous diffusion models for collider physics [20, 52–55], is encoded to a higher dimensional space using a time embedding layer. The time embedding consists Fourier features [56] followed by two multi-layer perceptrons (MLPs) with GELU activation function [57]. Unless otherwise stated, all MLP layers used in this work are followed by a GELU non-linear activation. Contrary to previous diffusion models, we modify the time embedding by multiplying the output of the Fourier features by the time parameter, such that the output of the time embedding is zero when the input time is also zero. This choice ensures the time embedding is effectively turned off when the model is evaluated in classifier mode. The next step is to combine the time information with the input particle information. Datasets store different levels of information for each particle. The most basic information, described by the kinematic information of each particle, is always stored. However, additional information such as particle identification (PID) and vertex information for charged particles is only available in specific datasets, with the latter only provided in the JetClass dataset [41] among benchmark, public jet datasets. To avoid training multiple models to accommodate each dataset, and thus defeating the purpose of a generalized model, we instead adopt a feature drop approach. During training, we consider as inputs both the kinematic information for each particle and their respective PID[1]. With a probability $p = 0.2$ we drop the PID information by replacing it with zeros. This approach is similar to dropout layers [58] that encourage the network to learn a useful representation both in the presence and absence of these features. After the feature drop, the inputs are encoded to a higher dimensional space using a feature embedding consisting of two MLP layers. The outputs of the feature embedding are then combined with the time information though a shift and scaling operation. Before the transformer block we introduce a positional token to encode the geometrical information of the neighborhood surrounding each particle inside the jet. Even though transformers are capable of learning general correlations between particles, the addition of local information can generally improve performance [40], creating a better latent representation that is aware of the distances between particles. We create the local encoding using dynamic graph convolution (DGCNNs) layers where the neighborhood is defined using a k-nearest neighbor algorithm with number of neighbors fixed to 10. The distances are calculated in the pseudorapidity-azimuthal angle space. For each of the k-neighbors, edge features are defined based on the particle features concatenated with the subtraction between the particle features and each of the respective neighbors. An MLP is used over all edges before an average pooling operation over the neighbor dimen-

[1] The vertex information while present in the JetClass dataset is not used during training. While that could also be included in the training methodology, our focus is the application to multiple datasets without this information, hence for simplicity we skip these features.

sion. The result of the operation represents the new particle features. A second DGCNN layer is then created with distances calculated based on the Euclidean distances between particles after the updated features. This second operation allows the dynamic construction of edges between particles in a learned geometrical space that pulls relevant particles closer together and far apart otherwise. The particle embedding equipped with local information is then passed through multiple transformer blocks. The transformer block closely follows the original proposal [59], combining multi-head attention modules with additional skip connections. Our main modification is the addition of LayerScale [60] layers. The LayerScale operation introduces a learnable multiplicative factor to each skip connection in the transformer block. Using a small initial value (10^{-5} in our implementation), the operation improves the stability and convergence of the model by allowing each transformer block to learn the relevance of each attention block in the transformer architecture. The outputs of the transformer blocks are then added to the original particle embedding after the combination with the local tokens, improving the information flow over the entire model. The outputs are then used as inputs of the task-specific blocks. Since our proposed backbone architecture combines edge creation with transformer modules we refer to it as Point-Edge Transformer (PET), noting that OmniLearn refers to the join training strategy, while PET refers to the specific neural network architecture used to implement OmniLearn.

B. Classifier Head

The classification task across different jet types is accomplished through a dedicated classification head that takes as inputs the particles after the shared network, now referred to as the PET body. Additionally, we also include the overall information from the jet kinematics, including the jet mass, transverse momentum p_T, pseudorapidity η, and particle multiplicity. Even though the additional information is partially redundant compared to the initial particle features, the addition of the jet kinematic information helps the model converge faster when evaluated over datasets covering different fiducial regions than the ones used during training. The jet information is embedded in a higher dimensional space using a jet embedding layer that includes two MLP layers of same size as the current particle embedding dimensionality. This information is then combined with the outputs of the PET body through a scaling and shift operations. A trainable class token [61] is then used to summarize the information of the particle embeddings before the classification output. The class token is essentially interpreted as an additional particle, concatenated to the true particle inputs. Inside the transformer block, the outputs of the PET body are not updated but only the class token is allowed to change at the end of each transformer block. The output predictions are then determined by passing the updated class tokens over one last MLP with output size determined by the number of classes in the dataset.

C. Generator Head

Similarly to the classification head, the generator head takes as inputs both particle embeddings and jet kinematic information. Additionally, we include the time information and the set of class labels to condition the generator over the jet types to be simulated. The time and jet information are embedded in a higher dimensional space using the same encoding blocks used in the PET body and classifier head, respectively. This information is then combined through an addition operation followed by an MLP. The classification labels are also mapped to a higher dimensional space using a single MLP. The outputs of the MLP are then passed through a layer drop operation that, similarly to the feature drop, has a probability $p = 0.1$ to replace the entire output of the MLP with zeros. This choice is motivated by two observations: when the model is used during downstream tasks, the classes used to condition the PET architecture are hardly going to be the same as the ones used during training. Randomly ignoring the class labels encourages the entire architecture to learn both a general and specialized representation, leading to quicker convergence when adapted to other datasets. Second, this technique is similar to classifier-free guidance, observed to improve the generation quality of diffusion models [62]. The results of the layer drop operation are then added to the outputs of the combined jet and time embeddings. The result of this combination is then used as a diffusion token, where similarly to the classification token, is tasked to summarize the particle embedding information inside the transformer block. However, while the classification token is interpreted as an additional particle, we interpret the diffusion token as a conditional shift of the particle embeddings produced by the PET body. Initially, all particles are simultaneously shifted by the diffusion token created from the combined class labels, time, and jet information. The diffusion tokens are then updated with every transformer layer. The diffusion prediction is then the sum of the original PET body outputs with the learned diffusion tokens.

D. Loss Function

The loss function of OmniLearn consists of multiple terms designed to combine both the classification and generation tasks. Each of the terms is shown in Equation 2.

$$\mathcal{L} = \mathcal{L}_{\text{class}} + \mathcal{L}_{\text{gen}} + \mathcal{L}_{\text{class smear}} \tag{1}$$

$$= \text{CE}(y, y_{\text{pred}}) + \|\mathbf{v} - \mathbf{v}_{\text{pred}}\|^2 + \alpha^2 \text{CE}(y, \hat{y}_{\text{pred}})\,. \tag{2}$$

For input data $\mathbf{x}$, the cross-entropy (CE) loss is calculated using the output of the PET classifier y_{pred} and true class labels y. The PET generator takes as inputs perturbed data $\hat{\mathbf{x}} = \alpha(t)\mathbf{x} + \sigma(t)\boldsymbol{\epsilon}$ with time-dependent perturbation parameters $\alpha(t)$ and $\sigma(t)$ and predicts a velocity parameter $\mathbf{v}_{\text{pred}}$ that is compared with the true velocity value $\mathbf{v} = \alpha(t)\boldsymbol{\epsilon} - \sigma(t)\mathbf{x}$. Additionally, the perturbed inputs can also be interpreted as a form of data augmentation that can further improve the classifier performance. We evaluate the PET classifier over the perturbed inputs to get the predictions $\hat{y}_{\text{pred}}$ used in the calculation of the cross entropy loss. A weight of $\alpha(t)^2$ is applied to ensure that at $t = 0$, where $\alpha(0) = 1$ and $\sigma(0) = 0$, we recover the classifier loss over clean inputs and at $t = 1$, where $\alpha(1) = 0$ and $\sigma(1) = 1$, the completely corrupted data does not negatively impact the classification performance.

E. Training Details

We train OmniLearn using the JetClass dataset [41]. A total of 10 different jet classes are provided, simulated using MADGRAPH5_aMC@NLO [63] for the matrix element calculation and Pythia [64, 65] to perform the parton showering and hadronization. Detector effects are simulated using Delphes [66–68] with the CMS detector configuration. Jets are clustered using the anti-k_T algorithm with radius parameter of $R = 0.8$. [69–71]. Jets with transverse momentum between 500-1000 GeV and pseudorapidity $|\eta| < 2.0$ are saved. The training dataset consists of 100M jets, divided equally between each of the 10 classes. The kinematic information and PID for each particle is used as input features for the training and listed in Appendix A.

Up to 150 particles are saved per jet to be used during training. The training is carried out on the Perlmutter Supercomputer [72] using 128 GPUs simultaneously with Horovod [73] package for data distributed training. A local batch size of size 256 is used with model training up to 200 epochs. OmniLearn is implemented in TensorFlow [74] with Keras [75] backend. The cosine learning rate schedule [76] is used with an initial learning rate of 3×10^{-5}, increasing to $3\sqrt{128} \times 10^{-5}$ after three epochs and decreasing to 10^{-6} until the end of the training. The Lion optimizer [77] is used with parameters $\beta_1 = 0.95$ and $\beta_2 = 0.99$. The fine-tuning of OmniLearn across different datasets and tasks is performed by setting the learning rate of the PET body to be a factor 10 smaller than the learning rate used in the rest of the PET architecture. The PET body model has 1.3M trainable weights, while the classifier and generator heads have 268k and 416k trainable parameters, respectively.

In every application, all pre-trained weights are loaded unless there is a change in input dimensions. This is only the case for the output layer of the classification model and the input class labels in the generator model, where the number of classes used in each downstream task changes. In these cases, a new layer with correct input and output sizes replaces the trained weights and is initialized using random weights. We evaluate OmniLearn across 10 different datasets with results described in the following sections.

III. GENERALIZATION ACROSS JET TYPES

TABLE I. Comparison between the performance reported for different classification algorithms on the top tagging dataset. The uncertainty quoted corresponds to the standard deviation of five trainings with different random weight initialization. If the uncertainty is not quoted then the variation is negligible compared to the expected value. Bold results represent the algorithm with highest performance.

	Acc	AUC	$1/\epsilon_B$	
			$\epsilon_S = 0.5$	$\epsilon_S = 0.3$
ResNeXt-50 [37]	0.936	0.9837	302 ± 5	1147 ± 58
P-CNN [37]	0.930	0.9803	201 ± 4	759 ± 24
PFN [34]	-	0.9819	247 ± 3	888 ± 17
ParticleNet [37]	0.940	0.9858	397 ± 7	1615 ± 93
JEDI-net [36]	0.9300	0.9807	-	774.6
PCT [40]	0.940	0.9855	392 ± 11	1559 ± 98
LGN [78]	0.929	0.964	-	435 ± 95
rPCN [38]	-	0.9845	364 ± 9	1642 ± 93
LorentzNet [10]	0.942	0.9868	498 ± 18	2195 ± 173
PELICAN [79]	0.9425	0.9869	2289 ± 204	-
ParT [41]	0.940	0.9858	413 ± 16	1602 ± 81
ParT-f.t. [41]	**0.944**	**0.9877**	**691 ± 15**	**2766 ± 130**
PET Classifier	0.938	0.9848	340 ± 12	1318 ± 39
OmniLearn	0.942	0.9872	568 ± 9	**2647 ± 192**

TABLE II. Comparison between the performance reported for different classification algorithms on the quark and gluon dataset. The uncertainty quoted corresponds to the standard deviation of nine trainings with different random weight initialization. If the uncertainty is not quoted then the variation is negligible compared to the expected value. Bold results represent the algorithm with highest performance.

