SURFing to the fundamental limit of jet tagging

Ian Pang

May 12, 2026

Pheno 2026, Pittsburgh

[2511.15779] **IP,** D. Faroughy, D. Shih , R. Das, G. Kasieczka

ian.pang@physics.rutgers.edu

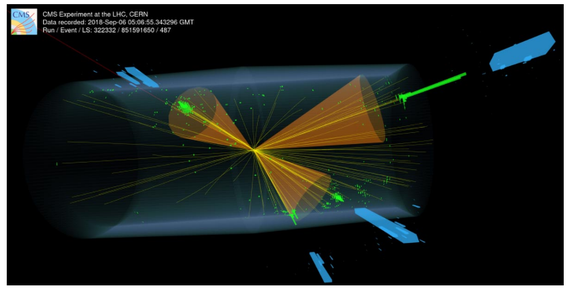
Outline

1. **Generation:** Jet generative models

2. **Inference:** Fundamental limit of jet tagging

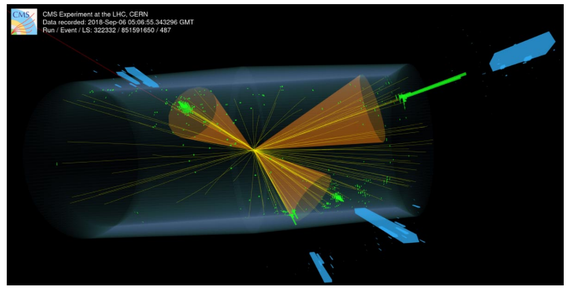
3. **Evaluation:** SURF method

Jet generative models

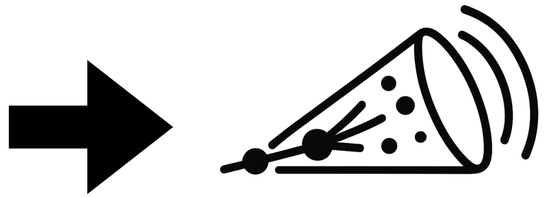


- A jet is a collimated spray of particles, each with features $(p_T, \Delta\eta, \Delta\phi)$

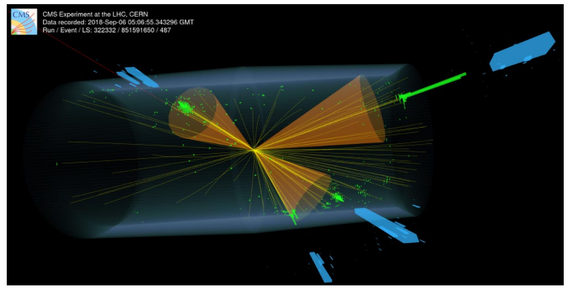
Jet generative models



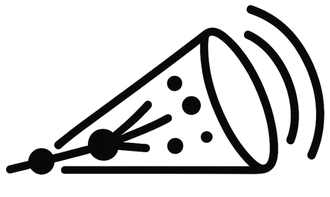
- A jet is a collimated spray of particles, each with features $(p_T, \Delta\eta, \Delta\phi)$

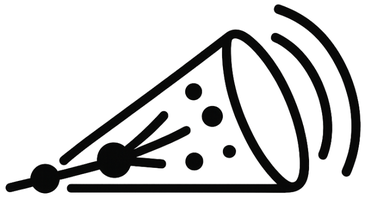
Generation: Generative model 

Jet generative models



- A jet is a collimated spray of particles, each with features $(p_T, \Delta\eta, \Delta\phi)$

Generation: Generative model → 

Likelihood computation:  → Generative model → Likelihood $p(\text{jet})$

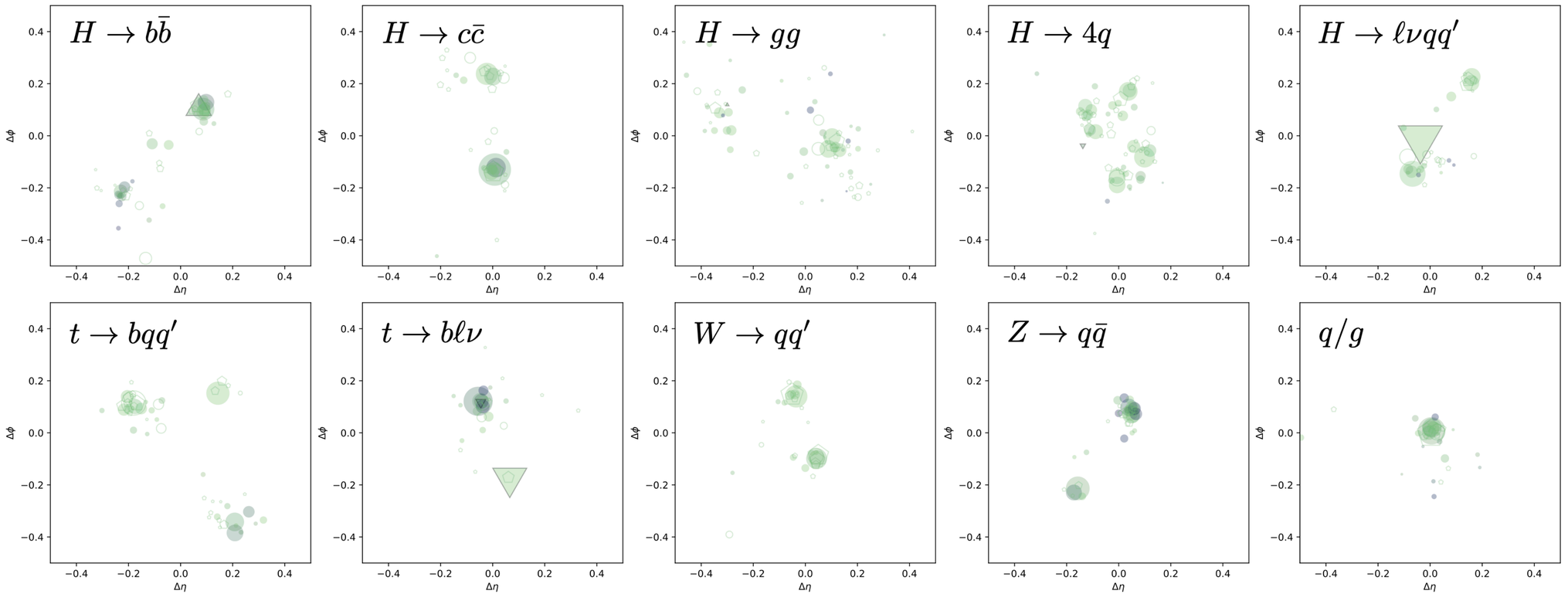
Jet tagging

Jet tagging

- **Jet tagging** is a key task in collider physics

Jet tagging

- **Jet tagging** is a key task in collider physics



https://github.com/jet-universe/particle_transformer

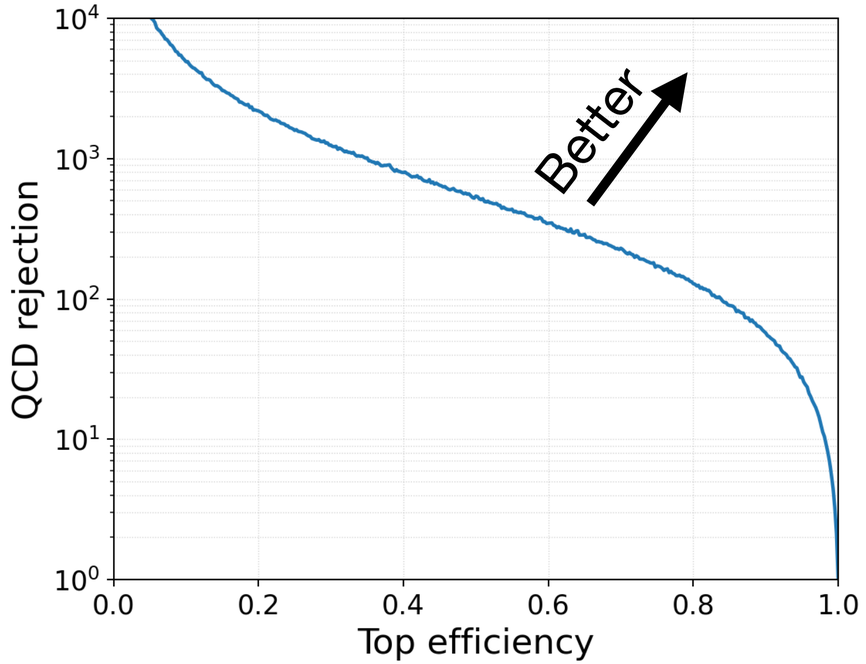
Jet tagging

- **Jet tagging** is a key task in collider physics
- Can think of it as (binary) classification task!

Jet tagging

- **Jet tagging** is a key task in collider physics
- Can think of it as (binary) classification task!

ROC curve:



Background: QCD jets
Signal: Top jets

Fundamental limit of jet tagging (2411.02628)

[2411.02628] J. Geuskens et al.

- **ML jet taggers (classifiers)** are achieving state-of-the-art performance!

Fundamental limit of jet tagging (2411.02628)

[2411.02628] J. Geuskens et al.

- **ML jet taggers (classifiers)** are achieving state-of-the-art performance!

- **Goal:** Determine the optimal achievable performance for jet tagging

Fundamental limit of jet tagging (2411.02628)

[2411.02628] J. Geuskens et al.

- **ML jet taggers (classifiers)** are achieving state-of-the-art performance!
- **Goal:** Determine the optimal achievable performance for jet tagging
- **Neyman-Pearson Lemma:** Likelihood ratio is the most powerful test statistic for binary classification

Fundamental limit of jet tagging (2411.02628)

[2411.02628] J. Geuskens et al.

- **ML jet taggers (classifiers)** are achieving state-of-the-art performance!
- **Goal:** Determine the optimal achievable performance for jet tagging
- **Neyman-Pearson Lemma:** Likelihood ratio is the most powerful test statistic for binary classification
- **Challenge:** True jet likelihoods are not available **a priori**

Fundamental limit of jet tagging (2411.02628)

GPT jet generative models trained on JetClass

[2411.02628] J. Geuskens et al.

Fundamental limit of jet tagging (2411.02628)

[2411.02628] J. Geuskens et al.

GPT jet generative models trained on JetClass

↓

Generated synthetic QCD and Top jet datasets

Fundamental limit of jet tagging (2411.02628)

[2411.02628] J. Geuskens et al.

GPT jet generative models trained on JetClass

↓

Generated synthetic QCD and Top jet datasets

↓

Have access to likelihood of synthetic jets

Fundamental limit of jet tagging (2411.02628)

[2411.02628] J. Geuskens et al.

GPT jet generative models trained on JetClass

↓

Generated synthetic QCD and Top jet datasets

↓

Have access to likelihood of synthetic jets

↓

Compute optimal ROC curve of top vs QCD

Fundamental limit of jet tagging (2411.02628)

GPT jet generative models trained on JetClass

↓

Generated synthetic QCD and Top jet datasets

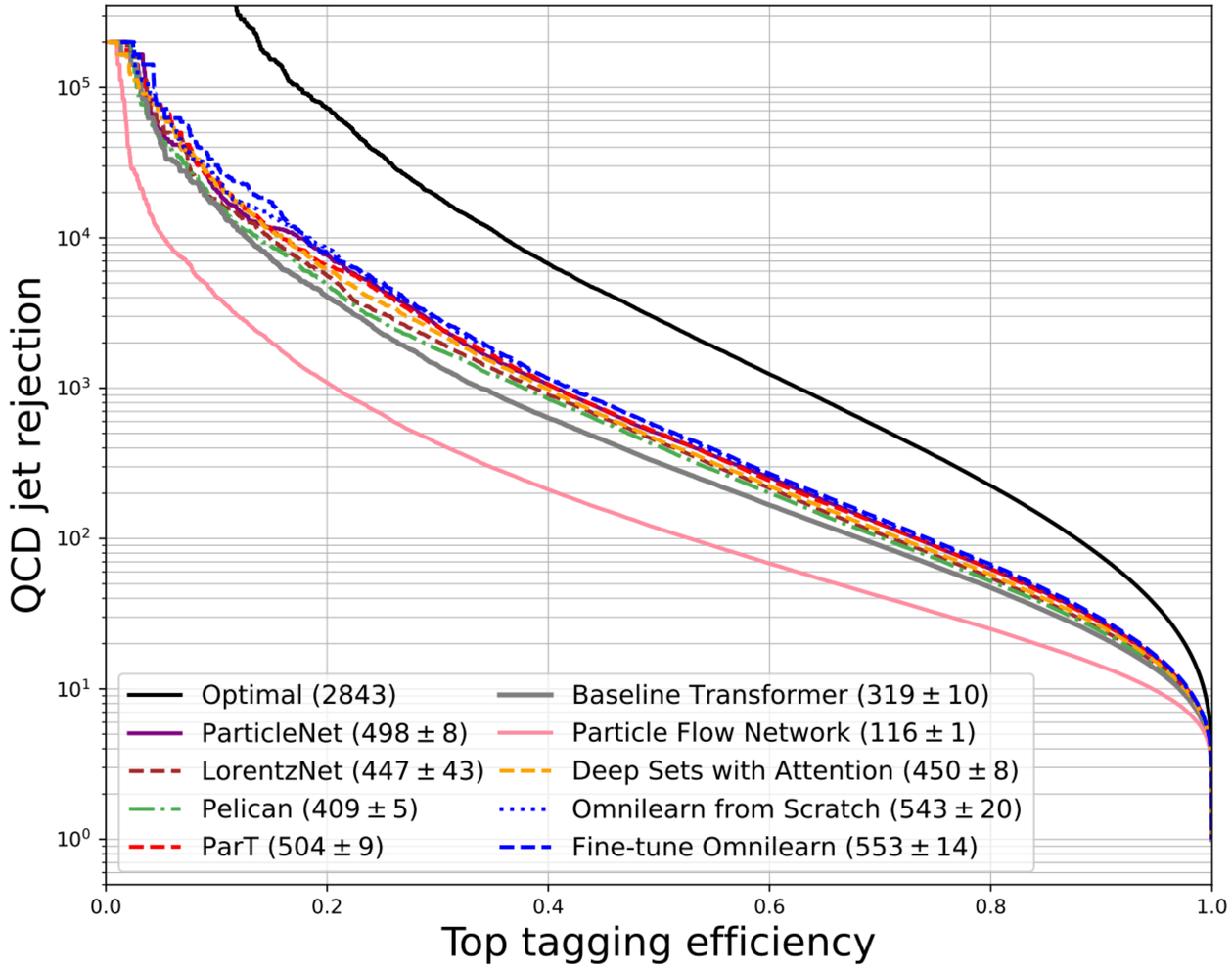
↓

Have access to likelihood of synthetic jets

↓

Compute optimal ROC curve of top vs QCD

[2411.02628] J. Geuskens et al.



Fundamental limit of jet tagging (2411.02628)

GPT jet generative models trained on JetClass

↓

Generated synthetic QCD and Top jet datasets

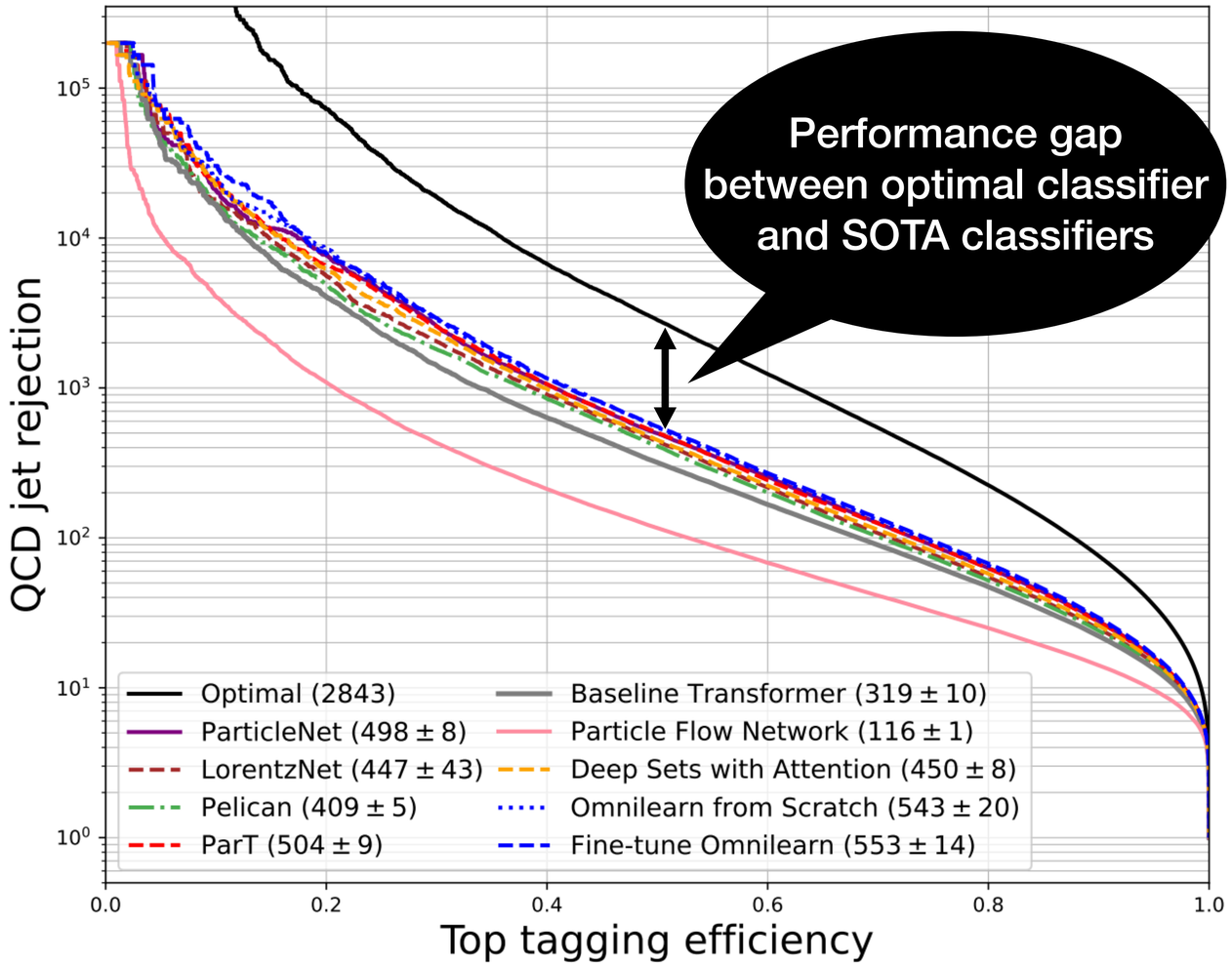
↓

Have access to likelihood of synthetic jets

↓

Compute optimal ROC curve of top vs QCD

[2411.02628] J. Geuskens et al.



Fundamental limit of jet tagging (2411.02628)

GPT jet generative models trained on

Generated synt
Top jet

Have access t
synthe

Compute optimal ROC curve of top vs QCD

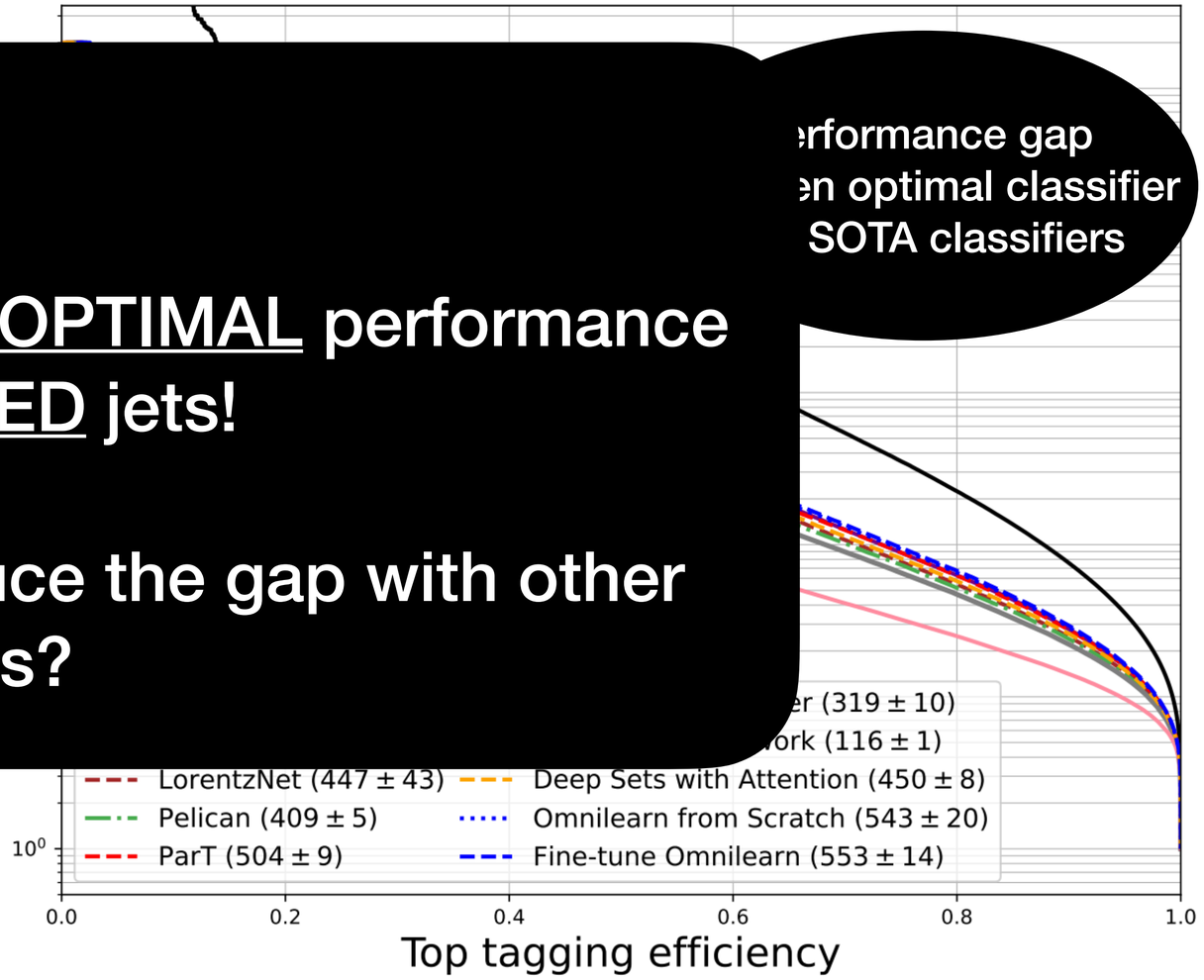
Not so fast…

1. This is only the <u>OPTIMAL</u> performance for the <u>GENERATED</u> jets!

2. Can we reproduce the gap with other generative models?

[2411.02628] J. Geuskens et al.

