Probing the fundamental limit of jet tagging with surrogate reference models

Ian Pang,[1,*] Darius A. Faroughy,[1,†] David Shih,[1,‡] Ranit Das,[1,2,§] and Gregor Kasieczka[3,‖]
[1]*NHETC, Department of Physics and Astronomy, Rutgers University, Piscataway, New Jersey 08854, USA*
[2]*Institute for Theoretical Physics, Universität Heidelberg, Germany*
[3]*Institut für Experimentalphysik, Universität Hamburg, 22761 Hamburg, Germany*

(Received 12 December 2025; accepted 10 August 2026; published 16 November 2026)

Beyond the practical goal of improving search and measurement sensitivity through better jet tagging algorithms, there is a deeper question: what are their upper performance limits? Generative surrogate models with learned likelihood functions offer a new approach to this problem, provided the surrogate correctly captures the underlying data distribution. In this work, we introduce the SUrrogate ReFerence (SURF) method, a new approach to validating generative models. This framework enables exact Neyman-Pearson tests by training the target model on samples from another tractable surrogate, which is itself trained on real data. We argue that the EPiC-FM generative model is a valid surrogate reference for JETCLASS jets and apply SURF to show that modern jet taggers may already be operating close to the true statistical limit. By contrast, we find that autoregressive generative pretrained transformer models unphysically exaggerate top vs QCD separation power encoded in the surrogate reference, implying that they are giving a misleading picture of the fundamental limit.

DOI: 10.1103/rcpy-nr72

I. INTRODUCTION

Jet tagging is a central task in collider physics. Over the past decade, machine learning has driven major advances in jet tagging, with increasingly sophisticated architectures achieving very high classification performance on simulated datasets [1–12]. This success naturally raises a key question: have current jet taggers already reached the fundamental limit of jet tagging, or does a gap remain between practical performance and the true statistical optimum?

The Neyman-Pearson (NP) limit, defined by the likelihood ratio, is the best possible discriminant between two different underlying physics processes—such as top and QCD jets—that any classifier could achieve if it had access to the exact data likelihoods [13]. In practice, however, this limit cannot be evaluated directly because the true likelihood of the data-generating process is unknown. It therefore remains unclear how close existing classifiers are to this ultimate bound.

Recently, Ref. [14] proposed using autoregressive generative pretrained transformer (GPT)-style generative models to probe this limit for top vs QCD jets from the JETCLASS dataset [15]. These models operate on discretized, tokenized representations of jet constituents and yield explicit log-likelihoods, enabling the computation of likelihood ratios between jet classes. When applied to top-QCD discrimination, GPT-based likelihoods were found to produce receiver-operator-characteristic (ROC) curves that appear to substantially exceed those of all existing classifiers trained on GPT samples. This was interpreted as evidence that current jet taggers fall well short of the true Neyman-Pearson (NP) limit.

In this work, we reevaluate this conclusion in several ways. First, we upgrade the GPT model of Refs. [14,16] with a number of enhancements, including voxel tokenization, positional encoding and weight tying. Despite this, we find that the improved GPT model trained on tops and QCD from JETCLASS still inflates their separation relative to state-of-the-art classifiers by more than an order of magnitude in rejection factor over a wide range of signal efficiencies. In fact, this is an even larger gap than the original one of Refs. [14,16]. Along the way, we show that GPT-generated jets are perfectly separable from continuous jets, while being close to the discretized jets that the GPT models were trained on. However, we show this cannot explain the inflated separation of the GPT jets—the discretized top and QCD jets are if anything *less* separable than their continuous counterparts.

*Contact author: ian.pang@physics.rutgers.edu
†Contact author: darius.faroughy@rutgers.edu
‡Contact author: shih@physics.rutgers.edu
§Contact author: das@thphys.uni-heidelberg.de
‖Contact author: gregor.kasieczka@uni-hamburg.de

Published by the American Physical Society under the terms of the Creative Commons Attribution 4.0 International license. Further distribution of this work must maintain attribution to the author(s) and the published article's title, journal citation, and DOI. Funded by SCOAP³.

Second, we assess the fundamental limit of jet tagging using an alternative generative modeling framework that also gives tractable likelihoods: conditional flow matching (CFM) [17–19] with permutation equivariance, in the form of the EPiC-FM architecture [20,21]. Unlike the GPT framework, EPiC-FM operates directly at the level of the continuous jet constituents and does not require any discretization or tokenization. Using the log-likelihood ratios derived from EPiC-FM trained on JETCLASS QCD and top jets, we find no evidence for an inflated fundamental limit: the ROC curve obtained from the exact NP test is fairly close (within a factor of two) to the state-of-the-art classifier performance on EPiC-FM samples.

These results raise the question—which jet generative model is giving the more accurate picture of the fundamental limit of jet tagging? Is there an enormous gap between the fundamental limit and our best classifiers, as the GPT models would suggest? Or is there essentially no gap between the two, as the EPiC-FM models would suggest?

To clarify the origin of this discrepancy and to provide a general framework for evaluating generative models, we introduce the SUrrogate ReFerence (SURF) method. Here the key idea is to use one generative model with tractable likelihoods (EPiC-FM) to evaluate the other generative model with tractable likelihoods (GPT). The first one serves as the *surrogate reference model* for the second. By various ways we validate EPiC-FM as a surrogate reference model, checking that it does not introduce spurious artifacts into the JETCLASS data that are hard to detect.[1] We replace JETCLASS samples with samples drawn from EPiC-FM, train the GPT model on these samples, and repeat all the studies and tests we want to do with the GPT model. But now with tractable likelihoods for both the surrogate reference and the target generative model, ground truth NP-optimal ROC curves can be calculated and compared against one another, giving us more information than when we had trained the GPT model on JETCLASS directly.

Within this framework, we find that the NP ROC curve derived from the GPT likelihoods far exceeds that of the EPiC-FM surrogate itself. This demonstrates unambiguously that the GPT model artificially inflates the apparent separation between top and QCD jets beyond the physical limit represented by the surrogate reference. Evidently, the GPT model is introducing unphysical artifacts into the EPiC-FM surrogate samples that it is trained on, leading them to be more separable than they should be. We investigate the reasons for this and find evidence that GPT models exhibit signs of overfitting to their training data. We speculate how this form of mismodeling might produce likelihood ratios that are inherently difficult for classifiers to reproduce, leading to artificially inflated ROC curves.

In the end, we find no evidence of an inflated "fundamental limit" of jet tagging. Instead, the overall picture, especially the concurrence of top vs QCD ROC curves from OmniLearn trained on JETCLASS and on EPiC-FM together with the exact likelihoods of EPiC-FM, suggests that the fundamental limit is not far away from state-of-the-art classifiers.

The outline of our paper is as follows. In Secs. II and III, we present the model descriptions, baseline evaluations, and top vs QCD separation results obtained using the GPT model and the EPiC-FM model, respectively. Section IV introduces the SURF method and its corresponding results. Possible reasons for the inflated GPT top vs QCD curve are examined in Sec. V. We conclude with an outlook in Sec. VI, while additional technical background and log-likelihood details are provided in the Appendixes.

II. GPT: JETS AS TOKENIZED SEQUENCES

A. Description of the model

One popular generative modeling framework for jets is to view them as p_T-ordered sequences of tokenized particles. We follow the approach of Ref. [14], where the kinematic features of each constituent are discretized into bins and mapped to tokens. The resulting sequences can then be modeled autoregressively with GPT-style autoregressive language models, which learn the joint distribution over jet constituents by factorizing it into a product of conditional probabilities for each tokenized particle given the preceding ones. We note that VQ-VAE-based tokenization is also possible [22,23]; however, obtaining a well-defined likelihood in physical space is nontrivial because the learned codebook partitions feature space into irregular regions without a known volume element. Establishing a principled mapping from sequence log-probabilities to physical-space log-densities in that setting is left to future work, so we adopt a fixed-bin vocabulary here.

Concretely, within each jet $\mathbf{x} = \{\mathbf{x}_i\}_{i=1}^{N}$ we order the constituents $\mathbf{x}_i = (\log p_T, \Delta\eta, \Delta\phi)_i$ by descending p_T and discretize the kinematic features into $N_{p_T} = 40$, $N_{\Delta\eta} = 30$, and $N_{\Delta\phi} = 30$ evenly spaced bins. Here $p_{T,i}$ is the transverse momenta of particle i, $\Delta\eta_i \equiv \eta_i - \eta_J$ is the pseudorapidity offset from the jet axis, and $\Delta\phi_i \equiv \phi_i - \phi_J$ is the azimuthal-angle offset. Each three-dimensional voxel in feature space is mapped to a unique token, giving rise to a vocabulary of size $V = N_{p_T} \cdot N_{\Delta\eta} \cdot N_{\Delta\phi} = 36,000$. We restrict each jet to the 40 highest-p_T particles before

[1] As we explain in detail in Sec. II, the same validation tests fail for the GPT model, indicating that it does not satisfy the requirements of a reliable surrogate reference. For this reason, we do not recommend running the SURF method in the opposite order.

discretization.[2] This choice is motivated by Ref. [14], which observed a sizable gap between the NP ROC curve and that of a trained classifier even under this 40-particle restriction. Increasing the number of particles only slightly enlarged the gap without altering the conclusion, and thus does not justify the additional computational cost. The pseudorapidity and azimuthal bins are restricted to the range $(-0.8, 0.8)$ with overflows. Each jet $\mathbf{x}$ is thus represented by a sequence $\mathbf{t_x} = (\mathtt{t}_1, \ldots, \mathtt{t}_N)$, where each token $\mathtt{t}_i$ represents an individual particle. Sequences are padded to a maximum length of $N_{\max} = 40$ using a dedicated `[PAD]` token. Additional special tokens, `[START]` and `[END]`, are appended to the vocabulary to indicate the beginning and end of each jet sequence.