	Acc	AUC	$1/\epsilon_B$	
			$\epsilon_S = 0.5$	$\epsilon_S = 0.3$
P-CNN [37]	0.827	0.9002	34.7	91.0
PFN [34]	-	0.9005	34.7±0.4	-
ParticleNet [37]	0.840	0.9116	39.8±0.2	98.6±1.3
rPCN [38]	-	0.9081	38.6 ± 0.5	-
ParT [41]	0.840	0.9121	41.3 ± 0.3	101.2 ± 1.1
ParT-f.t. [41]	0.843	0.9151	42.4 ± 0.2	**107.9 ± 0.5**
PET classifier	0.837	0.9110	39.92±0.1	104.9 ± 1.5
OmniLearn	**0.844**	**0.9159**	**43.7±0.3**	**107.7 ± 1.5**

We first evaluate OmniLearn on two widely-used benchmark datasets for jet tagging: the top quark tagging [9] and quark/gluon [34] classification. In the top quark tagging dataset, events are simulated using Pythia 8 and Delphes with the ATLAS configuration.

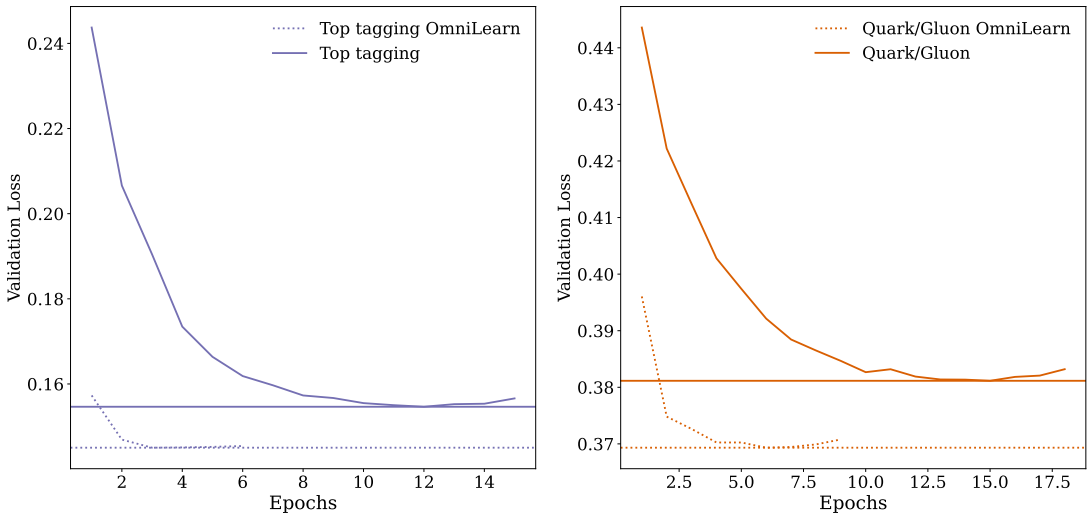

FIG. 2. Validation loss curves obtained in the top quark tagging (left) and quark/ gluon (rights) datasets. The OmniLearn validation loss is compared with the PET classifier trained from scratch.

The background consists of dijets produced via QCD and the signal consists of top quark pair production with all-hadronic decays. The default energy flow algorithm in Delphes is used to create jet constituents, which are clustered using the anti-k_T algorithm with $R = 0.8$. All jets in the range 550 GeV $< p_T <$ 650 GeV and $|\eta| < 2$ are saved. Note that while top quark and QCD categories are present in the JetClass dataset, the Delphes detector configuration and p_T ranges are different. The quark/gluon dataset consists of stable particles clustered into jets, excluding neutrinos, using the anti-k_T algorithm with radius R = 0.4. The quark-initiated sample (signal) is generated using a $Z(\nu\nu) + (u, d, s)$ while the gluon-initiated data (background) are generated using $Z(\nu\nu) + g$ processes. Both samples are generated using Pythia8 without detector effects. Jets are required to have transverse momentum $p_T \in [500, 550]$ GeV and rapidity $|y| < 1.7$ for the reconstruction. For each dataset we evaluate the results of adapting OmniLearn on each dataset and compare with the training carried out with the PET classifier architecture from scratch. The results are compared with other models using the same datasets in Tables I and II.

In both cases we observe the performance obtained by OmniLearn to be significantly better than other models trained from scratch, while matching and sometimes surpassing the performance observed by the fine-tuned version of the state-of-the-art model ParT [41]. In Figure. 2, we show the loss curve obtained during training in the validation set. After a single epoch, OmniLearn already reaches the performance from the PET classifier with convergence observed after only three epochs in both datasets, reducing the overall training time by a factor 3.

IV. GENERALIZATION ACROSS DETECTORS

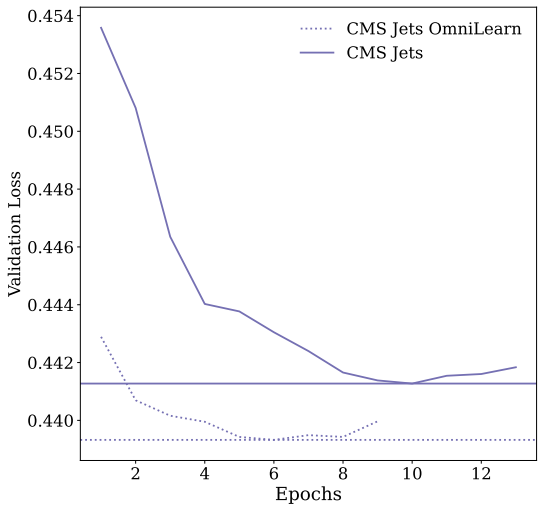

FIG. 3. Validation loss curves obtained in the CMS Open Data quark gluon tagging dataset. The OmniLearn validation loss is compared with the PET classifier trained from scratch.

A more realistic scenario is to consider a complete simulation of the detector response. In this scenario, the generalization power from OmniLearn trained on fast simulations could greatly reduce the need for large simulation samples with full detector simulation for the training of particle taggers at the LHC or be quickly deployed using open data released by the LHC Experiments. We investigate this scenario using the public CMS Open Data [80] and ATLAS Top tagging dataset [81]. We use the simulations from the CMS Open Data release for 2011A run period [82] processed by the MIT Open Data soft-

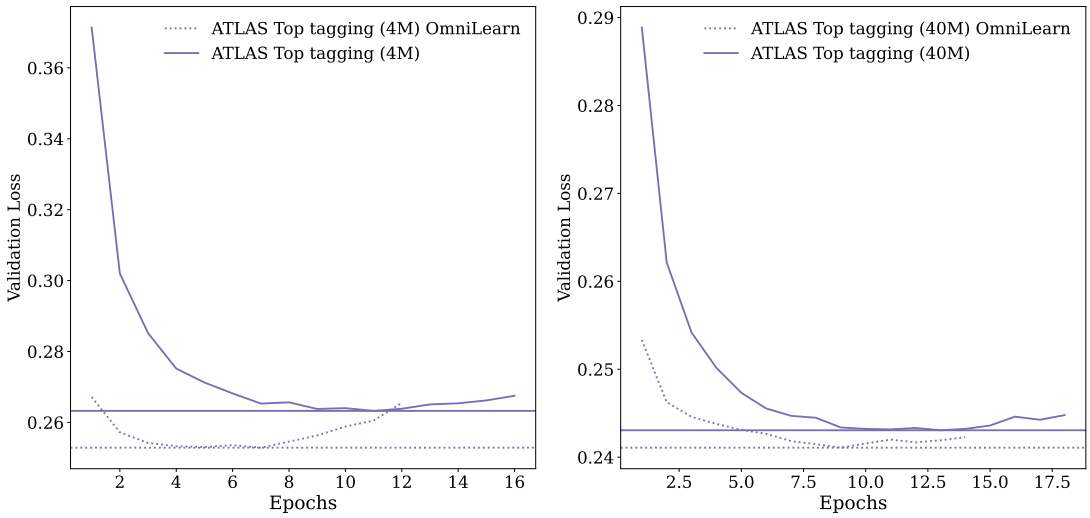

FIG. 4. Validation loss curves obtained in the ATLAS top quark tagging dataset trained using 4M (left) and 40M (rights) examples. The OmniLearn validation loss is compared with the PET classifier trained from scratch.

TABLE III. Comparison between the performance reported for different classification algorithms on the CMS Open Data dataset. Bold results represent the algorithm with highest performance.

	AUC	Acc	$1/\epsilon_B$	
			$\epsilon_S = 0.5$	$\epsilon_S = 0.8$
PET classifier	0.875	0.796	23.91 ± 0.07	4.770 ± 0.001
OmniLearn	**0.877**	**0.797**	**24.36 ± 0.01**	**4.836 ± 0.004**

TABLE IV. Comparison between the performance reported for different classification algorithms on the ATLAS top tagging dataset. Bold results represent the algorithm with highest performance.

	AUC	Acc	$1/\epsilon_B$	
			$\epsilon_S = 0.5$	$\epsilon_S = 0.8$
ResNet 50	0.885	0.803	21.4	5.13
EFN	0.901	0.819	26.6	6.12
hlDNN	0.938	0.863	51.5	10.5
DNN	0.942	0.868	67.7	12.0
PFN	0.954	0.882	108.0	15.9
ParticleNet	0.961	0.894	153.7	20.4
PET classifier (4M)	0.959	0.890	146.5	19.4
OmniLearn (4M)	0.961	0.894	172.1	20.8
PET classifier (40M)	0.964	0.898	201.4	23.6
OmniLearn (40M)	**0.965**	**0.899**	**207.30**	**24.10**

ware [83] to select simulated QCD jets in proton-proton collisions at a center-of-mass energy of $\sqrt{s} = 7$ TeV produced with Pythia6 [64] and Geant4 [84] for detector effects. Particle-Flow objects [85] are used to define the objects that are clustered into jets using the anti-k_T algorithm with radius parameter 0.5 with additional requirement to have $p_T > 375$ GeV to achieve high trigger efficiency. The jet flavor is defined by the hard parton associated to the jet. We define quark-initiated jets as jets associated to uds partons and save these jets for the classification task against gluon-initiated jets. For simplicity, we fix the number of quark and gluon jets to be the same and around 20M events in total. We train OmniLearn using 70% of the simulated jets and report the results compared to the PET classifier trained from scratch using a separate test set consisting of 10% of the events in Table III. We observe the performance achieved by OmniLearn to be better in all metrics compared to a classifier trained from scratch. This observation is encouraging since OmniLearn shows that classification performance can be enhanced even when available datasets are of similar size as the ones used to train OmniLearn. Moreover, OmniLearn is able to converge 2 times faster and to a better minimum than training from scratch as shown by the progression of the validation loss in Figure 3.

Next, we use the data released by the ATLAS Collaboration consisting of top quarks and QCD jets simulated using the ATLAS detector simulation based on Geant4. Events are generated with Pythia8 using the NNPDF2.3LO [86] set of parton distribution functions and the A14 [87] set of tuned parameters. Pileup effects are simulated by overlaying inelastic interactions on top of the underlying hard scattering process based on the 2017 data taking period. Hadronic boosted top quarks are obtained in simulated events containing the decay of a heavy Z' boson with mass of 2 TeV. The cross section of this process is reweighted to produce an approximately flat jet p_T distribution to efficiently populate the full kinematic region. Background QCD jets are obtained in simulated events containing pairs of light quarks or gluons. Unified Flow Objects [88] are used to determine the jet constituents. Jets are clustered using anti-k_T algorithm with R=1.0 with additional pileup mitigation algorithms [89–91] applied. The Soft-Drop algorithm [92] is also applied to remove soft and wide-angle radiation.

