Streamlining latent spaces in machine learning using moment pooling

Rikab Gambhir,[1,2,*] Athis Osathapan,[2,3,†] and Jesse Thaler[1,2,‡]
[1]*Center for Theoretical Physics, Massachusetts Institute of Technology, Cambridge, Massachusetts 02139, USA*
[2]*The NSF AI Institute for Artificial Intelligence and Fundamental Interactions, USA*
[3]*Bowdoin College, Brunswick, Maine 04011, USA*

(Received 29 March 2024; accepted 11 September 2024; published 15 October 2024)

Many machine learning applications involve learning a latent representation of data, which is often high-dimensional and difficult to directly interpret. In this work, we propose "moment pooling," a natural extension of deep sets networks which drastically decreases the latent space dimensionality of these networks while maintaining or even improving performance. Moment pooling generalizes the summation in deep sets to arbitrary multivariate moments, which enables the model to achieve a much higher effective latent dimensionality for a fixed learned latent space dimension. We demonstrate moment pooling on the collider physics task of quark/gluon jet classification by extending energy flow networks (EFNs) to moment EFNs. We find that moment EFNs with latent dimensions as small as 1 perform similarly to ordinary EFNs with higher latent dimension. This small latent dimension allows for the internal representation to be directly visualized and interpreted, which in turn enables the learned internal jet representation to be extracted in closed form.

DOI: 10.1103/PhysRevD.110.074020

I. INTRODUCTION

As modern machine learning (ML) models and their applications continue to grow in size and scope, their internal representations of data become increasingly more complex and difficult to decipher. While there are a variety of ways to interpret what is "learned" in an ML model [1–10], it is often difficult to draw concrete, first-principles conclusions on how these models internally represent learned data, as the latent space tends to be high-dimensional and complex. This, in turn, makes it more difficult not only to trust ML models when applied outside their original training sets, but also to understand what additional domain insights may be driving the improved performance of these models.

ML methods have been gaining interest in collider physics, and have shown to perform remarkably well in a variety of collider physics and jet substructure tasks [11–33]. Recently, the energy flow network (EFN) [34] has emerged as a promising model, performing relatively well on jet tagging [35] while being more robust than other models with respect to training set simulation choice [36]. EFNs are a generalization of deep sets [37],[1] which use a set-based representation of the event, $\mathcal{P}$, to construct observables with the ansatz,

$$\mathcal{O}(\mathcal{P}) = F(\langle \Phi^a \rangle_{\mathcal{P}}), \tag{1}$$

where $\langle \Phi \rangle$ is the expectation value of Φ over the event $\mathcal{P}$, defined below. The function $\Phi\colon \mathbb{R}^d \to \mathbb{R}^L$, usually parametrized as a dense neural network, is a per-particle L-dimensional latent representation of x, with the latent dimension indexed by $a = 1, \ldots, L$. The function F (another dense neural network) is then a function of this representation, which converts the latent representation into the observable $\mathcal{O}$. The deep sets theorem, as discussed in Refs. [34,37], guarantees that any [infrared and collinear (IRC)-safe] observable can be approximated arbitrarily well for a sufficiently expressive F and Φ, and large enough L. However, the theorem makes no guarantees on the complexity of Φ or F, and may require a very large L.

In this paper, we introduce moment pooling, a natural extension of deep sets architectures that significantly reduces the number of latent dimensions L needed while maintaining or improving its performance. The moment pooling operation generalizes the expectation value of Φ in

*Contact author: rikab@mit.edu
†Contact author: aosathap@bowdoin.edu
‡Contact author: jthaler@mit.edu

[1]EFNs generalize deep sets in the sense that EFNs reduce to deep sets when weights are removed, discussed more below.

Eq. (1) to higher-order multivariate moments,

$$\mathcal{O}_k(\mathcal{P}) \equiv F(\langle\Phi^a\rangle_{\mathcal{P}}, \langle\Phi^{a_1}\Phi^{a_2}\rangle_{\mathcal{P}}, \ldots \langle\Phi^{a_1}\ldots\Phi^{a_k}\rangle_{\mathcal{P}}), \quad (2)$$

where k is the highest order moment considered. This procedure is inspired by histogram pooling [38], in which the Φ are histograms binned in x. We focus primarily on applying moment pooling to EFNs in the collider physics context, where Eq. (2) defines an order-k *moment EFN*, which reduces to the ordinary EFN when $k = 1$. Alternative modifications of EFNs are discussed in Refs. [39,40].

We show that for $k > 1$, a moment EFN enables the same or better performance on quark/gluon jet classification as an EFN, but with a much smaller latent dimension L, allowing the same machine-learned observables to be constructed using fewer base functions. With fewer latent dimensions, it is much easier to directly visualize the model's internal representations and therefore easier to directly interpret and find closed-form expressions for the learned observable. As a concrete example, an order $k = 4$ moment EFN with a *single* latent dimension achieves comparable performance on quark/gluon jet classification to an ordinary EFN with four latent dimensions. We are able to directly plot this latent dimension and find that it takes a remarkably simple closed form, the "log angularity" observable, which bears many similarities to jet angularities [41,42].

The rest of the paper is organized as follows. In Sec. II, we give an overview of moment pooling and the moment EFN architecture, show how it naturally arises as a generalization of deep sets, and introduce the idea of effective latent dimensions. In Sec. III, we demonstrate how the moment EFN may be used for quark/gluon discrimination, and how moment EFNs outperform ordinary EFNs as L and k are varied. In Sec. IV, we analyze the latent spaces of small-L moment EFNs and attempt to understand them in terms of simple closed-form fits, allowing for analytic observables to be extracted from the model. Finally, in Sec. V, we present our conclusions and outlook. Implementation details of the architecture may be found in Appendix A. An additional study involving regression on jet angularities, rather than classification, using moment EFNs may be found in Appendix B. Additional studies complementing Sec. III, involving top/QCD discrimination and moment particle flow networks (PFNs) rather than EFNs, may be found in Appendix C.

II. MOMENT POOLING

We begin with the construction of the moment pooling operation. We first define moment pooling as an extension of deep sets and apply it to EFNs, a form of weighted deep sets, to produce moment EFNs in Sec. II A. Then, in Sec. II B, we discuss how moment pooling is capable of reducing the latent dimension of EFNs through the concept of effective latent dimensions.

A. The moment energy flow network

The moment pooling operation, as given by Eq. (2), is a generalization of deep sets-style architectures. The form of Eq. (2) is motivated by the observation that the summation step over the latent representation Φ^a in deep sets architectures, generalized to weighted sums in EFNs, can be regarded as taking an *expectation* value of the L-dimensional random variable $\Phi^a(x)$ defined over a base space $\mathbb{R}^d$, taken over $\mathcal{P}$,

$$\langle\Phi^a\rangle_{\mathcal{P}} \equiv \sum_{i\in\mathcal{P}} z_i \Phi^a(x_i), \quad (3)$$

where z_i are weights and $x_i \in \mathbb{R}^d$. In the collider physics context, z_i are (normalized) particle energies and $x_i = (y_i, \phi_i)$ are particle's rapidity y and azimuthal angle ϕ on the detector, and $\mathcal{P}$ is a probability distribution of energy on detector space, or an energy flow [43–47], over which we can take expectation values.[2]

Applying Eq. (3) to Eq. (1), we find,

$$\mathcal{O}(\mathcal{P}) = F\left(\sum_{i\in\mathcal{P}} z_i \Phi^a(x_i)\right), \quad (4)$$

which is how an EFN is typically written [34]. Note that an ordinary deep sets network, as presented in Ref. [37], is simply a special case of the EFN where $z_i = 1$ for all i.

Given that EFNs are functions F of the expectation value of Φ^a, it is natural to extend them to also include higher-order moments of Φ^a, arriving at the moment energy flow network. More precisely, the moment EFN of Eq. (2) simply extends F from being a function of only the expectation value of Φ^a to a function of up to k moments of Φ^a, which reduces to the ordinary EFN for $k = 1$. As an explicit example, the $k = 2$ moment EFN takes the form,

$$\mathcal{O}_2(\mathcal{P}) = F(\langle\Phi^a\rangle_{\mathcal{P}}, \langle\Phi^{a_1}\Phi^{a_2}\rangle_{\mathcal{P}}), \quad (5)$$

where $\langle\Phi^{a_1}\Phi^{a_2}\rangle$ is the second moment of Φ, which is

$$\langle\Phi^{a_1}\Phi^{a_2}\rangle_{\mathcal{P}} = \sum_{i\in\mathcal{P}} z_i \Phi^{a_1}(x_i)\Phi^{a_2}(x_i). \quad (6)$$

This quantity is related to the covariance between the random variables Φ^{a_1} and Φ^{a_2},

$$\langle\Phi^{a_1}\Phi^{a_2}\rangle_{\mathcal{P}} = [\mathrm{Cov}(\Phi,\Phi)]^{a_1 a_2}_{\mathcal{P}} + \langle\Phi^{a_1}\rangle_{\mathcal{P}}\langle\Phi^{a_2}\rangle_{\mathcal{P}}. \quad (7)$$

[2]To align with the notation of Ref. [47], we have $\langle\Phi^a\rangle_{\mathcal{P}} = \langle\mathcal{P}, \Phi^a\rangle$.

Similarly, the $k = 3$ and $k = 4$ moment EFNs contain the skew and kurtosis of Φ^a, respectively. In principle, it is also possible to instead define a "cumulant EFN" with "cumulant pooling," where F is a function of the first k cumulants rather than the first k moments, though we will not pursue this here. In general, keeping only the first k moments can be thought of as an unpixelated generalization of max- or mean-pooling procedure in convolutional neural networks, wherein one "coarse grains" the distribution Φ, hence the term "moment pooling." For example, by keeping only the first two moments of Φ, we have effectively smoothed out the higher-moment information of our point cloud and kept only the Gaussian component.

It is important to emphasize that Φ^a remains a function of a single particle, and that the moments are taken over the set of particles, *not* pairs or m-tuples of particles. In other words, $\Phi^a(x_i)$ only provides information about the ith particle, and the moments $\langle \prod \Phi^a \rangle$ describe only how that information is distributed across an event, *not* explicit interparticle correlations. This is in contrast to graph-based approaches, such as ParticleNet [15], IRC-safe graph networks [32,33], or energy flow polynomials [48], which explicitly construct interparticle correlations of the form $h(x_i, x_j, x_k, \ldots)$. As a final point of contrast, for an event with N particles, a graph-based approach with m edges has to consider $\mathcal{O}(N^m)$ terms, while the moment EFN still only has N terms in its sum for each moment.

The above discussion has focused on extending EFNs to have multiple moments. However, it is straightforward to drop IRC-safety and generalize to the particle flow network (PFN) [34], or indeed any realization of deep sets architectures.[3] This can be accomplished by simply modifying the definition of the expectation value Eq. (3) to remove the energy weighting,

$$\langle \Phi \rangle_{\mathcal{P}}^{\text{PFN}} = \sum_{i \in \mathcal{P}} \Phi(p_i), \tag{8}$$

where Φ is a function of the per-particle information p, which can include the particle's energy, momentum, charge, flavor, and other information. Here, $\mathcal{P}$ can no longer be regarded as the distribution of energy over the detector space, but rather just as an abstract unnormalized distribution of particle information. Example studies involving moment PFNs rather than moment EFNs may be found in Appendix C.

Finally, some notes about our conventions and notation for the rest of the paper. First, we will use the terms "energy" and transverse momentum "p_T" interchangeably, as nothing we say here depends on this distinction; our studies here focus on the Large Hadron Collider (LHC), where it is typical to speak of transverse momenta rather than energies. Second, although detectors are often cylindrical or spherical and these models can be extended to accommodate this, we will only consider local rectangular patches $\sim \mathbb{R}^2$ in the rapidity-azimuth plane. Third, we will always implicitly include the $k = 0$ moment, $\langle 1 \rangle_{\mathcal{P}}$, which is the total energy of the event. For normalized events, this contains no information, but we find it convenient to include. When we speak of a $k = 1$ moment EFN, or equivalently an "ordinary" EFN, we are still including the $k = 0$ moment, which differs from the conventions of Ref. [34] slightly. Practically speaking, this makes no numeric difference. Finally, we will occasionally find it convenient to speak of Φ^a not as a single L-dimensional function of x, but as L separate one-dimensional functions Φ, and suppress the a indices.

B. The effective latent dimension

Given that, by the deep sets theorem [37], EFNs are already capable of approximating any IRC-safe observable arbitrarily well, why should we bother making them more complicated by adding moments? A moment EFN is able to approximate the same observable with a much smaller learned latent space dimension than an ordinary EFN, by taking advantage of its large effective latent dimension. The effective latent dimension of an order-k moment EFN with L latent dimensions is the total number of distinct inputs to the function F, and is given by

$$L_{\text{eff}} = \binom{L+k}{k}, \tag{9}$$

which asymptotically goes as $L_{\text{eff}} \sim \frac{L^k}{k!}$ for large L.

An order k moment EFN with L different Φ functions (indexed by a) acts like an ordinary EFN with L_{eff} different Φ' functions (indexed by A), in the sense that if we identify,

$$\begin{aligned}
&\Phi'^0(x) = 1, \\
&\Phi'^A(x) = \Phi^a \quad \text{for } A = 1, \ldots, L, \\
&\Phi'^A(x) = \Phi^{a_1}\Phi^{a_2} \quad \text{for } A = L+1, \ldots, \frac{L^2+3L}{2}, \\
&\qquad \cdots \\
&\Phi'^A(x) = \prod_{i=1}^{k} \Phi^{a_i} \quad \text{for } A = \binom{L+k-1}{k-1}, \ldots, \binom{L+k}{k} - 1,
\end{aligned} \tag{10}$$

and assign both models exactly the same F network, then the two models are completely identical. The moment EFN only needs L different learnable functions Φ to express this, while the ordinary EFN needs L_{eff} different learnable functions Φ', with $L \ll L_{\text{eff}}$. We can think of a moment EFN as effectively "compressing" or "encoding" L_{eff} pieces of information into just L functions, with Eq. (10) being the "decoder."

[3] A note on nomenclature; a PFN is identical to an ordinary deep sets network. We use the term "PFN" to refer to this architecture in the particle physics context, and "deep sets" to refer generically to sets-based architectures.

Equation (10) highlights another distinguishing feature of the moment EFN; explicit nonlinear products. Typically, Φ is parametrized as a dense neural network, which consists of affine transformations interleaved with nonlinear activation functions. It is difficult for these models to approximate highly nonlinear functions as it is a lot easier for a dense neural network to learn $\Phi(x) = x$ than to learn $\Phi(x) = x^k$. Moment EFNs involve an explicit product of Φ functions, which directly enable functions of the type x^k to be easily represented. See Appendix B for a concrete example of how this multiplication structure can aid in learning jet angularities, which are observables involving nonlinear powers of particle coordinates.