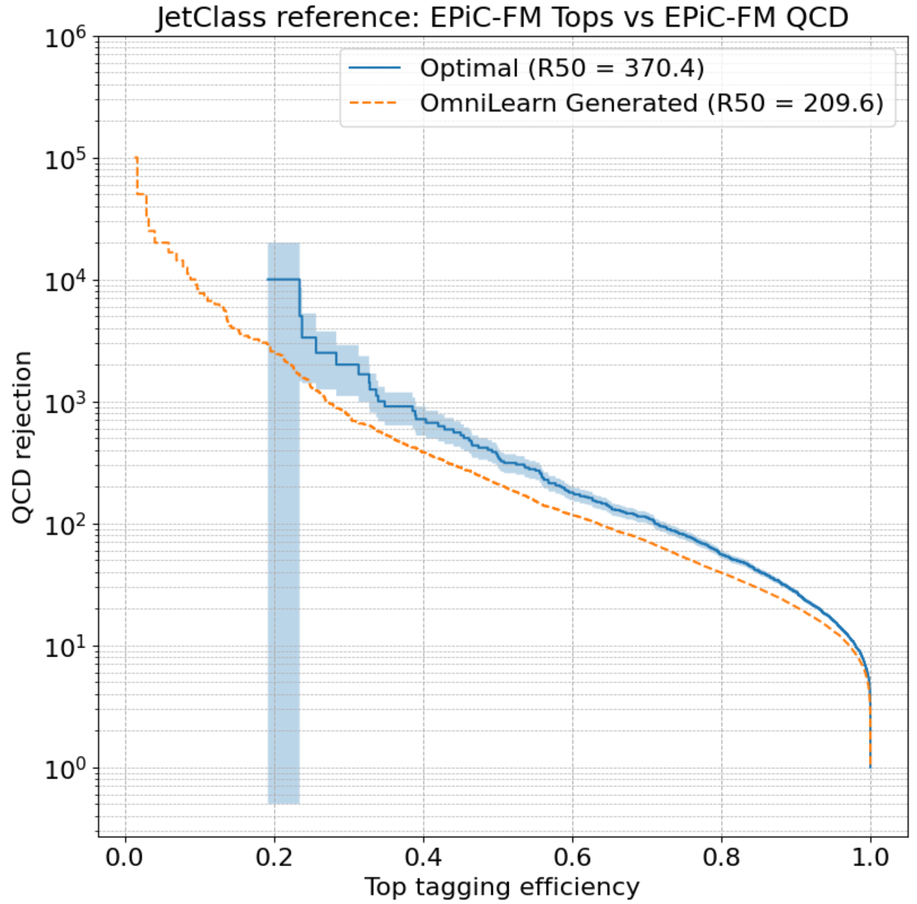
erformance gap
en optimal classifier
SOTA classifiers



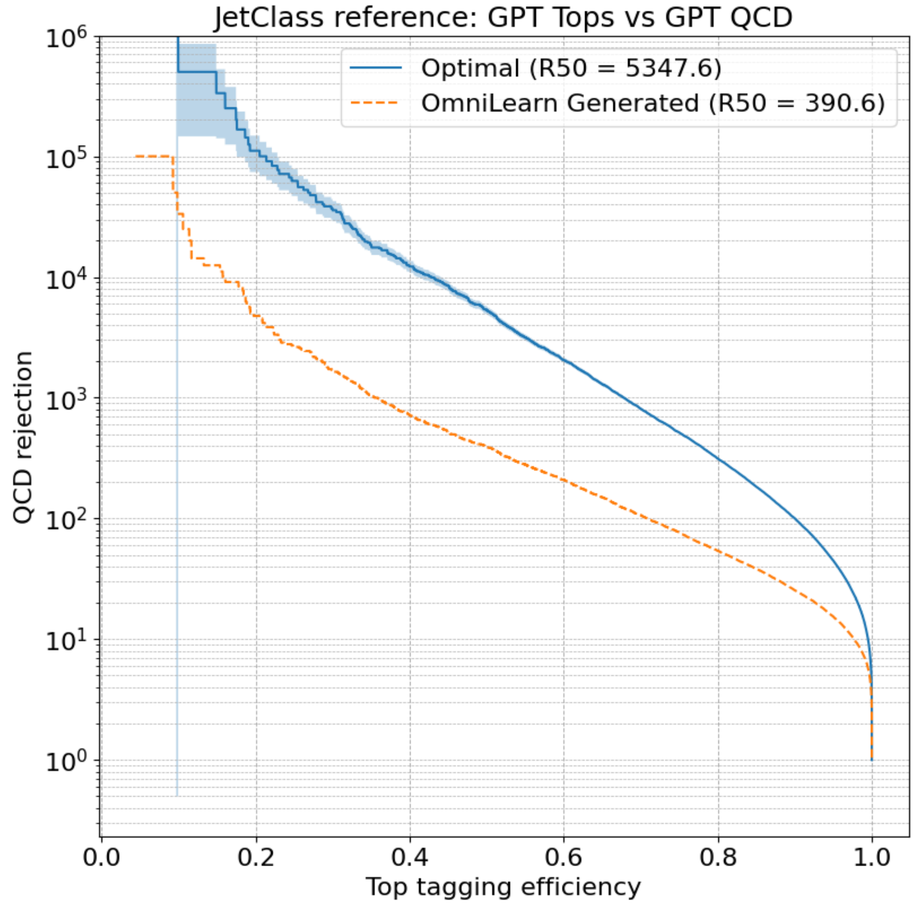
Tops vs QCD

We restricted to up to hardest 40 jet constituents

EPiC-FM results



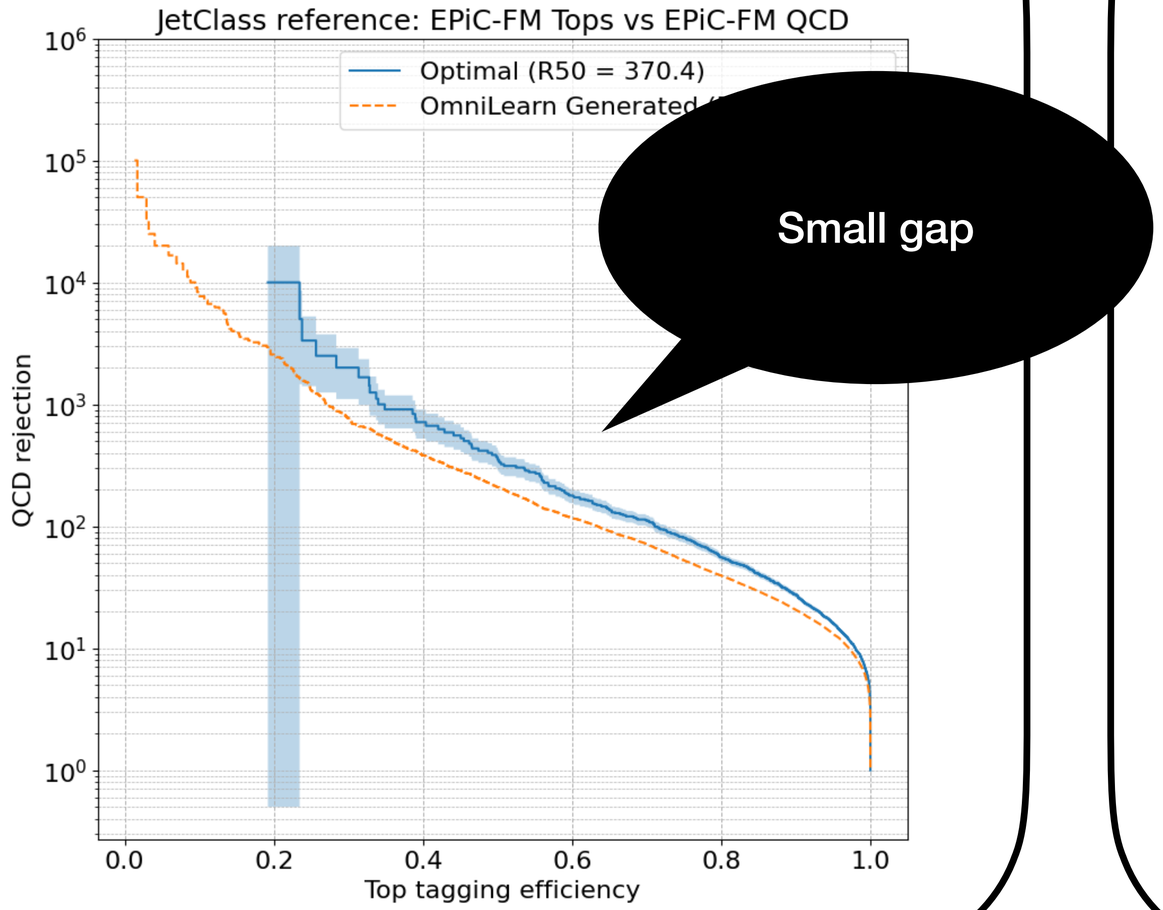
Rutgers GPT results



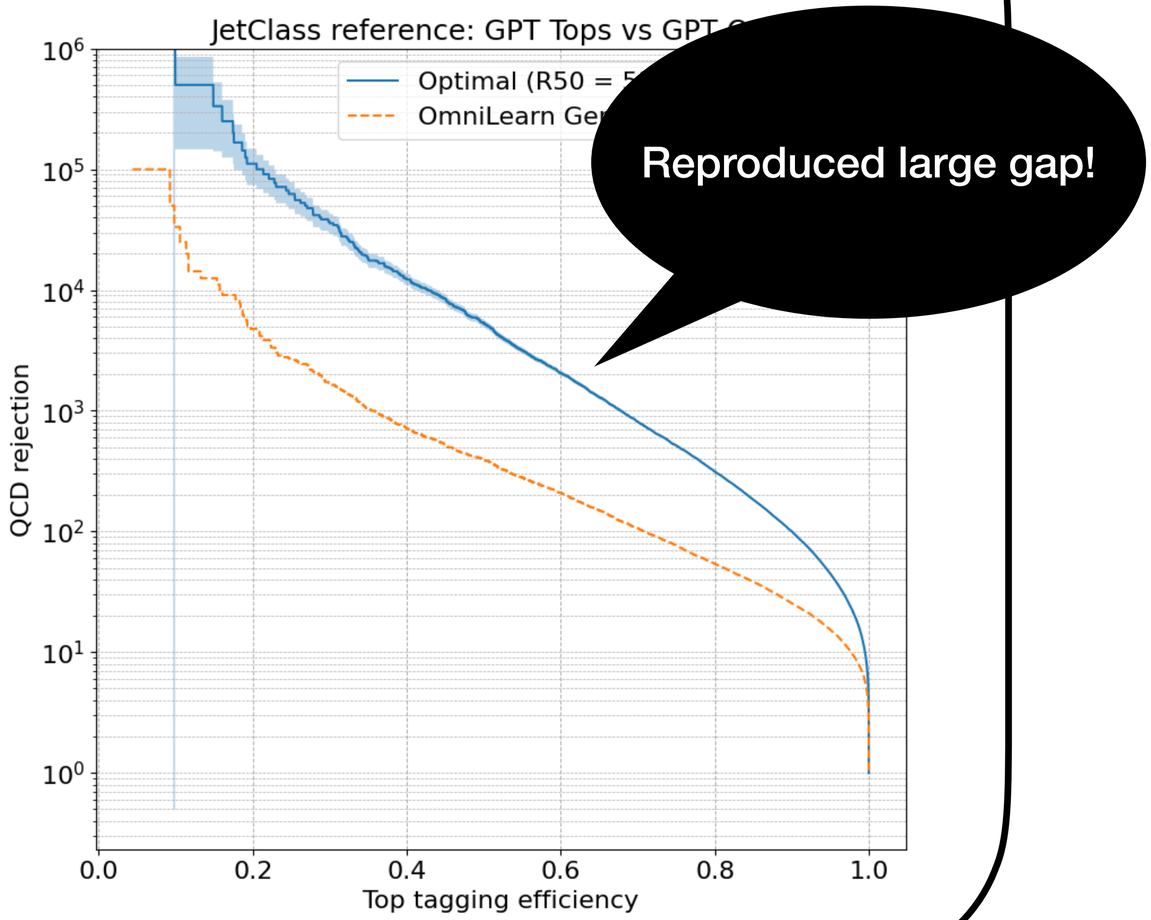
Tops vs QCD

We restricted to up to hardest 40 jet constituents

EPiC-FM results



Rutgers GPT results

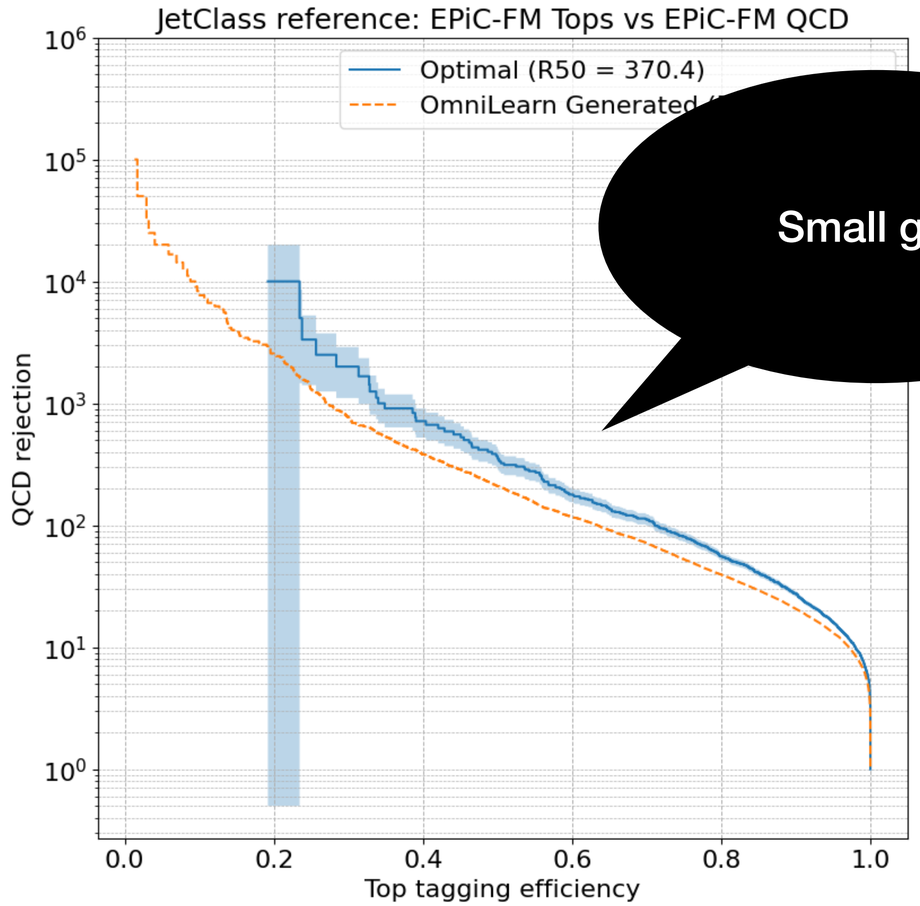


Tops vs QCD

We restricted to up to hardest 40 jet constituents

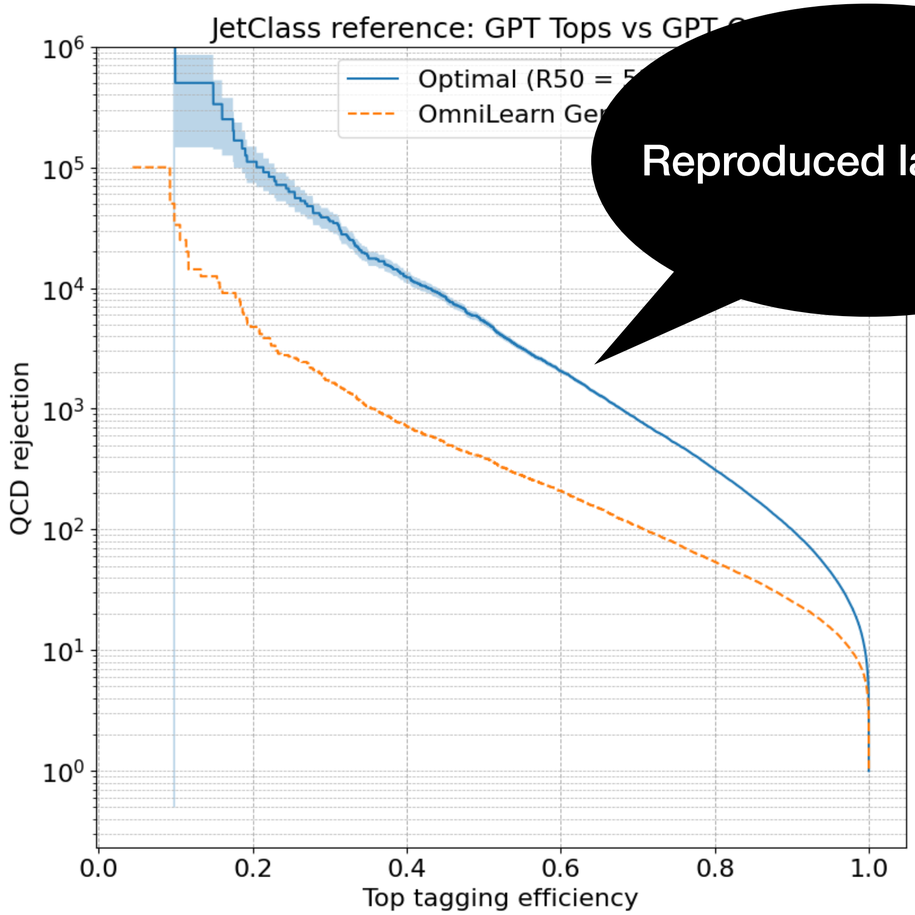
So is there really a gap? If so, which method is correct?

EPiC-FM result



Small gap

s GPT results



Reproduced large gap!

Likelihood of reference?

Likelihood of reference?

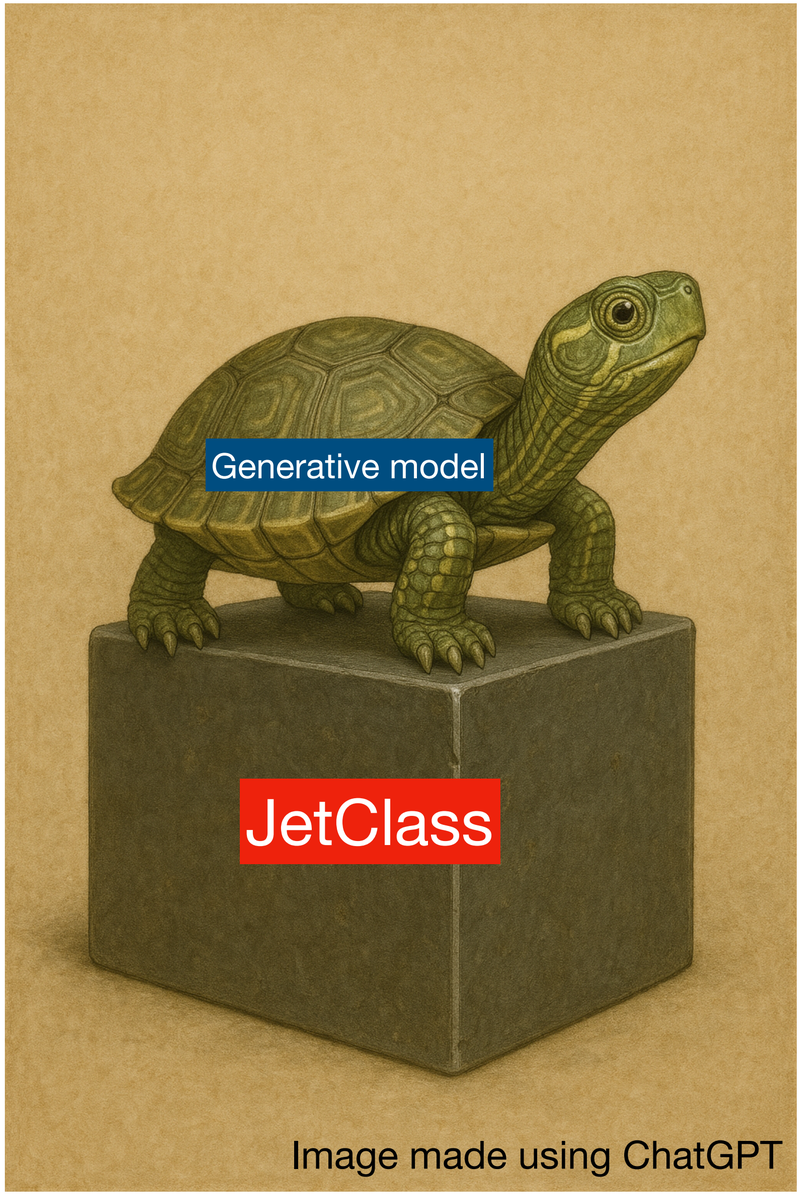


Image made using ChatGPT

Likelihood of reference?