We investigate two scenarios. One where we fine-tune OmniLearn using all 40M training events available in the dataset and one where we evaluate the fine-tuning performance using only 4M events. Results are shown in Table. IV compared to other architectures trained using the same dataset and with the PET classifier trained from scratch.

Once again, OmniLearn is able to achieve state-of-the-art performance, surpassing the results reported by all previous models trained using the same dataset. The difference between the OmniLearn and PET classifiers trained over the entire dataset is less significant since the 40M training events are only a factor 2 smaller than the entire JetClass dataset. Conversely, OmniLearn fine-tuned on 4M events already matches the performance of all previous models trained from scratch using the entire dataset, requiring less examples to achieve the same performance. In Figure. 4 we show the validation loss curves obtained during the training. In both cases, OmniLearn already starts from a lower loss and converges quicker to a lower value than the classifier trained from scratch.

V. GENERALIZATION ACROSS COLLISION SYSTEMS

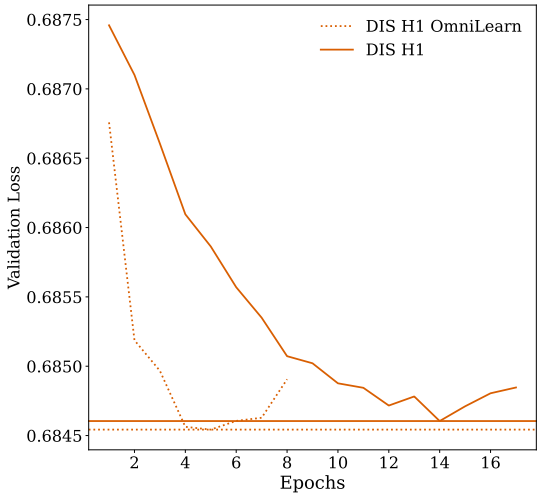

FIG. 5. Validation loss curves obtained in the DIS dataset. The OmniLearn validation loss is compared with the PET classifier trained from scratch.

TABLE V. Comparison between the performance reported for different classification algorithms on the DIS dataset. Bold results represent the algorithm with highest performance.

	AUC	Acc	$1/\epsilon_B$	
			$\epsilon_S = 0.1$	$\epsilon_S = 0.5$
PET classifier	0.5691	0.547	17.73±0.04	2.467±0.002
OmniLearn	0.5695	0.547	17.78±0.06	2.470±0.003

We also evaluate the generalization capability of OmniLearn for jets measured in different collision systems, covering completely different fiducial regions of the phase space compared to the LHC. We use simulations of neutral-current deep inelastic scattering (DIS) generated using the Rapgap 3.1 [93] generator for electron-proton collisions with electron and proton beam energies of 27.6 GeV and 920 GeV, respectively. The simulations are provided by the H1 Collaboration using the Heracles routines [94–96] for QED radiation, CTEQ6L PDF set [97], and the Lund hadronization model [98]. The detector simulation is performed using the Geant3 [99] package. An energy-flow algorithm [100–102] is then used to reconstruct the particles clustered into jets using the k_T algorithm with R=1.0. A second simulation using the Djangoh 1.4 [103] generator is used during the classification. The task is to separate jets between the two different simulations. Notice that both simulations target the description of DIS events, hence their differences are more subtle than the previous classification tasks investigated thus far. This choice of classification problem is motivated by previous studies on the unfolding of jet substructure observables [104]. One of the leading uncertainties, the closure test performed between two different simulations, is carried out using the same simulation routines and classification task used in this study, thus any improvements driven by OmniLearn could also lead to better unfolding algorithms to be developed for current and future unfolding analyses. A total of 2.5M events are generated for each simulation and used during the training. The performance obtained is listed in Table V.

We observe the same level of performance between OmniLearn and the PET classifier. Compared to the previous classification tasks, both accuracy and AUC values are much lower, evidencing the challenge of distinguishing jets from the two simulations apart. While the final performance is the same, we observe a much quicker convergence from OmniLearn as shown in the validation loss curve in Figure 5, resulting in a factor 3.5 faster training compared to starting from scratch. As described in [104], the full determination of the uncertainties require the training of thousands of classifiers, making any improvements in training speed an important asset for the applicability of the algorithm. These results also highlight the capability of OmniLearn to quickly adapt to new datasets, providing a strong asset for future experimental facilities such as the EIC.

VI. CONDITIONAL GENERATION

Next, we evaluate the generation quality obtained by OmniLearn using the JetNet [44] datasets consisting of jets initiated by light-quarks, gluons, top quarks, W and Z bosons. The jets are generated with transverse momenta p_T around 1 TeV and are clustered using the anti-k_t algorithm with a radius parameter of 0.4. Each jet has a maximum number of particles stored fixed to 30 [105]

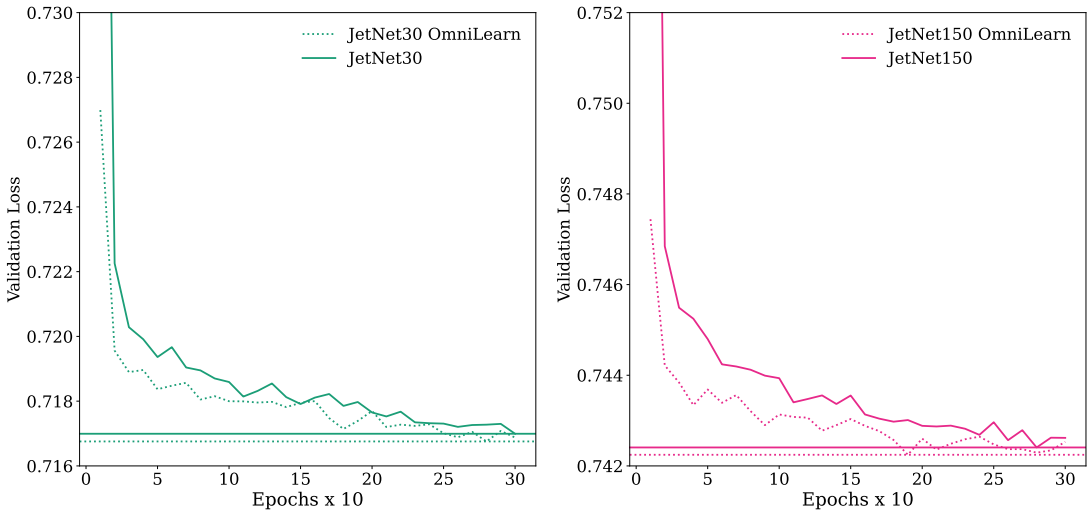

FIG. 6. Validation loss curves obtained in the JetNet dataset with 30 particles (left) 150 particles. The OmniLearn validation loss is compared with the PET classifier trained from scratch.

or 150 [106]. For each jet, the four-momentum information $(p_{T_{jet}}, \eta_{jet}, \phi_{jet}, m_{jet})$ is provided, as well as the particle multiplicity. We adopt the two-model strategy presented in [52], training a model that only learns the kinematic information of the jets and using that information as conditional information for the diffusion model trained using particles as inputs. Note that in this case, OmniLearn is used only to learn the particle information while the overall jet kinematic information model is always trained from scratch. The performance of the generation is evaluated using multiple physics-based metrics proposed in [44] and listed in Tables VI and VII. We also provide the results of OmniLearn and the PET generator in the ideal case, where the jet information, used to condition the particle generation model, is taken directly from the validation set of the JetNet dataset, effectively separating the impact of the jet generation, that does not benefit from OmniLearn, from the particle generation process.

In all metrics investigated in this study, OmniLearn shows similar or improved performance compared to previous models for both datasets consisting of 30 and 150 particles. We also notice that differences between the idealized version of OmniLearn and PET generator are more significant than at full generation level where the jet generation quality also affects the overall model performance. Additionally, statistical uncertainties determined from the bootstrapping method are often above the 20% level for some metrics, requiring access to larger evaluation sample sizes to provide a more precise comparison between models. In Figure 6 we show the validation loss curve for the PET generator and OmniLearn training in the JetNet dataset. Similarly to previous results, the OmniLearn training starts from a lower value of the loss function and is able to converge quicker, requiring roughly 20% and 30% fewer training epochs than a model trained from scratch.

VII. REWEIGHING AND UNFOLDING

Unfolding, or correcting observables for detector effects, is a fundamental task in collider physics to enable efficient comparisons between measurements and theory predictions. Traditional unfolding methods use histograms to estimate the unfolded response through a regularized inversion of the detector response matrix [107–113]. Machine learning greatly increase the flexibility and potential of unfolding by allowing the simultaneous correction of multiple distributions without the use of histograms [17]. The OmniFold algorithm [12, 16] introduced an iterative approach for unfolding based on reweighting functions. In the first iteration of the method, a reweighting function that corrects simulations towards measured data, is determined using a classifier. We examine the potential of OmniLearn applied to high-dimensional reweighting and complete unfolding using all available features and the same dataset introduced in Ref. [12], available on Zenodo [114]. The dataset consists of proton-proton collisions producing a Z boson, generated at a center-of-mass energy of $\sqrt{s} = 14$ TeV. A sample used as the 'data' representative is simulated using particle collisions with the default tune of Herwig 7.1.5 [115–117]. A second dataset, representative of the 'simulation' we want to correct, is simulated using Pythia8 with Tune 26 [118]. Detector distortions are simulated with Delphes and the CMS tune that uses a particle flow reconstruction. Jets are clustered using all particle flow objects at detector level and all stable non-neutrino truth particles at particle level. They are defined by the anti-k_T algorithm with radius parameter $R = 0.4$ as implemented in FastJet 3.3.2. The Z bosons

TABLE VI. Comparison of the results obtained between different generative models in the task of particle property generation in the dataset consisting of 30 particles. Lower is better for all metrics except Cov. FPND metrics are not available for W and Z bosons, hence omitted.