We implement our GPT model with the HuggingFace Transformers library [24]. Model and training details can be found in the Appendix B. When compared to Ref. [14], our setup differs in three key ways. (i) *Voxel tokenization:* we use a single, joint vocabulary over $(\log p_T, \Delta\eta, \Delta\phi)$ voxels and embed each token directly into $\mathbb{R}^{d_{\text{embd}}}$, whereas Ref. [14] tokenizes each binned feature separately and sums three feature-specific embeddings to form the per-particle representation. (ii) *Positional information:* we add learned positional embeddings (reflecting the p_T ordering) to the token embeddings before the self-attention layers; Ref. [14] omits positional encodings. (iii) *Weight tying:* we tie the last linear layer (language head) to the input token embedding matrix, using the same weights for input embedding and output logits. Weight tying reduces the number of trainable parameters and has been shown to improve generalization in language models [25,26]. Empirically, we find these choices improve the generative performance of our GPT model.

Drawing a sample from a trained GPT produces tokens $\mathbf{t_x} = (\mathtt{t}_0 = [\mathtt{START}], \mathtt{t}_1, \ldots, \mathtt{t}_N, \mathtt{t}_{N+1} = [\mathtt{END}])$ which correspond to binned, p_T-ordered jet constituents. The log-likelihood of a tokenized jet is computed by summing the next-token probabilities

$$\log p_\theta(\mathbf{t_x}) = \sum_{n=1}^{N+1} \log p_\theta(\mathtt{t}_n | \mathtt{t}_0, \ldots, \mathtt{t}_{n-1}), \tag{1}$$

along the sequence and stopping at the `[END]` symbol.

The tokenized jets can be mapped back to the physical space of continuous jet constituents by identifying each constituent token with its corresponding voxel in $(\log p_T, \Delta\eta, \Delta\phi)$ space and then smearing it within its voxel according to a uniform density. The corresponding log-likelihood in the physical space is then given by

$$\log p_\theta^{\text{cont}}(\mathbf{x}) = \log p_\theta(\mathbf{t_x}) - N \log dV + \sum_k \log N_k! - \log N! \tag{2}$$

Here the second term is the correction to the log-likelihood from the uniform smearing (dV is the volume of the voxel), and the third and fourth terms are necessary to remove the spurious effects of p_T-ordering in the sequence (N_k is the number of constituents with identical discretized p_T).

Clearly, with the GPT models there are two choices for what we consider as the "original data"—we can use either the original continuous JETCLASS samples or their bin-smeared counterparts as the baseline. Ostensibly only the former case is of interest—we are ultimately interested in generative models for continuous jets. However, the latter is also a useful comparison point, and we will consider both in the following.

TABLE I. AUC scores for distinguishing between jets from JETCLASS, bin-smeared JETCLASS, and GPT trained on JETCLASS, using the OmniLearn classifier. The NP-optimal classifier scores cannot be computed because JETCLASS likelihoods are unavailable.

Comparison	Method	QCD	Top
JETCLASS vs GPT	OmniLearn	0.9949	0.9734
	NP-optimal	···	···
JETCLASS-bs vs GPT	OmniLearn	0.5564	0.5965
	NP-optimal	···	···
JETCLASS vs JETCLASS-bs	OmniLearn	0.9940	0.9714
	NP-optimal	···	···

B. Baseline evaluation of the model

We first perform a baseline evaluation of GPT models trained directly on JETCLASS data. For brevity, we evaluate performance solely using the area under the curve (AUC) of a classifier trained to distinguish generated samples from real data [27,28], although other evaluation strategies are possible [29–32].

We train an OmniLearn classifier[3] to separate JETCLASS data from GPT-generated jets. We find that OmniLearn separates both datasets almost perfectly (AUC = 0.9949 for QCD and 0.9734 for tops; Table I, first row). The near-perfect separability indicates that GPT samples lie far from the continuous data manifold.

Is this due to an issue with the GPT models, or is it due to the binning of the JETCLASS data itself? We answer this

[2]We have also performed a minimal study using up to 128 particles per jet and observe consistent behavior, with no qualitative change in our conclusions. These results are included in the Appendix C.

[3]For all our results, we use OmniLearn from Refs. [8,9] to represent current state-of-the-art classifiers. We pretrain it on the entire JETCLASS training dataset and fine-tune it on the jets used in our study. In particular, we use 1M jets from each class, with an 80/10/10 split into training, validation, and test sets.

question with two more classifier tests: GPT samples vs bin-smeared JETCLASS, and continuous JETCLASS vs bin-smeared JETCLASS. The first is shown in Table I, second row, indicating OmniLearn AUCs of 0.5564 and 0.5965 for QCD and tops respectively. This reproduces the good performance achieved by the original Aachen GPT-style jet generator [16] and in fact improves upon it.[4] Meanwhile the latter is shown in the third row of Table I, indicating OmniLearn AUCs of 0.9940 and 0.9714 for QCD and tops respectively. We conclude that the separability of GPT jets from original continuous JETCLASS is driven entirely by the binning of the data—this alone takes the data off-manifold with respect to the original continuous samples. By contrast, GPT jets are quite close to the binned JETCLASS that they were trained on.

C. Top vs QCD separation with GPT

We next turn to an evaluation of how well the GPT model reproduces the separation of tops vs QCD. We compare three classifiers in Fig. 2 (each classifier is evaluated on the samples from the generative process it is defined/trained with):

(1) "GPT optimal": The GPT model's likelihood ratio discriminator. Note that here it does not matter whether one uses the autoregressive likelihood for tokens $\log p_\theta(\mathbf{t}_\mathbf{x})$ or the bin-smeared likelihood in the continuous physical space $\log p_\theta^{\text{cont}}(\mathbf{x})$—the extra correction factors in (2) cancel out.
(2) "JetClass OmniLearn": An OmniLearn classifier trained on JETCLASS jets.
(3) "GPT OmniLearn": An OmniLearn classifier trained on samples drawn from the GPT model.

We see that while the OmniLearn classifiers trained on original JETCLASS and on GPT outputs agree, the NP-optimal classifier derived from the GPT model gives a much higher ROC curve. Evidently, the GPT tops vs QCD jets are highly separable, and in a way that the OmniLearn classifier cannot detect. This reproduces the observation of Ref. [14], and in our setup the separability is even more pronounced.

However, unlike that work, we point out that there are two possible interpretations of this result: either

(a) The NP-optimal classifier between GPT tops and QCD is representative of the NP-optimal classifier between JETCLASS tops and QCD, in which case there is a major gap between state-of-the-art jet taggers and the true "fundamental limit" of jet tagging (the conclusion reached in [14]),

or

(b) the GPT model introduces unphysical artifacts into JETCLASS that make the GPT tops much more separable from GPT QCD than their JETCLASS counterparts, and in a way that the classifier is somehow unable to learn. Then the fact that the state-of-the-art classifiers cannot reproduce the NP-optimal result on GPT-generated jets is unfortunate, but possibly beside the point if this result is due to unphysical artifacts.

We next turn to another popular generative modeling framework for jets—permutation-equivariant conditional flow matching—and explore what it has to say about the separation of top vs QCD jets.

III. EPIC-FM: JETS AS PARTICLE CLOUDS

A. Description of the model

We represent jets as *unordered* particle clouds with kinematic features $\mathbf{x} = \{\mathbf{x}_i\}_{i=1}^N$, where each constituent carries $\mathbf{x}_i = (p_T, \Delta\eta, \Delta\phi)_i$. We restrict each jet to at most the 40 highest-p_T constituents (i.e. $N_{\max} = 40$), discarding softer particles beyond this cutoff. We use the conditional flow-matching (CFM) paradigm to train a continuous normalizing flow equipped with a time-dependent EPiC architecture [20] that approximates the velocity field $\mathbf{u}_t$ of the ODE trajectories connecting source to target samples. The setup for our architecture closely follows the one in [21], with the main difference being that the only conditional model inputs that we consider are the time variable $t \in [0, 1]$, the jet-type (QCD and top), and the number of constituents in each jet. For further details on the model and training see Appendix A 1.

Once trained, we generate jets of a specified class by integrating the ODE $\dot{\mathbf{x}}_t = \mathbf{u}_t^\theta(\mathbf{x}_t)$ from $t = 0$ to $t = 1$ with a forward Euler integrator with step size $\Delta t = 3.33 \times 10^{-3}$. We estimate the log-density of the generated jets by integrating the dynamics with backward Euler steps in the reverse time direction. Writing the total derivative as $d/dt = \partial_t + \mathbf{u}_t^\theta \cdot \nabla_\mathbf{x}$, the log-density evolves according to

$$\frac{d}{dt}\log p_t(\mathbf{x}) = -\mathrm{Tr}\left[\frac{\partial \mathbf{u}_t^\theta}{\partial \mathbf{x}}\right], \tag{3}$$

which integrates to

$$\log p_1(\mathbf{x}_1) = \log p_0(\mathbf{x}_0) - \int_0^1 dt \mathrm{Tr}\left[\frac{\partial \mathbf{u}_t^\theta}{\partial \mathbf{x}}\right]. \tag{4}$$

If the model is well trained, then this solution is expected to approximate the target log-density. In practice, we accumulate the Jacobian-trace term via automatic differentiation; numerical details and validation (fixed-point refinement, step-size scans) are provided in the Appendix A 2.

As described in Sec. II A, tokenized jets are reconstructed in physical space by assigning each token to its

[4] Retraining OmniLearn on bin-smeared Aachen GPT-generated jets from Ref. [33] vs bin-smeared JETCLASS jets yields AUC values of 0.63 for QCD jets and 0.75 for top jets, considering the 40 hardest constituents of each jet.

corresponding $(\log p_T, \Delta\eta, \Delta\phi)$ bin and uniformly smearing it within that bin. Although EPiC-FM generates continuous jets rather than tokenized jets, its likelihood can still be evaluated on tokenized jets by mapping them into the continuous space through this bin-smearing procedure.

To approximate the log-likelihood of tokenized EPiC-FM generated jets, we apply the same procedure to the continuous EPiC-FM samples. Specifically, we first bin the constituent features of the generated jets and map each constituent to the center of its corresponding three-dimensional bin, producing the binned jet $\hat{\mathbf{x}}_1$. We then uniformly smear each constituent within its bin by adding noise $\boldsymbol{\epsilon}_k$, sampled independently for each constituent.