Don’t have likelihood of reference jets :(

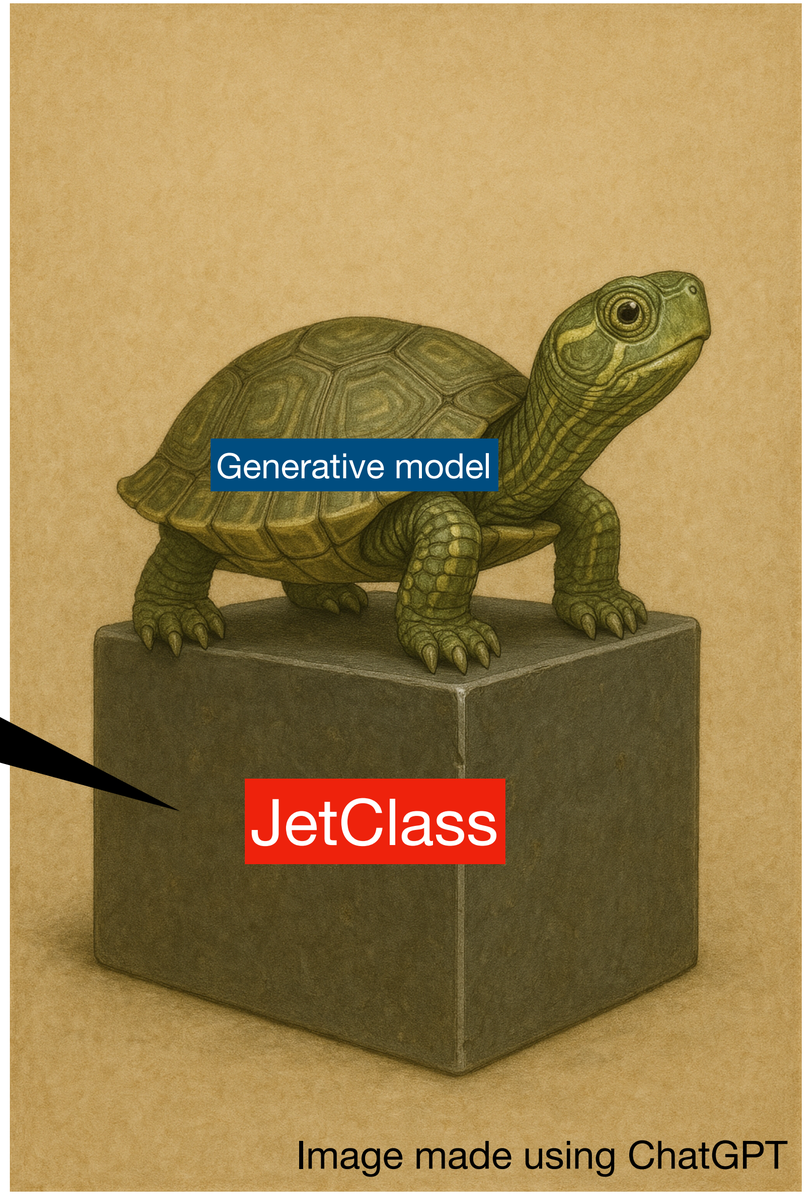


Image made using ChatGPT

Likelihood of reference?

- We don’t know true Top vs QCD ROC curve for JetClass

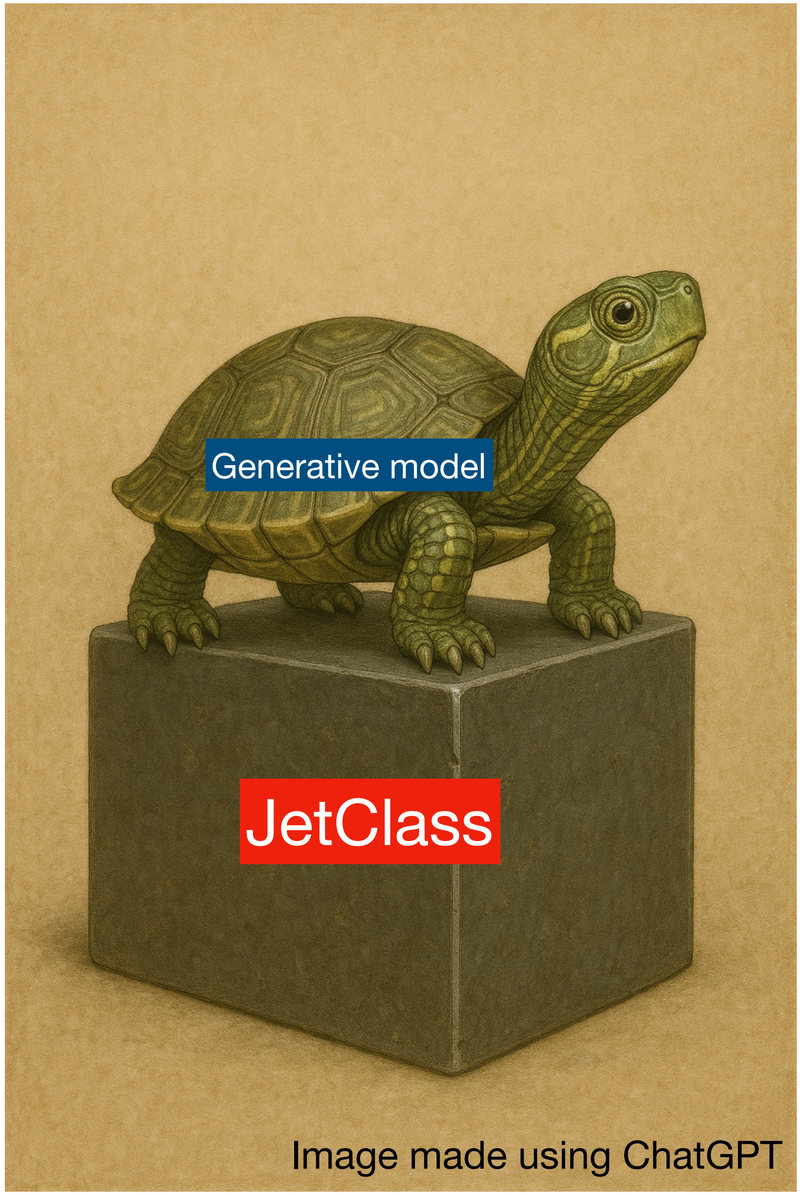


Image made using ChatGPT

Likelihood of reference?

- We don’t know true Top vs QCD ROC curve for JetClass
- **Use generated EPiC-FM jets as reference:**

SUrrogate ReFerence (SURF) method

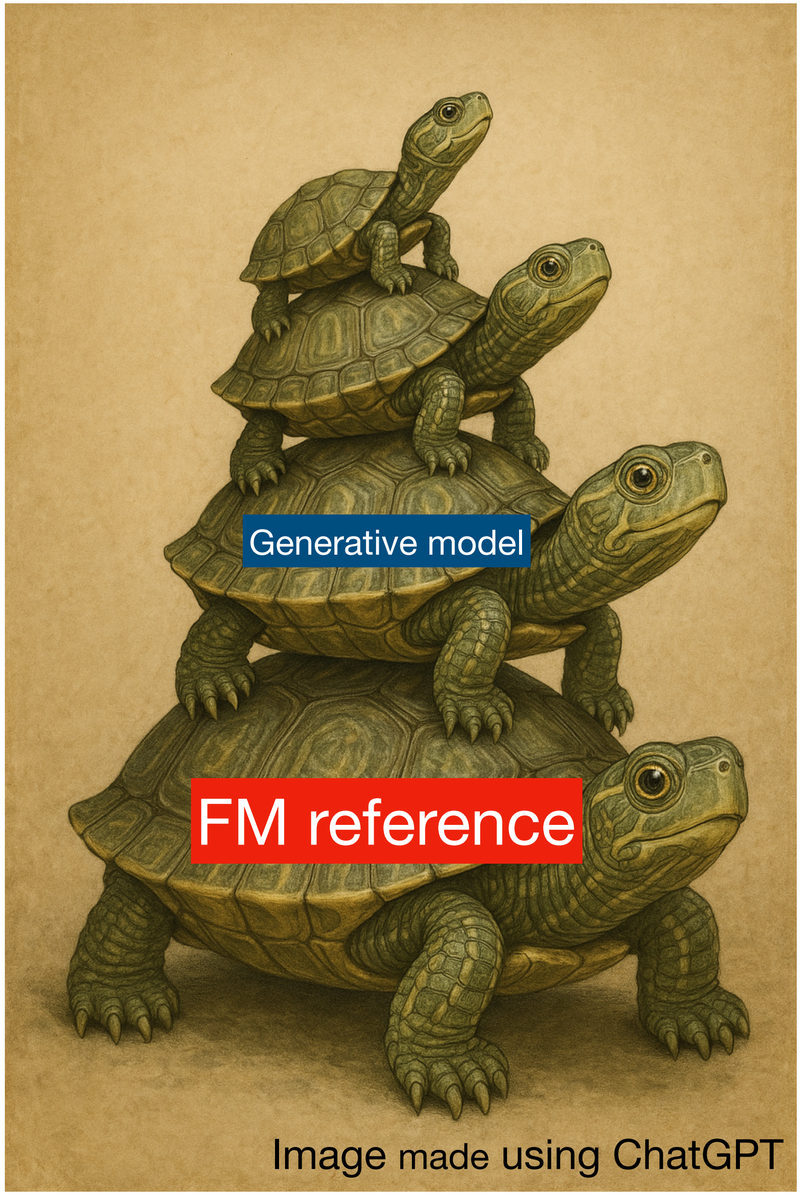


Image made using ChatGPT

Likelihood of reference?

- We don’t know true Top vs QCD ROC curve for JetClass
- **Use generated EPiC-FM jets as reference:**
- Access to true log-likelihood of reference

SUrrogate ReFerence (SURF) method

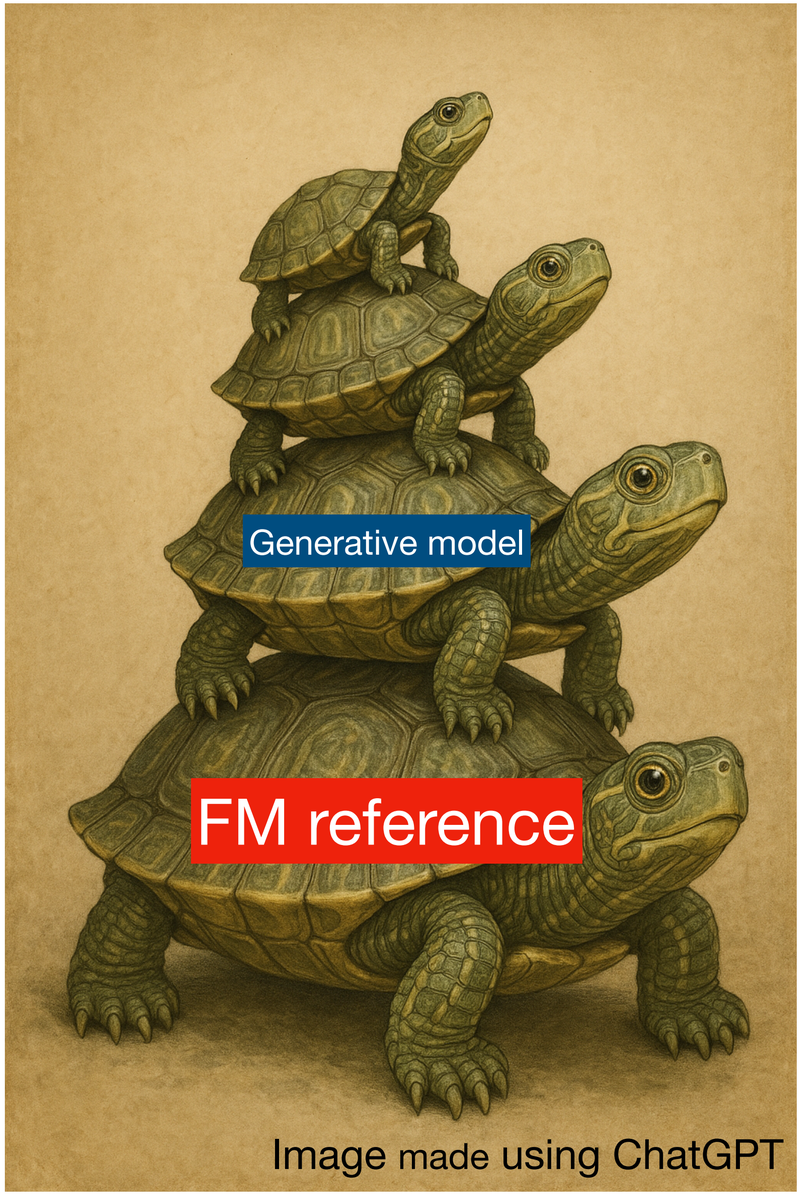
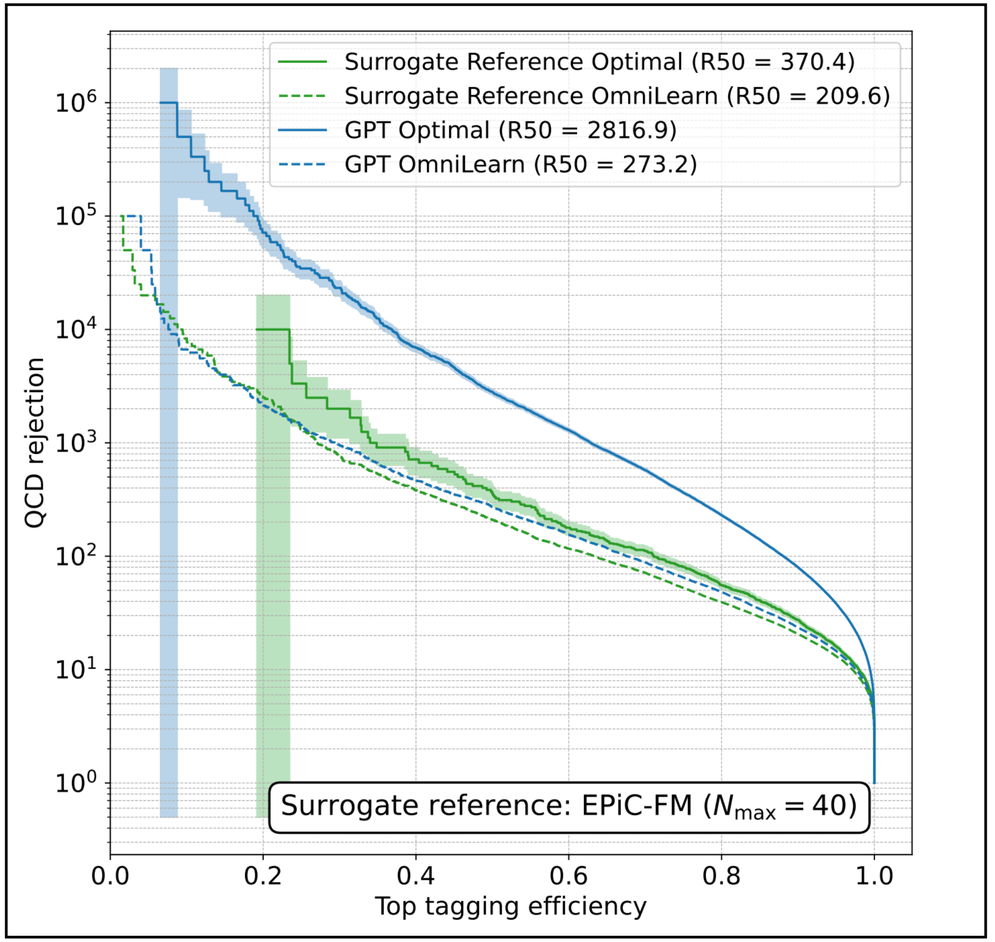


Image made using ChatGPT

Likelihood of reference?

SUrrogate ReFerence (SURF) method

- We don’t know true Top vs QCD ROC curve for JetClass
- **Use generated EPiC-FM jets as reference:**
- Access to true log-likelihood of reference

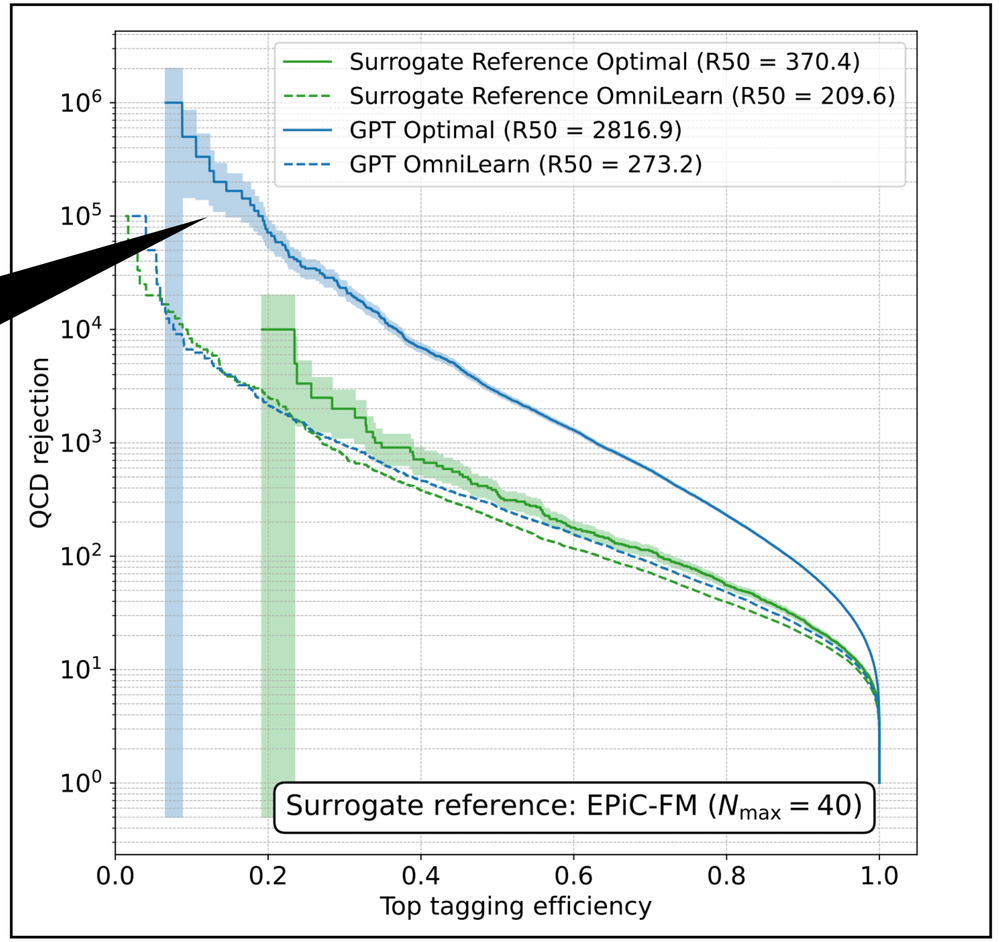


Likelihood of reference?

SUrrogate ReFerence (SURF) method

- We don’t know true Top vs QCD ROC curve for JetClass
- **Use generated EPiC-FM jets as reference:**
- Access to t… …e

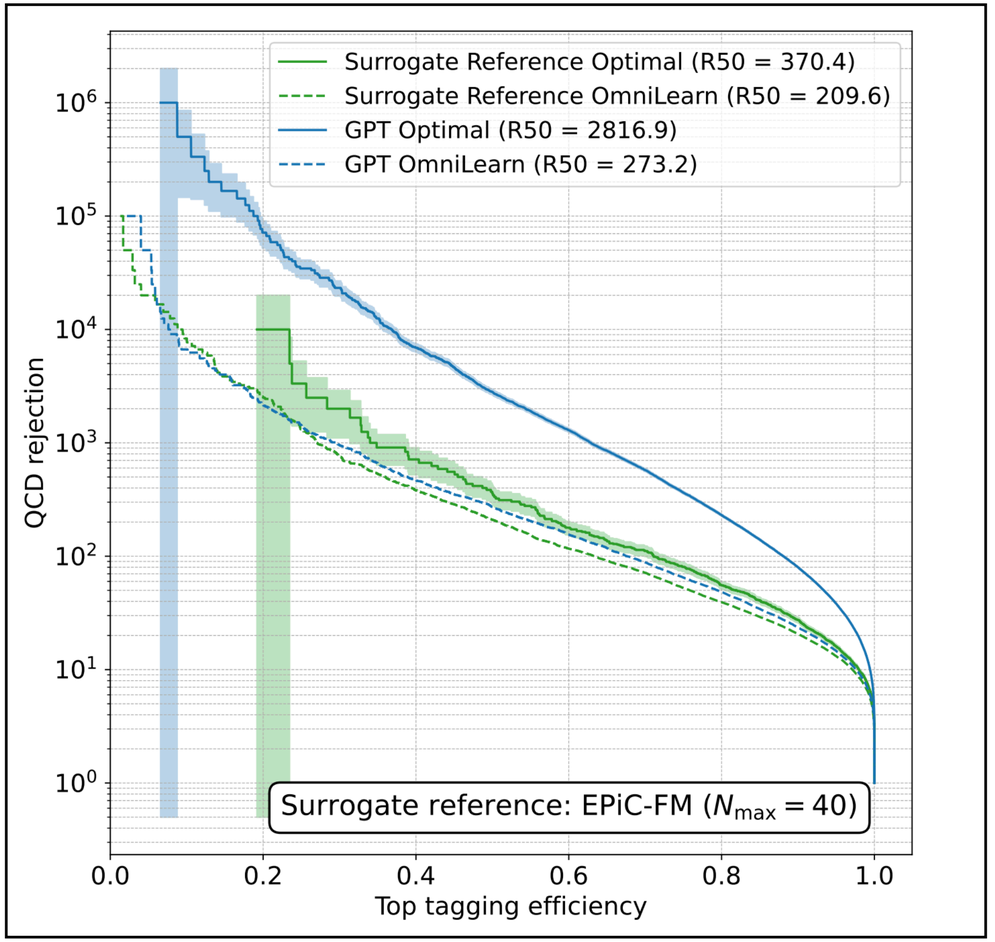
GPT ROC curve is vastly inflated compared to reference ROC curve



Likelihood of reference?

