OmniLearn: Facilitating All Jet Physics Tasks

vmikuni@lbl.gov

vinicius-mikuni

Vinicius M. Mikuni

Foundational Models

- Foundational models are everywhere now
- In essence, these models are trained on large datasets and can be used for multiple tasks
- How does a **foundational model for science** looks like?

ChatGPT

Claude

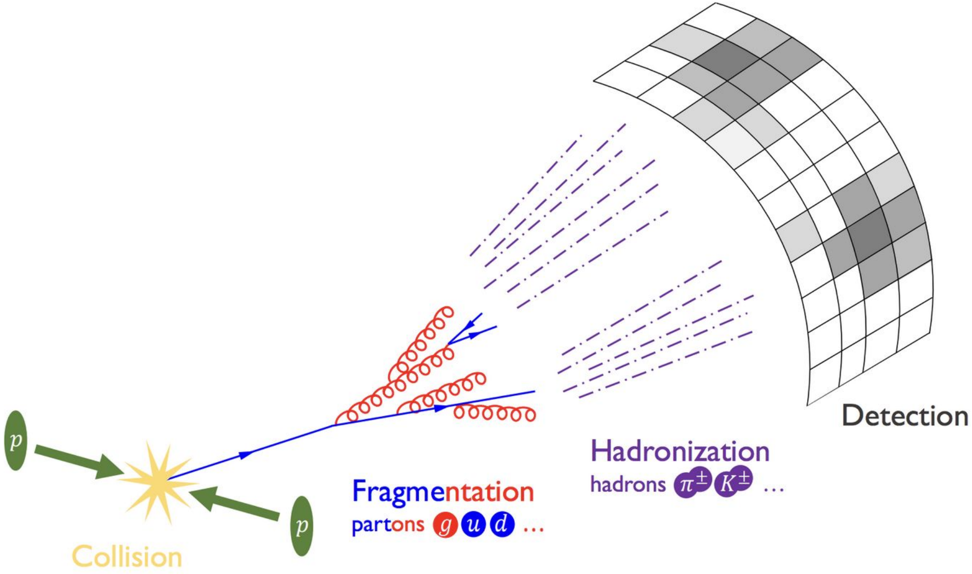
BY ANTHROP\C

Data

Model

Learning

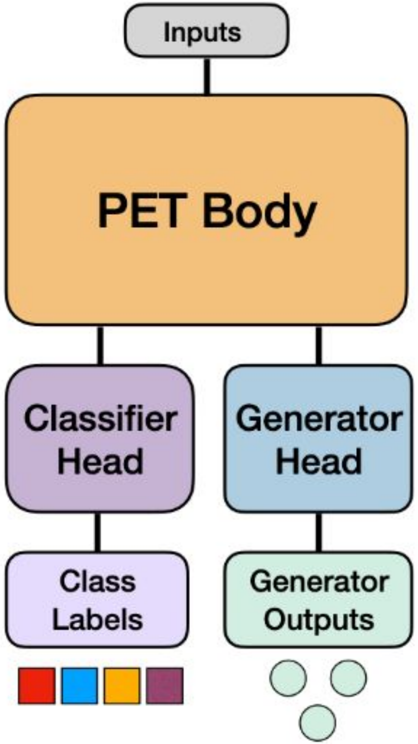
The Data



Jets are the most common signatures at the LHC

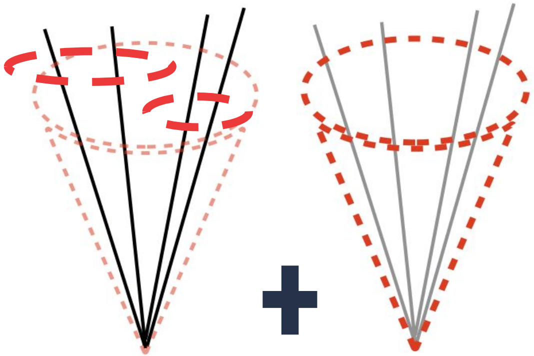
- Complicated signature: O(10-100) particles are clustered in each jet
- Everywhere: Jets are used in almost any analysis at the LHC

The Model



Point-Edge Transformer (**PET**)

- Combine local information with graphs
- Learn global information with Transformers: **3M parameters**



The Model

BERKELEY LAB

Not all datasets contain the same information:

- Let the model learn with and without some features
- **Feature Dropout**: With fixed probability, set some of the input features to 0