The log-likelihood is approximated by averaging over M such smearing realizations,

$$\log p_{bs}(\mathbf{x}_1) = \log\left(\frac{1}{M}\sum_{k=1}^{M} p_1(\hat{\mathbf{x}}_1 + \boldsymbol{\epsilon}_k)\right), \tag{5}$$

where $\hat{\mathbf{x}}_1$ denotes the binned EPiC-FM generated jet with constituent features mapped to the centers of the $(\log p_T, \Delta\eta, \Delta\phi)$ bins, and $\boldsymbol{\epsilon}_k$ represents the noise that uniformly smears each constituent within its bin. This estimator becomes exact in the limit $M \to \infty$. In our implementation we choose $M = 100$.

B. Baseline evaluation of the model

We again perform a 2-sample classifier test between EPiC-FM generated jets and JETCLASS jets. We find AUCs of 0.6729 for QCD and 0.7703 for tops. These results indicate that EPiC-FM captures the main features of the simulation with moderate levels of mismodeling likely due to finite training data and finite model capacity.

C. Top vs QCD separation with EPiC-FM

Next we investigate whether EPiC-FM accurately models the separation of top vs QCD jets. In Fig. 2 we compare the ROC curves of three top vs QCD classifiers (each classifier is evaluated on the samples from the generative process it is defined/trained with):

(1) "EPiC-FM optimal": The NP-optimal EPiC-FM likelihood ratio discriminator
(2) "JetClass OmniLearn": An OmniLearn classifier trained on JETCLASS jets.
(3) "EPiC-FM OmniLearn": An OmniLearn classifier trained on samples from the EPiC-FM model.

We see from Fig. 2 that all three ROC curves are fairly consistent with one another. We point out that there are two possible interpretations of these results: either

(a) The NP-optimal classifier between EPiC-FM tops and QCD is representative of the NP-optimal classifier between JETCLASS tops and QCD, in which case there is a small gap between state-of-the-art jet taggers and the true "fundamental limit" of jet tagging,

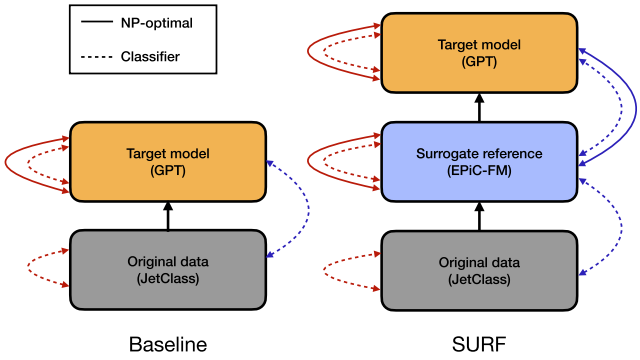


FIG. 1. Illustration of the SURF method. The conventional baseline approach is shown on the left and the SURF approach is shown on the right. The red lines denote interclass (e.g. top vs QCD) tests and the blue lines denote 2-sample tests. The solid lines correspond to NP-optimal tests, while dashed lines correspond to trained classifier tests (e.g. OmniLearn). Here we see the NP-optimal tests enabled by the SURF method.

or

(b) the EPiC-FM model is smearing out real physical features in JETCLASS which results in smaller separability between EPiC-FM tops and QCD.

While we cannot totally rule out the possibility that JETCLASS has subtle features that neither EPiC-FM nor OmniLearn can detect, the concurrence of JETCLASS OmniLearn with both EPiC-FM ROC curves is highly suggestive that the NP-optimal JETCLASS ROC curve is also close by, and that the SOTA classifiers are close to the fundamental limit.

In any case, the OmniLearn classifier is evidently powerful enough to learn the true separation of the EPiC-FM top and QCD samples, unlike what we saw for the GPT jets.[5] Together with the 2-sample classifier tests described in the previous subsection, this gives us confidence that EPiC-FM is not introducing unphysical artifacts into JETCLASS and thus constitutes a valid surrogate reference model.

IV. SURF METHOD

We have two generative models that give different possibilities for the "fundamental limit" of jet tagging—GPT suggests a huge gap between the true NP-optimal ROC curve of tops vs QCD and the performance of state-of-the-art classifiers, while EPiC-FM suggests almost no gap. Which is correct?

Here we propose a new method that provides a plausible answer to this puzzle: the SURF method. As illustrated in Fig. 1, the core idea is to use a generative model with

[5]The difference in R50 values between EPiC-FM Optimal and EPiC-FM OmniLearn is within a factor of 2. This is consistent with recent gains in top tagging from improved architectures and more data [7,12].

TABLE II. Same as Table I but with JETCLASS replaced with the EPiC-FM surrogate reference. The NP-optimal classifier AUC is now available thanks to the tractable likelihoods of the surrogate reference.

Comparison	Method	QCD	Top
EPiC-FM vs GPT	OmniLearn	0.9955	0.9746
	NP-optimal	1.000	1.000
EPiC-FM-bs vs GPT	OmniLearn	0.5505	0.5579
	NP-optimal	0.5640	0.6406
EPiC-FM vs EPiC-FM-bs	OmniLearn	0.9950	0.9733
	NP-optimal	0.9980	0.9967

tractable likelihoods—the surrogate reference model—to evaluate another target generative model whose likelihood is likewise tractable. We achieve this by first training the surrogate reference model on the original data, and then training the target model on samples generated by the surrogate reference model.

We have seen that EPiC-FM seems to be a valid surrogate for JETCLASS, in that it does not seem to introduce any subtle artifacts into JETCLASS that OmniLearn cannot detect. So the SURF method in this context is to use samples from EPiC-FM as a new reference dataset to train GPT on. Then we will have access to all the classifiers as before, plus a fourth one: the true NP-optimal ROC curve for the reference dataset. This is what's lacking for JETCLASS, and is the value added by the SURF method.

As a first application of the SURF method, we perform the 2-sample classifier tests for top and QCD jets. These are summarized in the first and second rows of Table II for continuous and bin-smeared jets respectively. We confirm using the EPiC-FM surrogate reference that the GPT-generated jets are well-separated from continuous jets, and that the OmniLearn classifier is able to pick this up. We also confirm that the GPT-generated jets are quite close to bin-smeared jets, so the OmniLearn classifier was not

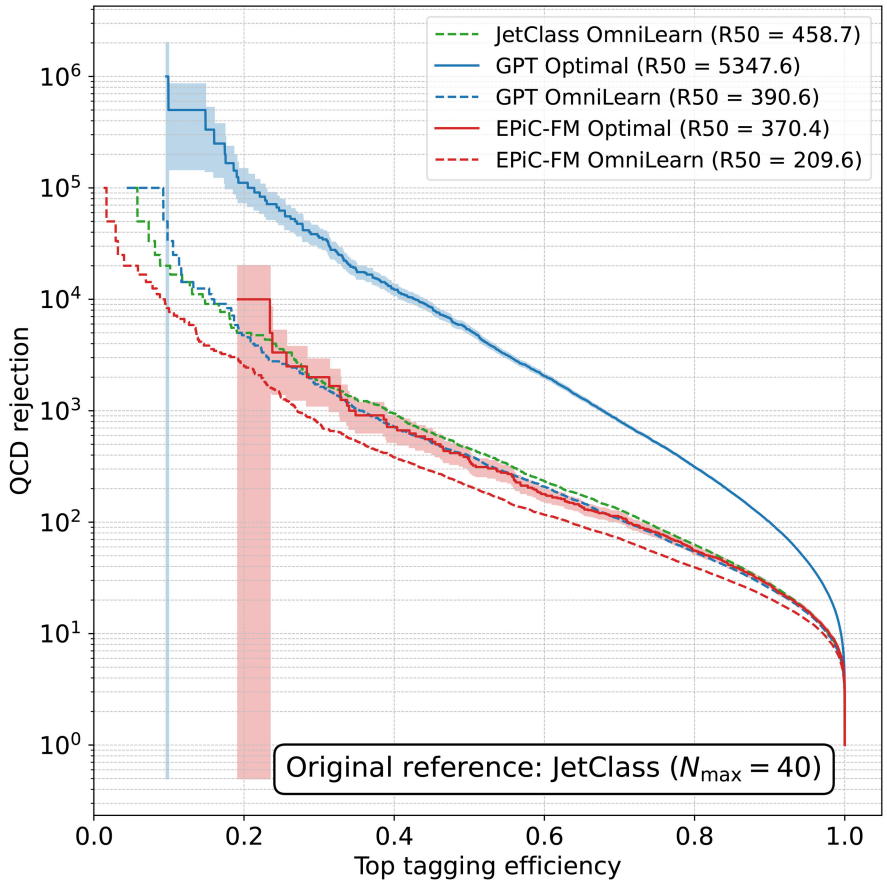


FIG. 2. ROC curves for top vs QCD jet classification derived from the JETCLASS dataset and the generative models trained on it. Note that, although physical jets can contain more than 40 constituents, in this study each jet is represented only by its 40 hardest constituents (i.e. $N_{max} = 40$). Solid lines indicate the performance of the NP-optimal classifier obtained directly from the true log-likelihood ratio. Dashed lines correspond to classifiers trained with OmniLearn. The QCD rejection at a top tagging efficiency of 50% (R50) is shown in parentheses for each curve. Shaded bands indicate the statistical uncertainty on the optimal ROC curves, estimated from binomial counting errors on the background sample and propagated to the QCD rejection axis.