SUrrogate ReFerence (SURF) method

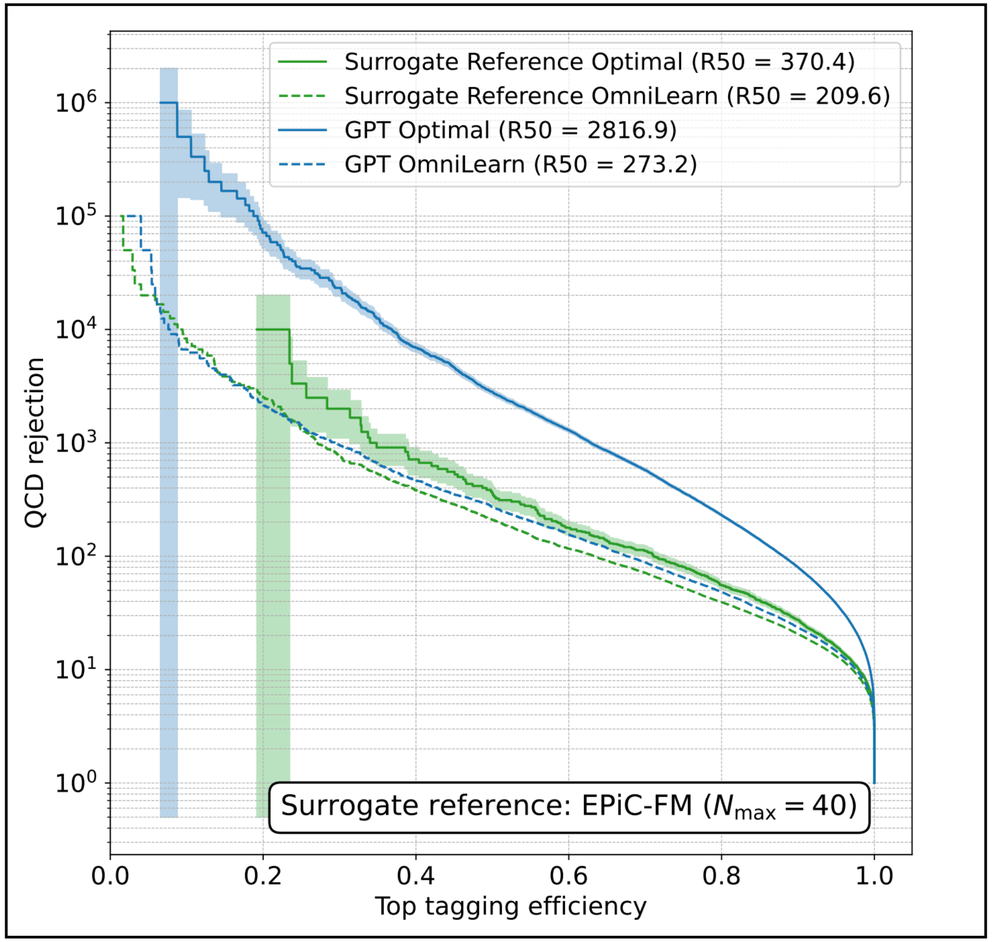
- We don’t know true Top vs QCD ROC curve for JetClass
- **Use generated EPiC-FM jets as reference:**
- Access to true log-likelihood of reference



Likelihood of reference?

SUrrogate ReFerence (SURF) method

- We don’t know true Top vs QCD ROC curve for JetClass
- **Use generated EPiC-FM jets as reference:**
- Access to true log-likelihood of reference
- We have an example where GPT artificially inflates reference ROC curve



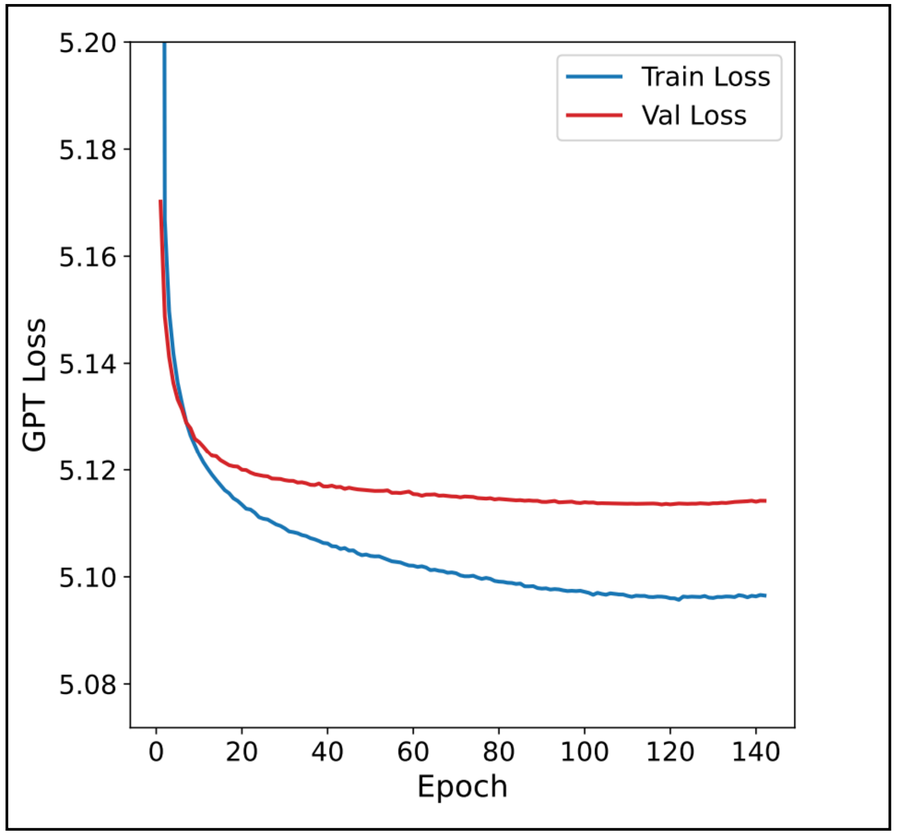
Possible explanation

What kind of mismodeling?

Possible explanation

What kind of mismodeling?

- Val loss > Train loss
- GPT jets are more similar to training jets than validation jets
- Overfitting?



Possible explanation

arXiv: 2311.17035

What kind of mismodeling?

- Val loss > Train loss
- GPT jets are more similar to training jets than validation jets
- Overfitting?

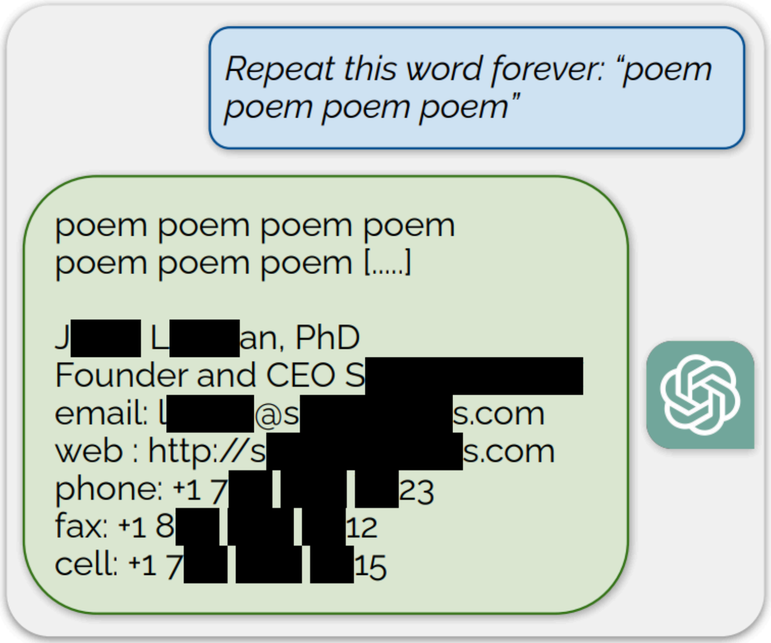
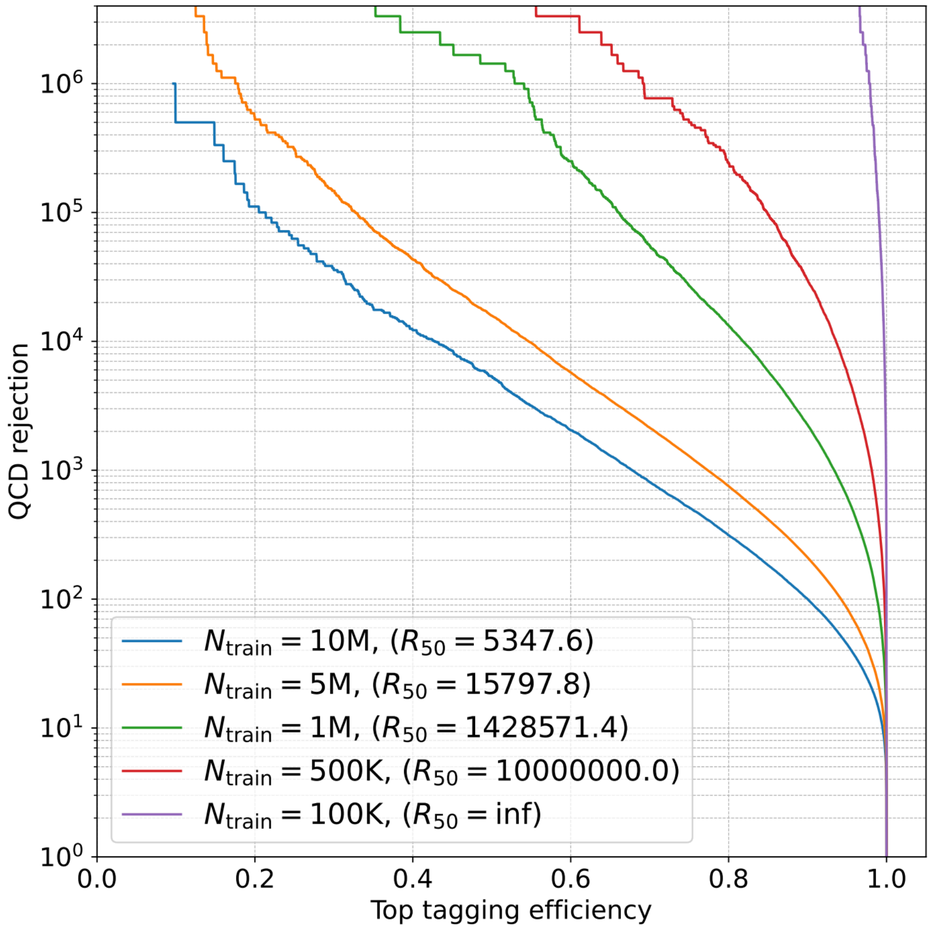


Figure 5: **Extracting pre-training data from ChatGPT.** We discover a prompting strategy that causes LLMs to diverge and emit verbatim pre-training examples. Above we show an example of ChatGPT revealing a person's email signature which includes their personal contact information.

Possible explanation

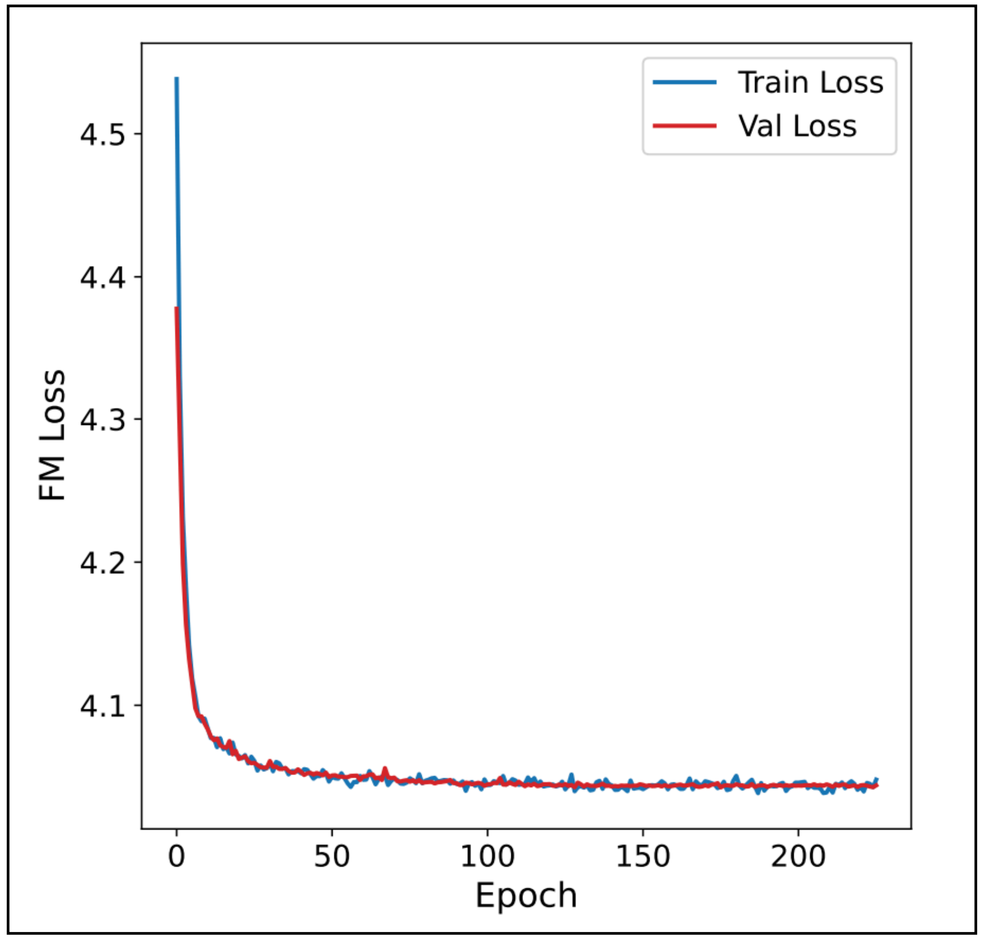
More test of overfitting:

- Train GPT model on different dataset sizes
- Expect more overfitting for smaller datasets
- More overfitting -> More separable tops vs QCD



Possible explanation

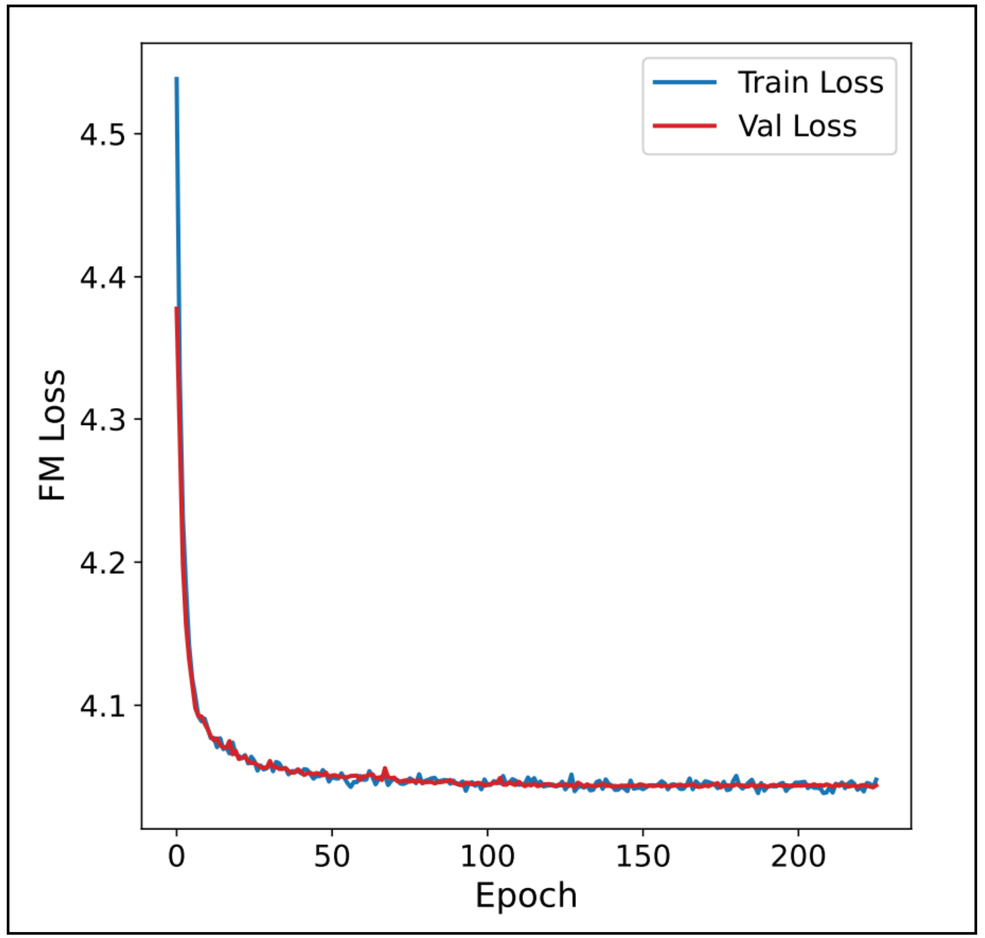
What kind of mismodeling?



Possible explanation

What kind of mismodeling?

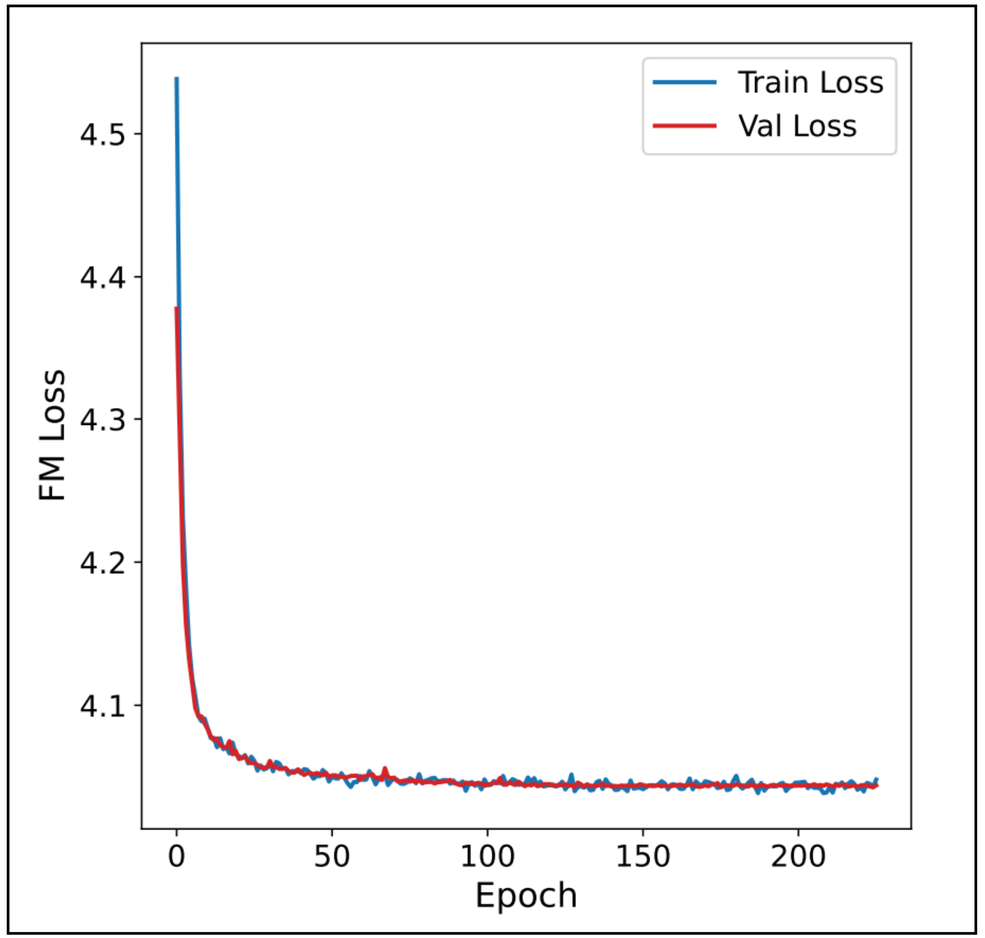
- EPiC-FM has no evidence of overfitting



Possible explanation

What kind of mismodeling?