Jet class	Model	W_1^{PM} ($\times 10^{-3}$)	W_1^{P} ($\times 10^{-3}$)	W_1^{PEFP} ($\times 10^{-5}$)	FPND	Cov↑	MMD
Gluon	FPCD [52]	**0.36 ± 0.08**	**0.34 ± 0.09**	0.47 ± 0.13	0.07	0.55	0.03
	FPCD 1 [52]	0.65 ± 0.11	**0.34 ± 0.06**	0.60 ± 0.09	0.11	0.55	0.03
	MP-GAN [44]	0.69 ± 0.07	1.8 ± 0.2	0.9 ± 0.6	0.20	0.54	0.037
	EPiC-GAN [45]	**0.3 ± 0.1**	1.6 ± 0.2	0.4 ± 0.2	1.01 ± 0.07	-	-
	PET generator	0.42 ± 0.10	**0.36 ± 0.08**	**0.35 ± 0.08**	0.04	0.55	0.03
	PET generator (Ideal)	**0.36 ± 0.08**	**0.34 ± 0.09**	0.47 ± 0.13	0.07	0.55	0.03
	OmniLearn	**0.38 ± 0.08**	**0.33 ± 0.07**	**0.33 ± 0.09**	**0.02**	0.55	0.03
	OmniLearn (Ideal)	**0.33 ± 0.06**	**0.29 ± 0.08**	**0.30 ± 0.07**	**0.02**	0.55	0.03
Light Quark	FPCD [52]	0.52 ± 0.07	**0.27 ± 0.06**	0.38 ± 0.11	0.08	0.49	0.02
	FPCD 1 [52]	0.59 ± 0.08	0.36 ± 0.08	0.50 ± 0.08	0.09	0.48	0.02
	MP-GAN [44]	0.6 ± 0.2	4.9 ± 0.5	0.7 ± 0.4	0.35	0.50	0.026
	EPiC-GAN [45]	0.5 ± 0.1	4.0 ± 0.4	0.8 ± 0.4	0.43 ± 0.03	-	-
	PET generator	0.39 ± 0.12	0.35 ± 0.06	**0.24 ± 0.10**	0.03	**0.54**	0.02
	PET generator (Ideal)	0.31 ± 0.08	0.38 ± 0.10	**0.23 ± 0.07**	0.03	0.53	0.02
	OmniLearn	**0.24 ± 0.03**	**0.32 ± 0.07**	**0.24 ± 0.08**	0.02	**0.54**	0.02
	OmniLearn (Ideal)	0.31 ± 0.08	**0.30 ± 0.09**	**0.26 ± 0.08**	**0.01**	**0.54**	0.02
Top Quark	FPCD [52]	0.51 ± 0.07	0.41 ± 0.12	1.25 ± 0.19	0.17	0.58	0.05
	FPCD 1 [52]	1.22 ± 0.09	0.46 ± 0.10	2.66 ± 0.26	0.56	0.57	0.05
	MP-GAN [44]	0.6 ± 0.2	2.3 ± 0.3	2 ± 1	0.37	0.57	0.071
	EPiC-GAN [45]	0.5 ± 0.1	2.1 ± 0.1	1.7 ± 0.3	0.31 ± 0.037	-	-
	PET generator	0.44 ± 0.03	**0.29 ± 0.07**	**1.09 ± 0.23**	0.07	0.58	0.05
	PET generator (Ideal)	**0.41 ± 0.07**	**0.34 ± 0.08**	**1.22 ± 0.23**	0.07	0.58	0.05
	OmniLearn	0.43 ± 0.06	**0.30 ± 0.07**	1.31 ± 0.18	0.04	0.58	0.05
	OmniLearn (Ideal)	**0.36 ± 0.05**	0.41 ± 0.08	**1.02 ± 0.20**	**0.03**	0.58	0.05
W Boson	FPCD [52]	0.26 ± 0.03	0.39 ± 0.08	0.15 ± 0.02	-	0.56	0.02
	FPCD 1 [52]	0.94 ± 0.06	0.42 ± 0.09	0.35 ± 0.03	-	0.56	0.02
	PET generator	**0.17 ± 0.04**	**0.26 ± 0.05**	**0.11 ± 0.02**	-	0.56	0.02
	PET generator (Ideal)	**0.15 ± 0.02**	**0.31 ± 0.07**	**0.12 ± 0.03**	-	**0.57**	0.02
	OmniLearn	**0.19 ± 0.03**	**0.27 ± 0.07**	**0.10 ± 0.02**	-	**0.57**	0.02
	OmniLearn (Ideal)	**0.16 ± 0.06**	**0.28 ± 0.04**	**0.10 ± 0.02**	-	**0.57**	0.02
Z Boson	FPCD [52]	**0.21 ± 0.04**	0.40 ± 0.13	0.18 ± 0.03	-	0.56	0.02
	FPCD 1 [52]	0.99 ± 0.05	0.35 ± 0.06	0.49 ± 0.03	-	0.56	0.02
	PET generator	**0.22 ± 0.04**	**0.32 ± 0.07**	0.20 ± 0.04	-	**0.57**	0.02
	PET generator (Ideal)	**0.18 ± 0.10**	**0.30 ± 0.08**	**0.14 ± 0.02**	-	0.56	0.02
	OmniLearn	**0.19 ± 0.07**	**0.32 ± 0.09**	**0.12 ± 0.03**	-	**0.57**	0.02
	OmniLearn (Ideal)	**0.22 ± 0.05**	**0.27 ± 0.06**	**0.13 ± 0.02**	-	**0.57**	0.02

are required to have $p_T > 200$ GeV in order to mitigate acceptance effects. While all clustered particles are used during the training, we report the performance obtained by different algorithms using the same set of observables reported in the original OmniFold publication.

First, we investigate the capability of OmniLearn to determine the full phase space reweighting function by fine-tuning the classifier. We also compare with the results obtained from a classifier trained from scratch (PET classifier). The results of the reweighted distributions at reconstruction level are shown in Figure 7. From a visual inspection we see that both OmniLearn and the PET classifier are able to correctly reweight the distributions of all high-level observables we investigated, however OmniLearn shows a better agreement with the 'data'. We quantify the improvement brought by OmniLearn by calculating the triangular discriminator [119–121] for each observable with results reported in Table VIII. In all cases, the values obtained by OmniLearn are significantly better than the baseline training.

In Figure 8, we show the validation loss curve obtained during the reweighting training for both OmniLearn and PET classifier. Similarly to previous classifier results, OmniLearn starts from a lower validation and and converges faster to a lower minimum compared to a classifier trained from scratch.

Next, we investigate the impact of using OmniLearn during the entire unfolding process. We follow the OmniFold methodology training the model for five iterations, similarly to the original OmniFold publication. In the first iteration, we either start from a model trained from scratch, in the case of the PET classifier baseline, or use OmniLearn as the starting point. Each following iteration starts from the trained model of the previous iteration. The results of the unfolded distributions are shown in Figure 9 with numerical comparison of the triangular discriminator shown in Table IX. Results are compared

TABLE VII. Comparison of the results obtained between different generative models in the task of particle property generation in the dataset consisting of 150 particles. Baseline FPCD [52] uses 512 time steps during sampling. Distilled models are listed alongside number of time steps used. Lower is better for all metrics except Cov.

Jet class	Model	W_1^{PM} ($\times 10^{-3}$)	W_1^{P} ($\times 10^{-3}$)	W_1^{PEFP} ($\times 10^{-5}$)	Cov↑	MMD
Gluon	FPCD [52]	0.44 ± 0.11	**0.28 ± 0.05**	0.91 ± 0.16	**0.56**	0.03
	FPCD 1 [52]	0.65 ± 0.12	0.58 ± 0.03	1.49 ± 0.34	0.55	0.03
	EPiC-GAN [45]	0.4 ± 0.1	3.2 ± 0.2	1.1 ± 0.7	-	-
	PET generator	**0.32 ± 0.09**	0.34 ± 0.07	1.19 ± 0.26	**0.56**	**0.02**
	PET generator (Ideal)	**0.32 ± 0.18**	0.29 ± 0.07	0.95 ± 0.41	**0.56**	**0.02**
	OmniLearn	0.47 ± 0.18	0.31 ± 0.11	1.05 ± 0.23	**0.56**	**0.02**
	OmniLearn (Ideal)	**0.32 ± 0.09**	**0.22 ± 0.06**	**0.65 ± 0.20**	0.55	**0.02**
Light Quark	FPCD [52]	0.46 ± 0.05	**0.24 ± 0.02**	**0.43 ± 0.09**	0.54	0.02
	FPCD 1 [52]	**0.39 ± 0.04**	0.61 ± 0.03	0.57 ± 0.10	0.54	0.02
	EPiC-GAN [45]	**0.4 ± 0.1**	3.9 ± 0.3	0.7 ± 0.4	-	-
	PET generator	**0.41 ± 0.04**	0.34 ± 0.08	0.74 ± 0.18	**0.55**	0.02
	PET generator (Ideal)	**0.34 ± 0.09**	0.34 ± 0.12	**0.50 ± 0.17**	**0.55**	0.02
	OmniLearn	0.46 ± 0.13	0.39 ± 0.11	0.54 ± 0.14	0.53	0.02
	OmniLearn (Ideal)	**0.34 ± 0.15**	0.41 ± 0.11	**0.41 ± 0.12**	0.54	0.02
Top Quark	FPCD [52]	0.40 ± 0.07	**0.30 ± 0.03**	2.23 ± 0.16	**0.58**	0.05
	FPCD 1 [52]	0.85 ± 0.09	0.87 ± 0.03	3.82 ± 0.24	**0.58**	0.05
	EPiC-GAN [45]	0.6 ± 0.1	3.7 ± 0.3	2.8 ± 0.7	-	-
	PET generator	0.40 ± 0.08	**0.28 ± 0.08**	1.81 ± 0.33	0.57	**0.04**
	PET generator (Ideal)	**0.29 ± 0.07**	0.36 ± 0.05	**1.27 ± 0.30**	0.57	**0.04**
	OmniLearn	0.38 ± 0.05	**0.30 ± 0.07**	1.84 ± 0.30	0.57	**0.04**
	OmniLearn (Ideal)	**0.30 ± 0.07**	**0.28 ± 0.07**	**1.16 ± 0.39**	0.57	**0.04**
W Boson	FPCD [52]	0.29 ± 0.02	**0.23 ± 0.02**	0.22 ± 0.04	0.55	0.02
	FPCD 1 [52]	0.93 ± 0.04	0.67 ± 0.01	0.37 ± 0.03	**0.56**	0.02
	PET generator	**0.15 ± 0.02**	0.27 ± 0.07	**0.12 ± 0.03**	0.55	0.02
	PET generator (Ideal)	**0.12 ± 0.03**	**0.24 ± 0.06**	**0.13 ± 0.03**	0.55	0.02
	OmniLearn	0.18 ± 0.01	0.27 ± 0.05	**0.14 ± 0.04**	**0.56**	0.02
	OmniLearn (Ideal)	**0.13 ± 0.04**	**0.26 ± 0.04**	**0.11 ± 0.03**	0.55	0.02
Z Boson	FPCD [52]	0.28 ± 0.05	**0.22 ± 0.03**	0.23 ± 0.03	0.55	0.02
	FPCD 1 [52]	1.04 ± 0.08	0.69 ± 0.02	0.62 ± 0.06	**0.57**	0.02
	PET generator	0.24 ± 0.06	0.35 ± 0.06	0.20 ± 0.04	0.55	0.02
	PET generator (Ideal)	**0.13 ± 0.02**	**0.22 ± 0.06**	**0.16 ± 0.04**	0.55	0.02
	OmniLearn	0.19 ± 0.05	0.38 ± 0.11	0.19 ± 0.03	0.56	0.02
	OmniLearn (Ideal)	**0.12 ± 0.03**	0.28 ± 0.09	**0.14 ± 0.04**	0.55	0.02

TABLE VIII. Comparison of the triangular discriminator between different algorithms for reweighting. Uncertainties from PET and OmniLearn are taken from 100 histogram variations within the statistical uncertainty of the prediction. Quantities in bold represent the method with best performance.