For a fixed L, a moment EFN with greater k is always at least as expressive as moment EFN with a smaller k, and therefore should perform at least as well. This is because F can be chosen to simply ignore the extra effective latent dimensions. Practically speaking, increasing k with L fixed makes the models slightly more complex to train, as there are more parameters to optimize over, so this monotonicity in performance may be imperfect in practice. To mitigate this, we employ a "pretraining" procedure on our models so that extra effective latent dimensions do not damage performance (see Appendix A for details).

On the other hand, if we fix L_{eff} and the function F, there is a small loss of expressivity when using moments, and therefore not all observables can have their representations efficiently compressed this way. A moment EFN with L latent dimensions is not quite as expressive as the equivalent ordinary EFN with L_{eff} latent dimensions, so the identification in Eq. (10) is not always possible. This is because the latent dimensions of an ordinary EFN are completely uncorrelated, whereas the moments of a random variable Φ are correlated; for example, it is always the case that $\langle\Phi^2\rangle > \langle\Phi\rangle^2$. As an explicit counterexample, suppose we were interested in constructing the energy-weighted average position of a jet in detector space; the observable $\mathcal{O}(\mathcal{P}) \equiv (y_{\text{avg}}, \phi_{\text{avg}})$. This is easy to accomplish with an $L = 2$ ordinary EFN, with $\Phi'(x_i) = (y_i, \phi_i)$ and trivial F. However, this is difficult to accomplish with the equivalent order $k = 2$ moment EFN with $L = 1$, even if we allowed F to vary, because y_{avg} and ϕ_{avg} are independent, whereas any possible $\Phi(x)$ we construct would have $\langle\Phi\rangle$ and $\langle\Phi^2\rangle$ correlated.[4]

To summarize: While one should always expect a higher-order moment EFN with L latent dimensions to more accurately approximate an observable than a lower order moment EFN with L latent dimensions, it is not guaranteed that a higher-order moment EFN with L_{eff} effective latent dimensions will outperform a lower-order EFN with L_{eff} effective latent dimensions. When this does happen, this is a statement that the observable being estimated has some simpler structure, which we will see is the case for IRC-safe quark/gluon jet discrimination in Sec. III. Interestingly, this is *not* the case for top/QCD jet discrimination (see Appendix C for a concrete example).

As a brief aside, the explicit product structure of moment EFNs is reminiscent of the self-attention mechanism [49,50] in transformer models [26,51]. Schematically, in the $k = 2$ product $\Phi^{a_1}(x)\Phi^{a_2}(x)$ we can think of $\Phi^{a_1}(x)$ as telling the network how much to "pay attention" to $\Phi^{a_2}(x)$ (and vice versa). Similarly, the self-attention mechanism in transformers is of the schematic form,

$$\text{softmax}(Q(x)K(x)) \cdot V(x), \tag{11}$$

which, ignoring the softmax (primarily used to interpret the result as a weight), is a cubic product of the form $\Phi^{a_1}(x)\Phi^{a_2}(x)\Phi^{a_3}(x)$.

III. CASE STUDY: QUARK/GLUON DISCRIMINATION

We now apply the moment EFN to the task of quark/gluon jet tagging [52,53], to show how its performance varies with different choices of L and k compared to the ordinary EFN. Additional details about the model specifications and training procedures can be found in Appendix A, and similar studies using a moment PFN instead of a moment EFN and for discriminating top-initiated jets from QCD jets can be found in Appendix C.

A. Dataset

We use the same quark and gluon jet dataset as described in Ref. [34]. This dataset consists of Z plus jet events at $\sqrt{s} = 14$ TeV generated using Pythia 8.226 [54,55] with multiple parton interactions turned on. The Z is forced to decay invisibly to neutrinos, and the remaining particles at then clustered into $R = 0.4$ anti-k_T [56] (AK4) jets using FastJet 3.3.0 [57]. Only jets with transverse momentum $p_T \in [500, 550]$ GeV and rapidity $|y| < 2$ are kept. Each jet is then labeled as a quark or gluon depending on the underlying hard process that generated it, with quarks generated using the `WeakBosonAndParton:qg2gmZq` process and gluons generated from the `WeakBosonAndParton:qqbar2gmZg` process.[5] No detector simulation is applied. Each jet is then prepossessed, such that the sum of the particle p_T's is normalized to 1 and the p_T-weighted average position of the jet is (0, 0) in the rapidity-azimuth plane. In the following studies, we use 1 M total jets to train, and 50 k jets each for validation and testing.

[4]This is technically possible if Φ is a space-filling curve, but not only this discontinuous and therefore not IRC-safe, it would be incredibly difficult to learn.

[5]These quark and gluon labels are technically unphysical, and there exist more physical operational definitions of the quark and gluon content of a jet [58,59], but this is largely unimportant for our study here.

TABLE I. The AUC and gluon rejection factor at quark efficiencies of 0.3 and 0.5 for the $k = 1$, 2, 3, and 4 moment EFNs trained in Sec. III. Here, we show results for the $L = 1$ networks, alongside the $k = 1$, $L = 4$ ordinary EFN, which achieves comparable results to the $k = 4$, $L = 1$ moment EFN. We also show the highest latent dimension L considered at each order ($L = 128$, 64, 32, and 16, respectively). For each metric, the model with the best performance is bolded.

Model	AUC	$1/\epsilon_g$ at $\epsilon_q = 0.3$	$1/\epsilon_g$ at $\epsilon_q = 0.5$	Trainable parameters
$k = 1$, $L = 1$ EFN	0.743 ± 0.001	54.2 ± 1.7	12.0 ± 0.1	31,106
$k = 2$, $L = 1$ moment EFN	0.802 ± 0.002	53.9 ± 3.5	16.7 ± 0.2	31,206
$k = 3$, $L = 1$ moment EFN	0.831 ± 0.000	52.2 ± 2.6	18.5 ± 0.2	31,306
$k = 4$, $L = 1$ moment EFN	$\mathbf{0.841 \pm 0.004}$	$\mathbf{61.6 \pm 2.8}$	$\mathbf{21.3 \pm 0.1}$	31,406
$k = 1$, $L = 4$ EFN	0.843 ± 0.004	64.0 ± 2.1	24.5 ± 1.0	31,745
$k = 1$, $L = 128$ EFN	0.879 ± 0.001	69.2 ± 1.8	31.4 ± 0.8	89,653
$k = 2$, $L = 64$ moment EFN	0.886 ± 0.001	$\mathbf{83.0 \pm 1.3}$	30.7 ± 0.6	260,085
$k = 3$, $L = 32$ moment EFN	0.886 ± 0.001	72.6 ± 2.1	$\mathbf{33.6 \pm 1.0}$	69,0645
$k = 4$, $L = 16$ moment EFN	$\mathbf{0.887 \pm 0.001}$	81.6 ± 3.4	32.7 ± 0.4	51,7461

B. Performance

For orders $k = 1$ through 4, we train moment EFNs for a wide range of latent dimensions L from 1 to 128 in powers of 2. Due to memory and training time considerations, we consider only up to $L = 2^{8-k}$, since otherwise L_{eff} becomes prohibitively large. For each model, we report its performance using the "area under curve" (AUC)[6] metric across three retrainings. We also report the gluon rejection factor at quark efficiencies of 0.3 and 0.5 for the largest L and smallest L models in Table I for ease of comparison with other quark/gluon discrimination studies [23,26,31,60–72].

Each moment EFN model is trained as an ordinary classifier to minimize the binary cross entropy between the quark and gluon classes. Both classes appear in the dataset with 50% probability. The specific details of the models and training procedure may be found in Appendix A.

The resulting model performances on the quark/gluon discrimination task, as a function of L and L_{eff}, are shown in Figs. 1(a) and 1(b), respectively. From these plots we can make four key observations:

(1) *At fixed L, AUC improves with k*: As expected, increasing the order of the moment EFN improves its performance for fixed L, since the ansatz is more expressive. This effect is particularly pronounced near $L = 1$, with the AUC improving from 0.75 to 0.84 from $k = 1$ to $k = 4$;

(2) *Higher k saturates faster*: As k increases, the value of L required to saturate performance drops. The ordinary EFN saturates around $L = 128$, whereas the order $k = 4$ saturates around $L = 16$. To achieve peak performance, you don't need as high an L with a moment EFN;

(3) *Peak AUC improves with k*: The highest AUC achieved by these models improves slightly with k, as indicated by the dashed lines in Fig. 1(a). In Fig. 1(b), we can see that this is primarily driven by the extremely high effective latent dimensions reached by higher k moment EFNs.

(4) *L_{eff} drives performance*: The AUC correlates very strongly with L_{eff}, regardless of the order k. This suggests that "encoding" and "decoding," as per Eq. (10), is occurring, and that the quark/gluon discriminant is "compressible" into fewer elementary functions. In particular, if an ordinary EFN requires L_{eff} latent dimensions to achieve a desired performance in quark/gluon discrimination, an order-k moment EFN would only require $L \sim k! L_{\text{eff}}^{1/k}$ latent dimensions to achieve the same performance.

All of these observations point to moment EFNs being able to achieve the same (or better) performance as ordinary EFNs but with a significantly smaller learned latent space dimension. The quark/gluon discriminator can be efficiently compressed, with the peak classifier going from being composed of ∼128 functions to only ∼16 functions while gaining a slight performance bump in the process. It is also especially remarkable that an order $k = 4$ moment EFN is able to achieve an AUC of 0.84 with just a *single* latent dimension, equivalent to an ordinary EFN with four latent dimensions, and only a few points away from the best possible EFN score of 0.88. We have checked for all studies shown here that going to $k = 5$ and beyond does not offer any significant improvement over $k = 4$.

Note that these four observations are *not* generically true across different classification tasks. As shown in Appendix C, the improvement in Observation 1 is not always perfectly monotonic in k, especially if the models fail to converge, and Observation 4 especially is not true for top/QCD jet discrimination. As noted in Sec. II B, a higher-k moment EFN may be less expressive than a lower-k moment EFN with the same L_{eff}, causing performance to potentially worsen with k for fixed L_{eff}.

[6]More precisely, if $P_i(x)$ is the cumulative distribution function for the model output x assuming the distribution i (either q or g in this case), then the AUC is defined to be $1 - \int_0^1 d\lambda P_g(P_q^{-1}(\lambda))$. An AUC of 0.5 indicates random guessing, and an AUC of 1.0 is a perfect classifier.

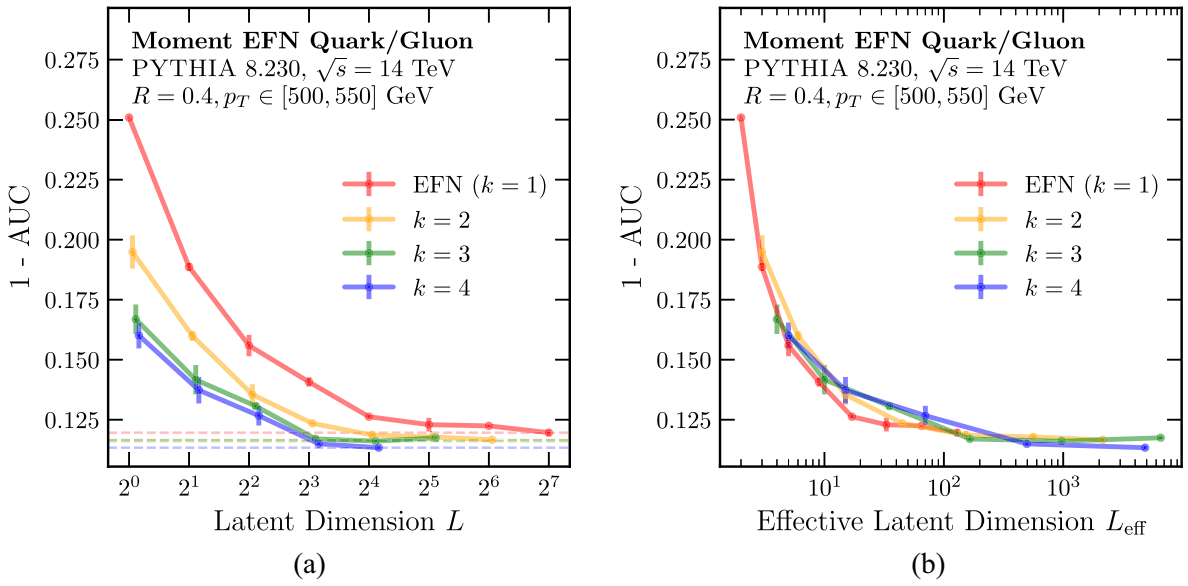

FIG. 1. The performance (AUC) on quark/gluon discrimination of the k moment EFN as a function of the (a) latent dimension L and (b) effective latent dimension L_{eff} for different values of k. The thin horizontal dashed lines indicate the best value of the AUC achieved. For each model, the standard deviation and mean of the AUC across three trainings is shown.

IV. OPENING THE BLACK BOX

One practical advantage of the smaller latent dimension afforded by moment EFNs is that lower-dimensional spaces are easier to visualize and interpret. An $L = 4$ ordinary EFN involves the complex interplay of four independent functions on detector space, whereas the equivalent order $k = 4$, $L = 1$ moment EFN achieving the same performance only has a single function to look at (and moreover, we will see that this single function is radially symmetric). For small L, we can even obtain closed form expressions for the latent spaces of moment EFNs, and due to the effective latent space, we can use this to extend our understanding of ordinary EFNs for larger values of L than we could have otherwise.

The rest of this section proceeds as follows. In Sec. IV A, we take the order $k = 4$, $L = 1$ models trained in Sec. III, visualize their internal representations, and find a closed-form expression for their latent spaces, resulting in observables we call "log angularities." In Sec. IV B, we show to what extent the F network can also be cast into closed form. Finally, in Sec. IV C, we briefly discuss $L \geq 2$ models.

A. $L = 1$ and log angularities

When $L = 1$, it is feasible to study exactly what the model learned and extract a single closed-form, one-dimensional observable representing the entire latent space [34,73]. The order $k = 4$, $L = 1$ moment EFN is able to achieve an AUC of 0.84 using a single learned representation, and our goal is to understand and extract this representation. Because of the effective latent space, interpreting the latent space of the order $k = 4$, $L = 1$ moment EFN is equivalent to interpreting *all four* of the latent dimensions of an $L = 4$ ordinary EFN. This study is modeled after the study performed in Ref. [34], which constructs *two* independent observable using an $L = 2$ ordinary EFN and achieves an AUC ~ 0.80.

Since $\Phi(x)$ is a function of the rapidity-azimuth plane, we can directly plot the latent spaces of the best $L = 1$ networks in 2D following the procedure outlined in Ref. [34], where it is possible to visualize the entire latent space at once. We show examples of this in Fig. 2 for $k = 1$ through 4. We first note that while the $k = 1$, 2 and 4 networks learn a radially symmetric latent space, the $k = 3$ network does not—it instead exhibits an approximate mirror symmetry. This is not a feature unique to $k = 3$, as this mirror symmetric latent space occasionally occurs for $k = 2$ and $k = 4$ networks as well in some retrainings, though seemingly without loss in performance. Since QCD jets are approximately radially symmetric [74,75], the fact that this "symmetry breaking" does not affect performance is not surprising, since only radial information is necessary for classification. The precise mechanism that causes this to occur likely is sensitive to the training dynamics of the models.