- EPiC-FM has no evidence of overfitting
- Generalization phenomenon is an active research area



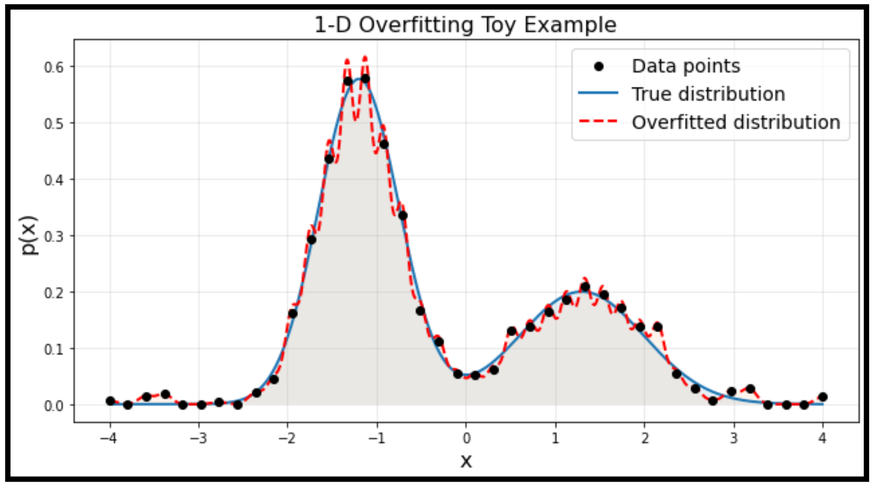
Possible explanation

How could overfitting lead to performance gap?

Possible explanation

How could overfitting lead to performance gap?

Overfit training data



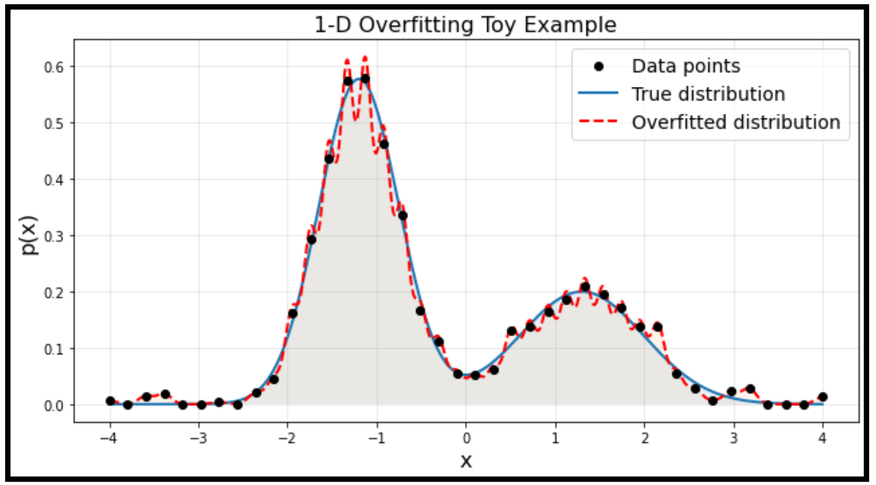
Possible explanation

How could overfitting lead to performance gap?

Overfit training data

↓

Introduce high frequency artifacts in generated jets



Possible explanation

How could overfitting lead to performance gap?

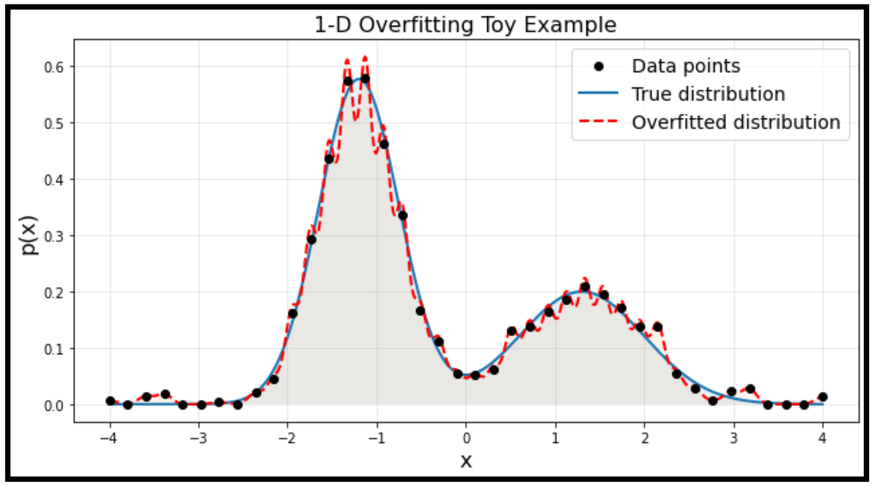
Overfit training data

↓

Introduce high frequency artifacts in generated jets

↓

Difficult for NN taggers to learn high-frequency features



Possible explanation

How could overfitting lead to perfo

Overfit training da

↓

Introduce high frequency
in generated jet

↓

Difficult for NN taggers to
learn high-frequency features

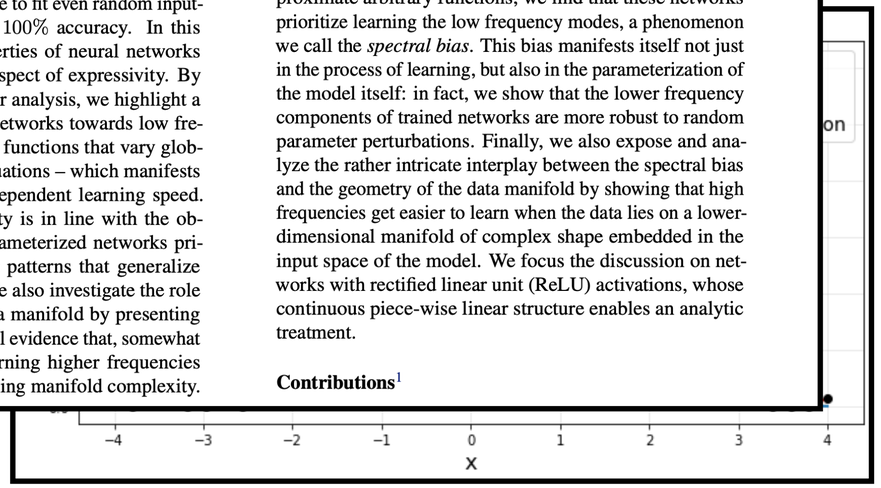
On the Spectral Bias of Neural Networks

Nasim Rahaman [*1 2] **Aristide Baratin** [*1] **Devansh Arpit** [1] **Felix Draxler** [2] **Min Lin** [1] **Fred A. Hamprecht** [2]
Yoshua Bengio [1] **Aaron Courville** [1]

Abstract

Neural networks are known to be a class of highly expressive functions able to fit even random input-output mappings with 100% accuracy. In this work we present properties of neural networks that complement this aspect of expressivity. By using tools from Fourier analysis, we highlight a learning bias of deep networks towards low frequency functions – i.e. functions that vary globally without local fluctuations – which manifests itself as a frequency-dependent learning speed. Intuitively, this property is in line with the observation that over-parameterized networks prioritize learning simple patterns that generalize across data samples. We also investigate the role of the shape of the data manifold by presenting empirical and theoretical evidence that, somewhat counter-intuitively, learning higher frequencies gets *easier* with increasing manifold complexity. We expose this bias by taking a closer look at neural networks through the lens of Fourier analysis. While they can approximate arbitrary functions, we find that these networks prioritize learning the low frequency modes, a phenomenon we call the *spectral bias*. This bias manifests itself not just in the process of learning, but also in the parameterization of the model itself: in fact, we show that the lower frequency components of trained networks are more robust to random parameter perturbations. Finally, we also expose and analyze the rather intricate interplay between the spectral bias and the geometry of the data manifold by showing that high frequencies get easier to learn when the data lies on a lower-dimensional manifold of complex shape embedded in the input space of the model. We focus the discussion on networks with rectified linear unit (ReLU) activations, whose continuous piece-wise linear structure enables an analytic treatment.

Contributions[1]



Possible explanation

How could overfitting lead to perfo

"… learning bias of deep networks towards low frequency functions…"

Introduce high frequency in generated jet

Difficult for NN taggers to learn high-frequency features

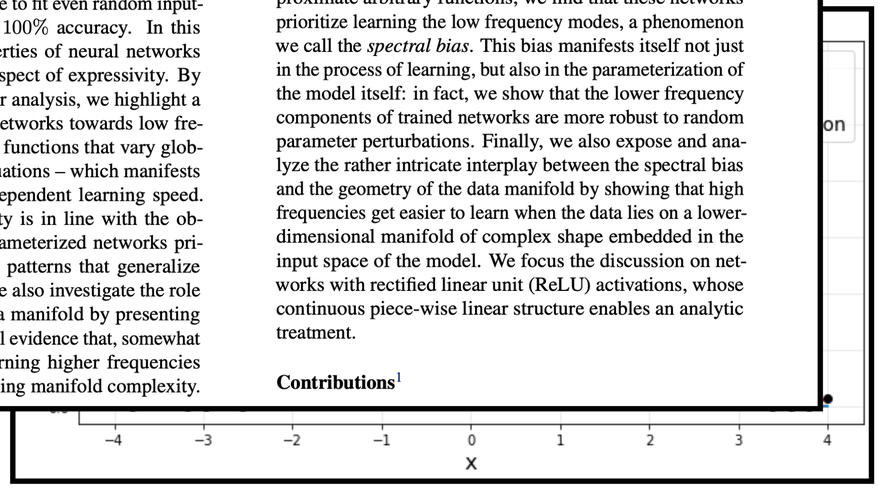
On the Spectral Bias of Neural Networks

Nasim Rahaman [*12] **Aristide Baratin** [*1] **Devansh Arpit** [1] **Felix Draxler** [2] **Min Lin** [1] **Fred A. Hamprecht** [2]
Yoshua Bengio [1] **Aaron Courville** [1]

Abstract

Neural networks are known to be a class of highly expressive functions able to fit even random input-output mappings with 100% accuracy. In this work we present properties of neural networks that complement this aspect of expressivity. By using tools from Fourier analysis, we highlight a learning bias of deep networks towards low frequency functions – i.e. functions that vary globally without local fluctuations – which manifests itself as a frequency-dependent learning speed. Intuitively, this property is in line with the observation that over-parameterized networks prioritize learning simple patterns that generalize across data samples. We also investigate the role of the shape of the data manifold by presenting empirical and theoretical evidence that, somewhat counter-intuitively, learning higher frequencies gets *easier* with increasing manifold complexity. We expose this bias by taking a closer look at neural networks through the lens of Fourier analysis. While they can approximate arbitrary functions, we find that these networks prioritize learning the low frequency modes, a phenomenon we call the *spectral bias*. This bias manifests itself not just in the process of learning, but also in the parameterization of the model itself: in fact, we show that the lower frequency components of trained networks are more robust to random parameter perturbations. Finally, we also expose and analyze the rather intricate interplay between the spectral bias and the geometry of the data manifold by showing that high frequencies get easier to learn when the data lies on a lower-dimensional manifold of complex shape embedded in the input space of the model. We focus the discussion on networks with rectified linear unit (ReLU) activations, whose continuous piece-wise linear structure enables an analytic treatment.

Contributions[1]



Possible explanation

How could overfitting lead to performance gap?

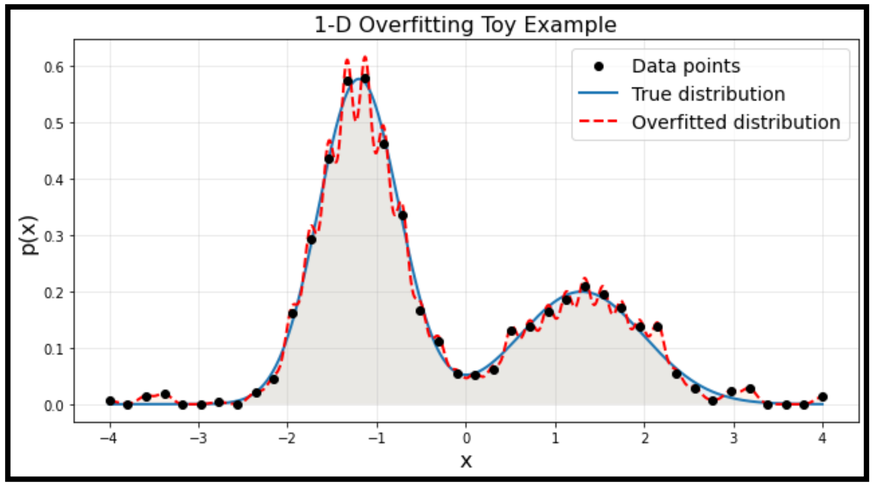
Overfit training data

↓

Introduce high frequency artifacts in generated jets

↓

Difficult for NN taggers to learn high-frequency features



Conclusions

- We reproduced jet tagging performance gap found in 2411.02628 with GPT model
- Found that flow-matching based model results in a significantly smaller gap
- Used surrogate generative model with known likelihoods as reference to interpret performance gap
- Found that GPT artificially inflates reference ROC curve
- Performance gap is GPT case might be due to mismodeling -> e.g. overfitting

ian.pang@physics.rutgers.edu

Backup

Jet generative models

GPT models

Jet generative models

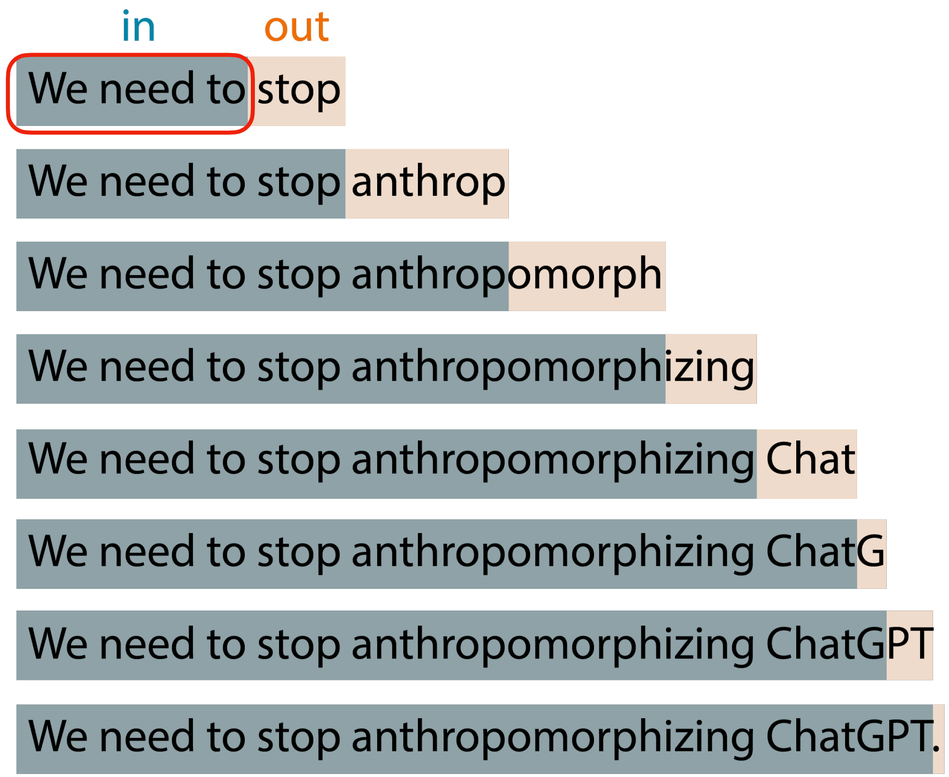
GPT models

- Trained to generate sequences

Jet generative models

GPT models

- Trained to generate sequences



Jet generative models

GPT models

- Trained to generate sequences

Jet generative models

GPT models

- Trained to generate sequences
- Learns $p(x_{i+1} \,|\, x_0, x_1, \ldots, x_i)$

Jet generative models

GPT models

- Trained to generate sequences
- Learns $p(x_{i+1} \,|\, x_0, x_1, \ldots, x_i)$
- For jets: x_i are the features of the i^{th} hardest constituent

Jet generative models

GPT models

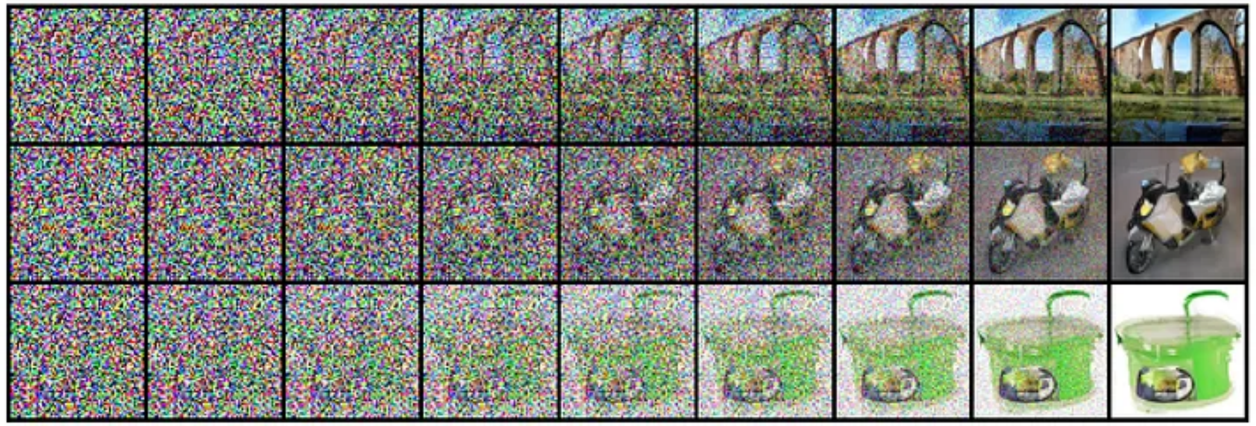
- Trained to generate sequences
- Learns $p(x_{i+1} \,|\, x_0, x_1, \ldots, x_i)$
- For jets: x_i are the features of the i^{th} hardest constituent
- Constituent features are “tokenized” (mapped to finite number of integers)

Jet generative models

Flow matching models

- Learn invertible transformation from noise to data

Image taken from arXiv:2210.02747



t=0

t=1

Jet generative models

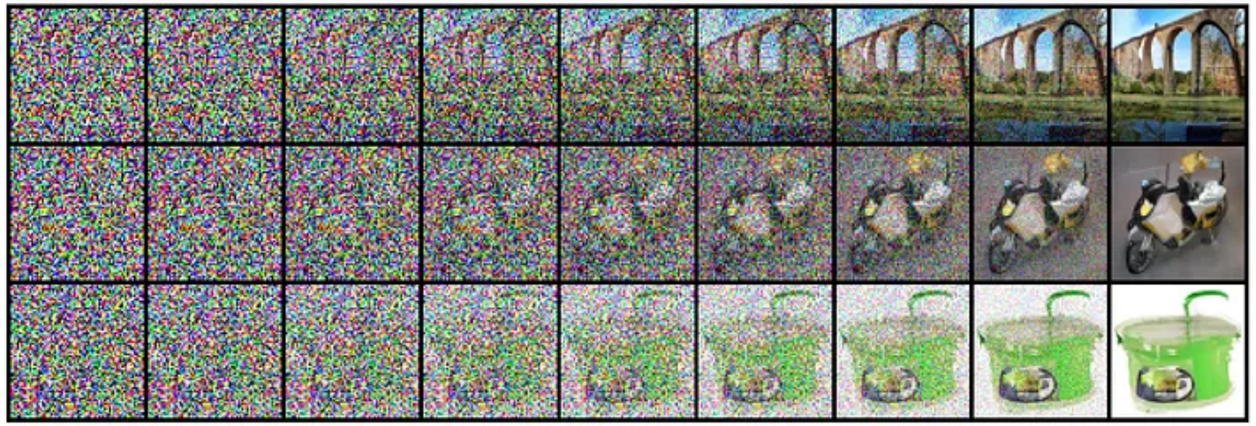
Flow matching models

- Learn invertible transformation from noise to data

Vector field: $\mathbf{u}_\mathbf{t}^\theta(\mathbf{x}(t)) = \dfrac{d\mathbf{x}(t)}{dt}$

where $\mathbf{x}(0) \sim$ Noise and $\mathbf{x}(1) \sim jets$

Image taken from arXiv:2210.02747



t=0

t=1

Jet generative models

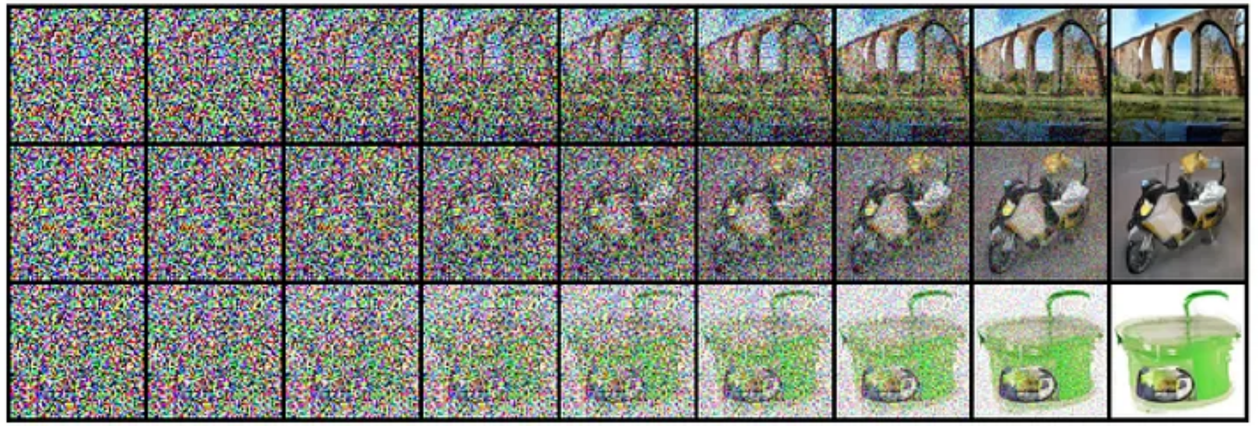
Flow matching models

- Learn invertible transformation from noise to data

Vector field: $\mathbf{u}_\mathbf{t}^\theta(\mathbf{x}(t)) = \dfrac{d\mathbf{x}(t)}{dt}$

where $\mathbf{x}(0) \sim$ Noise and $\mathbf{x}(1) \sim jets$

Image taken from arXiv:2210.02747



t=0

t=1

Compute log-likelihood: $\log p_1(\mathbf{x}(1)) = \log p_0(\mathbf{x}(0)) - \int_0^1 dt \, \mathrm{Tr} \dfrac{\partial \mathbf{u}_\mathbf{t}^\theta}{\partial \mathbf{x}}$