	PET classifier	OmniLearn
Jet mass	0.13±0.03	**0.027±0.008**
N	0.13±0.03	**0.05±0.02**
Jet Width	0.09±0.02	**0.02±0.01**
$\log \rho$	0.08±0.02	**0.03±0.01**
τ_{21}	0.08±0.03	**0.02±0.01**
z_g	0.04±0.01	**0.001±0.004**

with the values reported in the original OmniFold publication, where particle-level unfolding was accomplished using a DeepSets architecture [122].

Once again OmniLearn shows an improvement in performance compared to other methods and same classifier architecture trained from scratch. These results are encouraging since data availability is strictly limited at collider experiments, presenting a limiting factor to the applicability of OmniFold. However, using OmniLearn as the starting point, we can mitigate this issue to achieve better performance.

VIII. WEAK SUPERVISION AND RESONANT ANOMALY DETECTION

The search for new particle interactions is a challenging task at collider experiments. Expected to be rare, signs for new physics might be subtle, affecting only very specific observables. While traditional approaches rely on theory predictions to narrow down possible new physics scenarios, the current lack of new physics discoveries at the LHC motivates new strategies to search for new phenomena. Anomaly detection brings a change in this paradigm [123–125]. Unexpected data structures can be automatically identified by algorithms as possible hints for new physics. One well-studied approach designed for resonant anomaly detection is based on Classification

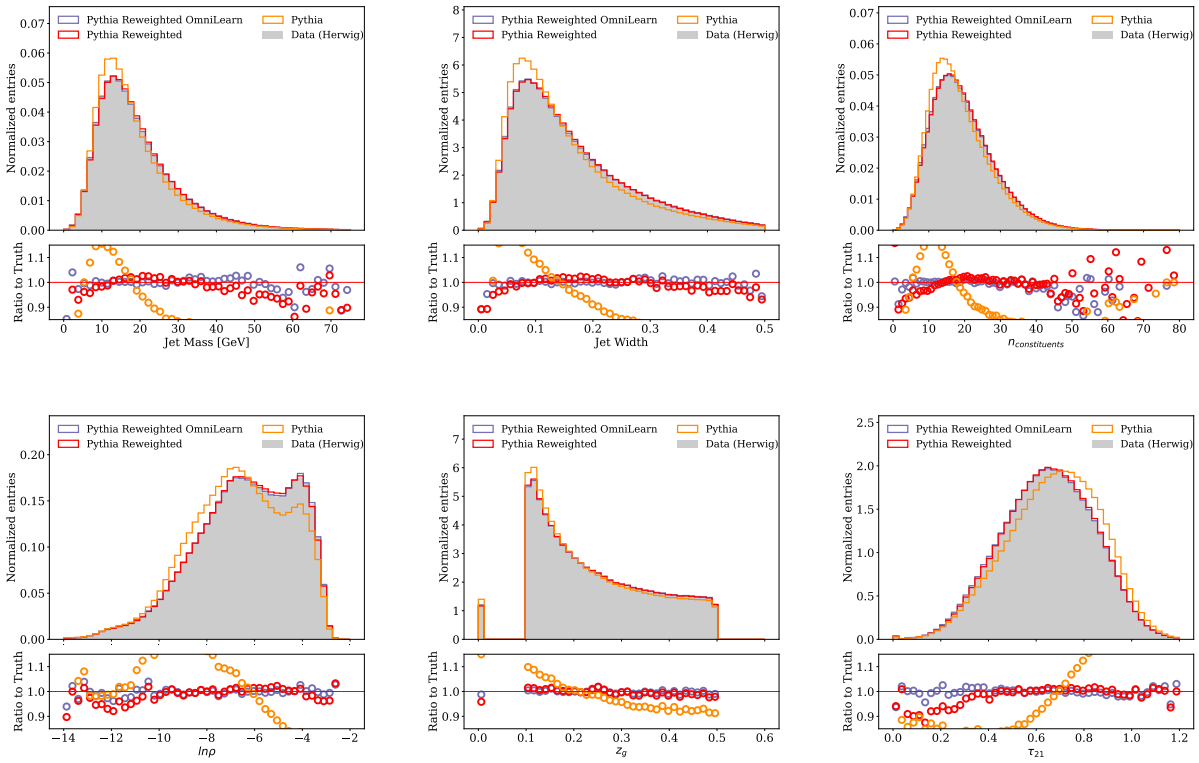

FIG. 7. Reweighted distributions for six different physics observables obtained from the OmniLearn and PET classifier.

TABLE IX. Comparison of the triangular discriminator between different algorithms for unfolding. Uncertainties from PET and OmniLearn are taken from 100 histogram variations within the statistical uncertainty of the prediction. Quantities in bold represent the method with best performance.

Metric	MultiFold	UniFold	IBU	OmniFold		
				DeepSets	PET classifier	OmniLearn
Jet mass	3.80	8.82	9.31	**2.77**	**2.8±0.9**	**2.6±0.8**
N	0.89	1.46	1.51	**0.33**	0.50±0.15	**0.34±0.1**
Jet Width	0.09	0.15	0.11	0.10	0.09±0.02	**0.07±0.01**
$\log\rho$	0.37	0.59	0.71	0.35	0.23±0.07	**0.14±0.03**
τ_{21}	0.26	1.11	1.10	0.53	0.13±0.03	**0.05±0.01**
z_g	**0.15**	0.59	0.37	0.68	0.19±0.03	0.21±0.04

Without Labels (CWoLa) framework [126–128], where weakly-supervised learning enables training directly on (unlabeled) data. In the CWoLa approach, samples with mixed fractions of a possible signal and background are used to train a classifier whose goal is to identify the origin of the sample. In the best case, we can imagine a sample consisting of only the background process, possibly as part of a background simulation or derived from a control region, that is then used to train the classifier against data possibly containing new particle interactions in addition to the background process. We examine the benefits of using OmniLearn during the classification process and evaluate the performance using the R&D dataset from the LHC Olympics data challenge [123, 129]. The background consists of dijet final states from QCD production while the signal is a resonant boson production $A \to B(\to qq')C(\to qq')$ with masses $m_A, m_B, m_C = 3.5,\ 0.5,\ 0.1$ TeV, respectively. Signal and background vents are generated with Pythia8 interfaced with Delphes3.4.1 for detector simulation. Jets are defined using the anti-k_T algorithm as implemented in FastJet with $R = 1$. We focus on the two leading jets in transverse momentum space and require the leading jet to have $p_T > 1.2$ TeV. After selection, we save all particles associated to the two most energetic jets, resulting in a maximum particle multiplicity of 279 particles per jet.

During weakly-supervised training, we follow previous studies [53, 130–142] and define the signal region of interest for events with dijet mass 3300 GeV $< m_{jj} <$ 3700 GeV. In the region of interest, we call 'data' the combination of 100k background events with varying

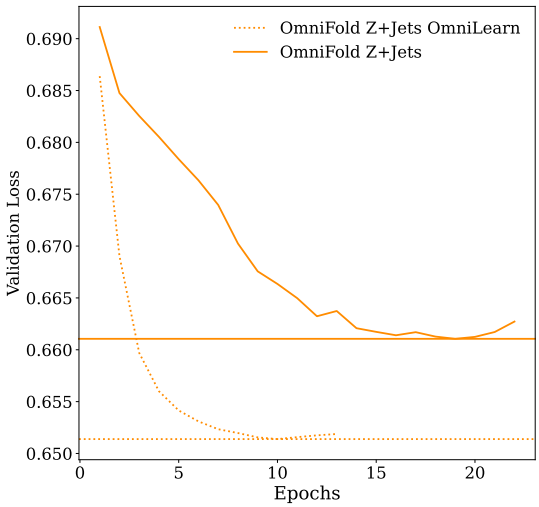

FIG. 8. Validation loss curves obtained in the first iteration and first step of the OmniFold dataset. The OmniLearn validation loss is compared with the PET classifier trained from scratch.

amounts of signal events. The background-only distribution consists of 350k independently simulated background events. Differently to previous applications of OmniLearn described in this paper, here, we need to include the information of both jets in the classifier. We modify the PET classifier to accommodate the changes in the dataset while also preserving the permutation equivariance of the complete network. This is achieved by first passing the particles present in each jet through the PET body independently, such that particles belonging to different jets do not interact with each other. The outputs of the PET body are then shifted and scaled by the outputs of the jet embedding block that takes as input the kinematic information of each jet. This strategy allows us to maintain the permutation equivariance of the model while giving jet specific information to each particle. The shifted and scaled particles are then passed to the classifier head, reshaped as if all particles belonged to the same jet. Since the shift and scaling operations are jet-dependent, the reshaping operation allows all particles to be conditionally mapped to the same space without loss of information. The classifier head is unchanged, with a class token used to summarize the information of all particles before the classification output. We use the output of the classifier as the anomaly score to determine the sensitivity to this specific new physics scenario. We quantify the performance based on the maximum value of the significance improvement characteristic curve (SIC) defined as the signal efficiency divided by the square root of the background efficiency versus the signal efficiency. The SIC represents a multiplicative factor by which the initial significance of a signal present in the data would increase when a particular threshold of the classifier output is chosen. Maximum SIC values above unity indicate value added. We show the results in Figure 10 and compare the results obtained by OmniLearn and PET classifier with the results reported in [53]. Since the reference background process is statistically identical to the background presented in the 'data' construction, we call this scenario idealized[2].

OmniLearn shows non-negligible signal sensitivity for signals injections above 600, corresponding to an initial significance $S/\sqrt{B} \sim 2$, representing a large increase in sensitivity compared to previous results where signal sensitivity was only achieved for signal injections above 1500 ($S/\sqrt{B} \sim 5$). Compared to previous results, we also observe the performance of PET classifier to be similar at lower signal injections to the results reported in [53] and performing worse at higher signal injections. The reason for this difference is due to the limited amount of data in the signal region (around 100k). Even with the larger generated background of 350k events, the dataset size of this application is at least 2 times smaller than all previous datasets investigated so far. In the low data regimes, data efficient models are often observed to perform better than large transformer models, a limitation that is mitigated by OmniLearn.

In general, a pure background process in the region of interest is often not readily available, requiring alternative strategies to produce a background-only description. In the resonant case, a natural strategy is to use sidebands around the region of interest to determine the properties of the background process. As in a number of previous weakly supervised studies [53, 131, 132, 135–140, 142, 143], a generative model conditioned on the resonant variable and trained only in the sidebands can be used to interpolate the background description in the region of interest. We use this strategy to train OmniLearn in the sidebands using the dijet mass value as a conditional variable replacing the class labels as inputs to the PET generator head. After training of the generative model, predicted background events in the region of interest are created by generating 350k background events. The same classifier used in the weakly supervision case is used to separate events from the generated background and 'data', created using true background events and different amounts of signal injected. Even though the training of the generative model is carried out using all clustered particles, we require particles in the generated and data samples to have $p_T > 1$ GeV, to reduce the impact of low energy particles driving the classifier performance. Results are shown in Figure 11 with idealized results also included for completion.

Similarly to the idealized scenario, OmniLearn increases the reach in signal sensitivity to signal injections above 700 ($S/\sqrt{B} \sim 2.2$) and greatly improving upon previous results. While the Cathode method, using 6

[2] The authors of Ref. [140] show that it is possible to achieve even better performance if the functional form is known - it would be interesting to see such a strategy combined with OmniLearn in the future.