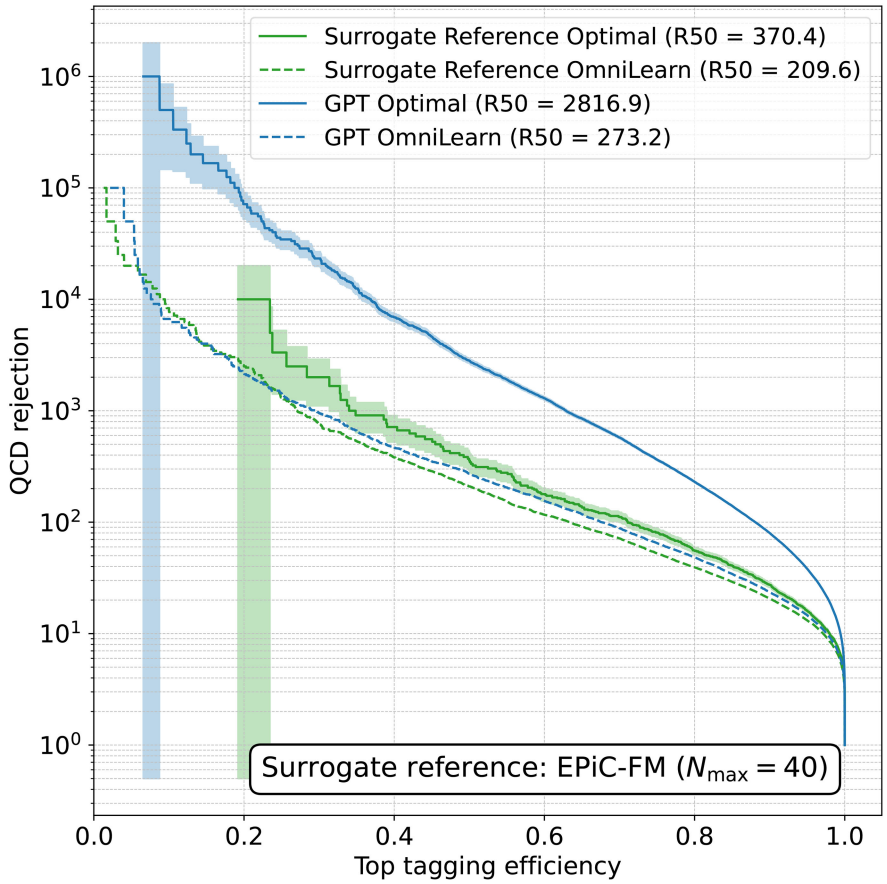


FIG. 3. ROC curves for top vs QCD jet classification derived from the EPiC-FM surrogate reference samples and the GPT models trained on them. Note that, although physical jets can contain more than 40 constituents, in this study each jet is represented only by its 40 hardest constituents (i.e. $N_{\text{max}} = 40$). Solid lines indicate the performance of the NP-optimal classifier obtained directly from the true log-likelihood ratio. Dashed lines correspond to classifiers trained with OmniLearn. The QCD rejection at a top tagging efficiency of 50% (R50) is shown in parentheses for each curve. Shaded bands indicate the statistical uncertainty on the optimal ROC curves, estimated from binomial counting errors on the background sample and propagated to the QCD rejection axis.

misleading on this front. As shown in the third row of Table II, we observe the same behavior previously identified for JETCLASS: bin-smeared EPiC-FM jets are nearly perfectly separable from their continuous counterparts.

Next we turn to the main question of interest: whether these GPT models accurately represent the true degree of top vs QCD separation or not. The relevant ROC curves are shown in Fig. 3. Here in addition to the analogs of the three classifiers considered previously—GPT optimal, surrogate reference OmniLearn and GPT OmniLearn, we are able to add a fourth ROC curve, surrogate reference optimal, thanks to the tractable likelihoods of the EPiC-FM surrogate reference model.

We see that the GPT model vastly inflates the ground-truth ROC curve of the surrogate reference model. By contrast, the OmniLearn classifier curves follow the surrogate reference optimal curve closely, showing that classifier-based discrimination remains consistent with the physically meaningful separation encoded in the EPiC-FM surrogate. This confirms that the inflated NP-optimal behavior originates from artifacts in the GPT likelihood landscape, not from deficiencies in classifier performance as was claimed in Ref. [14].

As expected, we observe the same behavior—both qualitatively and quantitatively—as in the baseline GPT evaluation (Fig. 2), where we found the NP-optimal ROC curve for GPT jets to be substantially larger than the OmniLearn classifiers trained on the reference data and on GPT samples. This cross-validation builds further confidence in our interpretation of the hugely inflated NP-optimal ROC curve from GPT tops and QCD.

V. INVESTIGATING ROC INFLATION IN THE GPT MODEL

Since top and QCD jets are already extremely well separated (AUC ≥ 0.98 with state-of-the-art taggers), the ROC curve of any approximate model becomes highly sensitive to even small mismodelings of the jet distributions.

We illustrate this sensitivity using a 10D Gaussian toy model with background given by $\mathcal{N}(\mathbf{0}, I_{10})$ and signal given by $\mathcal{N}(\mathbf{1}, I_{10})$. To emulate mismodeling, we introduce

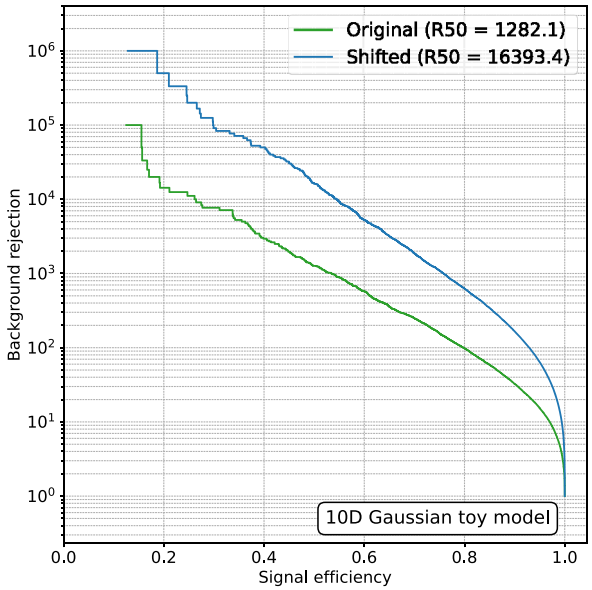


FIG. 4. ROC curves from the 10-dimensional Gaussian toy model. Although the shifted signal and background distributions differ only slightly from their original counterparts, their mutual ROC curve appears highly inflated. Here, "original" denote the original signal and background, while "shifted" indicate the slightly displaced versions. The background rejection at a signal efficiency of 50% (R50) is shown in parentheses for each curve. This demonstrates how small mismodelings of well-separated classes can artificially exaggerate separation power.

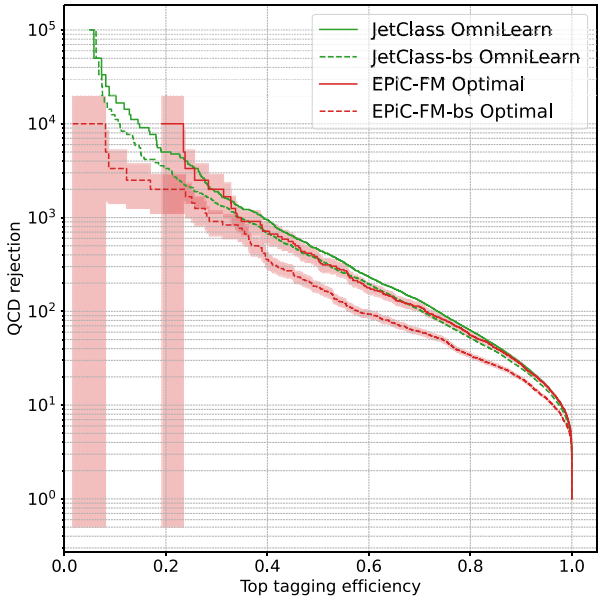


FIG. 5. Comparison of top vs QCD discrimination for both EPiC-FM surrogate reference jets and JETCLASS jets. The EPiC-FM curves, shown in red (continuous) and red dashed (bin-smeared, labeled "bs"), represent the true NP optimal. Shaded bands indicate the statistical uncertainty on the optimal ROC curves, estimated from binomial counting errors on the background sample and propagated to the QCD rejection axis. For JETCLASS, where the true likelihood is not available, the solid and dashed green curves show the corresponding OmniLearn classifiers trained on continuous and bin-smeared ("bs") JETCLASS samples, respectively. Across both EPiC-FM and JETCLASS jets, bin-smearing leads to slightly reduced separability between top and QCD jets.

shifted versions of the background and signal, described by Gaussians with slightly displaced means: $\mathcal{N}(-0.1\mathbf{1}_{10}, I_{10})$ and $\mathcal{N}(1.1\mathbf{1}_{10}, I_{10})$.

While the AUCs between the original and shifted distributions are around 0.6, the ROC curve between the shifted signal and background appears highly inflated, as shown in Fig. 4. Because the original signal and background already exhibit minimal overlap, even a small shift in the likelihood ratios can have an outsized impact on the ROC curve. This example highlights how minor deviations in otherwise well-separated classes can artificially enhance apparent separation power.

Next we turn to the question of what kind of mismodeling could be causing the inflated ROC curve in the GPT-generated jets.

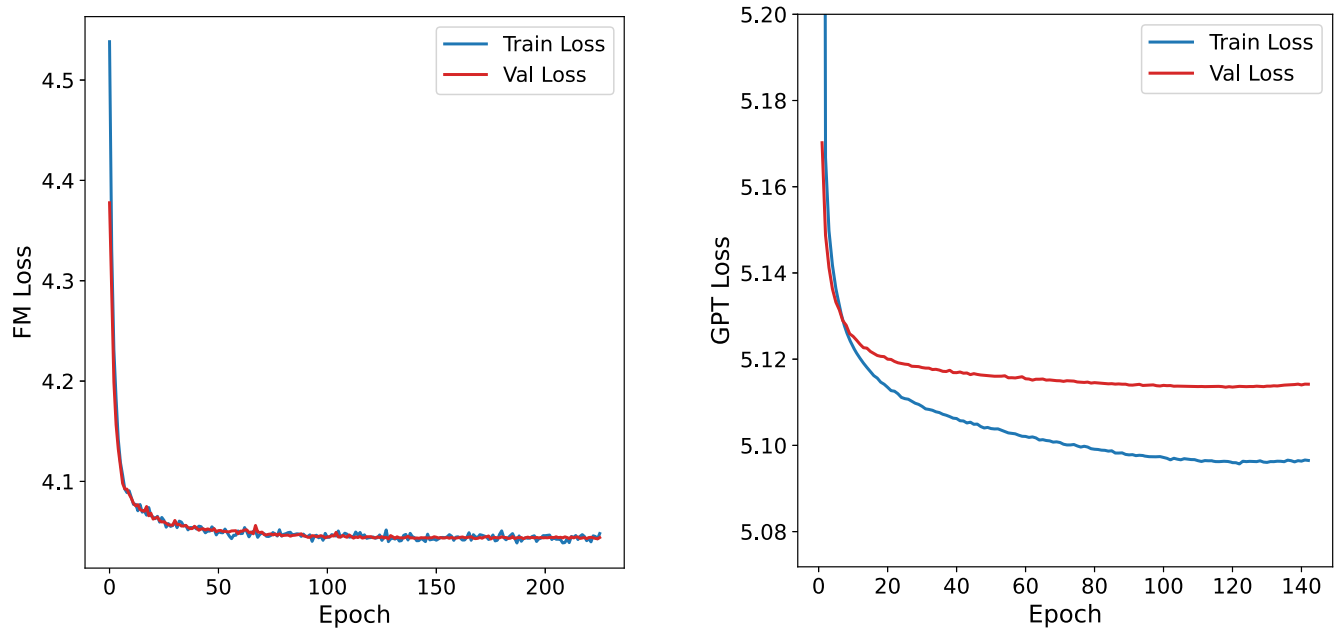
A. Bin-smearing?