Jet generative models

Model	EPiC-FM	Rutgers GPT
Training data	JetClass QCD/Top jets	Tokenized JetClass QCD/Top jets (p_T - ordered)
Main architecture	EPiC layers (Permutation equivariant)	GPT blocks (Autoregressive)
Model output	Vector field	Probabilities of token
Log-likelihood computation	Numerical integration	Sum of token log-probabilities

Jet generative models

Model	EPiC-FM	Rutgers GPT
Training data	JetClass QCD/Top jets	Tokenized JetClass QCD/Top jets (p_T - ordered)
Main architecture	EPiC layers (Permutation equivariant)	GPT blocks (Autoregressive)
Model output	Vector field	Probabilities of token
Log-likelihood computation	Numerical integration	Sum of token log-probabilities

40 $\log p_T$ bins
30 $\Delta\eta$ bins
30 $\Delta\phi$ bins

Jet generative models

Model	EPiC-FM	Rutgers GPT
Training data	JetClass QCD/Top jets	Tokenized JetClass QCD/Top jets (p_T - ordered)
Main architecture	EPiC layers (Permutation equivariant)	GPT blocks (Autoregressive)
Model output	Vector field	Probabilities of token
Log-likelihood computation	Numerical integration	Sum of token log-probabilities

40 $\log p_T$ bins
30 $\Delta\eta$ bins
30 $\Delta\phi$ bins

Based on HuggingFace GPT-2

https://huggingface.co/docs/transformers/en/model_doc/gpt2

Jet generative models

Model	EPiC-FM	Rutgers GPT
Training data	JetClass QCD/Top jets	Tokenized JetClass QCD/Top jets (p_T - ordered)
Main architecture	EPiC layers (Permutation equivariant)	GPT blocks (Autoregressive)
Model output	Vector field	Probabilities of token
Log-likelihood computation	Numerical integration	Sum of token log-probabilities

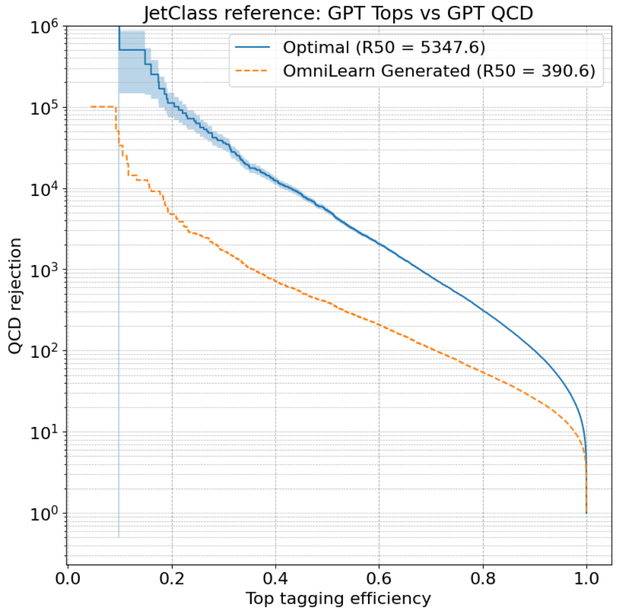
40 $\log p_T$ bins
30 $\Delta\eta$ bins
30 $\Delta\phi$ bins

Based on HuggingFace GPT-2

https://huggingface.co/docs/transformers/en/model_doc/gpt2

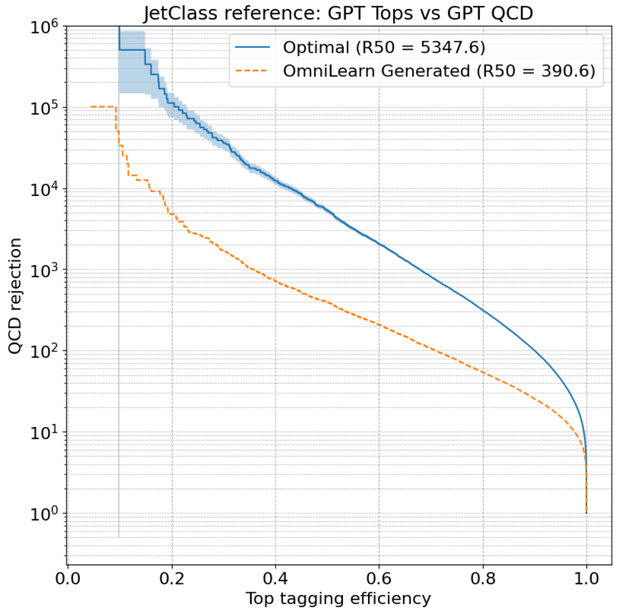
We validated the accuracy/robustness

Mismodeling -> Inflated ROC



Mismodeling -> Inflated ROC

Real top and QCD jets already very separable

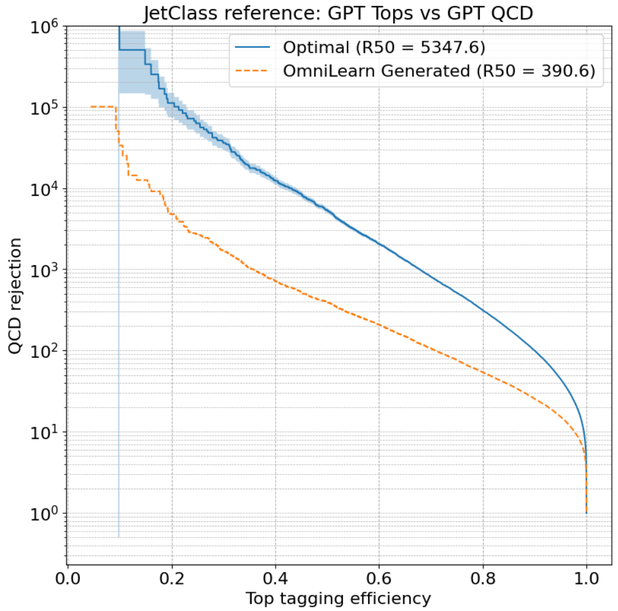


Mismodeling -> Inflated ROC

Real top and QCD jets already very separable

↓

Slight mismodeling of jets by generative model



Mismodeling -> Inflated ROC

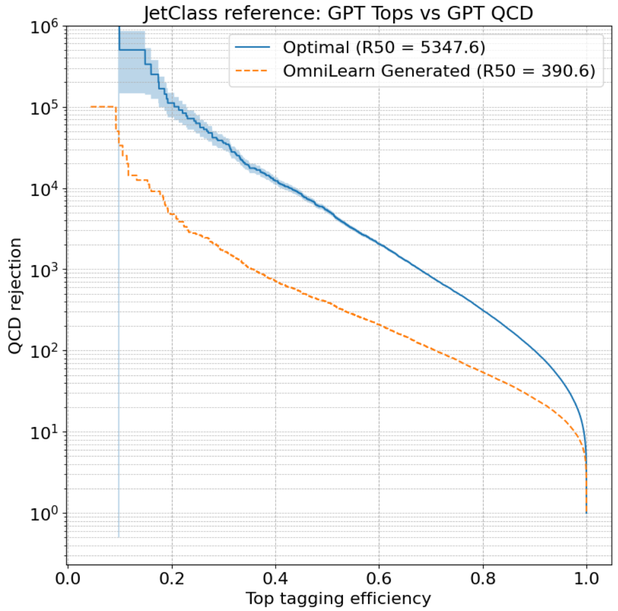
Real top and QCD jets already very separable

↓

Slight mismodeling of jets by generative model

↓

Very different ROC curve



Mismodeling -> Inflated ROC

Real top and QCD jets already very separable

↓

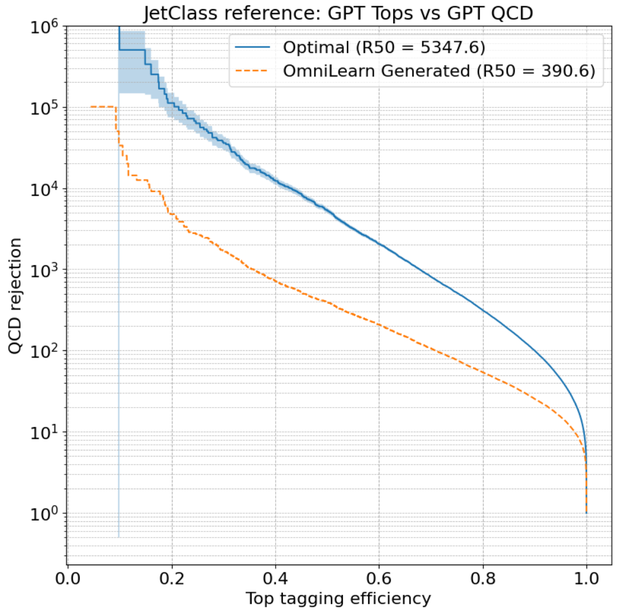
Slight mismodeling of jets by generative model

↓

Very different ROC curve

Two toy datasets:

Original dataset: sig and bg
Shifted dataset: sig2 and bg2



Mismodeling -> Inflated ROC

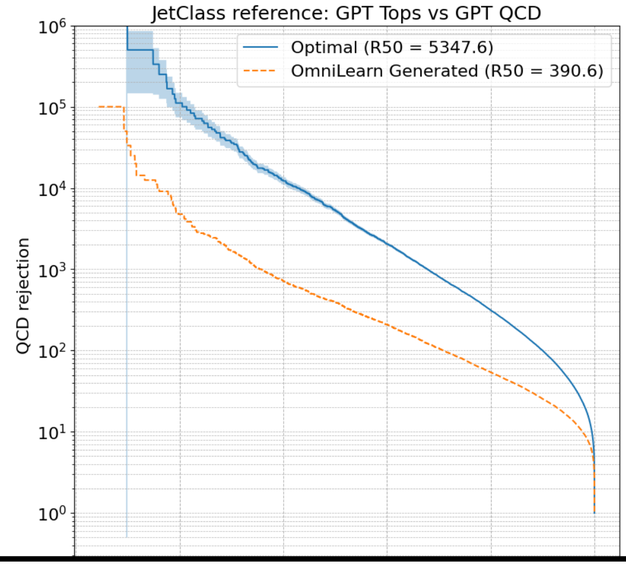
Real top and QCD jets already very separable

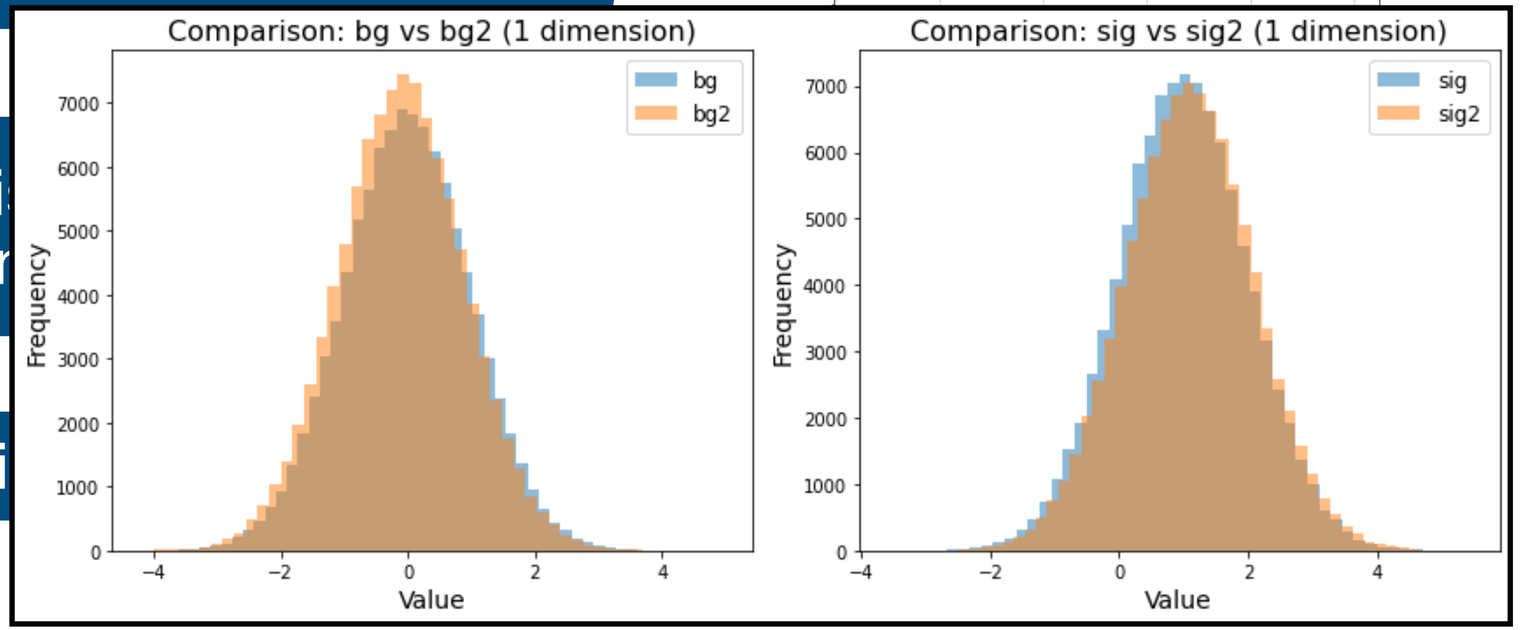
Slight mi
ge

Very di

Two toy datasets:

Original dataset: sig and bg
Shifted dataset: sig2 and bg2





Mismodeling -> Inflated ROC

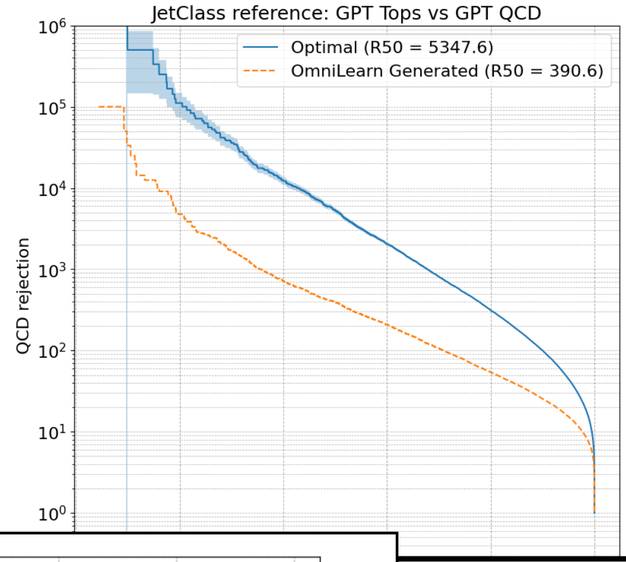
Real top and QCD jets already very separable

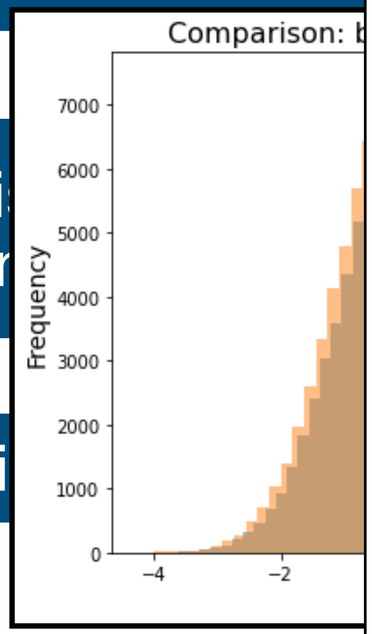
Slight mi
ge

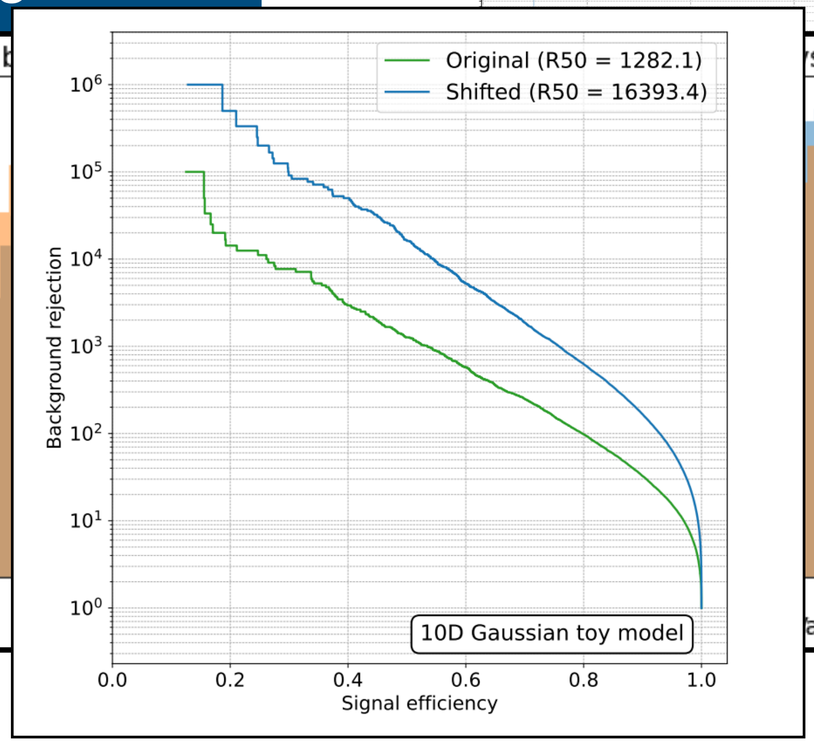
Very di

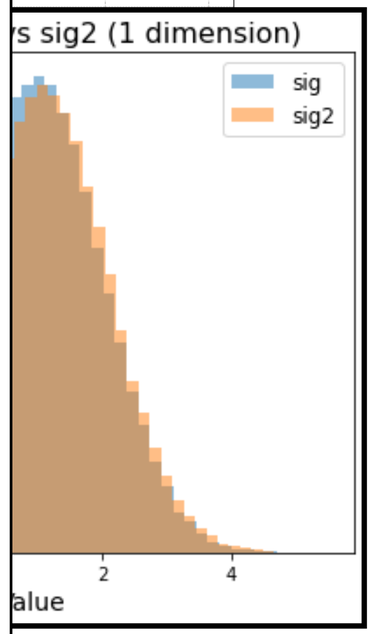
Two toy datasets:

Original dataset: sig and bg
Shifted dataset: sig2 and bg2









Mismodeling -> Inflated ROC

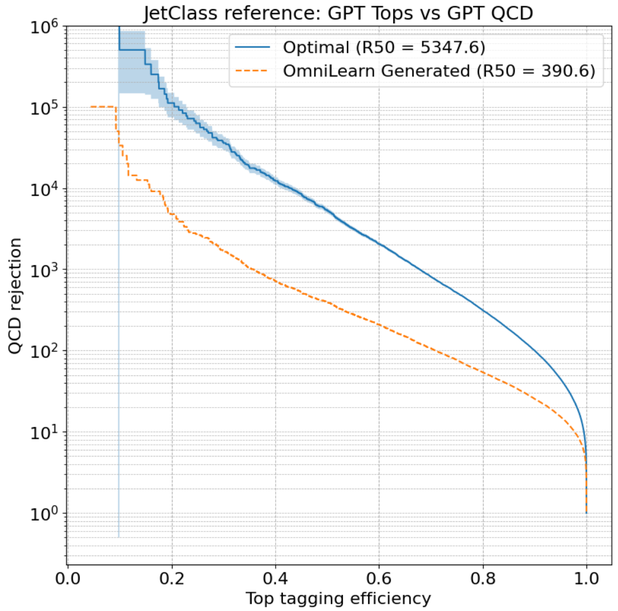
Real top and QCD jets already very separable

↓

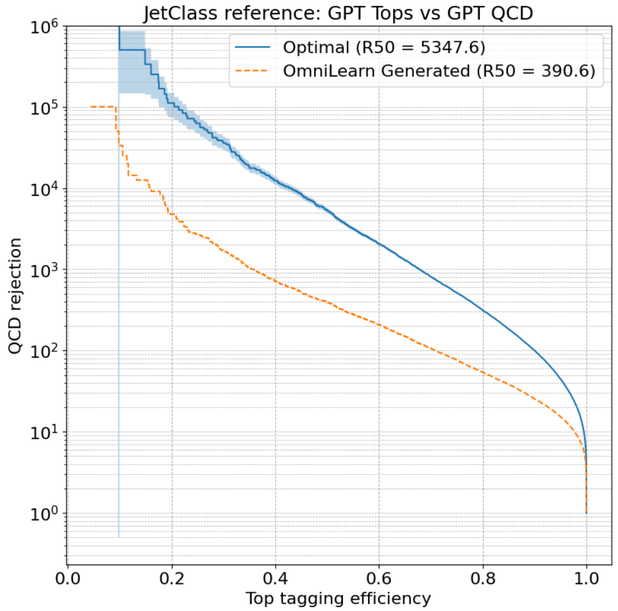
Slight mismodeling of jets by generative model

↓

Very different ROC curve



Mismodeling -> Inflated ROC



Real top and QCD jets already very separable

↓

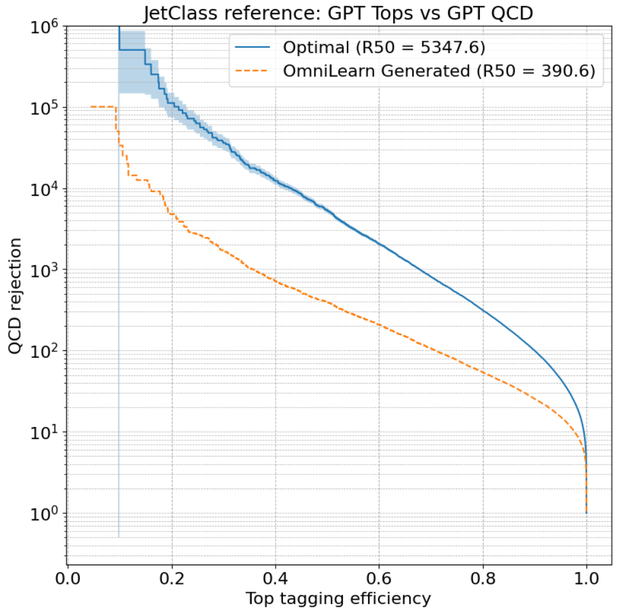
Slight mismodeling of jets by generative model

↓

Very different ROC curve

But why can’t SOTA jet taggers reproduce the optimal ROC curve?