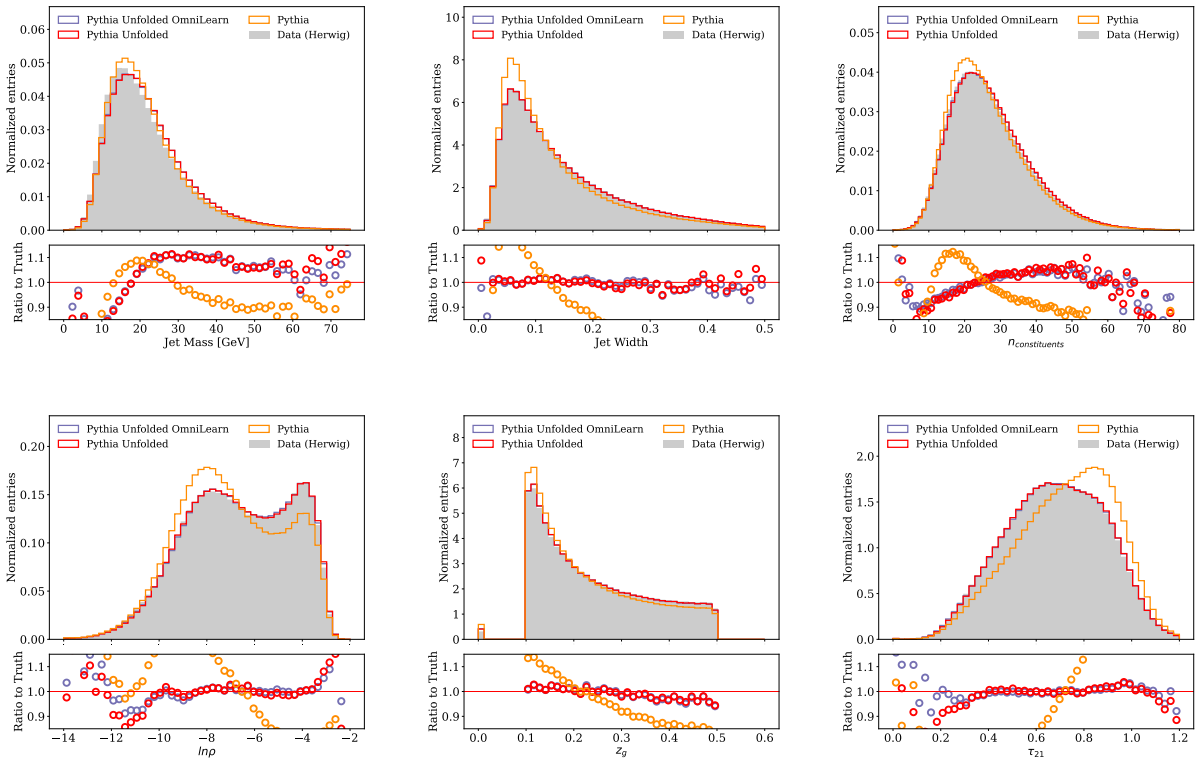

FIG. 9. Unfolded distributions for six different physics observables obtained from the OmniLearn and PET classifier.

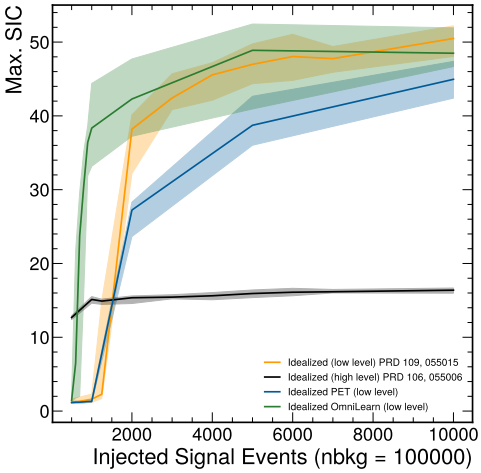

FIG. 10. Maximum values of the SIC curve evaluated over different values of injected signal. OmniLearn and PET classifier results are compared with other algorithms used for the same task.

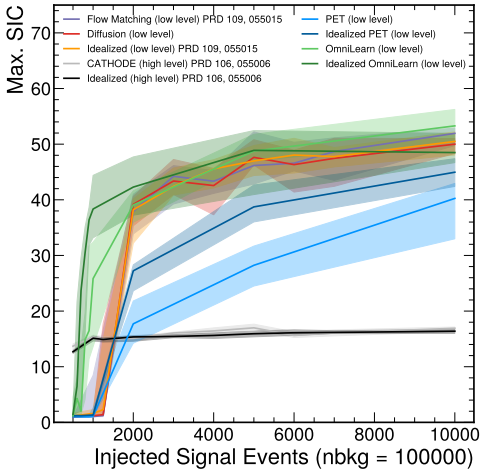

FIG. 11. Maximum values of the SIC curve evaluated over different values of injected signal. OmniLearn and PET classifier results are compared with other algorithms used for the same task.

high level observables [132], is still more sensitive to the specific new physics scenario at lower signal injection values, we expect the use of all particles enhanced by the OmniLearn model to attain sensitivity to a more general class of possible new physics scenarios, that would otherwise not be identified by the specific set of high level

observables chosen. This application highlights the potential of OmniLearn for anomaly detection, where the data used to train the generative model and classifiers is limited by the experimental setup and thus cannot be increased arbitrarily.

Finally, we also investigate the changes in validation loss as the training of the generative model progresses. The results are shown in Figure 12.

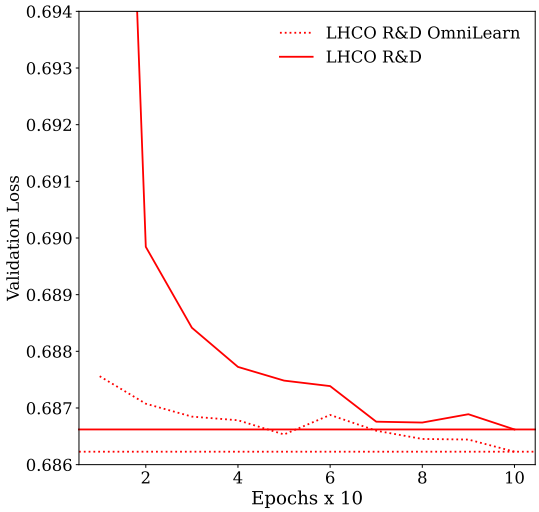

FIG. 12. Validation loss curves obtained in the LHCO dataset. The OmniLearn validation loss is compared with the PET generator trained from scratch.

Not only do we observe OmniLearn starting from a considerably lower loss, but OmniLearn also reaches an overall lower validation loss and matches the minimum validation loss from the PET generator in half the number of epochs. Reducing the training time for the generator is important for realistic applications at the LHC, where a scan using multiple signal windows is necessary to cover the entire dijet mass spectrum, requiring the training of several generative models for each signal region of interest.

IX. CONCLUSION AND OUTLOOK

In this paper, we have introduced a foundation model for jet physics called OmniLearn. This machine learning approach is a neural network capable of advancing a wide variety of research areas within jet physics. We have shown that even though this model was trained on one specific dataset, it accelerates and/or improves the accuracy of other classification tasks and generation quality evaluated over nine additional datasets across initial states, final states, and simulation levels as well as of tasks in jet generation, reweighting/unfolding, and anomaly detection. While our neural network has millions of trainable parameters, it is much smaller than many other foundation models (i.e. it is not 'large' in the sense of an LLM). We view this as a strength for usability and we hypothesize that this is possible because our model had a training task close to the application tasks.

In all tasks, OmniLearn is nearly always as good or (much) better than previous dedicated models. This is potentially transformative in a number of applications and we highlight three to illustrate the immense potential of this new approach:

1. For tasks with **expensive simulations**, we have shown that OmniLearn can effectively amplify the training statistics to improve the performance. For example, on a full detector simulation dataset for top tagging from ATLAS [81], we show that we can achieve (or exceed) state-of-the-art performance with only 10% of the training dataset. This could lead to significant computational savings for the experiments when developing new taggers.

2. For full phase space **unfolding**, OmniLearn is not only more precise, but also much faster - converging in about half the time as a dedicated approach. This is critical for practical applications of unfolding, where ensembling and statistical uncertainties require the training of a computationally expensive number of models [104, 144–148].

3. For **anomaly detection**, we have shown for the first time that full phase space methods are capable of non-trivial discoveries. Previous work in resonant anomaly detection [53] had shown that full phase-space methods could significantly amplify injected signals, but only if their starting significance was well above 2. With OmniLearn, this is pushed down to 2.

Our methodology and trained model are publicly available. While we have focused on jet physics, it would be exciting to see OmniLearn applied in other, related areas with similar challenges and thus also similar potential rewards.

CODE AVAILABILITY

The code for this paper can be found at https://github.com/ViniciusMikuni/OmniLearn.

ACKNOWLEDGMENTS

We thank F. Dreyer, Ming Fong, R. Grabarczyk, K. Greif, P.F. Monni, and D. Whiteson for interesting discussions related to transfer learning. We also thank J. Birk, A. Hallin , and G. Kasieczka for comments on the manuscript Additionally, we thank our colleagues from the H1 Collaboration for allowing us to use the simulated MC event samples. We also thank DESY-IT and

the MPI für Physik for providing computing infrastructure and supporting the data preservation project of the HERA experiments. VM, and BN are supported by the U.S. Department of Energy (DOE), Office of Science under contract DE-AC02-05CH11231. This research used resources of the National Energy Research Scientific Computing Center, a DOE Office of Science User Facility supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC02-05CH11231 using NERSC awards HEP-ERCAP0021099 and HEP-ERCAP0028249.

Appendix A: Input Variables

The input features used to train OmniLearn are described in Table X.

TABLE X. Input features used during the OmniLearn training. Binary flags consisting of charge and PID information are included in the 'is *' observables.