The fact that the GPT-generated jets are attempting to match bin-smeared jets, and the bin-smeared jets are off-manifold compared to continuous jets, could be a significant source of mismodeling.

However, Fig. 5 shows that this is not the issue. Using the EPiC-FM model as the reference, we can compute the true likelihood of both continuous and bin-smeared jets, so we can obtain the true ROC curve between continuous tops vs QCD as well as bin-smeared tops vs QCD. We see that if anything, the bin-smeared jets are *less* separable than their original continuous counterparts. For JETCLASS, where the true likelihood is not available and we instead rely on an OmniLearn classifier, we observe the same qualitative behavior that bin-smearing reduces separability. So the binning of the jets, despite being a major deviation from the continuous jets, cannot be the cause of the inflated top vs QCD ROC curve.

B. Overfitting

Finally, we consider another potential source of the mismodeling that affects GPT-generated jets but not EPiC-FM-generated jets: overfitting. By overfitting, we refer to the situation where the model achieves higher performance on the training set than on the validation set. In our case, an overfitted model produces jets that more closely resemble those in the training dataset than those in the validation dataset. Shown in Fig. 6 are comparisons of the train vs validation loss curves for GPT and EPiC-FM models. We see that the GPT jets overfit almost immediately (increasing gap between train vs val losses), despite their val loss continuing to decrease slowly. Overfitting in GPT-style models is a known phenomenon, discussed already in the



(a) EPiC-FM train and validation loss curves.

(b) GPT (tops) train and validation loss curves.

FIG. 6. Training vs validation loss curves for EPiC-FM and GPT. The GPT plot is shown for tops, with qualitatively similar behavior observed for QCD. GPT overfits almost immediately (diverging train/val losses), while EPiC-FM maintains closer alignment.

original GPT-2 paper [34]. Meanwhile the EPiC-FM loss curves are in perfect agreement between train vs validation. Hence, we do not find any evidence for the EPiC-FM model overfitting.[6] More broadly, flow-based models such as diffusion and flow-matching have often been observed to generalize well in machine learning applications, though the underlying reasons remain an active area of investigation (see e.g., Refs. [35–37]).

Finally, we directly examine the effects of model overfitting on the optimal likelihood ratio performance. We train separate GPT and EPiC-FM models at fixed capacity on subsets of JETCLASS ranging from 10^5 to 10^7 jets for tops and QCD. As the training set size decreases—particularly when the number of examples becomes small relative to the number of model parameters and in the absence of strong regularization—GPT overfitting becomes more severe. Indeed, as shown in Fig. 7, the GPT models trained on smaller datasets exhibit a strongly inflated NP-optimal R_{80}, spanning several orders of magnitude, indicating that overfitting amplifies the apparent separation. The EPiC-FM models, by contrast, show only mild inflation, consistent with the loss curve comparison in Fig. 6 that finds no evidence of EPiC-FM overfitting.

The overfitting hypothesis could also potentially explain why state-of-the-art classifiers cannot detect the inflated ROC curve for GPT jets. Overfitting in generative models means that the model is too close to the training data, starting to memorize the training data and become a lookup table, i.e. the model is reducing to a sum of (smeared out)

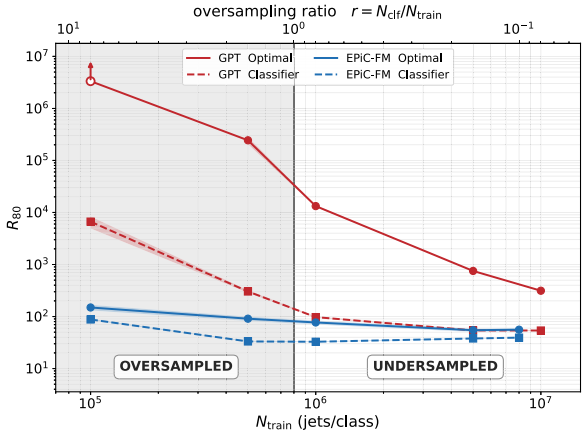


FIG. 7. QCD rejection at 80% top efficiency (R_{80}) for GPT and EPiC-FM models trained on increasing numbers of jets per class, N_{train}. Solid lines show the NP-optimal values, while dashed lines show the corresponding OmniLearn classifier values. The OmniLearn classifiers are trained on a fixed $N_{clf} = 8 \times 10^5$ generated jets per class, independent of N_{train}. The top axis gives the oversampling ratio $r = N_{clf}/N_{train}$, and the shaded region marks $r > 1$, where the classifier is trained on more generated jets than the generator itself saw. At the smallest training size, no GPT background jets survive the cut, so its optimal R_{80} is shown as a 95% CL lower limit.

[6]Although the EPiC-FM loss is not equivalent to the log-likelihood, we checked that the EPiC-FM model with the best validation loss gives consistent log-likelihoods on the train and validation set.

delta functions centered around the training data. Compared to the smooth distribution describing the reference data, a sum of smeared out delta functions is a high-frequency deformation. In the GPT case, this shows up as memorization of specific token cooccurrence patterns within the training jets, in the extreme, near-copying of jet (sub)sequences. Such behavior can distort multiparticle (higher-order) correlations, manifesting as "high-frequency" artifacts in the data manifold. Neural networks are well-known to have difficulty learning high frequency modes; see Refs. [38,39] for discussions and studies of this. We speculate that this could be the ultimate reason why the state-of-the-art classifiers cannot detect the inflated GPT ROC curves.

To investigate this explanation, we trained OmniLearn classifiers on the jets generated by the GPT and EPiC-FM models used in Fig. 7, using 8×10^5 jets per class for training, with a further 10^5 per class each for validation and testing. We observe the following behavior. For GPT, the NP-optimal R_{80} inflates monotonically as N_{train} decreases, by orders of magnitude across the full range. The classifier R_{80}, by contrast, remains roughly constant at large N_{train}, indicating that the dominant features of the learned distribution, which are all the classifier can access in this regime, are stable. It inflates only once N_{train} falls below the classifier training size, and even then far less than the NP-optimal value, so the gap between the two grows sharply. For EPiC-FM, both the NP-optimal and classifier R_{80} inflate only mildly and stay close to one another throughout.

This behavior is consistent with our interpretation. When a generative model overfits on limited training data, it introduces fine-grained structure in the learned density. Such structure is present in the likelihood ratio, and therefore in the NP-optimal ROC curve, but a trained classifier can only reliably exploit it if it is sufficiently sampled. The classifier is trained on 8×10^5 generated jets per class, so for generators trained on fewer jets we are in an oversampling regime, where these fine-grained features are repeatedly sampled and can be partially learned; the classifier R_{80} inflates accordingly, only once this regime is entered. Taken together with the loss curve comparison, the much weaker inflation of the EPiC-FM curves suggests that the continuous flow model overfits far less than the discrete GPT model, if at all. We caution, however, that the absence of detectable overfitting does not establish that EPiC-FM does not overfit, only that we find no evidence for it with the diagnostics used here.

VI. CONCLUSION

We revisited recent claims that autoregressive GPT-style jet generators expose a large performance gap between ML-based jet taggers and the NP optimal "fundamental limit" of top vs QCD discrimination [14]. Using our own independently trained GPT-based jet generator—with an improved architecture from that used in the original study—we reproduced their reported behavior. However, when comparing with the tractable EPiC-FM flow-matching model, which provides explicit likelihoods for unordered particle clouds, we find no evidence for such an inflated limit.

To resolve the apparent tension between the EPiC-FM and GPT results, we introduced the SURF method, which enables exact NP tests even when the real data likelihood is intractable. By training the target GPT model on samples from a tractable EPiC-FM surrogate, SURF allows direct NP comparisons between the surrogate and the target. We find that the GPT likelihood ratios dramatically overstate the true top vs QCD separation encoded in the EPiC-FM surrogate. The "fundamental limit" ROCs obtained from GPT models therefore reflect artifacts of the generative process rather than performance limitations of the jet taggers.

We speculate that the inflated GPT ROC curve could be due to overfitting, and more generally high frequency artifacts introduced by the GPT model. Such high-frequency modes can be exploited by exact NP tests but are difficult for neural-network classifiers to learn, given their well-known low-frequency spectral bias [38,39]. In future work it would be interesting to explore these high frequency artifacts in more detail. Existing work in the ML literature (e.g., [40]) suggests potential methods for making such artifacts more explicit. It would also be interesting to develop new classifier architectures that can detect these high frequency artifacts. This would lead to more sensitive classifier metric tests.

Furthermore, our study shows that GPT models trained on tokenized jets are faithful only to their discretized references, while being nearly perfectly separable from the original continuous jets. These results indicate that discretization can introduce significant off-manifold effects, which may limit the fidelity of generative models based on binning and tokenization if not carefully mitigated. Alternative discretization schemes could potentially reduce the degree of separability from the continuous reference. For example, Refs. [22,23] use VQ-VAEs to tokenize jet constituents. Exploring such strategies, and their impact on preserving on-manifold fidelity, would be an interesting direction for future study.

Overall, the consistent picture between JETCLASS and the EPiC-FM surrogate reference, particularly the concurrence of three ROC curves—classifiers trained on samples from JETCLASS and EPiC-FM samples, and the NP-optimal likelihood ratio from EPiC-FM—point to the fundamental limit of top tagging being not far from the performance of state-of-the-art classifiers. However, we cannot rule out the possibility that there remains a gap between the fundamental limit of top tagging and SOTA classifiers—the EPiC-FM surrogate and OmniLearn may both miss subtle features in JETCLASS that make tops and QCD more separable than they seem.