Mismodeling -> Inflated ROC



Real top and QCD jets already very separable

↓

Slight mismodeling of jets by generative model

↓

Very different ROC curve

But why can’t SOTA jet taggers reproduce the optimal ROC curve?

Maybe GPT introduces artifacts that are hard to learn for the SOTA jet taggers?

Mismodeling -> Inflated ROC

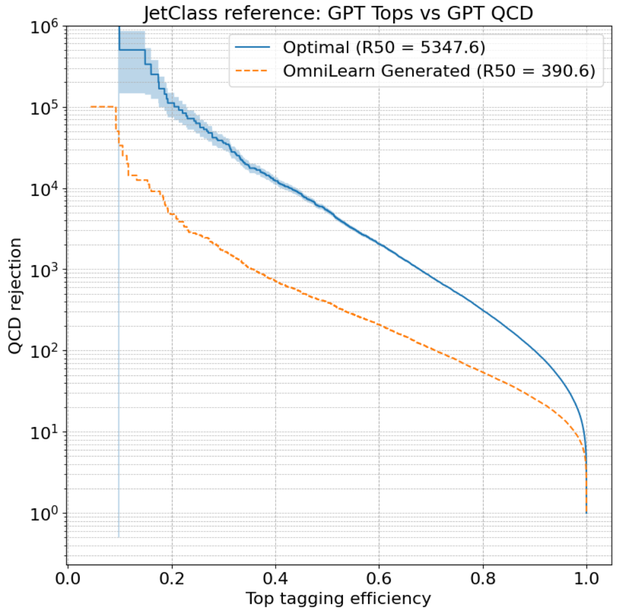
Real top and QCD jets already very separable

↓

Slight mismodeling of jets by generative model

↓

Very different ROC curve



Obtaining FM log-likelihoods

Obtaining FM log-likelihoods

Vector field: $\mathbf{u}_\mathbf{t}^\theta(\mathbf{x}(t)) = \dfrac{d\mathbf{x}(t)}{dt}$

where $\mathbf{x}(0) \sim$ Noise and $\mathbf{x}(1) \sim jets$

Obtaining FM log-likelihoods

Vector field: $\mathbf{u}_\mathbf{t}^\theta(\mathbf{x}(t)) = \dfrac{d\mathbf{x}(t)}{dt}$

where $\mathbf{x}(0) \sim$ Noise and $\mathbf{x}(1) \sim jets$

Compute log-likelihood: $\log p_1(\mathbf{x}(1)) = \log p_0(\mathbf{x}(0)) - \int_0^1 dt \, \mathrm{Tr} \dfrac{\partial \mathbf{u}_\mathbf{t}^\theta}{\partial \mathbf{x}}$

Obtaining FM log-likelihoods

Vector field: $\mathbf{u}_\mathbf{t}^\theta(\mathbf{x}(t)) = \dfrac{d\mathbf{x}(t)}{dt}$

where $\mathbf{x}(0) \sim$ Noise and $\mathbf{x}(1) \sim jets$

Compute log-likelihood: $\log p_1(\mathbf{x}(1)) = \log p_0(\mathbf{x}(0)) - \int_0^1 dt \, \mathrm{Tr} \dfrac{\partial \mathbf{u}_\mathbf{t}^\theta}{\partial \mathbf{x}}$

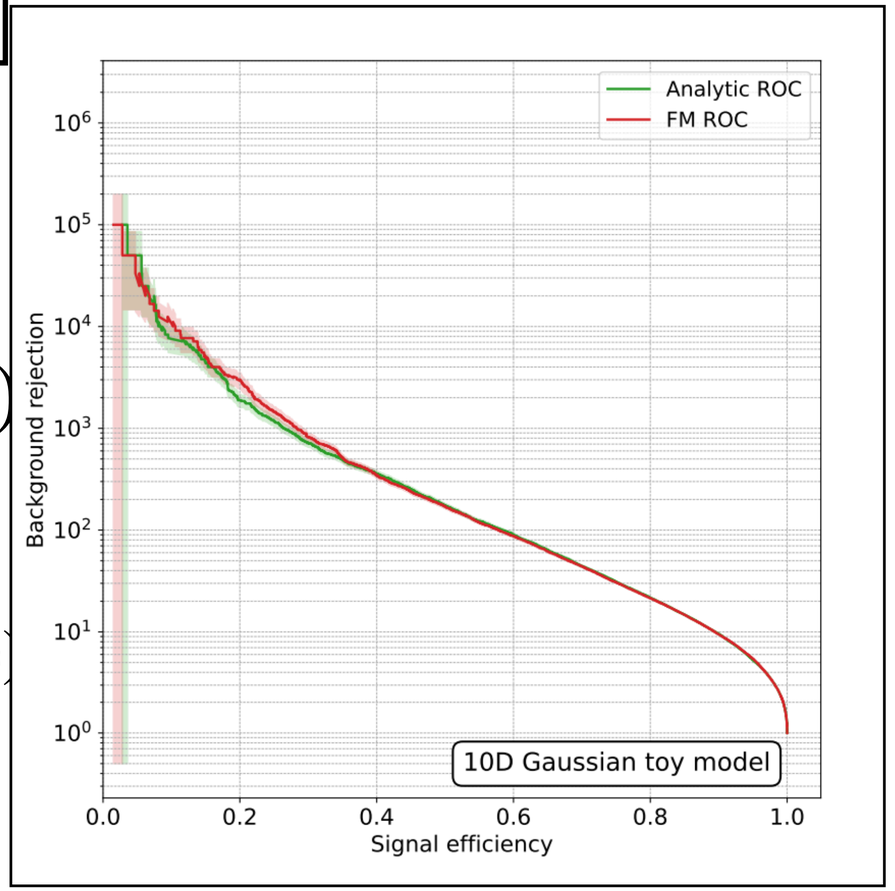
Numerical integration: $\log p_{t-\Delta t}(\mathbf{x}(t-\Delta t)) = \log p_t(\mathbf{x}(t)) - \Delta t \, \mathrm{Tr}(\mathrm{Jac}[\mathbf{u}_{t-\Delta t}^\theta(\mathbf{x}(t)])$

Obtaining FM log-likelihoods

Vector field: $\mathbf{u}_\mathbf{t}^\theta(\mathbf{x}(t)) = \dfrac{d\mathbf{x}(t)}{dt}$

where $\mathbf{x}(0) \sim$ Noise and $\mathbf{x}(1)$

Compute log-likelihood: $\log p_1(\mathbf{x}(1)) = \log p_0(\mathbf{x}(0))$



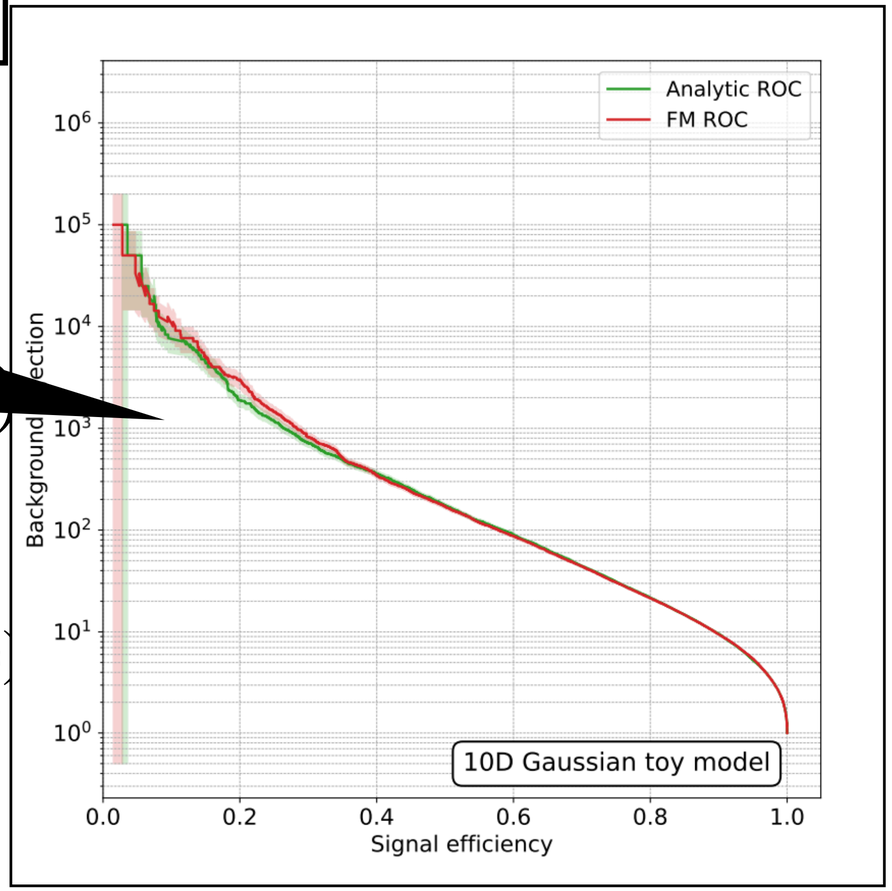
Numerical integration: $\log p_{t-\Delta t}(\mathbf{x}(t-\Delta t)) = \log p_t(\mathbf{x}(t)) - \Delta t \,\mathrm{Tr}(\mathrm{Jac}[\mathbf{u}^\theta_{t-\Delta t}(\mathbf{x}(t)])$

Obtaining FM log-likelihoods

Vecto

wh

Can reproduce ROC curve for analytic toy model



Compute log-likelihood:

$$\log p_1(\mathbf{x}(1)) = \log p_0(\mathbf{x}(0))$$

Numerical integration: $\log p_{t-\Delta t}(\mathbf{x}(t-\Delta t)) = \log p_t(\mathbf{x}(t)) - \Delta t \,\mathrm{Tr}(\mathrm{Jac}[\mathbf{u}^{\theta}_{t-\Delta t}(\mathbf{x}(t)])$

Validating FM log-likelihoods

Validating FM log-likelihoods

Forward: Noise -> Jets (Generation)

Backward: Jets -> Noise (Density estimation)

Want forward and backward to be exact inverses!

Validating FM log-likelihoods

Naive forward: $\mathbf{x}(t + \Delta t) = \mathbf{x}(t) + \Delta t\, \mathbf{u}_t^\theta(\mathbf{x}(t))$

Naive backward: $\mathbf{x}'(t) = \mathbf{x}(t + \Delta t) - \Delta t\, \mathbf{u}_t^\theta(\mathbf{x}(t + \Delta t))$

Validating FM log-likelihoods

Naive forward: $\mathbf{x}(t + \Delta t) = \mathbf{x}(t) + \Delta t\, \mathbf{u}_t^\theta(\mathbf{x}(t))$

Naive backward: $\mathbf{x}'(t) = \mathbf{x}(t + \Delta t) - \Delta t\, \mathbf{u}_t^\theta(\mathbf{x}(t + \Delta t))$

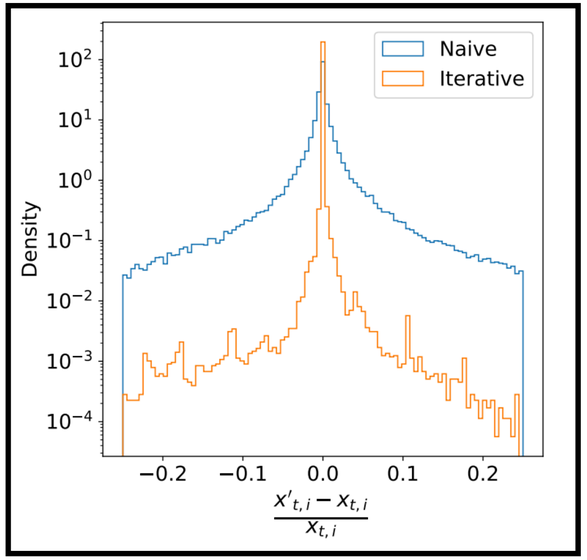
But… $\mathbf{x}'(t) - \mathbf{x}(t) = \Delta t\Big(\mathbf{u}_t^\theta(\mathbf{x}(t + \Delta t)) - \mathbf{u}_t^\theta(\mathbf{x}(t))\Big) \neq 0$

Validating FM log-likelihoods

Naive forward: $\mathbf{x}(t+\Delta t) = \mathbf{x}(t) + \Delta t\, \mathbf{u}_t^\theta(\mathbf{x}(t))$

Naive backward: $\mathbf{x}'(t) = \mathbf{x}(t+\Delta t) - \Delta t\, \mathbf{u}_t^\theta(\mathbf{x}(t+\Delta t))$

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

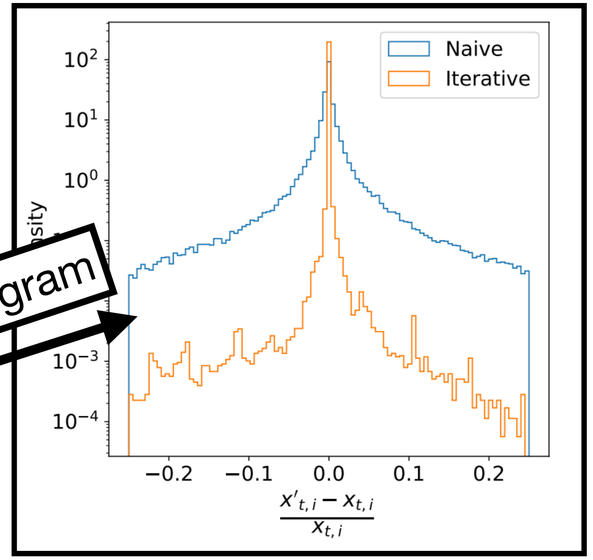


Validating FM log-likelihoods

Naive forward: $\mathbf{x}(t+\Delta t) = \mathbf{x}(t) + \Delta t\, \mathbf{u}_t^\theta(\mathbf{x}(t))$

Naive backward: $\mathbf{x}'(t) = \mathbf{x}(t+\Delta t) - \Delta t\, \mathbf{u}_t^\theta(\mathbf{x}(t+\Delta t))$

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

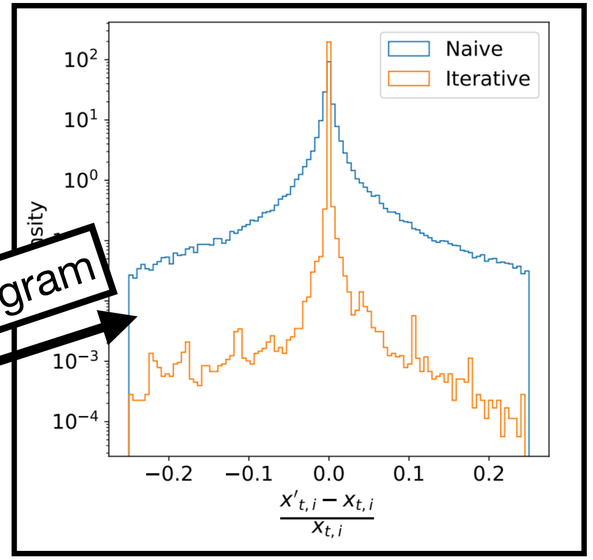


Validating FM log-likelihoods

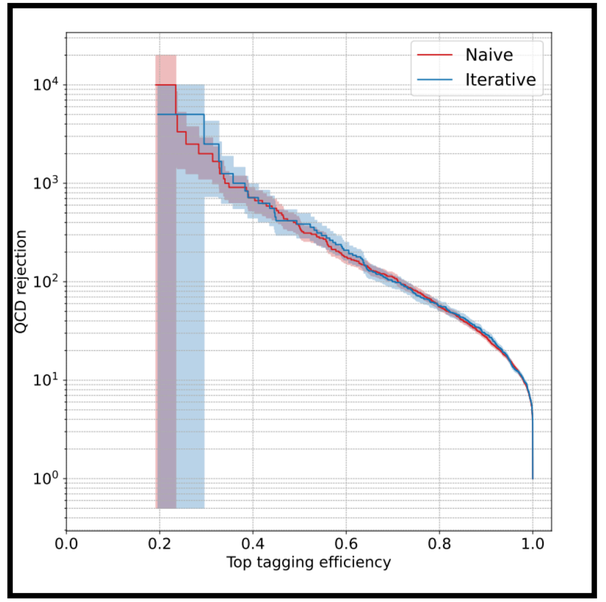
Naive forward: $\mathbf{x}(t+\Delta t) = \mathbf{x}(t) + \Delta t\, \mathbf{u}_t^\theta(\mathbf{x}(t))$

Naive backward: $\mathbf{x}'(t) = \mathbf{x}(t+\Delta t) - \Delta t\, \mathbf{u}_t^\theta(\mathbf{x}(t+\Delta t))$

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



Can correct with fixed-point iteration!

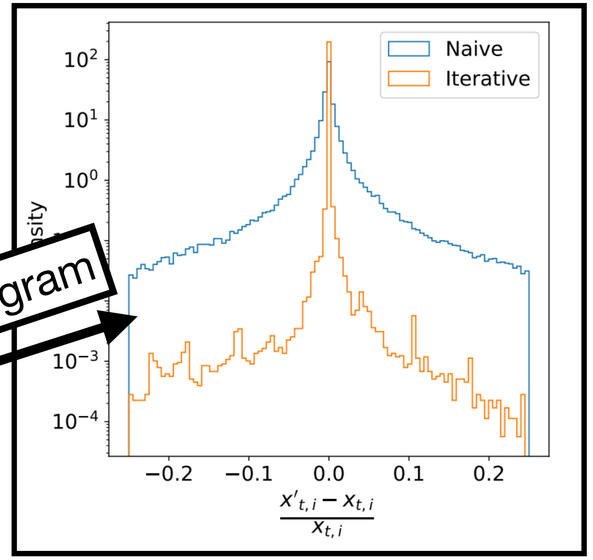


Validating FM log-likelihoods

Naive forward: $\mathbf{x}(t+\Delta t) = \mathbf{x}(t) + \Delta t\, \mathbf{u}_t^\theta(\mathbf{x}(t))$

Naive backward: $\mathbf{x}'(t) = \mathbf{x}(t+\Delta t) - \Delta t\, \mathbf{u}_t^\theta(\mathbf{x}(t+\Delta t))$

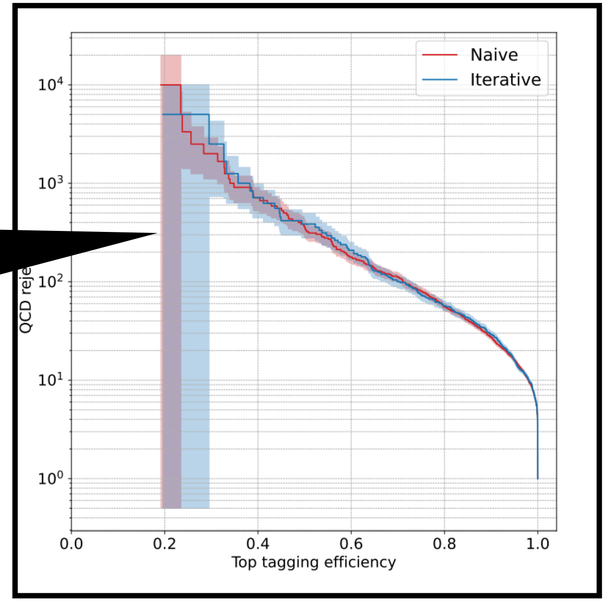
But… $\mathbf{x}'(t) - \mathbf{x}(t) = \Delta t\left(\mathbf{u}_t^\theta(\mathbf{x}(t+\Delta t)) - \mathbf{u}_t^\theta(\mathbf{x}(t))\right) \neq 0$



Blue histogram

Can correct with fixed-point iteration!