We now focus our attention to the highest performing models: the order $k = 4$ moment EFNs. In Fig. 3, we show a radial slice of the $k = 4$, $L = 1$ latent space [shown fully in Fig. 2(d)]. The radial slice is taken at an azimuthal angle of 0 as a function of the rapidity y, though this choice is arbitrary. Even in cases where "symmetry breaking" occurs in the latent space, we find that the radial profile is largely the same up to normalization on any projection not along the mirror symmetry axis. Motivated by the form of the radial profile in Fig. 3, we fit the function,

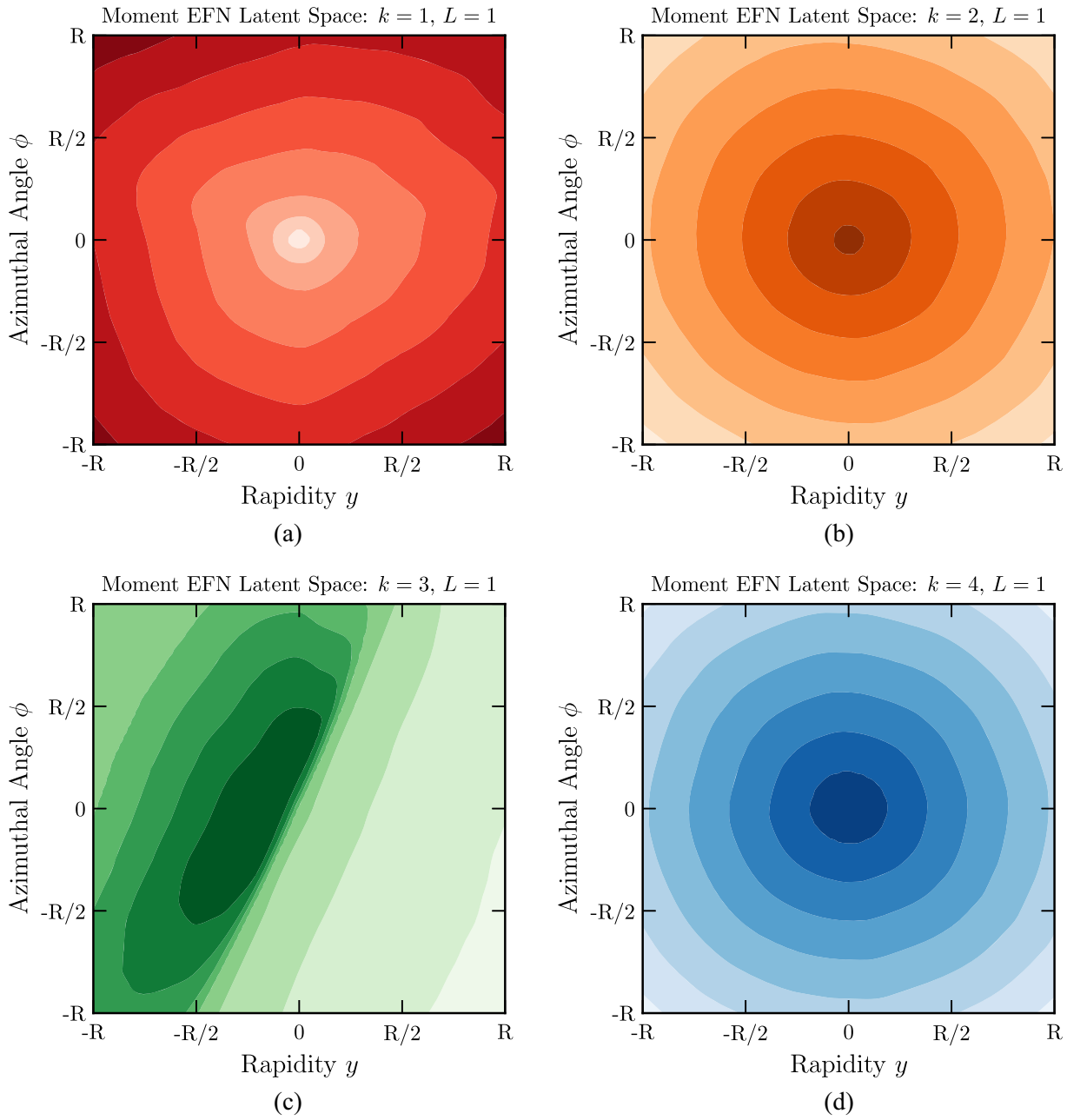

FIG. 2. Examples of learned moment EFN latent space embeddings $\Phi(x)$, for (a) $k = 1$, (b) $k = 2$, (c) $k = 3$, and (d) $k = 4$. Each figure represents the best model of the $L = 1$ trainings from Sec. III. The overall normalization is arbitrary. The $k = 3$ example features mirror rather than radial symmetry, which can generically occur for $k = 2$, 3, and 4.

$$\Phi_{\mathcal{L}}(r) = c_1 + c_2 \log(c_3 + r), \tag{12}$$

where r is the radial distance in the rapdity-azimuth plane from (0, 0), as defined by the energy-weighted average position of the jet.

This function provides an excellent fit to the latent function with $c_1 = -3.584$, $c_2 = -0.847$, and $c_3 = 0.005$. The values of c_1 and c_2 are largely unimportant, since they will be subject to affine transformations within the first layer of the F network, and these parameters vary significantly across retrainings. On the other hand, c_3 is consistently a small number in the range of 0.002 to 0.01, and is embedded in a logarithm which is more nontrivial for F to unravel.

The function $\Phi_{\mathcal{L}}$ has a divergence as $r \to 0$ (i.e., as particles become collinear with the jet center), but this divergence is regulated by the c_3 parameter. Interestingly, c_3 is within an $\mathcal{O}(1)$ factor of $\frac{\Lambda_{\mathrm{QCD}}}{p_T R} \sim 0.001$, suggesting that

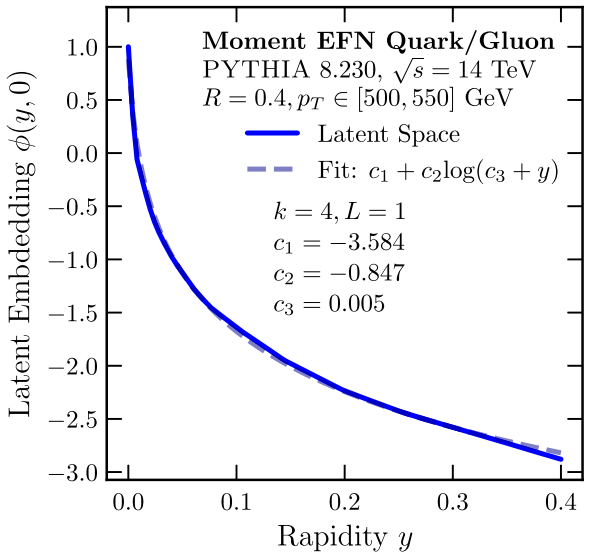

FIG. 3. A radial slice of the latent space for the best performing $k = 4$, $L = 1$ moment EFN, as a function of rapidity (y). The radial slice is taken at an azimuthal angle of zero. The latent space is shown as a dark blue line, and the logarithmic fit from Eq. (12) is shown as a blue dashed line.

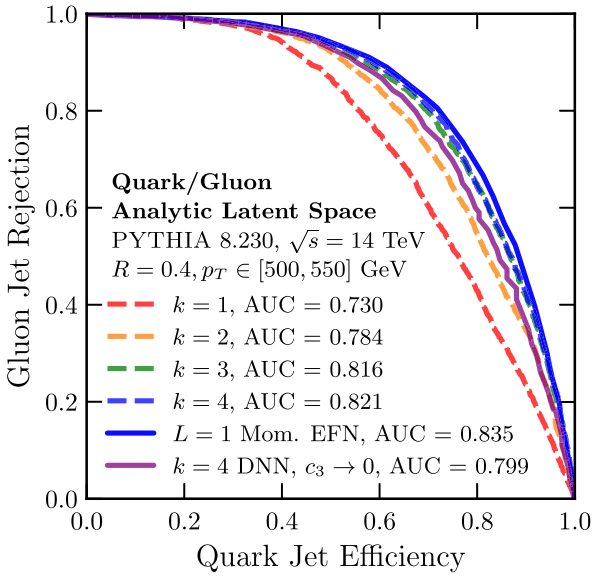

FIG. 4. ROC curves showing the performance of the analytic jet shape observables, as defined in Eq. (13), as a quark/gluon classifier. The jet shapes are passed into a dense neural net F as in Eq. (14). The ROC of the original $k = 4$, $L = 1$ moment (Mom.) EFN, from which the fits were derived, is shown in dark blue. Also shown in purple is a variant of the $k = 4$ DNN, except where the c_3 parameter is taken to zero.

the nonzero value of c_3 is due to genuine nonperturbative physics near the jet core learned by the moment EFN.

The moments of the function $\Phi_{\mathcal{L}}$ can be used to construct jet shape observables of the form,

$$\mathcal{L}^{(n)}(\mathcal{P}) = \langle \Phi_{\mathcal{L}}^n \rangle_{\mathcal{P}}, \\ = \sum_{i \in \mathcal{P}} z_i (c_1 + c_2 \log(c_3 + r_i))^n. \quad (13)$$

We call the observables $\mathcal{L}^{(n)}$ *log angularities*, since they resemble ordinary jet angularities $\lambda^\beta(\mathcal{P}) = \sum_{i \in \mathcal{P}} z_i r_i^\beta$ for $c_1 = c_2 = 0$. It is possible to generically set $c_1 = c_2 = 0$ by taking linear combinations of $\mathcal{L}^{(n)}$ for different n, but we elect to keep these parameters as it reduces the amount of total linear transformations our three hidden-layer dense networks have to do. These log angularities are interesting observables in their own right, especially in the $c_3 \to 0$ limit and are closely related to the $\beta \to 0$ limit of ordinary angularities [40,76], though we save a more in-depth theoretical discussion of log angularities for future work and here focus on their use as quark/gluon taggers.

We can use these analytic observables as inputs to a simple dense neural network classifier of the form,

$$F^{(k)}(\mathcal{L}^{(1)}, \ldots, \mathcal{L}^{(k)}). \quad (14)$$

If the $k = 4$ dense neural net classifier has the same performance as the full-order $k = 4$ moment EFN, then we can claim not only to have found a fully analytic form of the $k = 4$ latent space, but equivalently to have found a fully analytic form of the four different $L = 4$ ordinary EFN latent space dimensions.

In Fig. 4, we show ROC curves[7] of the classifier defined by Eq. (14) for $k = 1$ through 4. The F networks used here have precisely the same architecture and training procedure as those used for the moment EFNs in Sec. III, described in Appendix A. These results are also summarized in Table II. We also show, in purple, the $k = 4$ DNN classifier taking c_3 to 0. From this plot, we can make several observations:

(1) *The $k = 4$ dense model is as good as the $k = 4$ moment EFN*: We can replace the neural network latent dimension $\Phi(x)$ with the much simpler $\Phi_{\mathcal{L}}(r)$ when $L = 1$. Moreover, since the $k = 4$, $L = 1$ moment EFN is just as good as the $L = 4$ ordinary EFN, the single function $\Phi_{\mathcal{L}}(r)$ and its powers captures the same information as 4 dimensions worth of latent space in an ordinary EFN.

(2) *The $k = 1, 2$, and 3 dense models are not as good as their corresponding order-k moment EFNs*: The AUCs of the dense models are slightly lower than the corresponding $L = 1$ moment EFNs in Table I for $k < 4$. This suggests that while the *combination* of $\mathcal{L}^{(1)}, \mathcal{L}^{(2)}, \mathcal{L}^{(3)}, \mathcal{L}^{(4)}$ are optimal, individually they are not, and $\mathcal{L}^{(1)}$ by itself is not the most optimal

[7]The receiver operating characteristic (ROC) curve of a classifier quantifies the background rejection rate as a function of the signal acceptance rate, and is given by $\mathrm{ROC}(\lambda) = 1 - P_g(P_q^{-1}(\lambda))$.

TABLE II. The same as Table I, but with dense neural networks on log angularities as defined in Eq. (14). The "3+" and "2+" in the trainable parameters column refer to the parameters in the log angularity fit.

Model	AUC	$1/\epsilon_g$ at $\epsilon_q = 0.3$	$1/\epsilon_g$ at $\epsilon_q = 0.5$	Trainable parameters
$k = 1$ log angularity DNN	0.730 ± 0.001	50.5 ± 1.2	8.5 ± 0.9	$3 + 20702$
$k = 2$ log angularity DNN	0.784 ± 0.001	$\mathbf{72.0 \pm 1.7}$	13.5 ± 0.8	$3 + 20802$
$k = 3$ log angularity DNN	0.816 ± 0.001	59.9 ± 1.7	$\mathbf{19.4 \pm 1.1}$	$3 + 20902$
$k = 4$, log angularity DNN	$\mathbf{0.821 \pm 0.002}$	55.6 ± 2.0	18.5 ± 1.2	$3 + 21002$
$k = 4$, $c_3 \to 0$ DNN	0.799 ± 0.001	60.7 ± 1.8	15.5 ± 1.0	$2 + 21002$