Object	Observables
Particles	$\Delta\eta$
	$\Delta\phi$
	$\log \mathrm{p_T}$
	$\log \mathrm{E}$
	$\log\left(1 - \frac{\mathrm{p_T}}{\mathrm{p_T(jet)}}\right)$
	$\log\left(1 - \frac{\mathrm{E}}{\mathrm{E(jet)}}\right)$
	$\Delta\mathrm{R}$
	charge
	is electron
	is muon
	is photon
	is charged hadron
	is neutral hadron
Jets	$\mathrm{p_T}$
	η
	mass
	particle multiplicity

[1] A. Abdesselam *et al.*, Eur. Phys. J. C **71**, 1661 (2011), arXiv:1012.5412 [hep-ph].
[2] A. Altheimer *et al.*, J. Phys. G **39**, 063001 (2012), arXiv:1201.0008 [hep-ph].
[3] A. Altheimer *et al.*, Eur. Phys. J. C **74**, 2792 (2014), arXiv:1311.2708 [hep-ex].
[4] D. Adams *et al.*, Eur. Phys. J. C **75**, 409 (2015), arXiv:1504.00679 [hep-ph].
[5] A. J. Larkoski, I. Moult, and B. Nachman, Phys. Rept. **841**, 1 (2020), arXiv:1709.04464 [hep-ph].
[6] R. Kogler *et al.*, Rev. Mod. Phys. **91**, 045003 (2019), arXiv:1803.06991 [hep-ex].
[7] S. Marzani, G. Soyez, and M. Spannowsky, *Looking inside jets: an introduction to jet substructure and boosted-object phenomenology*, Vol. 958 (Springer, 2019) arXiv:1901.10342 [hep-ph].
[8] R. Kogler, *Advances in Jet Substructure at the LHC: Algorithms, Measurements and Searches for New Physical Phenomena*, Springer Tracts Mod. Phys., Vol. 284 (Springer, 2021).
[9] A. Butter *et al.*, SciPost Phys. **7**, 014 (2019), arXiv:1902.09914 [hep-ph].
[10] S. Gong, Q. Meng, J. Zhang, H. Qu, C. Li, S. Qian, W. Du, Z.-M. Ma, and T.-Y. Liu, JHEP **07**, 030 (2022), arXiv:2201.08187 [hep-ph].
[11] A. Bogatskiy, T. Hoffman, D. W. Miller, J. T. Offermann, and X. Liu, (2023), arXiv:2307.16506 [hep-ph].
[12] A. Andreassen, P. T. Komiske, E. M. Metodiev, B. Nachman, and J. Thaler, Phys. Rev. Lett. **124**, 182001 (2020), arXiv:1911.09107 [hep-ph].
[13] M. Bellagente, A. Butter, G. Kasieczka, T. Plehn, and R. Winterhalder, (2019), 10.21468/SciPostPhys.8.4.070, arXiv:1912.00477 [hep-ph].
[14] M. Bellagente, A. Butter, G. Kasieczka, T. Plehn, A. Rousselot, and R. Winterhalder, (2020), 10.21468/SciPostPhys.9.5.074, arXiv:2006.06685 [hep-ph].
[15] M. Vandegar, M. Kagan, A. Wehenkel, and G. Louppe, in *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics*, Proceedings of Machine Learning Research, Vol. 130, edited by A. Banerjee and K. Fukumizu (PMLR, 2021) pp. 2107–2115, arXiv:2011.05836 [stat.ML].
[16] A. Andreassen, P. T. Komiske, E. M. Metodiev, B. Nachman, A. Suresh, and J. Thaler, (2021), arXiv:2105.04448 [stat.ML].
[17] M. Arratia *et al.*, JINST **17**, P01024 (2021), arXiv:2109.13243 [hep-ph].
[18] J. N. Howard, S. Mandt, D. Whiteson, and Y. Yang, Sci.Rep. **12**, 7567 (2021), arXiv:2101.08944 [hep-ph].
[19] M. Backes, A. Butter, M. Dunford, and B. Malaescu, (2022), arXiv:2212.08674 [hep-ph].
[20] A. Shmakov, K. Greif, M. Fenton, A. Ghosh, P. Baldi, and D. Whiteson, (2023), arXiv:2305.10399 [hep-ex].
[21] A. Shmakov, K. Greif, M. J. Fenton, A. Ghosh, P. Baldi, and D. Whiteson, (2024), arXiv:2404.14332 [hep-ex].
[22] C. Zhou, Q. Li, C. Li, J. Yu, Y. Liu, G. Wang, K. Zhang, C. Ji, Q. Yan, L. He, H. Peng, J. Li, J. Wu, Z. Liu, P. Xie, C. Xiong, J. Pei, P. S. Yu, and L. Sun, "A comprehensive survey on pretrained foundation models: A history from bert to chatgpt," (2023), arXiv:2302.09419 [cs.AI].
[23] Z. Zhang *et al.*, (2024), arXiv:2404.08001 [hep-ph].
[24] B. M. Dillon, G. Kasieczka, H. Olischlager, T. Plehn, P. Sorrenson, and L. Vogel, SciPost Phys. **12**, 188 (2021), arXiv:2108.04253 [hep-ph].

[25] B. M. Dillon, R. Mastandrea, and B. Nachman, Phys.Rev.D **106**, 056005 (2022), arXiv:2205.10380 [hep-ph].

[26] B. M. Dillon, L. Favaro, F. Feiden, T. Modak, and T. Plehn, (2023), arXiv:2301.04660 [hep-ph].

[27] L. Heinrich, T. Golling, M. Kagan, S. Klein, M. Leigh, M. Osadchy, and J. A. Raine, (2024), arXiv:2401.13537 [hep-ph].

[28] P. Harris, M. Kagan, J. Krupa, B. Maier, and N. Woodward, (2024), arXiv:2403.07066 [hep-ph].

[29] M. P. Kuchera, R. Ramanujan, J. Z. Taylor, R. R. Strauss, D. Bazin, J. Bradt, and R. Chen, Nucl. Instrum. Meth. A **940**, 156 (2019), arXiv:1810.10350 [cs.CV].

[30] A. Chappell and L. H. Whitehead, Eur. Phys. J. C **82**, 1099 (2022), arXiv:2207.03139 [hep-ex].

[31] F. A. Dreyer, R. Grabarczyk, and P. F. Monni, Eur.Phys.J.C **82**, 564 (2022), arXiv:2203.06210 [hep-ph].

[32] H. Beauchesne, Z.-E. Chen, and C.-W. Chiang, (2023), arXiv:2312.06152 [hep-ph].

[33] J. Birk, A. Hallin, and G. Kasieczka, (2024), arXiv:2403.05618 [hep-ph].

[34] P. T. Komiske, E. M. Metodiev, and J. Thaler, JHEP **01**, 121 (2019), arXiv:1810.05165 [hep-ph].

[35] ATLAS Collaboration, *Deep Sets based Neural Networks for Impact Parameter Flavour Tagging in ATLAS*, Tech. Rep. ATL-PHYS-PUB-2020-014 (CERN, Geneva, 2020).

[36] 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, Eur. Phys. J. C **80**, 58 (2020), arXiv:1908.05318 [hep-ex].

[37] H. Qu and L. Gouskos, Phys. Rev. D **101**, 056019 (2020), arXiv:1902.08570 [hep-ph].

[38] C. Shimmin (2021) arXiv:2107.02908 [hep-ph].

[39] V. Mikuni and F. Canelli, Eur. Phys. J. Plus **135**, 463 (2020), arXiv:2001.05311 [physics.data-an].

[40] V. Mikuni and F. Canelli, Mach.Learn.Sci.Tech. **2**, 035027 (2021), arXiv:2102.05073 [physics.data-an].

[41] H. Qu, C. Li, and S. Qian, (2022), arXiv:2202.03772 [hep-ph].

[42] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, arXiv preprint arXiv:1710.10903 (2017).

[43] J. Kaplan, S. McCandlish, T. Henighan, T. B. Brown, B. Chess, R. Child, S. Gray, A. Radford, J. Wu, and D. Amodei, CoRR **abs/2001.08361** (2020), 2001.08361.

[44] R. Kansal, J. Duarte, H. Su, B. Orzari, T. Tomei, M. Pierini, M. Touranakou, J.-R. Vlimant, and D. Gunopulos, (2021), arXiv:2106.11535 [cs.LG].

[45] E. Buhmann, G. Kasieczka, and J. Thaler, SciPost Phys. **15**, 130 (2023), arXiv:2301.08128 [hep-ph].

[46] M. Touranakou, N. Chernyavskaya, J. Duarte, D. Gunopulos, R. Kansal, B. Orzari, M. Pierini, T. Tomei, and J.-R. Vlimant, Mach.Learn.Sci.Tech. **3**, 035003 (2022), arXiv:2203.00520 [physics.comp-ph].

[47] B. Käch, D. Krücker, I. Melzer-Pellmann, M. Scham, S. Schnake, and A. Verney-Provatas, (2022), arXiv:2211.13630 [hep-ex].

[48] R. Verheyen, SciPost Phys. **13**, 047 (2022), arXiv:2205.01697 [hep-ph].

[49] T. Finke, M. Krämer, A. Mück, and J. Tönshoff, JHEP **06**, 184 (2023), arXiv:2303.07364 [hep-ph].

[50] A. Butter, N. Huetsch, S. P. Schweitzer, T. Plehn, P. Sorrenson, and J. Spinner, (2023), arXiv:2305.10475 [hep-ph].

[51] J. Sohl-Dickstein, E. A. Weiss, N. Maheswaranathan, and S. Ganguli, "Deep unsupervised learning using nonequilibrium thermodynamics," (2015), arXiv:1503.03585 [cs.LG].

[52] V. Mikuni, B. Nachman, and M. Pettee, Phys.Rev.D **108**, 036025 (2023), arXiv:2304.01266 [hep-ph].

[53] E. Buhmann, C. Ewen, G. Kasieczka, V. Mikuni, B. Nachman, and D. Shih, (2023), arXiv:2310.06897 [hep-ph].

[54] V. Mikuni and B. Nachman, (2023), arXiv:2306.03933 [hep-ph].

[55] M. Leigh, D. Sengupta, G. Quétant, J. A. Raine, K. Zoch, and T. Golling, (2023), arXiv:2303.05376 [hep-ph].

[56] M. Tancik, P. Srinivasan, B. Mildenhall, S. Fridovich-Keil, N. Raghavan, U. Singhal, R. Ramamoorthi, J. Barron, and R. Ng, in *Advances in Neural Information Processing Systems*, Vol. 33, edited by H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin (Curran Associates, Inc., 2020) pp. 7537–7547.

[57] D. Hendrycks and K. Gimpel, arXiv preprint arXiv:1606.08415 (2016).

[58] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, Journal of Machine Learning Research **15**, 1929 (2014).

[59] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, CoRR **abs/1706.03762** (2017), 1706.03762.

[60] H. Touvron, M. Cord, A. Sablayrolles, G. Synnaeve, and H. Jégou, in *Proceedings of the IEEE/CVF international conference on computer vision* (2021) pp. 32–42.

[61] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, *et al.*, arXiv preprint arXiv:2010.11929 (2020).

[62] J. Ho and T. Salimans, arXiv preprint arXiv:2207.12598 (2022).

[63] J. Alwall, R. Frederix, S. Frixione, V. Hirschi, F. Maltoni, O. Mattelaer, H. S. Shao, T. Stelzer, P. Torrielli, and M. Zaro, JHEP **07**, 079 (2014), arXiv:1405.0301 [hep-ph].

[64] T. Sjöstrand, S. Mrenna, and P. Z. Skands, JHEP **05**, 026 (2006), arXiv:hep-ph/0603175 [hep-ph].

[65] T. Sjöstrand, S. Ask, J. R. Christiansen, R. Corke, N. Desai, P. Ilten, S. Mrenna, S. Prestel, C. O. Rasmussen, and P. Z. Skands, Comput. Phys. Commun. **191**, 159 (2015), arXiv:1410.3012 [hep-ph].

[66] J. de Favereau, C. Delaere, P. Demin, A. Giammanco, V. Lemaître, A. Mertens, and M. Selvaggi (DELPHES 3), JHEP **02**, 057 (2014), arXiv:1307.6346 [hep-ex].

[67] A. Mertens, J. Phys. Conf. Ser. **608**, 012045 (2015).

[68] M. Selvaggi, J. Phys. Conf. Ser. **523**, 012033 (2014).

[69] M. Cacciari and G. P. Salam, Phys. Lett. **B641**, 57 (2006), arXiv:hep-ph/0512210 [hep-ph].

[70] M. Cacciari, G. P. Salam, and G. Soyez, Eur. Phys. J. **C72**, 1896 (2012), arXiv:1111.6097 [hep-ph].