Similar ROC curve



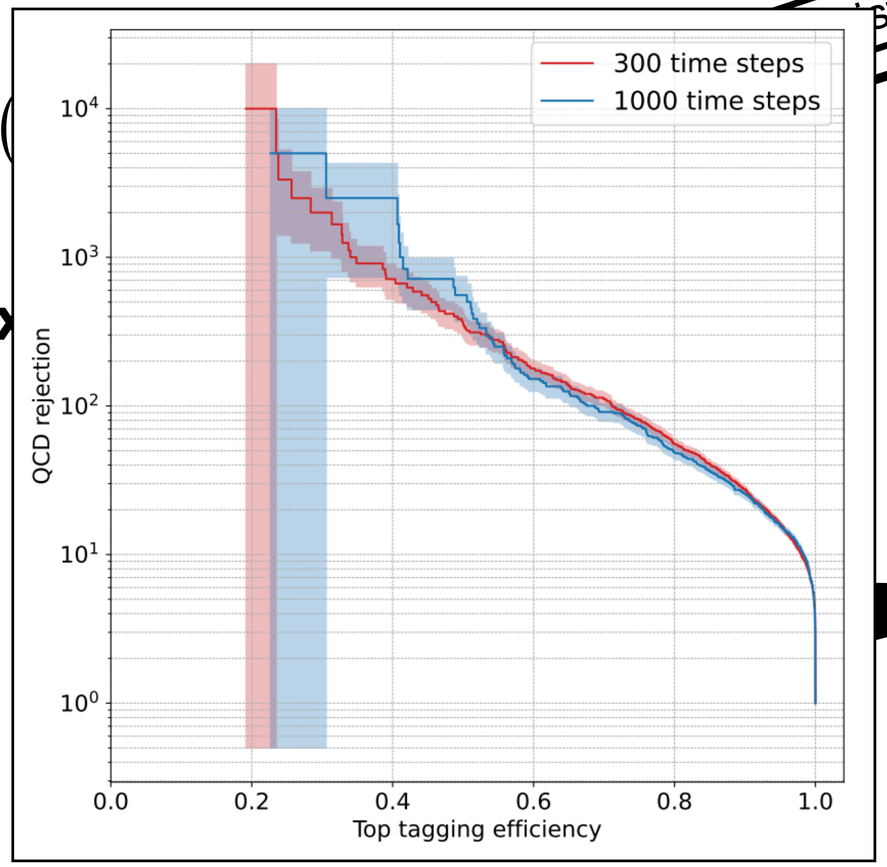
Validating FM log-likelihoods

Naive forward: $\mathbf{x}(t+\Delta t) = \mathbf{x}(t) + \Delta t\, \mathbf{u}_t^\theta(\mathbf{x}(t))$

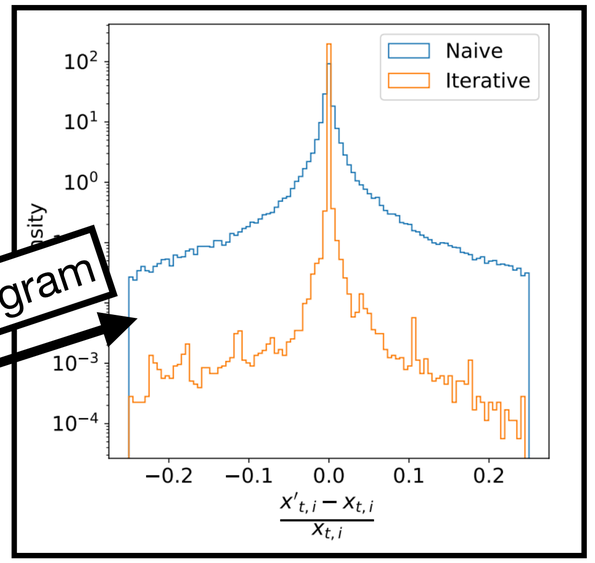
Naive backward: $\mathbf{x}'(t) = \mathbf{x}(t+\Delta t) - \Delta t\, \mathbf{u}_t^\theta(\mathbf{x}(t+\Delta t))$

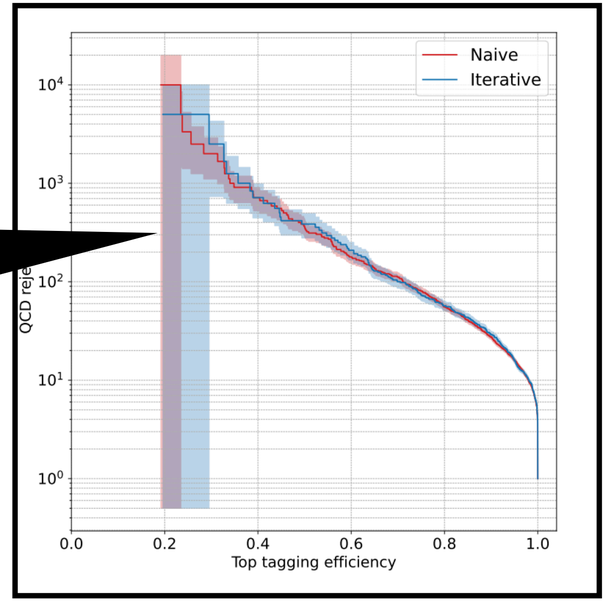
But… $\mathbf{x}'(t) - \mathbf{x}(t) = \Delta t($

Can correct with fix



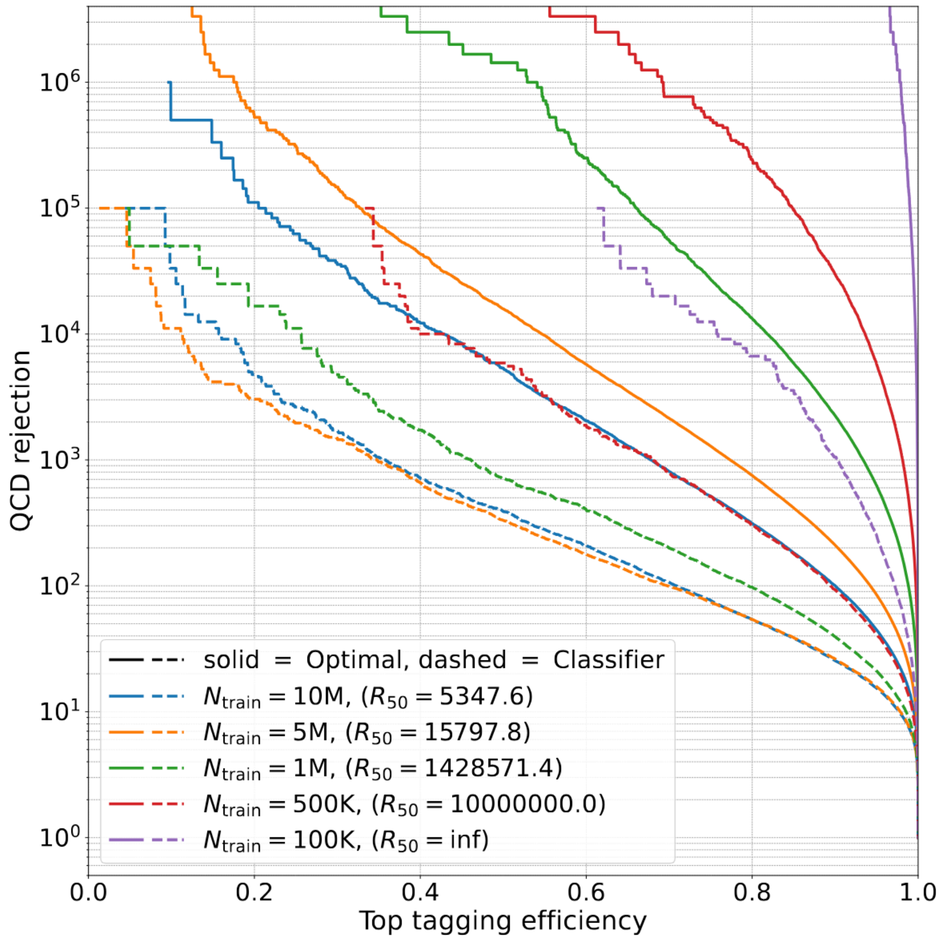
stogram



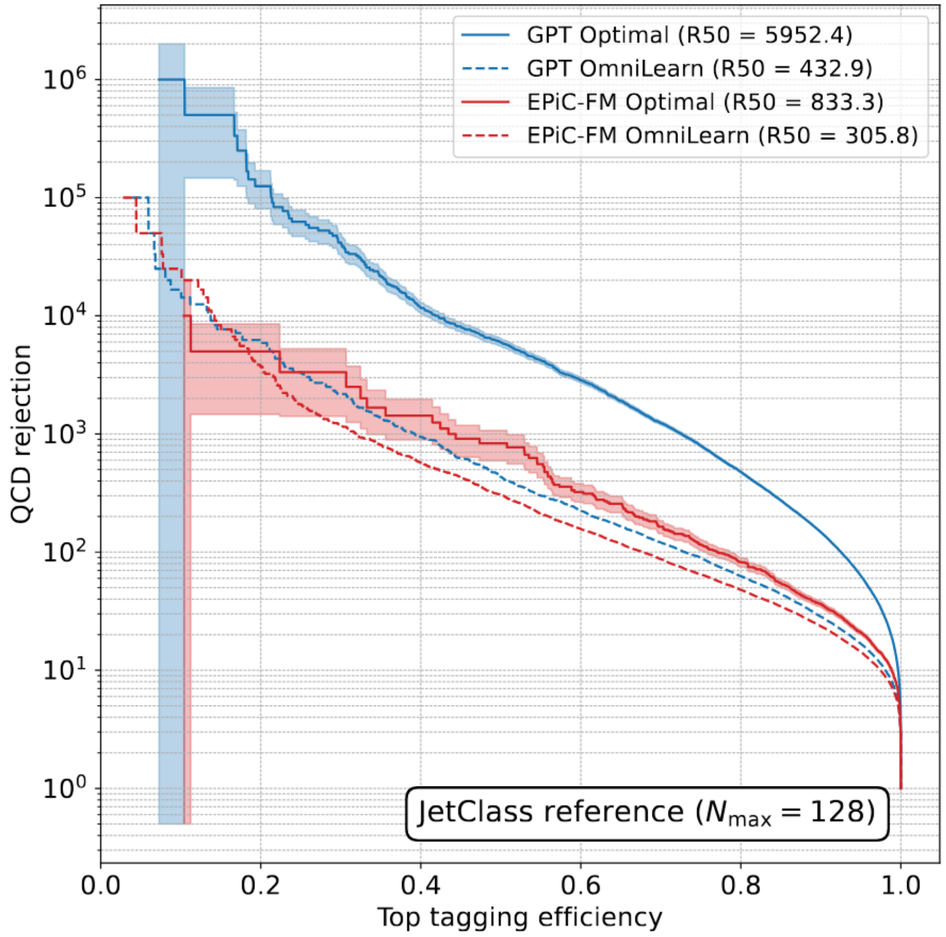


More tests of overfitting

- Train GPT model on different dataset sizes
- Expect more overfitting for smaller datasets
- More overfitting -> More separable tops vs QCD



Going to more constituents

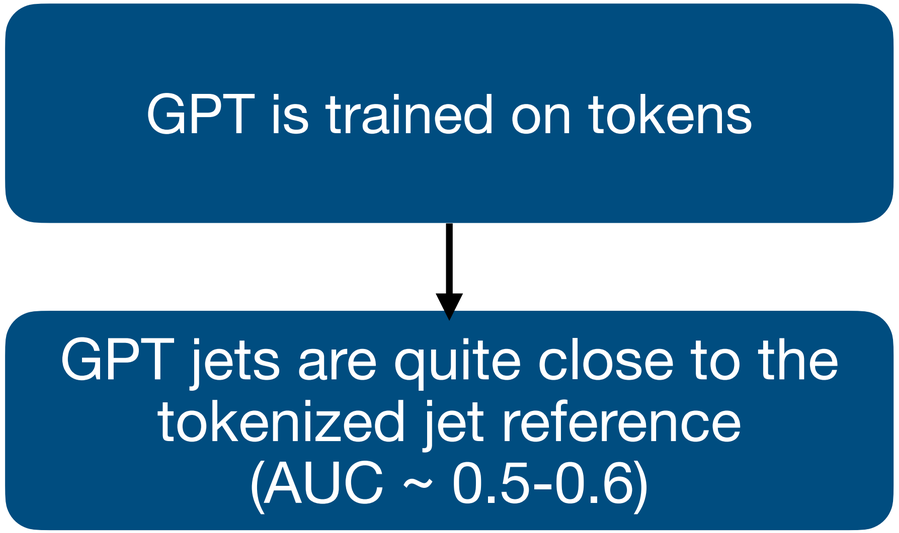


Possible explanation?

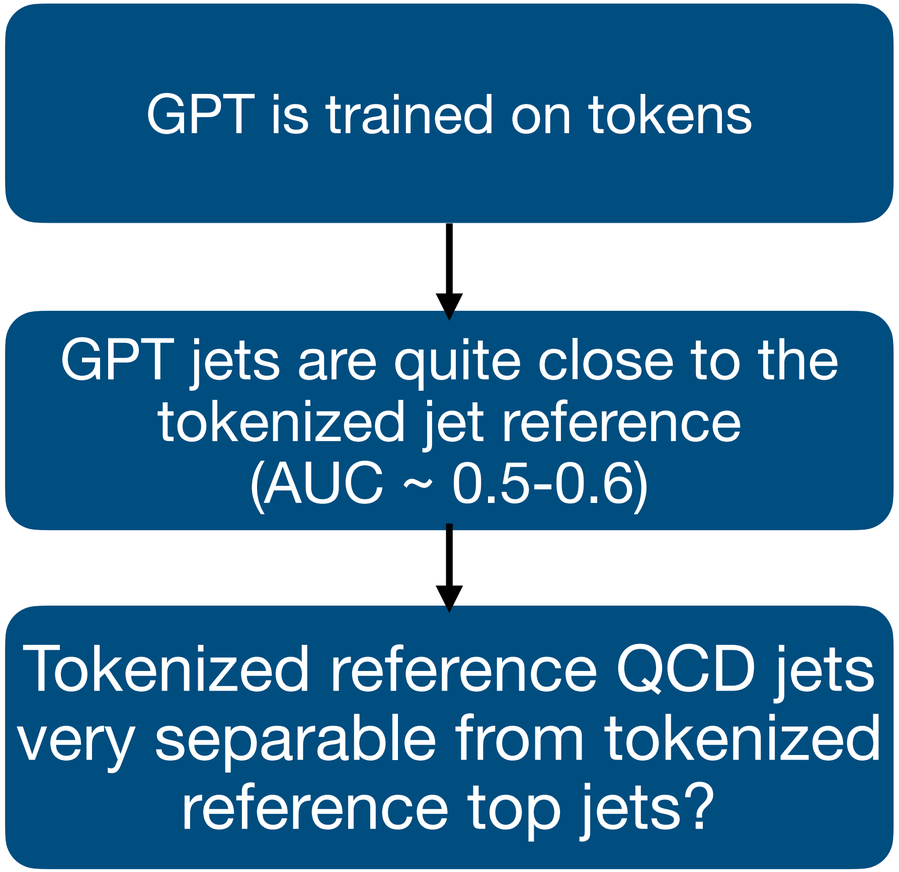
Possible explanation?

GPT is trained on tokens

Possible explanation?



Possible explanation?



Possible explanation?

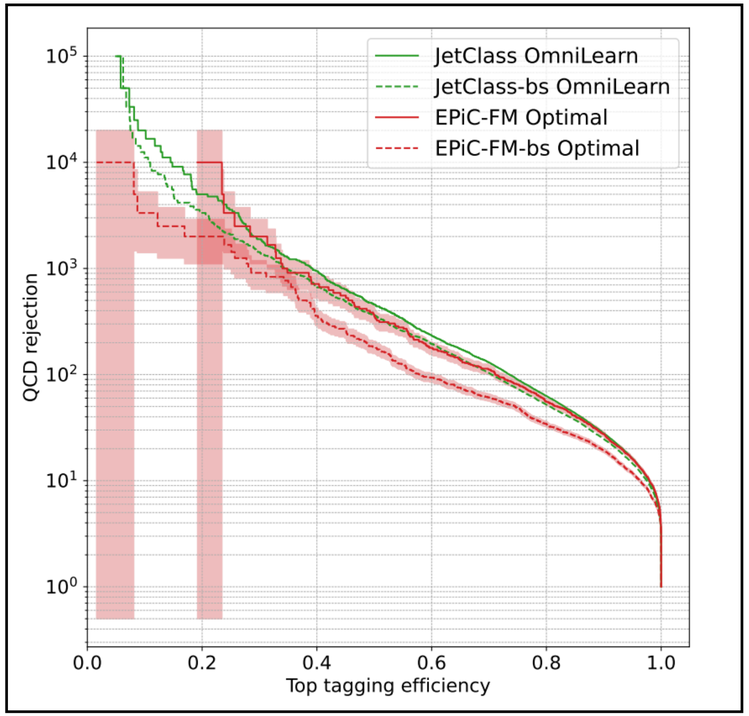
GPT is trained on tokens

↓

GPT jets are quite close to the tokenized jet reference (AUC ~ 0.5-0.6)

↓

Tokenized reference QCD jets very separable from tokenized reference top jets?



Possible explanation?

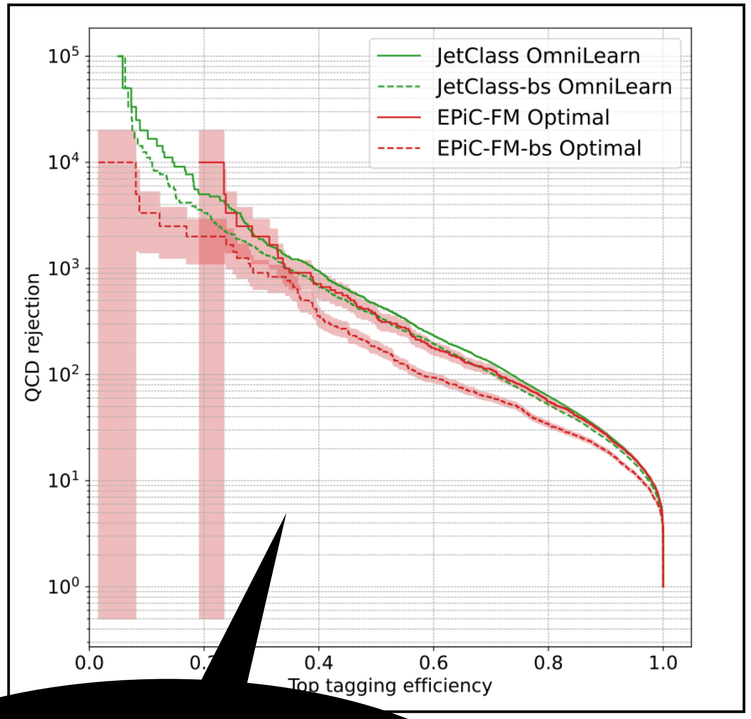
GPT is trained on tokens

↓

GPT jets are quite close to the tokenized jet reference (AUC ~ 0.5-0.6)

↓

Tokenized reference QCD jets very separable from tokenized reference top jets?



Tokenized (“bs”) jets are less separable

Possible explanation?

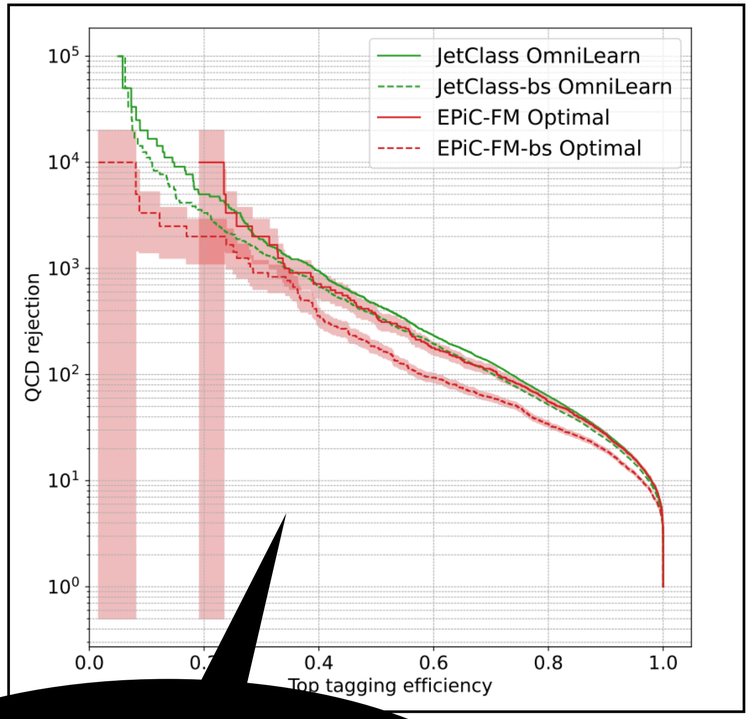
GPT is trained on tokens

↓

GPT jets are quite close to the tokenized jet reference (AUC ~ 0.5-0.6)

↓

Tokenized reference QCD jets very separable from tokenized reference top jets?



Tokenized (“bs”) jets are less separable