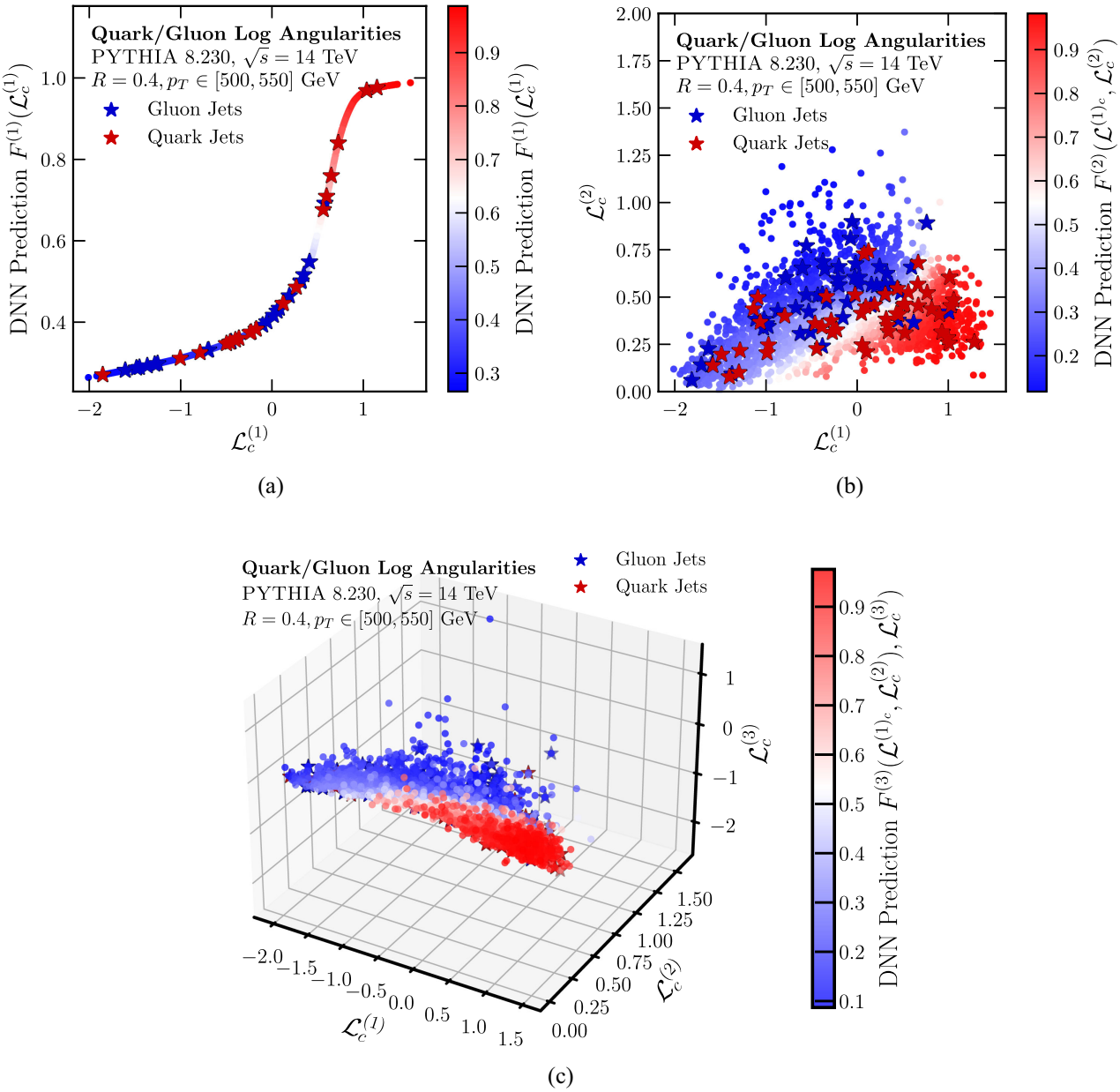

FIG. 5. The distribution of the dense neural network output (a) $F^{(1)}$, (b) $F^{(2)}$, and (c) $F^{(3)}$ as a function of the first, first two, and first three (cumulant) log angularities, respectively. The true quark/gluon label for several random jets are indicated with colored stars.

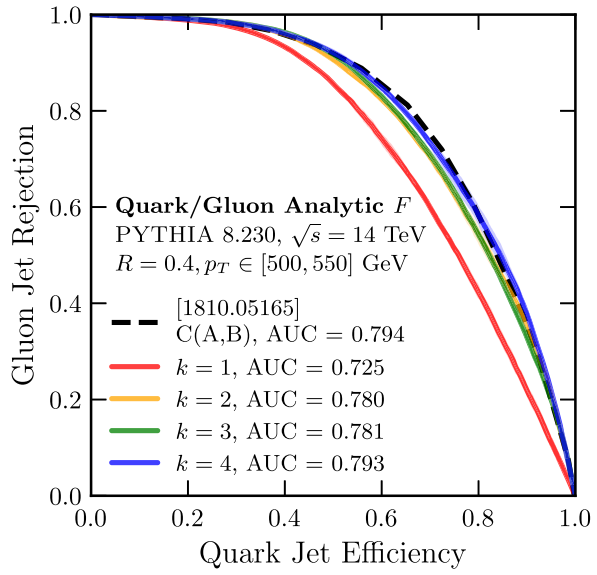

FIG. 6. ROC curves showing the performance of the closed form observables, as defined in Eq. (19), as a quark/gluon classifier. For comparison, we also show the ROC curve of the observable $C(A, B)$, a closed-form observable based on fits to an $L = 2$ EFN as defined in Ref. [34].

single-variable observable for IRC-safe quark/gluon discrimination.

(3) *The c_3 parameter matters*: Taking the parameter c_3 to zero reduces the AUC of the $k = 4$ models significantly. Interpreting c_3 as a nonperturbative parameter regulating a collinear divergence in the logarithm, this loss in performance can be viewed as the learned effect of nonperturbative physics in quark/gluon discrimination.

Thus, at least for $k = 4$, we have successfully cast the latent space of not only the moment EFN, but the equivalent $L = 4$ ordinary EFN, into closed form.

B. $L = 1$ F networks

Next, we attempt to go further by attempting to also find closed-form expressions for the dense neural network classifiers F, building off of the analysis performed in Sec. IV A where we found closed-form expressions for the latent space network Φ. This would result in a fully closed-form quark/gluon jet classifier. However, analyzing F is inherently more difficult than the individual Φ functions, as for the $k = 4$, $L = 1$ model of interest, F is a function of four inputs. Moreover, we know F must be nontrivial in all four inputs, since otherwise there would be no difference between the $L = 1$ networks at different values of k.

To aid in determining a functional form of F, we begin by plotting the output of the dense neural networks [Eq. (14)] considered in Sec. IV A as a function of $\mathcal{L}^{(k)}$, for $k = 1$, 2, and 3. To be explicit, we plot $F^{(1)}(\mathcal{L}^{(1)})$ in Fig. 5(a), plot $F^{(2)}(\mathcal{L}^{(1)}, \mathcal{L}^{(2)})$ in Fig. 5(b), and plot $F^{(3)}(\mathcal{L}^{(1)}, \mathcal{L}^{(2)}, \mathcal{L}^{(3)})$ in Fig. 5(c).[8] We will find it convenient to work with "cumulant" log angularities $\mathcal{L}_c$, rather than ordinary log angularities, as this makes the distributions in Fig. 5 and the resulting fits simpler. The cumulant log angularities are defined as

$$\mathcal{L}_c^{(1)} = \mathcal{L}^{(1)}, \tag{15}$$

$$\mathcal{L}_c^{(2)} = \mathcal{L}^{(2)} - [\mathcal{L}^{(1)}]^2, \tag{16}$$

$$\mathcal{L}_c^{(3)} = \mathcal{L}^{(3)} - 3\mathcal{L}^{(1)}\mathcal{L}^{(2)} + 2[\mathcal{L}^{(1)}]^3, \tag{17}$$

$$\mathcal{L}_c^{(4)} = \mathcal{L}^{(4)} - 4\mathcal{L}^{(1)}\mathcal{L}^{(3)} - 3[\mathcal{L}^{(2)}]^2 + 12\mathcal{L}^{(2)}[\mathcal{L}^{(1)}]^2 - 6[\mathcal{L}^{(1)}]^4. \tag{18}$$

In all three plots, we see that the DNN output is, for the most part, cleanly divided into distinct regions in $\mathcal{L}_c^{(i)}$ space. This motivates using a weighted distance from a learned reference point in $\mathcal{L}_c^{(i)}$ space as a classifier, with the ansatz,

$$F^{(k)}(\mathcal{L}^{(1)}, \ldots, \mathcal{L}^{(k)}) = \sigma\left(w_0 + \sum_{i=1}^{k} w_i (\mathcal{L}_c^{(i)} - b_i)^2\right), \tag{19}$$

where w_i and b_i are parameters to be minimized, and σ is the sigmoid function. This classifier arises naturally as the (sigmoid of) the log-likelihood if we assume that the $\mathcal{L}_c$ are Gaussian distributed with means b_i and variances $1/(2w_i)$, motivated by the observation that in Fig. 5 the distributions form rough contiguous blobs. The number of parameters in this classifier is naively $4 + 2k$, with 3 from the log angularity fit, 1 from w_0, and $2k$ from w_i and b_i. However, because any monotonic function of a classifier is an equally good classifier, this can be reduced to $2 + 2k$ by removing w_0 and an overall scale from the w_i's.

In Fig. 6, we show ROC curves corresponding to the observables defined in Eq. (19). We also summarize these results in Table III. For $k > 1$, the performance saturates around an AUC of 0.78–0.79, which is roughly the performance of an $L = 2$ EFN. Unlike the case where F was a dense network, however, the performance does not improve significantly beyond $k = 2$—moreover, these results are largely uncharged for various modifications to the functional form of Eq. (19), including extending to up to degree 4

[8]Unfortunately, we are unable to display the full plot of $F^{(4)}(\mathcal{L}^{(1)}, \mathcal{L}^{(2)}, \mathcal{L}^{(3)}, \mathcal{L}^{(4)})$, since the PDF file format is not yet available in more than two dimensions.

TABLE III. The same as Table I, but with the closed-form expressions defined in Eq. (19). The "3+" in the trainable parameters column refers to the parameters in the log angularity fit.

Model	AUC	$1/\epsilon_g$ at $\epsilon_q = 0.3$	$1/\epsilon_g$ at $\epsilon_q = 0.5$	Trainable parameters
$k = 1$ log angularity closed form	0.725 ± 0.001	36.2 ± 0.2	7.0 ± 0.0	$3 + 1$
$k = 2$ log angularity closed form	0.780 ± 0.002	57.4 ± 0.2	10.6 ± 0.1	$3 + 3$
$k = 3$ log angularity closed form	0.781 ± 0.002	$\mathbf{57.8 \pm 2.8}$	12.1 ± 0.1	$3 + 5$
$k = 4$ log angularity closed form	$\mathbf{0.793 \pm 0.002}$	54.6 ± 1.7	$\mathbf{12.7 \pm 0.4}$	$3 + 7$

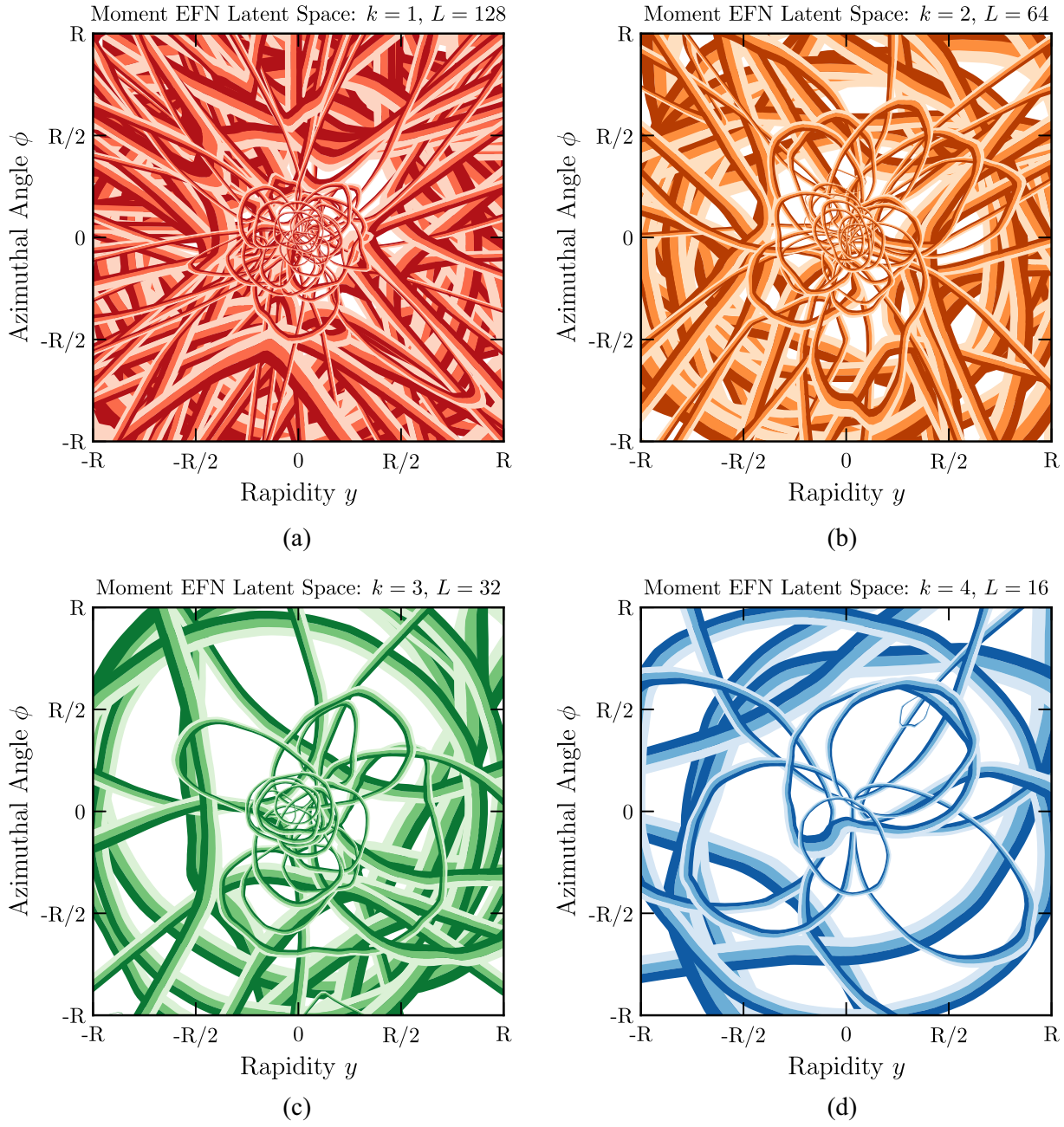

FIG. 7. The contours of the learned moment EFN latent space embeddings Φ, for (a) $k = 1$, (b) $k = 2$, (c) $k = 3$, and (d) $k = 4$. Each figure represents the best model for the highest value of L considered for each k; 128, 64, 32, and 16, respectively. Each curve represents the 45%–55% contours of each of the L different Φ functions. The overall normalization is arbitrary.