We can think of several avenues for further probing the fundamental limit of jet tagging. For example, improving the surrogate reference should be a high priority. Our EPiC-FM surrogate was far from perfect—our classifier tests showed that they are partially separable from the original continuous jets (AUC ≈ 0.7). It would be worthwhile to repeat this study with a more expressive surrogate reference (for example replacing EPiC with a transformer) that is closer to the JETCLASS data to confirm the conclusions reached here. It would also be good to repeat this study with a more faithful target generative model that does not inflate the true ROC curve of the surrogate reference. Taken together (a more faithful surrogate and target), this could start to reveal a gap between SOTA classifiers and the fundamental limit.

Recently some studies were initiated into the true theoretical limits on top tagging [41]. It would be interesting to bring the empirical, data-driven approach explored here into contact with the theory-driven approach and elucidate further the true fundamental limit of jet tagging.

We focused on just the momentum four-vectors of jet constituents in this work. For many jet tagging tasks, including top tagging, additional features are highly informative, such as track displacements and particle ID. These features are also included in JETCLASS, and it would be interesting to repeat this study for the full set of JETCLASS features to obtain an even fuller picture of the fundamental limit of jet tagging.

Finally, the SURF method provides a general route to perform exact NP tests using a tractable surrogate reference. Extending this approach to other classes of generative models, datasets, and applications—both within and beyond high-energy physics—would enable more systematic analysis of generative models.

ACKNOWLEDGMENTS

I. P., D. A. F., D. S., and R. D. are supported by DOE Grant No. DOE-SC0010008. R. D. is also supported by Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Grant No. 396021762–TRR 257 Particle Physics Phenomenology after the Higgs Discovery. G. K. is supported by the DFG under the German Excellence Initiative—EXC 2121 Quantum Universe—390833306. 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 Award No. HEP-ERCAP0027491.

DATA AVAILABILITY

The code for the GPT model and the EPiC-FM model can be found at [42,43], respectively.

APPENDIX A: DETAILS OF THE EPIC-FM MODEL

1. Architecture and training

The EPiC network is composed of EPiC layers [44]—a permutation-equivariant variant of Deep Sets [45], also known in high-energy physics as particle flow networks (PFNs) [46]. These layers couple per-particle embeddings to a learnable global context, enabling the model to capture particle correlations while exactly preserving the permutation symmetry of the constituents [20]. The setup for our architecture closely follows the one in [21], with the main difference being that the only conditional model inputs that we consider are the time variable $t \in [0, 1]$, the jet type (QCD and tops), and the number of constituents in each jet. Time is embedded into a vector space $\mathbb{R}^{16}$ using a sinusoidal embedding network and the jet-type labels are embedded into $\mathbb{R}^8$ using a linear lookup table. We stack 12 EPiC-Layers with a local hidden dimension of $h_{\text{loc}} = 300$ and a global hidden dimension $h_{\text{glob}} = 32$. The variable number of particles per jet is handled via binary masking. During training the multiplicity is given by the true number of constituents in each jet, while during generation it is sampled from an empirical distribution computed from the JETCLASS dataset.

During training, we construct source-target pairs by sampling the source from a three-dimensional Gaussian and randomly pairing it with a target jet (QCD or top) from the dataset. We use 10 million top jets and 10 million QCD jets, split 80/20 into train/validation.

Optimization uses Adam [47] with an initial learning rate of 10^{-4} and a reduce-on-plateau schedule triggered by the validation loss with a patience of 10 epochs. We train for a total of 1000 epochs with a patience window of 50 epochs. We select the best checkpoint with the lowest validation loss.

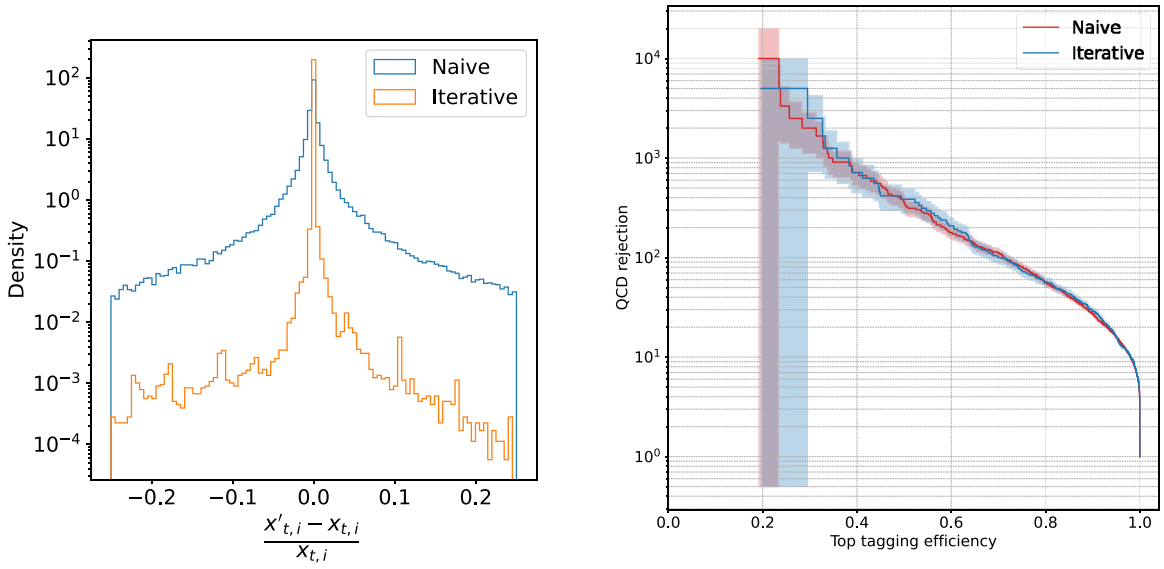
2. Validation of log-likelihood computation

With a trained FM model $\mathbf{u}^\theta$, we can calculate the log-likelihood iteratively by taking the following inference step

$$\log p_{t-\Delta t}(\mathbf{x}_{t-\Delta t}) = \log p_t(\mathbf{x}_t) - \Delta t \mathrm{Tr}(\texttt{Jac}[\mathbf{u}^\theta_{t-\Delta t}(\mathbf{x}_t)]),$$

at every time step Δt. We compute the Jacobian matrix `Jac[]` of the neural network in pytorch via `torch.func.jacrev`. We verified that using `torch.autograd.functional.jacobian` yields identical results.

A natural concern is whether flow matching models yield *reliable* likelihood *ratios*, especially in light of claims about the precision of likelihood computation in related models [14]. Constant offsets in log-likelihoods cancel in this ratio and therefore do not affect ROC curves. Operationally, we require numerical stability of *differences* in log-likelihoods accumulated along the FM trajectory. We assessed this with the following two tests.



(a) Error suppression (b) Classifier robustness

FIG. 8. Effect of fixed-point refinement in the backward update. "Naive" denotes the backward update computed without any fixed-point iterations; "Iterative" denotes the backward update performed with five fixed-point refinement iterations. (a) Component-wise relative error $\frac{x'_{t,i}-x_{t,i}}{x_{t,i}}$ is sharply peaked around zero when using fixed-point iterations. Here we show results for a few top jets; similar behavior is observed across larger jet samples, including QCD jets. (b) The ROC curves built from FM-based log-likelihood ratios with and without refinement nearly coincide, showing negligible impact on classifier performance. Shaded bands indicate the statistical uncertainty on the optimal ROC curves, estimated from binomial counting errors on the background sample and propagated to the QCD rejection axis.

a. Consistency of forward and backward Euler integration

As mentioned in Sec. III A, the forward Euler step is used for generation and the backward step is used for log-likelihood computation. Therefore, it is vital that the backward step is as close as possible to an exact inverse of the forward step. This guarantees consistent trajectories and preserves the theoretical assumption that the underlying flow is perfectly reversible. Any mismatch between the two steps violates this assumption and might lead to incorrect log-likelihood computation.

Unlike for regular normalizing flows, consistency of forward and backward steps is not guaranteed for continuous normalizing flows. The reason can be seen by the following:

$$\text{Forward step: } \mathbf{x}_{t+\Delta t} = \mathbf{x}_t + \Delta t \mathbf{u}_t^\theta(\mathbf{x}_t), \tag{A1}$$

$$\text{Backward step: } \mathbf{x}'_t = \mathbf{x}_{t+\Delta t} - \Delta t \mathbf{u}_t^\theta(\mathbf{x}_{t+\Delta t}). \tag{A2}$$

A backward update [Eq. (A2)] generally fails to recover the original point, since

$$\mathbf{x}'_t - \mathbf{x}_t = \Delta t(\mathbf{u}_t^\theta(\mathbf{x}_{t+\Delta t}) - \mathbf{u}_t^\theta(\mathbf{x}_t)) \neq 0$$

This means that the forward and backward updates are not exact inverses of each other; they only approach invertibility in the limit of infinitesimally small step sizes, $\Delta t \to 0$.

We observe that the backward step would be an exact inverse of the forward step if, in Eq. (A2), the vector field $\mathbf{u}_t$ is evaluated at $\mathbf{x}_t$ rather than at $\mathbf{x}_{t+\Delta t}$. However, this is precisely what is unavailable during the computation of the backward step, indeed this is meant to be the *output* of the backward step.

The problem is solved if we view the backward step as an *implicit equation* for $\mathbf{x}_t$:

$$\mathbf{x}_t = \mathbf{x}_{t+\Delta t} - \Delta t \mathbf{u}_t^\theta(\mathbf{x}_t), \tag{A3}$$

This can be solved for $\mathbf{x}_t$ via fixed-point iteration.

Figure 8(a) displays the distribution of the component-wise relative error

$$\frac{x'_{t,i} - x_{t,i}}{x_{t,i}},$$

where i indexes all components of the flattened jet (i.e., both constituent and feature). Results are shown for backward updates computed with and without fixed-point refinement over 300 integration time steps. When employing the iterative backward update, we performed five fixed-point refinement iterations. The iterative method yields a sharply peaked distribution around zero and strongly suppresses the heavy tails observed in the naive backward update.