[71] M. Cacciari, G. P. Salam, and G. Soyez, JHEP **04**, 063 (2008), arXiv:0802.1189 [hep-ph].

[72] "Perlmutter system," https://docs.nersc.gov/systems/perlmutter/system_details/, accessed: 2022-05-04.

[73] A. Sergeev and M. D. Balso, arXiv preprint arXiv:1802.05799 (2018).

[74] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, *et al.*, in *OSDI*, Vol. 16 (2016) pp. 265–283.

[75] F. Chollet, "Keras," https://github.com/fchollet/keras (2017).

[76] I. Loshchilov and F. Hutter, CoRR **abs/1608.03983** (2016), 1608.03983.

[77] X. Chen, C. Liang, D. Huang, E. Real, K. Wang, H. Pham, X. Dong, T. Luong, C.-J. Hsieh, Y. Lu, *et al.*, Advances in Neural Information Processing Systems **36** (2024).

[78] A. Bogatskiy, B. Anderson, J. T. Offermann, M. Roussi, D. W. Miller, and R. Kondor, (2020), arXiv:2006.04780 [hep-ph].

[79] A. Bogatskiy, T. Hoffman, D. W. Miller, and J. T. Offermann, (2022), arXiv:2211.00454 [hep-ph].

[80] "CERN Open Data Portal," http://opendata.cern.ch.

[81] *Constituent-Based Top-Quark Tagging with the ATLAS Detector*, Tech. Rep. (CERN, Geneva, 2022).

[82] CMS Collaboration, "Jet primary dataset in aod format from runa of 2011 (/jet/run2011a-12oct2013-v1/aod)," CERN Open Data Portal (2016).

[83] P. Komiske, R. Mastandrea, E. Metodiev, P. Naik, and J. Thaler, "CMS 2011A Open Data | Jet Primary Dataset | pT > 375 GeV | MOD HDF5 Format," (2019).

[84] S. Agostinelli *et al.* (GEANT4), Nucl. Instrum. Meth. A **506**, 250 (2003).

[85] A. M. Sirunyan *et al.* (CMS), JINST **12**, P10003 (2017), arXiv:1706.04965 [physics.ins-det].

[86] R. D. Ball *et al.*, Nucl. Phys. B **867**, 244 (2013), arXiv:1207.1303 [hep-ph].

[87] A. Buckley, in *6th International Workshop on Multiple Partonic Interactions at the LHC* (2014) p. 29.

[88] G. Aad *et al.* (ATLAS), Eur. Phys. J. C **81**, 334 (2021), arXiv:2009.04986 [hep-ex].

[89] P. Berta, M. Spousta, D. W. Miller, and R. Leitner, JHEP **06**, 092 (2014), arXiv:1403.3108 [hep-ex].

[90] P. Berta, L. Masetti, D. W. Miller, and M. Spousta, JHEP **08**, 175 (2019), arXiv:1905.03470 [hep-ph].

[91] M. Cacciari, G. P. Salam, and G. Soyez, Eur. Phys. J. C **75**, 59 (2015), arXiv:1407.0408 [hep-ph].

[92] A. J. Larkoski, S. Marzani, G. Soyez, and J. Thaler, JHEP **05**, 146 (2014), arXiv:1402.2657 [hep-ph].

[93] H. Jung, Comput. Phys. Commun. **86**, 147 (1995).

[94] H. Spiesberger *et al.*, in *Workshop on Physics at HERA* (1992).

[95] A. Kwiatkowski, H. Spiesberger, and H. J. Mohring, Z. Phys. C **50**, 165 (1991).

[96] A. Kwiatkowski, H. Spiesberger, and H. J. Mohring, Comput. Phys. Commun. **69**, 155 (1992).

[97] J. Pumplin, D. R. Stump, J. Huston, H. L. Lai, P. M. Nadolsky, and W. K. Tung, JHEP **07**, 012 (2002), arXiv:hep-ph/0201195.

[98] B. Andersson, G. Gustafson, G. Ingelman, and T. Sjostrand, Phys. Rept. **97**, 31 (1983).

[99] R. Brun, F. Bruyant, M. Maire, A. C. McPherson, and P. Zanarini, (1987).

[100] M. Peez, *Search for deviations from the standard model in high transverse energy processes at the electron proton collider HERA*, Other thesis (2003).

[101] S. Hellwig, *Untersuchung der $D^* - \pi_{slow}$ Double Tagging Methode in Charmanalysen*, Master's thesis, Hamburg U. (2004).

[102] B. Portheault, *First measurement of charged and neutral current cross sections with the polarized positron beam at HERA II and QCD-electroweak analyses*, Other thesis (2005).

[103] K. Charchula, G. A. Schuler, and H. Spiesberger, Comput. Phys. Commun. **81**, 381 (1994).

[104] H1 Collaboration, Phys.Lett.B **844**, 138101 (2023), arXiv:2303.13620 [hep-ex].

[105] R. Kansal, J. Duarte, H. Su, B. Orzari, T. Tomei, M. Pierini, M. Touranakou, J.-R. Vlimant, and D. Gunopulos, "Jetnet," (2022).

[106] R. Kansal, J. Duarte, H. Su, B. Orzari, T. Tomei, M. Pierini, M. Touranakou, J.-R. Vlimant, and D. Gunopulos, "Jetnet150," (2022).

[107] G. Cowan, Conf. Proc. **C0203181**, 248 (2002).

[108] V. Blobel, PHYSTAT2011 Proceedings , 240 (2011).

[109] V. Blobel, Data Analysis in High Energy Physics , 187 (2013).

[110] R. Balasubramanian, L. Brenner, C. Burgard, G. Cowan, V. Croft, W. Verkerke, and P. Verschuuren, (2019), arXiv:1910.14654 [physics.data-an].

[111] G. D'Agostini, Nucl. Instrum. Meth. **A362**, 487 (1995).

[112] A. Hocker and V. Kartvelishvili, Nucl. Instrum. Meth. **A372**, 469 (1996), arXiv:hep-ph/9509307 [hep-ph].

[113] S. Schmitt, JINST **7**, T10003 (2012), arXiv:1205.6201 [physics.data-an].

[114] A. Andreassen, P. Komiske, E. Metodiev, B. Nachman, and J. Thaler, "Pythia/Herwig + Delphes Jet Datasets for OmniFold Unfolding," (2019).

[115] M. Bahr *et al.*, Eur. Phys. J. **C58**, 639 (2008), arXiv:0803.0883 [hep-ph].

[116] J. Bellm *et al.*, Eur. Phys. J. **C76**, 196 (2016), arXiv:1512.01178 [hep-ph].

[117] J. Bellm *et al.*, (2017), arXiv:1705.06919 [hep-ph].

[118] ATLAS Collaboration, (2014), https://cds.cern.ch/record/1966419.

[119] F. Topsoe, IEEE Transactions on Information Theory **46**, 1602 (2000).

[120] P. Gras, S. Höche, D. Kar, A. Larkoski, L. Lönnblad, S. Plätzer, A. Siódmok, P. Skands, G. Soyez, and J. Thaler, JHEP **07**, 091 (2017), arXiv:1704.03878 [hep-ph].

[121] S. Bright-Thonney and B. Nachman, JHEP **03**, 098 (2019), arXiv:1810.05653 [hep-ph].

[122] M. Zaheer, S. Kottur, S. Ravanbakhsh, B. Póczos, R. Salakhutdinov, and A. J. Smola, in *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA* (2017) pp. 3391–3401.

[123] G. Kasieczka *et al.*, Rept.Prog.Phys. **84**, 124201 (2021), arXiv:2101.08320 [hep-ph].

[124] T. Aarrestad *et al.*, SciPost Phys. **12**, 043 (2021), arXiv:2105.14027 [hep-ph].

[125] G. Karagiorgi, G. Kasieczka, S. Kravitz, B. Nachman, and D. Shih, (2021), arXiv:2112.03769 [hep-ph].

[126] E. M. Metodiev, B. Nachman, and J. Thaler, JHEP **10**, 174 (2017), arXiv:1708.02949 [hep-ph].

[127] J. H. Collins, K. Howe, and B. Nachman, Phys. Rev. Lett. **121**, 241803 (2018), arXiv:1805.02664 [hep-ph].

[128] J. H. Collins, K. Howe, and B. Nachman, Phys. Rev. **D99**, 014038 (2019), arXiv:1902.02634 [hep-ph].

[129] G. Kasieczka, B. Nachman, and D. Shih, "Official Datasets for LHC Olympics 2020 Anomaly Detection Challenge (Version v6) [Data set]." (2019), https://doi.org/10.5281/zenodo.4536624.

[130] B. Nachman and D. Shih, Phys. Rev. D **101**, 075042 (2020), arXiv:2001.04990 [hep-ph].

[131] A. Andreassen, B. Nachman, and D. Shih, Phys. Rev. D **101**, 095004 (2020), arXiv:2001.05001 [hep-ph].

[132] A. Hallin, J. Isaacson, G. Kasieczka, C. Krause, B. Nachman, T. Quadfasel, M. Schlaffer, D. Shih, and M. Sommerhalder, Phys.Rev.D **106**, 055006 (2021), arXiv:2109.00546 [hep-ph].

[133] K. Benkendorfer, L. L. Pottier, and B. Nachman, Phys. Rev. D **104**, 035003 (2021), arXiv:2009.02205 [hep-ph].

[134] G. Kasieczka, B. Nachman, and D. Shih (2021) arXiv:2107.02821 [stat.ML].

[135] A. Hallin, G. Kasieczka, T. Quadfasel, D. Shih, and M. Sommerhalder, Phys.Rev.D **107**, 114012 (2022), arXiv:2210.14924 [hep-ph].

[136] D. Sengupta, S. Klein, J. A. Raine, and T. Golling, (2023), arXiv:2305.04646 [hep-ph].

[137] J. A. Raine, S. Klein, D. Sengupta, and T. Golling, Front.Big Data **6**, 899345 (2022), arXiv:2203.09470 [hep-ph].

[138] T. Golling, G. Kasieczka, C. Krause, R. Mastandrea, B. Nachman, J. A. Raine, D. Sengupta, D. Shih, and M. Sommerhalder, (2023), arXiv:2307.11157 [hep-ph].

[139] G. Bickendorf, M. Drees, G. Kasieczka, C. Krause, and D. Shih, (2023), arXiv:2309.12918 [hep-ph].

[140] R. Das, G. Kasieczka, and D. Shih, (2023), arXiv:2312.11629 [hep-ph].

[141] T. Finke, M. Hein, G. Kasieczka, M. Krämer, A. Mück, P. Prangchaikul, T. Quadfasel, D. Shih, and M. Sommerhalder, (2023), arXiv:2309.13111 [hep-ph].

[142] M. Freytsis, M. Perelstein, and Y. C. San, (2023), arXiv:2310.13057 [hep-ph].

[143] G. Stein, U. Seljak, and B. Dai, (2020), arXiv:2012.11638 [cs.LG].

[144] H1 Collaboration, Phys.Rev.Lett. **128**, 132002 (2021), arXiv:2108.12376 [hep-ex].

[145] H1 Collaboration, H1prelim-22-031 (2022).

[146] H1 Collaboration, H1prelim-23-031 (2023).

[147] (2022), arXiv:2208.11691 [hep-ex].

[148] Y. Song (STAR), (2023), arXiv:2307.07718 [nucl-ex].