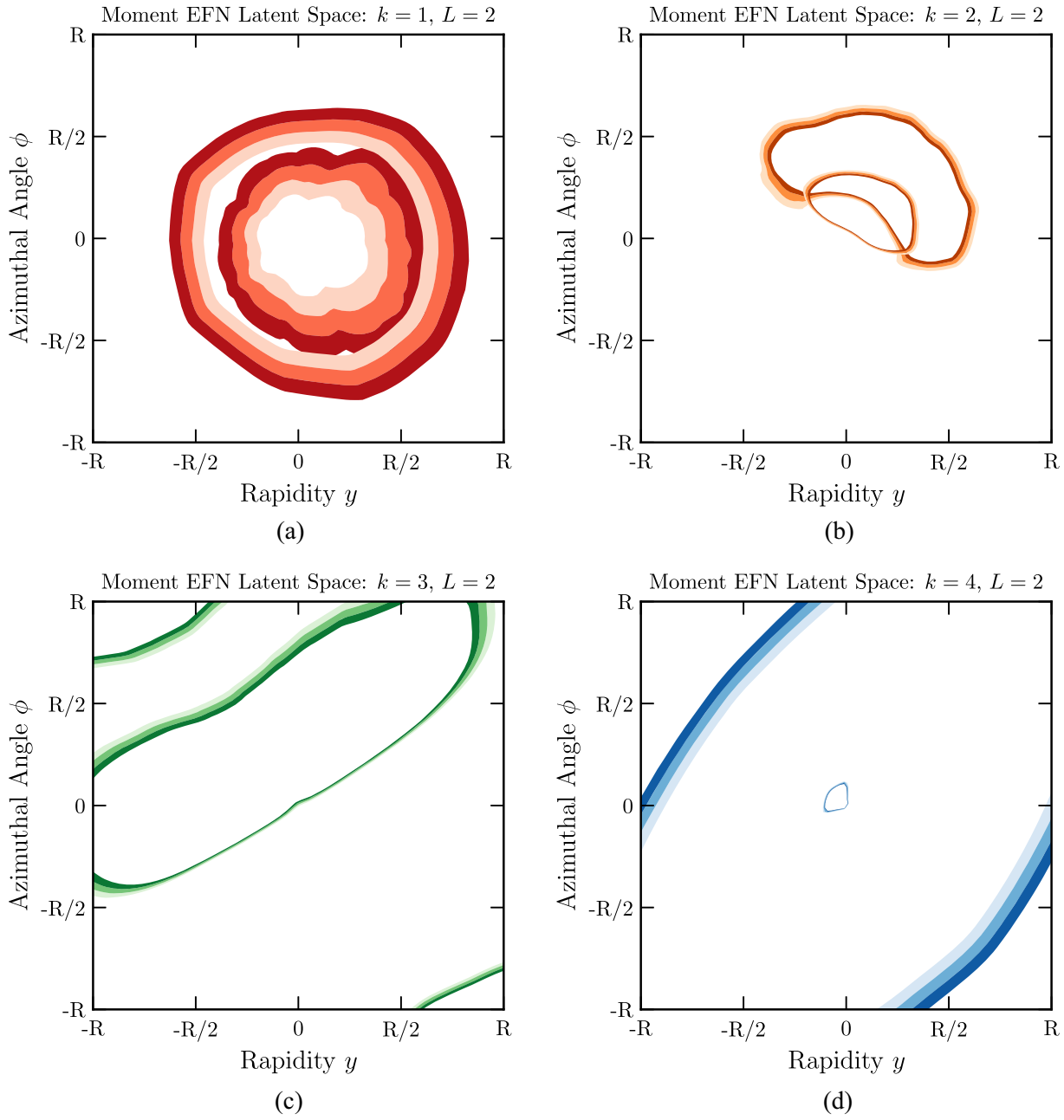

FIG. 8. The contours of the learned $L = 2$ moment EFN latent space embeddings Φ, for (a) $k = 1$, (b) $k = 2$, (c) $k = 3$, and (d) $k = 4$. Each curve represents the 45%–55% contours of each of the two different Φ functions. The overall normalization is arbitrary.

polynomials in $\mathcal{L}$ and $\mathcal{L}_c$ and even up to degree 4 rational polynomials.[9] This suggests that the contributions of higher-order moments beyond the first two are more complex and unable to be easily parametrized; that is, while it is easy to *encode* the latent space information into a smaller L via Eq. (10), the *decoding* of the effective latent space in F is nontrivial.

For comparison, we also show the ROC curve of the observable $C(A, B)$, an observable with a similar functional form defined in Ref. [34] based off of fits to an $L = 2$ ordinary EFN.[10] This observable also saturates at roughly

[9]There are a variety of ways one can combine cumulants one could try instead, for example, the Gram-Charlier A series or the Edgeworth Series to approximate probability distributions given cumulants [77].

[10]The EFNs in Ref. [34] use the ReLU rather than LeakyReLU [78] activation, which slightly changes the small-L behavior relative to the studies here.

the same AUC. Thus, while we have succeeded in developing a fully closed-form solution suitable for up to $k = 2$, equivalent to two latent dimensions of an ordinary EFN, finding a closed form expression for F beyond two effective latent dimensions is nontrivial.

C. Beyond $L = 1$

Finally, we briefly look at moment EFNs with $L > 1$ to see if we can attempt to gain some insight into them, just like the $L = 1$ moment EFNs. Unlike the case with $L = 1$ moment EFNs, the analysis of $L > 1$ models is much more difficult, both due to the higher dimension and because the product structure allows radial symmetry to be more easily broken, even for $L = 2$.

In order to visualize the latent spaces for $L > 1$, we follow the procedure outlined in Ref. [34]. We can overlay all L learned Φ functions of each model at once by drawing the 45%–55% contours of each Φ function. In these plots, the overall normalization and sign of the Φ functions is unimportant, as the F network easily learn to rescale its inputs via simple linear transformations. As an example, in Fig. 7, we show the $L = 128$, 64, 32, and 16 (the highest L considered for each k) dimensional latent spaces of the highest performing models for the order $k = 1$, 2, 3, and 4 moment EFNs, respectively. Like the EFN, the moment EFN is able to pick up on the collinear singularity of QCD, as filters closer to the center are more closely resolved. Note that while on the whole, the entire ensemble of contours appears to be radially symmetric, the individual contours are not—this in contrast with the $L = 1$ moment EFNs, where the Φ functions were genuinely radially symmetric (or at the very least, broken to mirror symmetric).

After $L = 1$ from Fig. 2, the next most natural thing to study are the $L = 2$ latent spaces, which we show in Fig. 8. Not only is $L = 2$ less radially symmetric than $L = 1$ in general, we also notice the $k > 1$, $L = 2$ moment EFNs exhibit less radial symmetry than the $L = 2$ ordinary EFN. With product structures, it is easier to form nontrivial representations of the rotation group that later combine to form the trivial representation. As an example, the functions $\Phi^1(x) = x^{(1)}$ and $\Phi^2(x) = x^{(2)}$ are themselves not radially symmetric, but the $k = 2$ moment combination $F(\Phi) = (\Phi^1)^2 + (\Phi^2)^2$ is. Nevertheless, some models occasionally still seem to retain some approximate radial symmetry, as is the case for the order $k = 4$ model in Fig. 8(d).

We now move to analyze and fit the radial projections of the order $k = 4$, $L = 2$ moment EFN in hopes of finding closed-form expressions for the (now two) latent dimensions, exactly as was done in Sec. IV A. This model has $L_{\text{eff}} = 15$, which is roughly equivalent to an ordinary EFN with 14 latent dimensions. In principle, fitting these two functions should therefore enable us to extract the same information as 14 latent dimensions worth of information in an ordinary EFN by taking moments. These radial projections are shown in Fig. 9. One of the two latent dimensions takes the form of a log angularity, as defined in Eq. (13), and the c_3 parameter is also in the expected range. We will call the associated jet shape observable $\mathcal{L}'$, with the prime indicative of the slightly different values of the fit parameters c_i. For the second latent dimension, we fit the form,

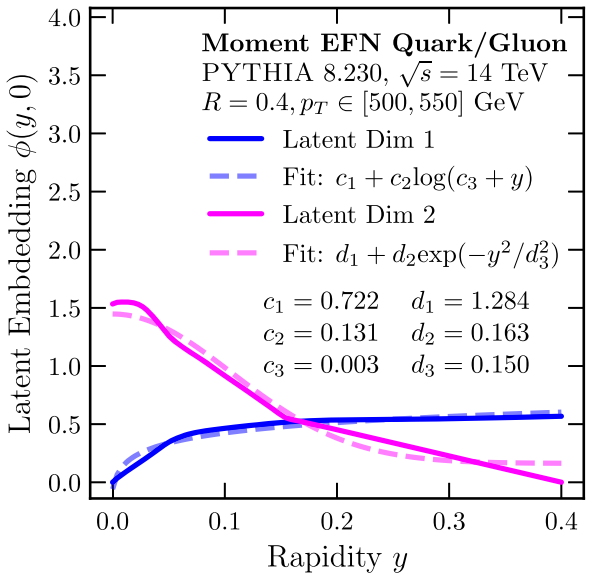

FIG. 9. The same as Fig. 3, but for the best performing $k = 4$, $L = 2$ moment EFN. The two latent dimensions are shown in blue and pink.

$$\Phi_{\mathcal{E}}(r) = d_1 + d_2 \exp(-r^2/d_3^2), \tag{20}$$

motivated both by the form of the plot and the use of a similar form in the analysis of the $L = 2$ EFN in Ref. [34]. Just as with log angularities, we can define moment-based jet shape observables based off this fit, which we will denote $\mathcal{E}^{(n)}$, and associated cumulants $\mathcal{E}_c^{(n)}$. We can also define mixed moments $\mathcal{M}^{(mn)}$ of the form $\mathcal{M}^{(mn)} \equiv \langle \Phi_{\mathcal{L}'}^m \Phi_{\mathcal{E}}^n \rangle$.

We can use the observables $\mathcal{L}'^{(n)}$ and $\mathcal{E}^{(n)}$ as inputs to dense neural networks F as a measure of how well the latent spaces are approximated by these fits. Here, the dense network $F^{(k)}$ takes in *all* $L_{\text{eff}}(k)$ multivariate moments of the two observables; for instance, the $k = 2$ dense network has the form $F^{(2)}(\mathcal{L}'^{(1)}, \mathcal{E}^{(1)}, \mathcal{L}'^{(2)}, \mathcal{E}^{(2)}, \mathcal{M}^{1,1})$. We show the performance of these networks in Fig. 10(a). We see here that unlike the $L = 1$ analysis performed in Sec. IV A, the two observables here are not enough to reproduce the full $L = 2$, $k = 4$ moment EFN; in fact, they are only about as good as the $L = 1, k = 4$ result at best. This means that the fits are not enough to capture the full latent space, suggesting that non radially symmetric information is important.

Finally, as was done in Sec. IV B, we may attempt to build closed-form taggers from these fits. To accommodate the second observable, we extend the form of our fit to

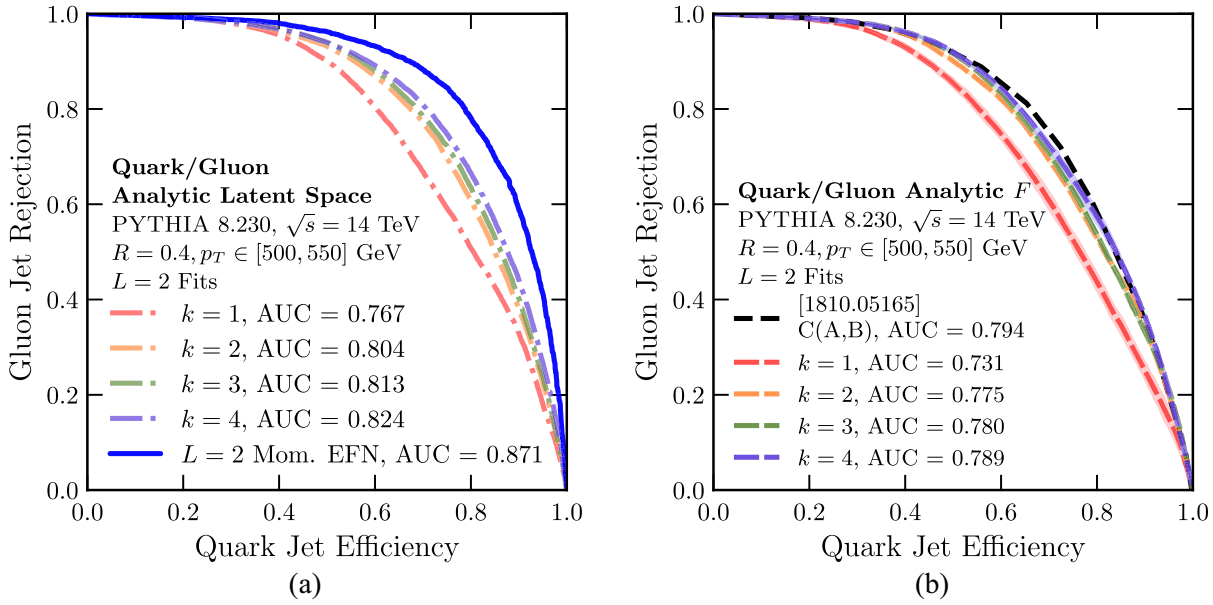

FIG. 10. (a) The same as Fig. 4, but with the $L = 2$ fits and dense networks. (b) The same as Fig. 6, but for the closed-form F defined using $L = 2$ in Eq. (21).

$$F^{(k)}(\mathcal{L}^{(i)}, \mathcal{E}^{(i)}) = \sigma\Bigg(w_0 + \sum_{i=1}^{k} w_i^{\mathcal{L}'}(\mathcal{L}_c^{(i)} - b_i^{\mathcal{L}'})^2 + \sum_{i=1}^{k} w_i^{\mathcal{E}}(\mathcal{E}_c^{(i)} - b_i^{\mathcal{E}})^2\Bigg). \tag{21}$$

For simplicity, we have not considered mixed moments between the two observables. We show the performance of these taggers in Fig. 10(b). The observables here are no better than the $L = 1$ analytic observables shown in Fig. 6; in fact, they are slightly worse, due to the extra numerical cost of optimizing additional parameters.[11] That is, unlike the EFN observable $C(A, B)$, the additional parametrization power granted by an additional latent dimension in a moment EFN is more complex than can be captured by a simple functional dependence.

V. CONCLUSIONS AND OUTLOOK

In this paper, we presented an extension of the deep sets framework in the form of moment pooling. Moment pooling arises when the summation operation in deep sets is generalized to arbitrary multivariate moments. We have shown an implementation of moment pooling in moment EFNs, a particular deep sets framework useful in particle physics. For a fixed number of latent dimensions L, an order-k moment EFN is able to reach a much higher *effective* latent dimension by recycling the same L functions in product structures, and conversely, a moment EFN is capable of reducing the L required to achieve the same performance as an ordinary EFN.

We find that moment EFNs are able to achieve the same (or better) performance as ordinary EFNs for quark/gluon discrimination, but with significantly fewer latent dimensions. In particular, an order $k = 4$ moment EFN with only 16 latent dimensions is able to achieve slightly better performance than an EFN with 128 latent dimensions, indicative of "data compression" and structure in the quark/gluon discrimination task. Similarly, an order $k = 4$ moment EFN with only a single latent dimension is able to achieve an AUC of 0.84, equivalent to an ordinary EFN with four latent dimensions. By analyzing the latent space of this model, we are able to find a simple, closed-form expression in the form of the log angularity shape observable, whose moments contain the same information as four latent dimensions of an ordinary EFN. However, we find that analyzing the F network in closed-form is difficult, and that simple parametrizations cannot go beyond the performance of up to an $L = 2$ model, which suggests some complexity in the way the information in log angularities is decoded.