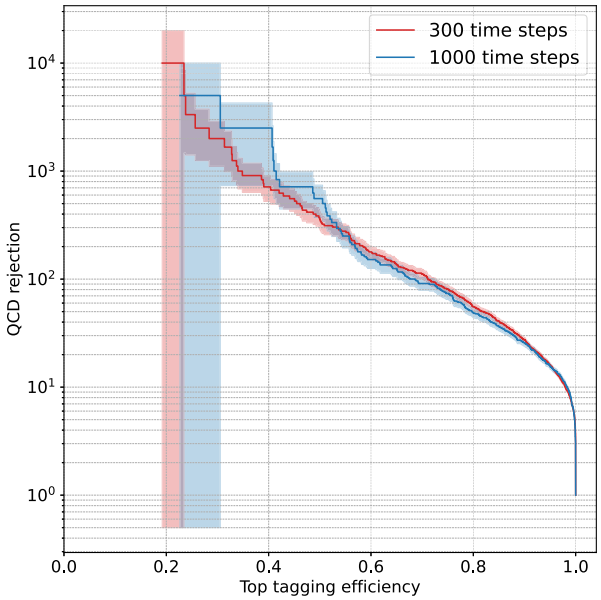


FIG. 9. Comparison of NP-optimal ROC curves with number of inference/sampling timesteps. The blue curve shows the performance when both inference and sample generation use 300 time steps, while the red curve corresponds to 1000 time steps. The shaded bands denote binomial counting errors on the background sample and propagated to the QCD rejection axis.

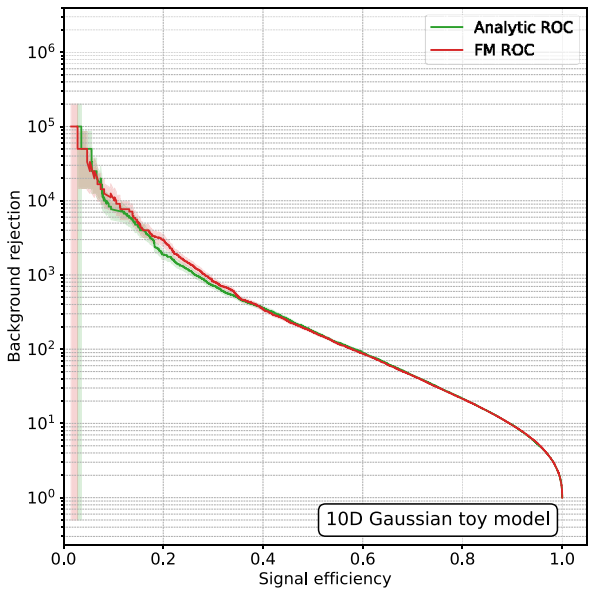


FIG. 10. ROC curves for 10D gaussian toy example used for flow matching log-likelihood validation. The shaded bands denote binomial counting errors on the background sample and propagated to the background rejection axis.

Nevertheless, we show in Fig. 8(b) that the error has minimal impact on approximate NP classifier performance computed based on log-likelihood values obtained with and without the iterative procedure.[7] This study demonstrates the robustness of the FM-based log-likelihood ratio computation to errors accumulated in the backward update. For the main results presented in the paper, we employed the naive backward update to efficiently generate a large training dataset for the jet taggers at reasonable computational costs.

b. Number of time steps

To assess the dependence of the approximate Neyman-Pearson classifier performance on the number of time steps used for sample generation and inference of the log-likelihood values, we compared two configurations—300 versus 1000 time steps. For each configuration, the same number of time steps was used for sampling and inference. No fixed-point iteration refinement was used in either configuration. We found that using 1000 time steps for generation results in slightly higher fidelity jets. Nevertheless, as shown in Fig. 9, both configurations yield statistically consistent classification performance. The larger error bands for the 1000-step case are from the smaller evaluation sample size—imposed by computational constraints—used to construct the ROC curve. For our main results in the paper, we used the 300-step configuration in order to obtain a large training dataset for the jet taggers while keeping computational costs manageable.

Beyond these internal consistency checks, we also verify correctness in a controlled toy setting where the analytic NP ROC is known. This provides an end-to-end demonstration that FM likelihood ratios reproduce exact NP behavior in a case with closed-form ground truth.

c. Validation on a toy 10D Gaussian model

We construct two ten-dimensional Gaussian distributions with shared unit covariance but shifted means:

$$p_{\mathrm{sig}} = \mathcal{N}(+0.4\mathbf{1}_{10}, I_{10}), \qquad p_{\mathrm{bg}} = \mathcal{N}(-0.4\mathbf{1}_{10}, I_{10}).$$

In this setting, the NP test has a closed-form solution, and the corresponding ROC curve can be derived analytically. We train separate FM models, implemented as simple MLP-based vector-field networks, on samples from each Gaussian. After training, we generate samples from the FM models and evaluate the NP log-likelihood ratio using the FM log-density estimators on these generated samples. Figure 10 shows that the ROC curve obtained from FM likelihood ratios coincides quite well with the analytic NP ROC and is mostly within statistical fluctuations.

This toy test confirms that flow matching recovers the correct ROC behavior in a regime where the ground truth NP statistic is exactly known, supporting the validity of our approach before applying it to more complex jet data.

[7]In the iterative case, error bands are wider, as computational constraints limited the evaluation sample size used to construct the ROC curve.

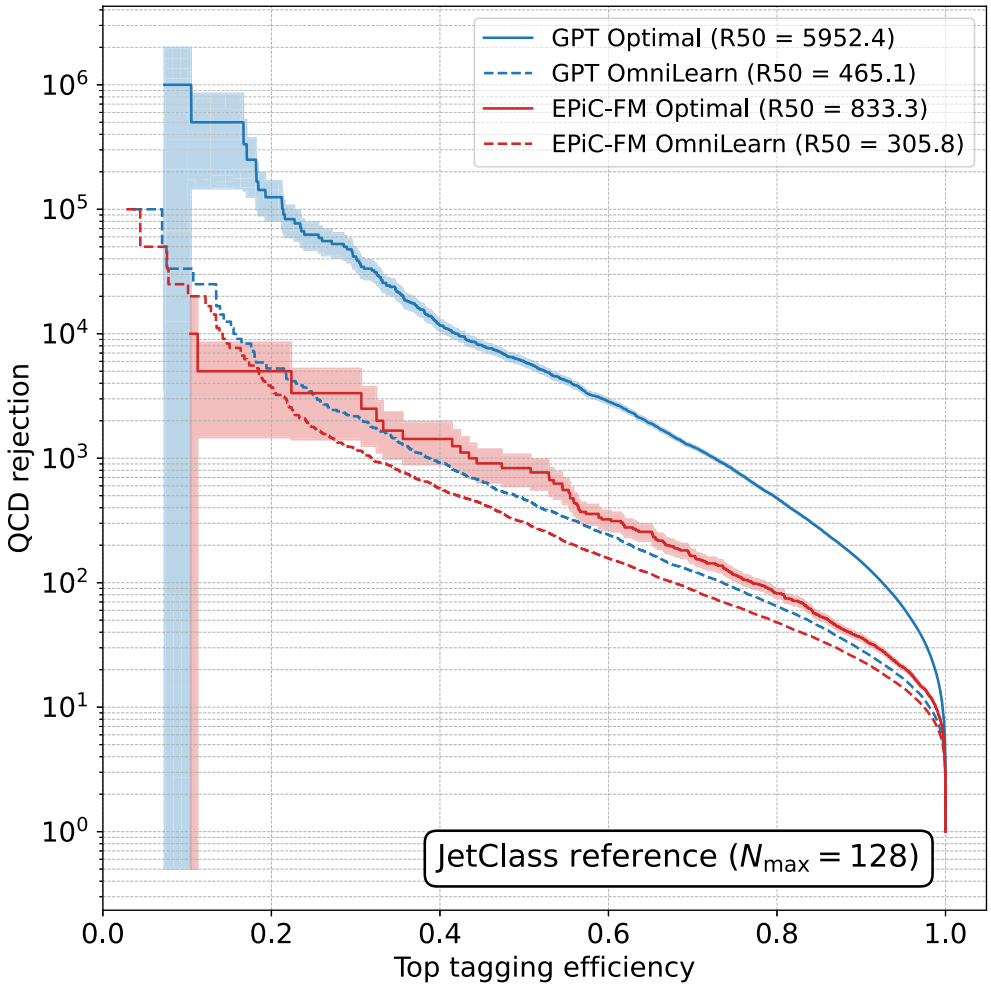


FIG. 11. ROC curves for top vs QCD jet classification derived from the JETCLASS dataset and the generative models trained on it. In contrast to the main study, each jet is represented by its 128 hardest constituents (i.e. $N_{max} = 128$). Solid lines indicate the performance of the NP-optimal classifier obtained directly from the true log-likelihood ratio. Dashed lines correspond to classifiers trained with OmniLearn. The QCD rejection at a top tagging efficiency of 50% (R50) is shown in parentheses for each curve. Shaded bands indicate the statistical uncertainty on the optimal ROC curves, estimated from binomial counting errors on the background sample and propagated to the QCD rejection axis.

Taken together, these checks validate that FM-based log-likelihood ratios are numerically stable under integrator choices and, in the Gaussian toy model, reproduce the exact NP ROC. They do not claim absolute calibration of log-densities; rather, they establish that the relative log-densities that define our ROC curves are robust for the analyses presented here.

APPENDIX B: DETAILS OF THE GPT MODEL

For our model, we use the `GPT2LMHeadModel` implementation of the GPT-2 transformer architecture [34] provided by HuggingFace [24], configured with an initial token embedding to $\mathbb{R}^{d_{\text{embd}}}$, followed by $n_{\text{layer}} = 12$ transformer blocks, each with $n_{\text{head}} = 8$ attention heads. We set $d_{\text{embd}} = 256$ for the embedding/hidden size, $d_{\text{inner}} = 1024$ for the feed-forward hidden size, $p_{\text{drop}} = 0.1$ for all dropout layers and `GELU` for all nonlinear activation functions. This resulting model contains approximately 16 million trainable parameters. In addition to the standard causal mask that enforces autoregressive ordering, we apply an attention mask to ignore `[PAD]` tokens. The final "language head" then maps the last hidden states to vocabulary logits through a linear layer.