We conclude by discussing possible avenues of further study. One can ask if the latent space structure granted by the moment architecture helps in learning richer jet representations, i.e., if the latent representation is a genuine representation of a jet from which multiple attributes and observables can be estimated, not just a quark/gluon discriminant. The latent spaces of EFNs are highly degenerate, and as we have seen, the four latent dimensions of an $L = 4$ EFN may be replaced by four moments of log angularities without any loss of performance on quark/gluon tagging. Second, we

[11]This holds for many modifications of Eq. (21), including rational polynomials.

have *only* considered moments, in the sense that we have restricted our architecture to only have terms of the form $\sum_{i\in\mathcal{P}} z_i(\Phi^1(x_i))^{w_1}(\Phi^2(x_i))^{w_2}\ldots(\Phi^L(x_i))^{w_L}$, for predetermined positive integers w_l such that $\sum_l w_l \leq k$. One could imagine generalizing this even further, for example by letting the powers w_l be real and negative, or even learned. Third, our study here primarily focused on IRC-safe observables and EFNs, but the moment pooling concept is applicable to any deep sets-style architecture. In particular, non-IRC safe observables such as multiplicity are known to aid in quark/gluon tagging, and indeed the studies in Appendix C additionally show there is potential for gain when applying moment pooling to non-IRC safe PFNs. It may be possible, therefore, to use moment pooling or related concepts to help compress the latent spaces of more sophisticated point-cloud based approaches, such as ParticleNet [15], Particle Transformer [26], or PELICAN [31].

The moment pooling operation and moment EFN are a step towards generalizing existing models while adding to their interpretability. We look forward to further developments in the direction of flexible models with interpretable internal representations.

The general moment EFN, along with several variations (including cumulant-based models and moment PFNs), is available at [79]. The code used to perform all the analyses and make all the figures featured in this paper is available at [80] in the form of Jupyter notebooks [81].

ACKNOWLEDGMENTS

We would like to thank Samuel Alipour-fard, Sean Benevedes, Miles Cranmer, Andrew Larkoski, Benjamin Nachman, and Sokratis Trifinopoulos for useful and interesting discussions and comments. R. G. and J. T. are supported by the National Science Foundation under Cooperative Agreement No. PHY-2019786 (The NSF AI Institute for Artificial Intelligence and Fundamental Interactions, [82]), and by the U.S. DOE Office of High Energy Physics under Grant No. DE-SC0012567. J. T. is additionally supported by the Simons Foundation through Investigator Grant No. 929241. A. O. was supported by the Bowdoin College Summer Internship Program.

APPENDIX A: MODEL AND TRAINING SPECIFICATIONS

In this appendix, we provide details for the models and training procedures used in Sec. III and Appendix C. All models are implemented as modified versions of the EFN/PFN models in the EnergyFlow Python package [34], built with Keras [83] using the TensorFlow [84] backend. Each training is performed using an NVIDIA A100.

The key difference between the moment EFN and the ordinary EFN is the addition of the `MomentPooling` layer between the Φ and F functions. The `MomentPooling` layer is a deterministic function $\mathbb{R}^L \to \mathbb{R}^{L_{\text{eff}}}$, where L_{eff} depends on both L and the order k, that maps the L-component Φ to the list of all multivariate moments up to order k, taken over the event $\mathcal{P}$:

$$\Phi^a \mapsto (\langle\Phi^a\rangle_{\mathcal{P}}, \langle\Phi^{a_1}\Phi^{a_2}\rangle_{\mathcal{P}}, \ldots\langle\Phi^{a_1}\ldots\Phi^{a_k}\rangle_{\mathcal{P}}). \tag{A1}$$

Implementation-wise, the `MomentPooling` layer takes in a TensorFlow tensor of shape $(N_{\text{batch}}, N_{\text{particles}}, L)$, where N_{batch} is the number of events to be computed in parallel, $N_{\text{particles}}$ is the maximum number of particles per event,[12] and L is the latent dimension; this tensor is obtained as the output of the Φ network. The moments are then computed recursively; we first define the $k=0$ moment tensor, a tensor of shape $(N_{\text{batch}}, N_{\text{particles}}, 1)$ with every entry equal to 1. Given the kth moment tensor, which is of shape $(N_{\text{batch}}, N_{\text{particles}}, L_{\text{eff}}(k))$, the $(k+1)$th moment tensor is obtained by performing the outer product of the original input tensor, to obtain a new tensor of shape $(N_{\text{batch}}, N_{\text{particles}}, L_{\text{eff}}(k), L)$. Note that this outer product will contain redundant moments, since the order of indices does not matter, thus, we only perform the outer product on the indices corresponding to the upper triangular part of the $k+1$-dimensional hypercube with L indices per dimension. The tensor is then flattened and concatenated to the kth moment tensor to form the $(k+1)$th moment tensor of shape $(N_{\text{batch}}, N_{\text{particles}}, L_{\text{eff}}(k+1))$. At the end of the recursive procedure, we perform the z-weighted sum over the $N_{\text{particles}}$ dimension, so that the final output has shape $(N_{\text{batch}}, L_{\text{eff}}(k))$. This recursive procedure allows for some computations to be reused when computing higher-order moments, simplifying the TensorFlow computational graphs and saving time on backpropagation versus recomputing all moments from scratch.

Following Ref. [34], all of our models consist of a Φ network with three layers of sizes 100, 100, and L, respectively (with L being the latent dimension), and an F network with four layers of size 100,100,100, and 1, respectively. For both networks, the final layer is the output layer. In between Φ and F is the `MomentPooling` layer. We use LeakyReLU [78] with $\alpha = 0.3$ for all activation functions,[13] except for the final layer of F, where we use a sigmoid function for the classifier output.

To aid our models in learning efficient representations, especially for $k > 1$, we use a "pretraining" procedure. This procedure helps to ensure that each additional moment added is used to learn "new" information that helps the model and to mitigate the effect of more nontrivial training for higher-order k moment EFNs. To train an order-k

[12]While deep sets, in theory, allows for an unbounded number of particles, it is more practical for speed and memory to have a fixed cap.

[13]This is to avoid the Dying ReLU problem [85], especially for smaller L.

moment EFN, first, we first train the order $k-1$ moment EFN with the exact same value of L and the same hyperparameters for the Φ and F networks, using the training procedure defined below. Then, we initialize an order-k network whose Φ and F weights are identical to the order $k-1$ model's weights, *except* for the weights attached to the `MomentPooling` layer, as the size of this layer has changed. The weights connecting the first $L_{\text{eff}}(k-1)$ outputs of the `MomentPooling` layer to the first F layer are the same as the $k-1$ network weights. Finally, the order k model can be trained. This pretraining procedure is recursive: to train the order $k-1$ model, we initialize its weights from an order $k-2$ model, and so on. The weights of the $k=1$ models, plus all other undetermined weights (namely the rest of the weights connecting the `MomentPooling` layer to the first F layer) are initialized using the default He-uniform [86] distribution. To ensure that any improvements are not the result of having more epochs to train, in our comparative studies lower k models have the training procedure described below applied to them multiple times so that all models train for the same total number of epochs. That is, since we study up to $k=4$ moment EFNs, all of our models are trained effectively for 4× the number of stated epochs.

Each model is trained for 50 epochs (times 4, with the recursive pretraining) with a batch size of 512; we allow for early stopping with a patience parameter of 8, though early stopping never occurred in any of our trainings. We have checked that allowing for a maximum of 150 epochs, rather than 50, per training round makes no significant difference in our results. We use 1M total jets for training, 50 k for validation, and 50 k for testing. We train to minimize the binary cross-entropy loss using the Adam optimizer [87] with a learning rate of 0.001. The training times of each network per epoch on an NVIDIA A100 are shown in Table IV. Each model is reinitialized and retrained three times, the ROC curves and AUC saved for each of the trainings. Note that we do not train models with both larger k and larger L, as the training time and memory requirements become excessive.

TABLE IV. The time per epoch, in seconds, for the $k=1$ through 4 moment EFNs to train on the quark/gluon discrimination dataset.

Latent dimension L	EFN	$k=2$	$k=3$	$k=4$
2^0	2s	3s	3s	4s
2^1	3s	3s	3s	4s
2^2	3s	3s	4s	5s
2^3	3s	4s	6s	11s
2^4	3s	6s	16s	66s
2^5	4s	13s	91s	
2^6	4s	38s		
2^7	5s			

APPENDIX B: REGRESSION WITH JET ANGULARITIES

In this appendix, we demonstrate the improved performance and reduced complexity of moment EFNs in regression tasks by exploring their relationship to jet angularities [41,42].

Jet angularities are well-studied QCD observables that quantify the radial distribution of energy within a jet. The product structure of the moment EFN is especially suited for angularities, since a jet angularity can be thought of as an energy-weighted radial moment. For a jet $\mathcal{P}$, the β-angularity of the jet, $\lambda^{(\beta)}$, is defined as

$$\lambda^{(\beta)}(\mathcal{P}) \equiv \sum_{i\in\mathcal{P}} z_i |x_i|^\beta, \tag{B1}$$

where $x_i = (y_i, \phi_i)$ are the particle coordinates defined from an appropriately defined jet center x_0, which we will take to be the energy-weighted average position of the jet.

In the language of moments, jet angularities take a very natural form,

$$\lambda^{(\beta)}(\mathcal{P}) = \langle |x_i|^\beta \rangle. \tag{B2}$$

If β is an even integer, then $\lambda^{(\beta)}$ can be expressed using a completely *linear* moment EFN with $k=\beta$ and $L=2$,

$$\Phi^1(x_i) = y_i, \qquad \Phi^2(x_i) = \phi_i, \tag{B3}$$

$$F(\langle\Phi\rangle_{\mathcal{P}}) = \sum_{a_1=1}^{2} \cdots \sum_{a_{\beta/2}=1}^{2} \langle (\Phi^{a_1})^2 \ldots (\Phi^{a_{\beta/2}})^2 \rangle_{\mathcal{P}}, \tag{B4}$$

where $a=1,2$ corresponds to the two dimensions of x. In particular, for $\beta=2$, this becomes

$$F(\langle\Phi\rangle_{\mathcal{P}}) = \langle y^2 \rangle + \langle \phi^2 \rangle. \tag{B5}$$

The $\beta=2$ angularity is especially nice, as it relates to the jet mass as

$$\lambda^{(2)} \approx \frac{m_J^2}{(\sum E_i)^2}. \tag{B6}$$

The fact that angularities can be represented as a linear function of moments is significant, as dense neural networks, especially those using variants of ReLU, are at their core are piecewise-linear approximators.[14] The nonlinear part of the angularities, namely the $|x|^\beta$ function, is encoded in the prespecified moment functions, leaving only a purely linear function to learn. Thus, one would expect a strictly linear moment EFN to learn even integer β angularities *exactly* (that is, with a mean squared error loss of zero) for

[14]This is still qualitatively true for other activation functions.

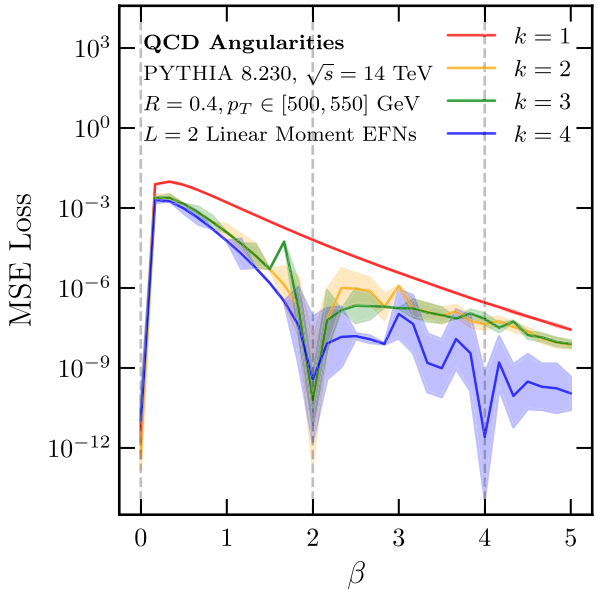

FIG. 11. The (average) MSE loss of the k moment EFN trained to regress the jet angularity $\lambda^{(\beta)}$ as a function of β. Each line is the average of the MSE across three retrainings (in log space), and the bands are the standard deviation (in log space).

$k \geq \beta$, and in general that $k > 1$ moment EFNs outperform $k = 1$ EFNs on generic regression tasks.

To illustrate the improved performance of the moment EFN on regression tasks, we train $L = 2$ moment EFNs from $k = 1$ through 4 to learn the angularity $\lambda^{(\beta)}$, for values of $\beta \in [0, 5]$. The moment EFNs are strictly linear; the functions Φ and F are *strictly linear* functions from $\mathbb{R}^2 \to \mathbb{R}^L$ and $\mathbb{R}^{L_{\text{eff}}} \to \mathbb{R}$, respectively, with no hidden layers, activation functions, or even layer biases. We use the exact same quark/gluon dataset described above in Sec. III A for this study, though we ignore the quark/gluon label. On each jet, we compute the β-angularity as defined in Eq. (B1), for β from 0 to 5 in increments of $\Delta\beta = \frac{1}{6}$. The models are trained to minimize the mean-squared error (MSE) over the training set.

In Fig. 11, we plot the average MSE of each model, taken over the three retrainings, on the test set as a function of β. First, we note that the $k > 1$ models are consistently better than the $k = 1$ model for $\beta > 0$. Second, there are large downward spikes in the loss: at $\beta = 0$ for all models; at $\beta = 2$ for $k = 2$, 3, and 4; and at $\beta = 4$ for $k = 4$. The first two spikes are especially close to zero, with an MSE loss of $\sim 10^{-12}$ approaching floating point precision. This is precisely the behavior expected by Eq. (B4), as the linear model is able to achieve an exact (up to machine precision) fit for even integer β with $k \geq \beta$.

APPENDIX C: ADDITIONAL COLLIDER CLASSIFICATION STUDIES

In this appendix, we present additional studies to supplement Sec. III, both by replacing the EFN in the moment architecture with a PFN, and by replacing the quark/gluon discrimination task with a top/QCD jet discrimination task.

The moment PFN models are identical to those described in Appendix A, except Φ is now a function of the particle (z_i, x_i) rather than just x_i, and Φ is no longer weighted by z. Importantly, this changes the definition of the moment pooling operation, since, as shown in Eq. (8) moments are *not* energy-weighted.

For our top/QCD dataset, we use the same top-tagging set as in Refs. [13,35], commonly used as a benchmark in

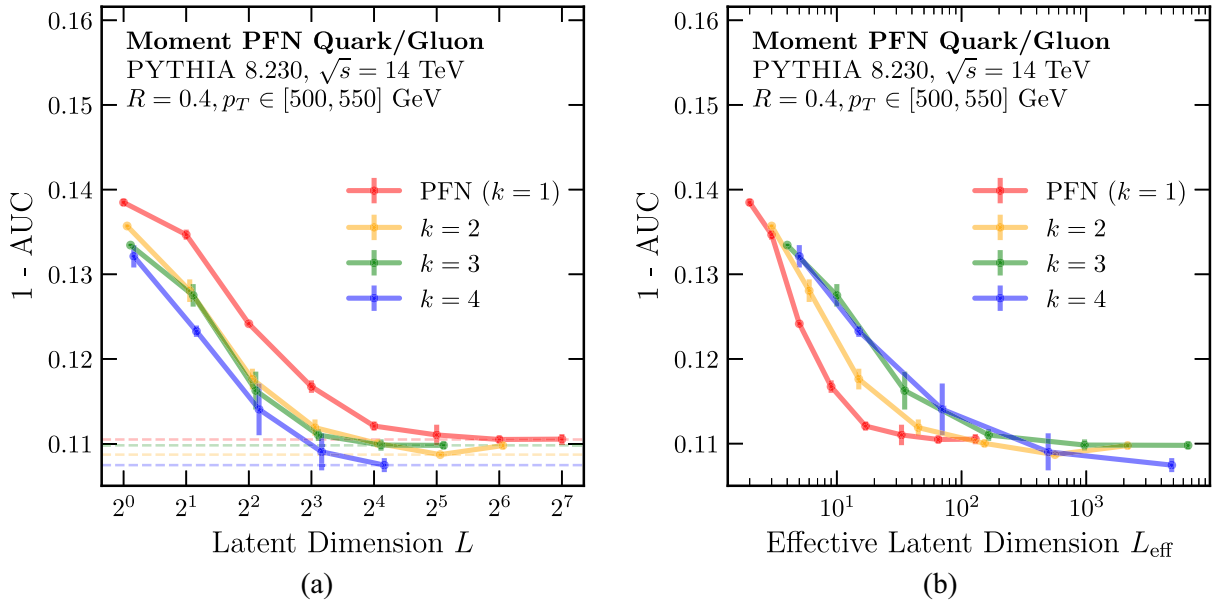

FIG. 12. (a) The same as Figs. 1(a) and (b) the same as 1(b), but with a moment PFN rather than a moment EFN.

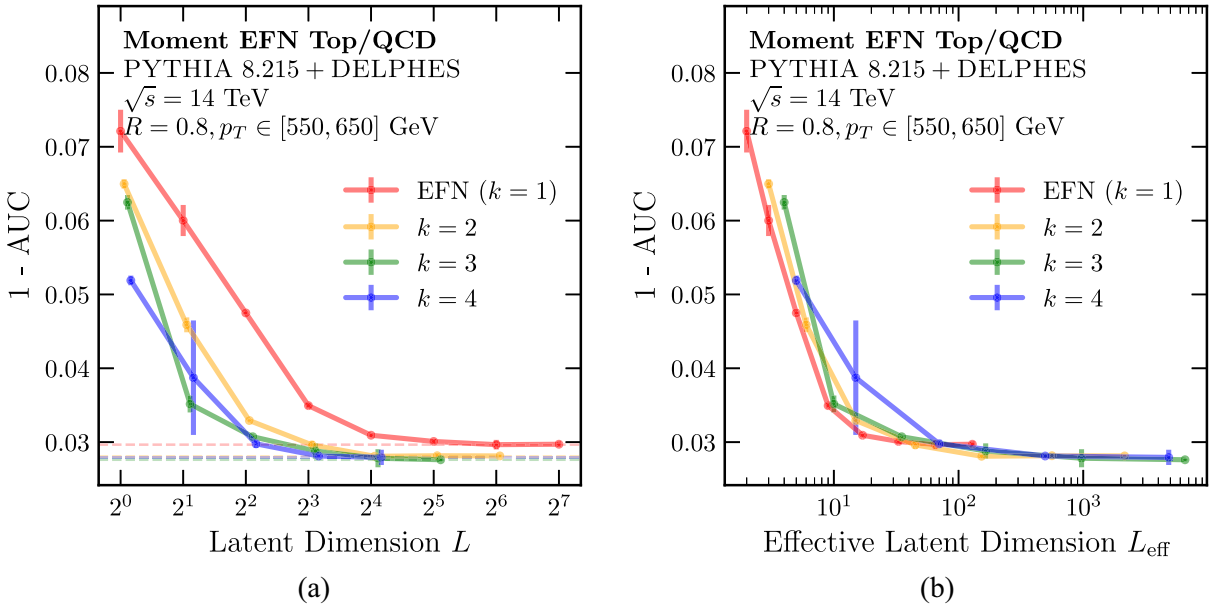

FIG. 13. (a) The same as Figs. 1(a) and (b) the same as 1(b), but with the top/QCD dataset rather than the quark/gluon dataset. Note the $k = 4$, $L = 2$ outlier, potentially due to a failed convergence.

tagger studies. This set consists of a top quark jet signal and a mixed light quark and gluon jet background, generated in Pythia 8.2.15 [54,55] at 14 TeV, and passed through the Delphes 3.3.2 [88] ATLAS detector simulation. The jets are clustered using the anti-k_T algorithm [56] with $R = 0.8$, and satisfy $p_T \in [550, 650]$ GeV and $|y| \leq 2$. We do not perform any additional preprocessing here; in particular, we do not rotate the jets in the rapidity-azimuth plane for these studies.

In Figs. 12–14, we show the AUC as a function of latent dimension and effective latent dimension for quark/gluon tagging with moment PFNs, top tagging with moment EFNs, and top tagging with moment PFNs, respectively. These figures are meant to complement Figs. 1(a) and 1(b) for quark/gluon tagging with moment EFNs. We also summarize the results of these classifiers in Tables V–VII for ease of comparison with other studies using the same datasets.

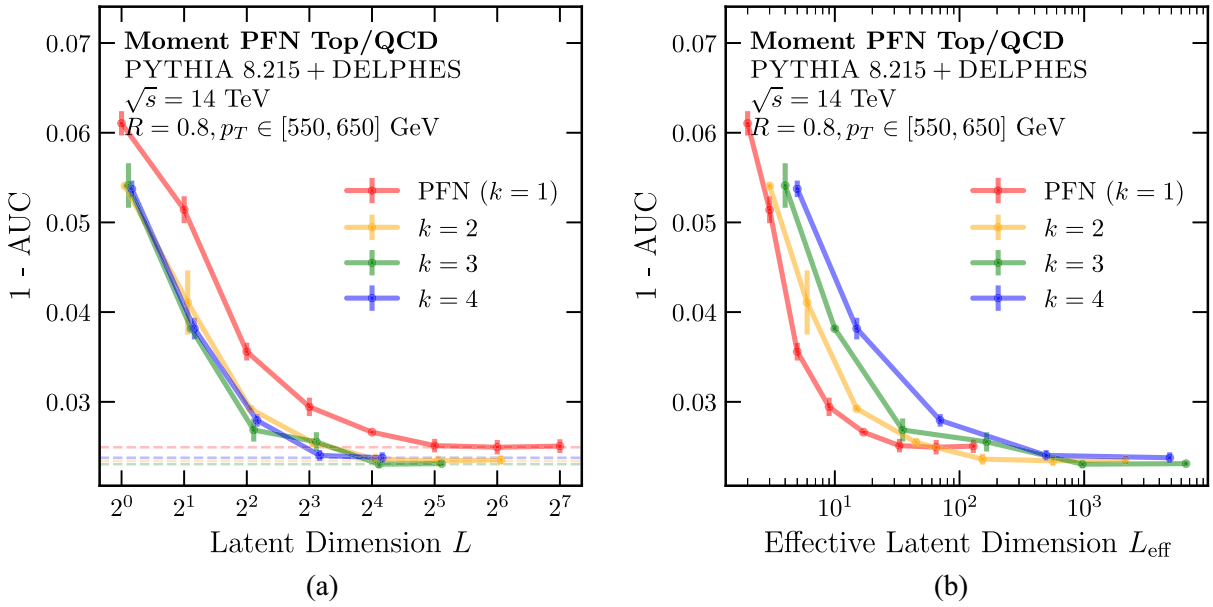

FIG. 14. (a) The same as Figs. 1(a) and (b) the same as 1(b), but with a moment PFN rather than a moment EFN and the top/QCD dataset rather than the quark/gluon dataset.

TABLE V. The same as Table I, but with a moment PFN rather than a moment EFN.

Model	AUC	$1/\epsilon_g$ at $\epsilon_q = 0.3$	$1/\epsilon_g$ at $\epsilon_q = 0.5$	Trainable parameters
$k = 1$, $L = 1$ PFN	0.862 ± 0.000	50.8 ± 1.0	19.4 ± 0.2	31206
$k = 2$, $L = 1$ moment PFN	0.864 ± 0.001	55.7 ± 0.9	20.3 ± 0.1	31306
$k = 3$, $L = 1$ moment PFN	0.866 ± 0.001	60.9 ± 2.3	21.3 ± 0.5	31406
$k = 4$, $L = 1$ moment PFN	$\mathbf{0.867 \pm 0.002}$	$\mathbf{62.8 \pm 1.2}$	$\mathbf{22.0 \pm 0.2}$	31506
$k = 1$, $L = 128$ PFN	0.889 ± 0.002	76.9 ± 0.6	31.6 ± 0.5	89753
$k = 2$, $L = 64$ moment PFN	0.890 ± 0.001	$\mathbf{83.0 \pm 1.3}$	32.2 ± 1.4	260185
$k = 3$, $L = 32$ moment PFN	0.890 ± 0.000	76.4 ± 2.7	33.2 ± 0.4	690745
$k = 4$, $L = 16$ moment PFN	$\mathbf{0.893 \pm 0.001}$	78.4 ± 4.0	$\mathbf{34.1 \pm 0.8}$	517561

TABLE VI. The same as Table I, but with the top/QCD dataset rather than the quark/gluon dataset.

Model	AUC	$1/\epsilon_{\mathrm{QCD}}$ at $\epsilon_{\mathrm{top}} = 0.3$	$1/\epsilon_{\mathrm{QCD}}$ at $\epsilon_{\mathrm{top}} = 0.5$	Trainable parameters
$k = 1$, $L = 1$ EFN	0.922 ± 0.002	29.5 ± 0.7	16.8 ± 0.3	31106
$k = 2$, $L = 1$ moment EFN	0.930 ± 0.002	37.6 ± 0.6	20.2 ± 0.4	31206
$k = 3$, $L = 1$ moment EFN	0.933 ± 0.001	41.2 ± 2.3	20.6 ± 0.6	31306
$k = 4$, $L = 1$ moment EFN	$\mathbf{0.946 \pm 0.001}$	$\mathbf{70.3 \pm 5.8}$	$\mathbf{29.1 \pm 1.1}$	31406
$k = 1$, $L = 128$ EFN	0.970 ± 0.000	465.0 ± 40.0	108.8 ± 2.4	89653
$k = 2$, $L = 64$ moment EFN	0.972 ± 0.001	407.9 ± 16.7	131.0 ± 3.5	260085
$k = 3$, $L = 32$ moment EFN	$\mathbf{0.974 \pm 0.000}$	416.1 ± 40.0	$\mathbf{138.4 \pm 11.7}$	690645
$k = 4$, $L = 16$ moment EFN	0.972 ± 0.001	$\mathbf{579.3 \pm 118.4}$	132.2 ± 8.2	517461

TABLE VII. The same as Table I, but with a moment PFN rather than a moment EFN and the top/QCD dataset rather than the quark/gluon dataset.

Model	AUC	$1/\epsilon_{\mathrm{QCD}}$ at $\epsilon_{\mathrm{top}} = 0.3$	$1/\epsilon_{\mathrm{QCD}}$ at $\epsilon_{\mathrm{top}} = 0.5$	Trainable parameters
$k = 1$, $L = 1$ PFN	0.937 ± 0.001	43.3 ± 0.5	25.1 ± 0.1	31206
$k = 2$, $L = 1$ moment PFN	0.944 ± 0.000	59.8 ± 0.7	29.0 ± 0.2	31306
$k = 3$, $L = 1$ moment PFN	0.944 ± 0.001	$\mathbf{62.8 \pm 1.3}$	29.1 ± 0.9	31406
$k = 4$, $L = 1$ moment PFN	$\mathbf{0.944 \pm 0.000}$	62.2 ± 0.4	$\mathbf{30.4 \pm 0.9}$	31506
$k = 1$, $L = 128$ PFN	0.975 ± 0.001	351.8 ± 12.5	138.8 ± 7.0	89753
$k = 2$, $L = 64$ moment PFN	0.976 ± 0.001	303.2 ± 26.9	$\mathbf{156.4 \pm 11.1}$	260185
$k = 3$, $L = 32$ moment PFN	0.975 ± 0.000	321.2 ± 0.0	136.1 ± 6.2	690745
$k = 4$, $L = 16$ moment PFN	$\mathbf{0.976 \pm 0.001}$	$\mathbf{351.8 \pm 12.5}$	146.0 ± 17.0	517561

We first observe that $k \geq 2$ architectures perform the same or better as their $k = 1$ counterparts for a given value of L, though this improvement is not always monotonic. In particular, in Fig. 13, we can see that the $k = 4$, $L = 2$ moment EFNS are indeed slightly worse than the $k = 3$, $L = 2$ moment EFNs, with a high variance. This is likely driven by a failed network convergence, reflective of the fact that $k = 4$ models are difficult to train. Second, unlike the quark/gluon moment EFNs, performance per effective latent dimension is *not* independent of k. For the quark/gluon moment PFNs, the performance per effective latent dimension tends to *increase* with k, but for the top/QCD moment EFNs and moment PFNs, the performance per effective latent dimension tends to *decrease* with k. This would seem to imply that the top/QCD discriminant is not "compressible," in that its latent space cannot be easily factorized into products of just a few functions, and that many independent functions are genuinely required.