The model is trained by maximizing the autoregressive likelihood (1) with $t_0 \equiv$ [START], excluding `[PAD]` tokens. Unlike EPiC-FM, which conditions on jet type, we train separate class-specific GPT models—one for top jets and one for QCD jets. Optimization uses Adam [47] with an effective batch size of 256 jets for up to 150 epochs: a cosine-annealing schedule decays the learning rate from 10^{-3} to 10^{-4} over the first 100 epochs, followed by 50 epochs at a fixed rate of 10^{-4}. The final model is selected by the lowest validation loss.

APPENDIX C: RESULTS FOR $N_{\max} = 128$

In this Appendix, we present a minimal extension of our study to higher particle multiplicity, increasing the maximum number of constituents per jet from $N_{\max} = 40$ to $N_{\max} = 128$. For this study, the EPiC-FM model is trained on JETCLASS jets in their continuous representation including up to the 128 hardest constituents, while the GPT model is trained on the corresponding tokenized representation. In order to maintain generation quality at this higher dimensionality, we increase the capacity of the EPiC-FM model accordingly.

Figure 11 shows the corresponding ROC curves for top vs QCD classification. As in the main text, we compare the performance of the NP-optimal classifier, obtained from the log-likelihood ratio, with that of classifiers trained using OmniLearn.

We observe behavior consistent with our findings at $N_{\max} = 40$. In particular, the GPT-based models exhibit a sizable discrepancy between the NP-optimal and classifier ROC curves, while the EPiC-FM NP-optimal ROC curve remains in closer agreement with the classifier performance.

These results support the robustness of our analysis with respect to particle multiplicity and indicate that restricting to $N_{\max} = 40$ captures the essential features relevant for our study. Given the consistency of our findings, we do not find it necessary to perform a more extensive exploration of higher multiplicities.

[1] H. Qu and L. Gouskos, ParticleNet: Jet tagging via particle clouds, Phys. Rev. D **101**, 056019 (2020).

[2] A. Butter *et al.*, The machine learning landscape of top taggers, SciPost Phys. **7**, 014 (2019).

[3] A. M. Sirunyan *et al.* (CMS), Identification of heavy, energetic, hadronically decaying particles using machine-learning techniques, J. Instrum. **15**, P06005 (2020).

[4] S. Gong, Q. Meng, J. Zhang, H. Qu, C. Li, S. Qian, W. Du, Z.-M. Ma, and T.-Y. Liu, An efficient Lorentz equivariant graph neural network for jet tagging, J. High Energy Phys. 07 (2022) 030.

[5] A. Bogatskiy, T. Hoffman, D. W. Miller, J. T. Offermann, and X. Liu, Explainable equivariant neural networks for particle physics: PELICAN, J. High Energy Phys. 03 (2024) 113.

[6] H. Qu, C. Li, and S. Qian, Particle transformer for jet tagging, arXiv:2202.03772.

[7] J. Brehmer, V. Bresó, P. de Haan, T. Plehn, H. Qu, J. Spinner, and J. Thaler, A Lorentz-equivariant transformer for all of the LHC, SciPost Phys. **19**, 108 (2025).

[8] V. Mikuni and B. Nachman, Solving key challenges in collider physics with foundation models, Phys. Rev. D **111**, L051504 (2025).

[9] V. Mikuni and B. Nachman, A method to simultaneously facilitate all jet physics tasks, Phys. Rev. D **111**, 054015 (2025).

[10] ATLAS Collaboration, Accuracy versus precision in boosted top tagging with the ATLAS detector, J. Instrum. **19**, P08018 (2024).

[11] L. Favaro, G. Gerhartz, F. A. Hamprecht, P. Lippmann, S. Pitz, T. Plehn, H. Qu, and J. Spinner, Lorentz-equivariance without limitations, arXiv:2508.14898.

[12] W. Bhimji, C. Harris, V. Mikuni, and B. Nachman, OmniLearned: A foundation model framework for all tasks involving jet physics, Phys. Rev. D **113**, 032020 (2026).

[13] J. Neyman and E. S. Pearson, On the problem of the most efficient tests of statistical hypotheses, Phil. Trans. R. Soc. A **231**, 289 (1933).

[14] J. Geuskens, N. Gite, M. Krämer, V. Mikuni, A. Mück, B. Nachman, and H. Reyes-González, The fundamental limit of jet tagging, Phys. Rev. D **112**, L091901 (2025).

[15] H. Qu, C. Li, and S. Qian, JetClass: A large-scale dataset for deep learning in jet physics, 10.5281/zenodo.6619768 (2022).

[16] T. Finke, M. Krämer, A. Mück, and J. Tönshoff, Learning the language of QCD jets with transformers, J. High Energy Phys. 06 (2023) 184.

[17] M. S. Albergo and E. Vanden-Eijnden, Building normalizing flows with stochastic interpolants, arXiv:2209.15571.

[18] Y. Lipman, R. T. Chen, H. Ben-Hamu, M. Nickel, and M. Le, Flow matching for generative modeling, arXiv:2210.02747.

[19] A. Tong, K. Fatras, N. Malkin, G. Huguet, Y. Zhang, J. Rector-Brooks, G. Wolf, and Y. Bengio, Improving and generalizing flow-based generative models with minibatch optimal transport, arXiv:2302.00482.

[20] E. Buhmann, C. Ewen, D. A. Faroughy, T. Golling, G. Kasieczka, M. Leigh, G. Quétant, J. A. Raine, D. Sengupta, and D. Shih, EPiC-ly fast particle cloud generation with flow-matching and diffusion, arXiv:2310.00049.

[21] J. Birk, E. Buhmann, C. Ewen, G. Kasieczka, and D. Shih, Flow matching beyond kinematics: Generating jets with particle identification and trajectory displacement information, Phys. Rev. D **111**, 052008 (2025).

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

[23] J. Birk, A. Hallin, and G. Kasieczka, OmniJet-α: The first cross-task foundation model for particle physics, Mach. Learn. Sci. Tech. **5**, 035031 (2024).

[24] T. Wolf *et al.*, Transformers: State-of-the-art natural language processing, in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations* (Association for Computational Linguistics, 2020), pp. 38–45.

[25] O. Press and L. Wolf, Using the output embedding to improve language models, *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics* (2017).
[26] H. Inan, K. Khosravi, and R. Socher, Tying word vectors and word classifiers: A loss framework for language modeling, in *International Conference on Learning Representations (ICLR)* (2017).
[27] C. Krause and D. Shih, CaloFlow: Fast and accurate generation of calorimeter showers with normalizing flows, Phys. Rev. D **107**, 113003 (2021).
[28] O. Amram *et al.*, CaloChallenge 2022: A community challenge for fast calorimeter simulation, Rep. Prog. Phys. **88**, 116201 (2025).
[29] S. H. Lim, K. A. Raman, M. R. Buckley, and D. Shih, GalaxyFlow: Upsampling hydrodynamical simulations for realistic mock stellar catalogues, Mon. Not. R. Astron. Soc. **533**, 143 (2024).
[30] R. Kansal, A. Li, J. Duarte, N. Chernyavskaya, M. Pierini, B. Orzari, and T. Tomei, Evaluating generative models in high energy physics, Phys. Rev. D **107**, 076017 (2023).
[31] R. Das, L. Favaro, T. Heimel, C. Krause, T. Plehn, and D. Shih, How to understand limitations of generative networks, SciPost Phys. **16**, 031 (2023).
[32] P. Cappelli, G. Grosso, M. Letizia, H. Reyes-González, and M. Zanetti, Learning to validate generative models: A goodness-of-fit approach, Mach. Learn. Sci. Tech. **7**, 045011 (2026).
[33] H. Reyes-Gonzalez, M. Krämer, A. Mück, B. Nachman, V. Mikuni, J. Geuskens, and N. Gite, Resources for the fundamental limit of jet tagging, 10.5281/zenodo.14023638 (2024).
[34] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, Language models are unsupervised multitask learners (2019).
[35] Z. Kadkhodaie, F. Guth, E. P. Simoncelli, and S. Mallat, Generalization in diffusion models arises from geometry-adaptive harmonic representations, arXiv:2310.02557.
[36] J. J. Vastola, Generalization through variance: How noise shapes inductive biases in diffusion models, arXiv:2504.12532.
[37] Q. Bertrand, A. Gagneux, M. Massias, and R. Emonet, On the closed-form of flow matching: Generalization does not arise from target stochasticity, arXiv:2506.03719.
[38] N. Rahaman, A. Baratin, D. Arpit, F. Draxler, M. Lin, F. Hamprecht, Y. Bengio, and A. Courville, On the spectral bias of neural networks, in *International Conference on Machine Learning* (PMLR, 2019), pp. 5301–5310.
[39] B. Ronen, D. Jacobs, Y. Kasten, and S. Kritchman, The convergence rate of neural networks for learned functions of different frequencies, Adv. Neural Inf. Process. Syst. **32** (2019).
[40] Y. Zhang and X. Xu, Diffusion noise feature: Accurate and fast generated image detection, arXiv:2312.02625.
[41] A. J. Larkoski, Systematic interpretability and the likelihood for boosted top quark identification, arXiv:2411.00104.
[42] D. Faroughy, JetSequences, GitHub repository (2025), https://github.com/dfaroughy/JetSequences.
[43] I. Pang, jet_cfm, GitHub repository (2025), https://github.com/Ian-Pang/jet_cfm.
[44] E. Buhmann, G. Kasieczka, and J. Thaler, EPiC-GAN: Equivariant point cloud generation for particle jets, SciPost Phys. **15**, 130 (2023).
[45] M. Zaheer, S. Kottur, S. Ravanbakhsh, B. Poczos, R. R. Salakhutdinov, and A. J. Smola, Deep sets, Adv. Neural Inf. Process. Syst. **30** (2017).
[46] P. T. Komiske, E. M. Metodiev, and J. Thaler, Energy flow networks: Deep sets for particle jets, J. High Energy Phys. 01 (2019) 121.
[47] D. P. Kingma and J. Ba, Adam: A method for stochastic optimization, arXiv:1412.6980.