[1] Supriyo Chakraborty, Richard Tomsett, Ramya Raghavendra, Daniel Harborne, Moustafa Alzantot, Federico Cerutti, Mani Srivastava, Alun Preece, Simon Julier, Raghuveer M. Rao, Troy D. Kelley, Dave Braines, Murat Sensoy, Christopher J. Willis, and Prudhvi Gurram, Interpretability of deep learning models: A survey of results, in *2017 IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computed, Scalable Computing & Communications, Cloud & Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI)* (IEEE, San Francisco, 2017), pp. 1–6.
[2] Leilani H. Gilpin, David Bau, Ben Z. Yuan, Ayesha Bajwa, Michael Specter, and Lalana Kagal, Explaining explanations: An overview of interpretability of machine learning, in *2018 IEEE 5th International Conference on data science and advanced analytics (DSAA)* (2018).
[3] Yu Zhang, Peter Tino, Ales Leonardis, and Ke Tang, A survey on neural network interpretability, IEEE Trans. Emerging Topics Comput. Intell. **5**, 726 (2021).
[4] Christoph Molnar, Giuseppe Casalicchio, and Bernd Bischl, Interpretable machine learning—a brief history, state-of-the-art and challenges, in *Communications in Computer and Information Science* (Springer International Publishing, New York, 2020), pp. 417–431.
[5] Cynthia Rudin, Chaofan Chen, Zhi Chen, Haiyang Huang, Lesia Semenova, and Chudi Zhong, Interpretable machine learning: Fundamental principles and 10 grand challenges, arXiv:2103.11251.
[6] Barry M. Dillon, Gregor Kasieczka, Hans Olischlager, Tilman Plehn, Peter Sorrenson, and Lorenz Vogel, Symmetries, safety, and self-supervision, SciPost Phys. **12**, 188 (2022).
[7] Alexander Bogatskiy *et al.*, Symmetry group equivariant architectures for physics, in *2022 Snowmass Summer Study* (2022), arXiv:2203.06153.
[8] Lei Gao and Ling Guan, Interpretability of machine learning: Recent advances and future prospects, arXiv:2305.00537.
[9] Alexander Bogatskiy, Timothy Hoffman, David W. Miller, Jan T. Offermann, and Xiaoyang Liu, Explainable equivariant neural networks for particle physics: PELICAN, J. High Energy Phys. 03 (2024) 113.
[10] Jeremy Wayland, Corinna Coupette, and Bastian Rieck, Mapping the multiverse of latent representations, arXiv:2402.01514.
[11] Bruce H. Denby, Neural networks and cellular automata in experimental high-energy physics, Comput. Phys. Commun. **49**, 429 (1988).
[12] Dan Guest, Kyle Cranmer, and Daniel Whiteson, Deep learning and its application to LHC physics, Annu. Rev. Nucl. Part. Sci. **68**, 161 (2018).
[13] Anja Butter, Gregor Kasieczka, Tilman Plehn, and Michael Russell, Deep-learned top tagging with a Lorentz layer, SciPost Phys. **5**, 028 (2018).
[14] Kim Albertsson *et al.*, Machine learning in high energy physics community white paper, J. Phys. Conf. Ser. **1085**, 022008 (2018).
[15] Huilin Qu and Loukas Gouskos, Jet tagging via particle clouds, Phys. Rev. D **101**, 056019 (2020).
[16] Dimitri Bourilkov, Machine and deep learning applications in particle physics, Int. J. Mod. Phys. A **34**, 1930019 (2020).
[17] Shiqi Gong, Qi Meng, Jue Zhang, Huilin Qu, Congqiao Li, Sitian Qian, Weitao Du, Zhi-Ming Ma, and Tie-Yan Liu, An efficient Lorentz equivariant graph neural network for jet tagging, J. High Energy Phys. 07 (2022) 030.
[18] Jonathan Shlomi, Peter Battaglia, and Jean-Roch Vlimant, Graph neural networks in particle physics, Mach. Learn. **2**, 021001 (2021).
[19] Amit Chakraborty, Sung Hak Lim, Mihoko M. Nojiri, and Michihisa Takeuchi, Neural network-based top tagger with two-point energy correlations and geometry of soft emissions, J. High Energy Phys. 07 (2020) 111.
[20] Anja Butter and Tilman Plehn, Generative networks for LHC events, arXiv:2008.08558.
[21] Michael Kagan, Image-based jet analysis, arXiv:2012.09719.
[22] Sung Hak Lim and Mihoko M. Nojiri, Morphology for jet classification, Phys. Rev. D **105**, 014004 (2022).
[23] Frédéric A. Dreyer and Huilin Qu, Jet tagging in the Lund plane with graph networks, J. High Energy Phys. 03 (2021) 052.
[24] Georgia Karagiorgi, Gregor Kasieczka, Scott Kravitz, Benjamin Nachman, and David Shih, Machine learning in the search for new fundamental physics, arXiv:2112.03769.
[25] Matthew D. Schwartz, Modern machine learning and particle physics, arXiv:2103.12226.
[26] Huilin Qu, Congqiao Li, and Sitian Qian, Particle transformer for jet tagging, *Proceedings of the 39th International Conference on Machine Learning* (2022), arXiv:2202.03772.
[27] Pierre Baldi, Peter Sadowski, and Daniel Whiteson, Deep learning from four vectors, arXiv:2203.03067.
[28] Tilman Plehn, Anja Butter, Barry Dillon, and Claudius Krause, Modern machine learning for LHC physicists, arXiv:2211.01421.
[29] Giuseppe Carleo, Ignacio Cirac, Kyle Cranmer, Laurent Daudet, Maria Schuld, Naftali Tishby, Leslie Vogt-Maranto, and Lenka Zdeborová, Machine learning and the physical sciences, Rev. Mod. Phys. **91**, 045002 (2019).
[30] Simon Badger *et al.*, Machine learning and LHC event generation, SciPost Phys. **14**, 079 (2023).
[31] Alexander Bogatskiy, Timothy Hoffman, David W. Miller, and Jan T. Offermann, PELICAN: Permutation equivariant and Lorentz invariant or covariant aggregator network for particle physics, arXiv:2211.00454.
[32] Oliver Atkinson, Akanksha Bhardwaj, Christoph Englert, Partha Konar, Vishal S. Ngairangbam, and Michael Spannowsky, IRC-safe graph autoencoder for unsupervised anomaly detection, Front. Artif. Intell. **5**, 943135 (2022).
[33] Akanksha Bhardwaj, Christoph Englert, Wrishik Naskar, Vishal S. Ngairangbam, and Michael Spannowsky, Equivariant, safe and sensitive—graph networks for new physics, J. High Energy Phys. 07 (2024) 245.
[34] Patrick T. Komiske, Eric M. Metodiev, and Jesse Thaler, Energy flow networks: Deep sets for particle jets, J. High Energy Phys. 01 (2019) 121.
[35] Anja Butter *et al.*, The machine learning landscape of top taggers, SciPost Phys. **7**, 014 (2019).
[36] Constituent-based top-quark tagging with the ATLAS detector (2022), https://cds.cern.ch/record/2825328.

[37] Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Ruslan Salakhutdinov, and Alexander Smola, Deep sets, in *Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS'17)* (Curran Associates Inc., Red Hook, 2017).
[38] Miles Cranmer, Christina Kreisch, Alice Pisani, Francisco Villaescusa-Navarro, David N. Spergel, and Shirley Ho, Histogram pooling operators: An alternative for deep sets, in *ICLR 2021 SimDL Workshop* (2021).
[39] Wei Shen, Daohan Wang, and Jin Min Yang, Hierarchical high-point energy flow network for jet tagging, J. High Energy Phys. 09 (2023) 135.
[40] Samuel Bright-Thonney, Benjamin Nachman, and Jesse Thaler, Safe but Incalculable: Energy-weighting is not all you need, Phys. Rev. D **110**, 014029 (2024).
[41] Carola F. Berger, Tibor Kucs, and George F. Sterman, Event shape/ energy flow correlations, Phys. Rev. D **68**, 014012 (2003).
[42] Carola F. Berger and Lorenzo Magnea, Scaling of power corrections for angularities from dressed gluon exponentiation, Phys. Rev. D **70**, 094010 (2004).
[43] Fyodor V. Tkachov, Measuring multi—jet structure of hadronic energy flow or what is a jet?, Int. J. Mod. Phys. A **12**, 5411 (1997).
[44] N. A. Sveshnikov and F. V. Tkachov, Jets and quantum field theory, Phys. Lett. B **382**, 403 (1996).
[45] Fyodor V. Tkachov, A theory of jet definition, Int. J. Mod. Phys. A **17**, 2783 (2002).
[46] Diego M. Hofman and Juan Maldacena, Conformal collider physics: Energy and charge correlations, J. High Energy Phys. 05 (2008) 012.
[47] Demba Ba, Akshunna S. Dogra, Rikab Gambhir, Abiy Tasissa, and Jesse Thaler, SHAPER: Can you hear the shape of a jet?, J. High Energy Phys. 06 (2023) 195.
[48] Patrick T. Komiske, Eric M. Metodiev, and Jesse Thaler, Energy flow polynomials: A complete linear basis for jet substructure, J. High Energy Phys. 04 (2018) 013.
[49] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio, Neural machine translation by jointly learning to align and translate, arXiv:1409.0473.
[50] Jianpeng Cheng, Li Dong, and Mirella Lapata, Long short-term memory-networks for machine reading, arXiv:1601.06733.
[51] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin, Attention is all you need, arXiv:1706.03762.
[52] Jason Gallicchio and Matthew D. Schwartz, Quark and gluon tagging at the LHC, Phys. Rev. Lett. **107**, 172001 (2011).
[53] Philippe Gras, Stefan Höche, Deepak Kar, Andrew Larkoski, Leif Lönnblad, Simon Plätzer, Andrzej Siódmok, Peter Skands, Gregory Soyez, and Jesse Thaler, Systematics of quark/gluon tagging, J. High Energy Phys. 07 (2017) 091.
[54] Torbjorn Sjostrand, Stephen Mrenna, and Peter Z. Skands, Pythia 6.4 physics and manual, J. High Energy Phys. 05 (2006) 026.
[55] Torbjörn Sjöstrand, Stefan Ask, Jesper R. Christiansen, Richard Corke, Nishita Desai, Philip Ilten, Stephen Mrenna, Stefan Prestel, Christine O. Rasmussen, and Peter Z. Skands, An introduction to Pythia 8.2, Comput. Phys. Commun. **191**, 159 (2015).
[56] Matteo Cacciari, Gavin P. Salam, and Gregory Soyez, The anti-k_t jet clustering algorithm, J. High Energy Phys. 04 (2008) 063.
[57] Matteo Cacciari, Gavin P. Salam, and Gregory Soyez, FastJet user manual, Eur. Phys. J. C **72**, 1896 (2012).
[58] Eric M. Metodiev and Jesse Thaler, Jet topics: Disentangling quarks and gluons at colliders, Phys. Rev. Lett. **120**, 241602 (2018).
[59] Patrick T. Komiske, Eric M. Metodiev, and Jesse Thaler, An operational definition of quark and gluon jets, J. High Energy Phys. 11 (2018) 059.
[60] Patrick T. Komiske, Eric M. Metodiev, and Matthew D. Schwartz, Deep learning in color: Towards automated quark/gluon jet discrimination, J. High Energy Phys. 01 (2017) 110.
[61] ATLAS collaboration, Quark versus gluon jet tagging using jet images with the ATLAS detector, Technical Report No. ATL-PHYS-PUB-2017-017, CERN, Geneva, 2017.
[62] Taoli Cheng, Recursive neural networks in quark/gluon tagging, Comput. Software Big Sci. **2**, 3 (2018).
[63] Hui Luo, Ming-Xing Luo, Kai Wang, Tao Xu, and Guohuai Zhu, Quark jet versus gluon jet: Fully-connected neural networks with high-level features, Sci. China Phys. Mech. Astron. **62**, 991011 (2019).
[64] Gregor Kasieczka, Nicholas Kiefer, Tilman Plehn, and Jennifer M. Thompson, Quark-gluon tagging: Machine learning vs detector, SciPost Phys. **6**, 069 (2019).
[65] Patrick T. Komiske, Eric M. Metodiev, and Jesse Thaler, Energy flow networks: Deep sets for particle jets, J. High Energy Phys. 01 (2019) 121.
[66] Jason Sang Hun Lee, Inkyu Park, Ian James Watson, and Seungjin Yang, Quark-gluon jet discrimination using convolutional neural networks, J. Korean Phys. Soc. **74**, 219 (2019).
[67] Jason Sang Hun Lee, Sang Man Lee, Yunjae Lee, Inkyu Park, Ian James Watson, and Seungjin Yang, Quark gluon jet discrimination with weakly supervised learning, J. Korean Phys. Soc. **75**, 652 (2019).
[68] Eric A. Moreno, Olmo Cerri, Javier M. Duarte, Harvey B. Newman, Thong Q. Nguyen, Avikar Periwal, Maurizio Pierini, Aidana Serikova, Maria Spiropulu, and Jean-Roch Vlimant, JEDI-net: A jet identification algorithm based on interaction networks, Eur. Phys. J. C **80**, 58 (2020).
[69] Huilin Qu and Loukas Gouskos, ParticleNet: Jet tagging via particle clouds, Phys. Rev. D **101**, 056019 (2020).
[70] Vinicius Mikuni and Florencia Canelli, ABCNet: An attention-based method for particle tagging, Eur. Phys. J. Plus **135**, 463 (2020).
[71] Minxuan He and Daohan Wang, Quark/gluon discrimination and top tagging with dual attention transformer, Eur. Phys. J. C **83**, 1116 (2023).
[72] Matthew J. Dolan, John Gargalionis, and Ayodele Ore, Quark-versus-gluon tagging in CMS open data with CWoLa and TopicFlow, arXiv:2312.03434.
[73] Kaustuv Datta and Andrew J. Larkoski, Novel jet observables from machine learning, J. High Energy Phys. 03 (2018) 086.

[74] Stephen D. Ellis, Zoltan Kunszt, and Davison E. Soper, Jets at hadron colliders at order $\alpha - s^{3:}$ A look inside, Phys. Rev. Lett. **69**, 3615 (1992).
[75] F. Abe *et al.* (CDF Collaboration), A measurement of jet shapes in $p\bar{p}$ collisions at $\sqrt{s} = 1.8$ TeV, Phys. Rev. Lett. **70**, 713 (1993).
[76] Andrew J. Larkoski, Gavin P. Salam, and Jesse Thaler, Energy correlation functions for jet substructure, J. High Energy Phys. 06 (2013) 108.
[77] David L. Wallace, Asymptotic approximations to distributions, Ann. Math. Stat. **29**, 635 (1958).
[78] Andrew L. Maas, Rectifier nonlinearities improve neural network acoustic models, in *Proc. icml* (2013).
[79] A. Osathapan and R. Gambhir, Moment Pooling, https://github.com/athiso/moment.
[80] A. Osathapan and R. Gambhir, Moment Analysis, https://github.com/rikab/MomentAnalysis.
[81] Thomas Kluyver, Benjamin Ragan-Kelley, Fernando Pérez, Brian Granger, Matthias Bussonnier, Jonathan Frederic, Kyle Kelley, Jessica Hamrick, Jason Grout, Sylvain Corlay, Paul Ivanov, Damián Avila, Safia Abdalla, and Carol Willing, Jupyter notebooks—a publishing format for reproducible computational workflows, in *Positioning and Power in Academic Publishing: Players, Agents and Agendas*, edited by F. Loizides and B. Schmidt (IOS Press, Amsterdam, 2016), pp. 87–90.
[82] http://iaifi.org/
[83] Fraciois Chollet, Keras, GitHub repository (2017), https://github.com/fchollet/keras.
[84] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard *et al.*, TensorFlow: A system for large-scale machine learning, in *OSDI* (USENIX Association, Savannah, Georgia, 2016), Vol. 16, pp. 265–283.
[85] Lu Lu, Dying ReLU and initialization: Theory and numerical examples, Commun. Comput. Phys. **28**, 1671 (2020).
[86] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, Delving deep into rectifiers: Surpassing human-level performance on imagenet classification, *2015 IEEE International Conference on Computer Vision (ICCV)* (2015), arXiv:1502.01852.
[87] Diederik P. Kingma and Jimmy Ba, Adam: A method for stochastic optimization, arXiv:1412.6980.
[88] J. de Favereau, C. Delaere, P. Demin, A. Giammanco, V. Lemaître, A. Mertens, and M. Selvaggi (Delphes 3 Collaboration), Delphes 3, A modular framework for fast simulation of a generic collider experiment, J. High Energy Phys. 02 (2014) 057.