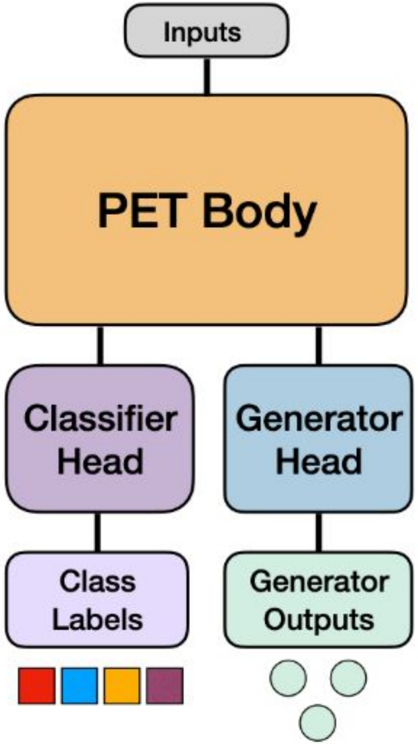
$$f_1, f_2, f_3, f_4, \boxed{f_5, f_6, f_7, f_8}$$

$\boxed{f_5, f_6, f_7, f_8}$ **p = 0.9**

$\boxed{0,0,0,0}$ **p = 0.1**

More details at: https://arxiv.org/abs/2404.16091

The Learning



Couple the network with multiple experts:

- **Classify jets**: learns the difference in radiation between jet types
- **Generate jets**: implicitly learn the likelihood of jets for different particles
- **Multi-objective** loss

The Learning

$$\mathcal{L} = \mathcal{L}_{\text{class}} + \mathcal{L}_{\text{gen}} + \mathcal{L}_{\text{class smear}}$$
$$= \text{CE}(y, y_{\text{pred}}) + \|\mathbf{v} - \mathbf{v}_{\text{pred}}\|^2 + \alpha^2 \text{CE}(y, \hat{y}_{\text{pred}})$$

Straightforward loss function:

- **Cross entropy** for each class
- Perturbed data prediction from the **diffusion loss**
- Classification over perturbed inputs: **data augmentation**!

More details at: https://arxiv.org/abs/2404.16091

Comparison Between Models

BERKELEY LAB

Language inspired models

- Data are tokenized
- Unsupervised and general pre-training
- Big models often required

OmniLearn

- Data are continuous
- HEP has one of the best simulators across all sciences: supervised pre-training
- Medium models that can fit on standard GPUs are still useful

Training

JetClass dataset used for training

- 100M jets
- **10 different jet categories, AK8 jets simulated in pp collisions with Madgraph + Pythia8 with CMS Delphes detector simulation**

Use the pre-trained model as the starting point and fine-tune using different datasets

Huilin Qu, Congqiao Li, Sitian Qian, arXiv:2202.03772

Evaluation

BERKELEY LAB

Evaluation datasets: **1**

2 different jet categories, AK8 jets simulated in pp collisions with ~~Madgraph +~~ Pythia8 with ATLAS Delphes detector simulation

	Acc	AUC	$1/\epsilon_B$	
			$\epsilon_S = 0.5$	$\epsilon_S = 0.3$
ResNeXt-50 [38]	0.936	0.9837	302 ± 5	1147 ± 58
P-CNN [38]	0.930	0.9803	201 ± 4	759 ± 24
PFN [35]	-	0.9819	247 ± 3	888 ± 17
ParticleNet [38]	0.940	0.9858	397 ± 7	1615 ± 93
JEDI-net [37]	0.9300	0.9807	-	774.6
PCT [41]	0.940	0.9855	392 ± 11	1559 ± 98
LGN [79]	0.929	0.964	-	435 ± 95
rPCN [39]	-	0.9845	364 ± 9	1642 ± 93
LorentzNet [10]	0.942	0.9868	498 ± 18	2195 ± 173
PELICAN [80]	0.9425	0.9869	-	2289 ± 204
ParT [42]	0.940	0.9858	413 ± 16	1602 ± 81
ParT-f.t. [42]	**0.944**	**0.9877**	**691 ± 15**	**2766 ± 130**
Mixer(HDBSCAN) [81]	-	0.9859	416	-
PET Classifier	0.938	0.9848	340 ± 12	1318 ± 39
OmniLearn	0.942	0.9872	568 ± 9	**2647 ± 192**

Better than all non-fine-tuned models and similar to ParT performance

Evaluation

Evaluation datasets: **2**

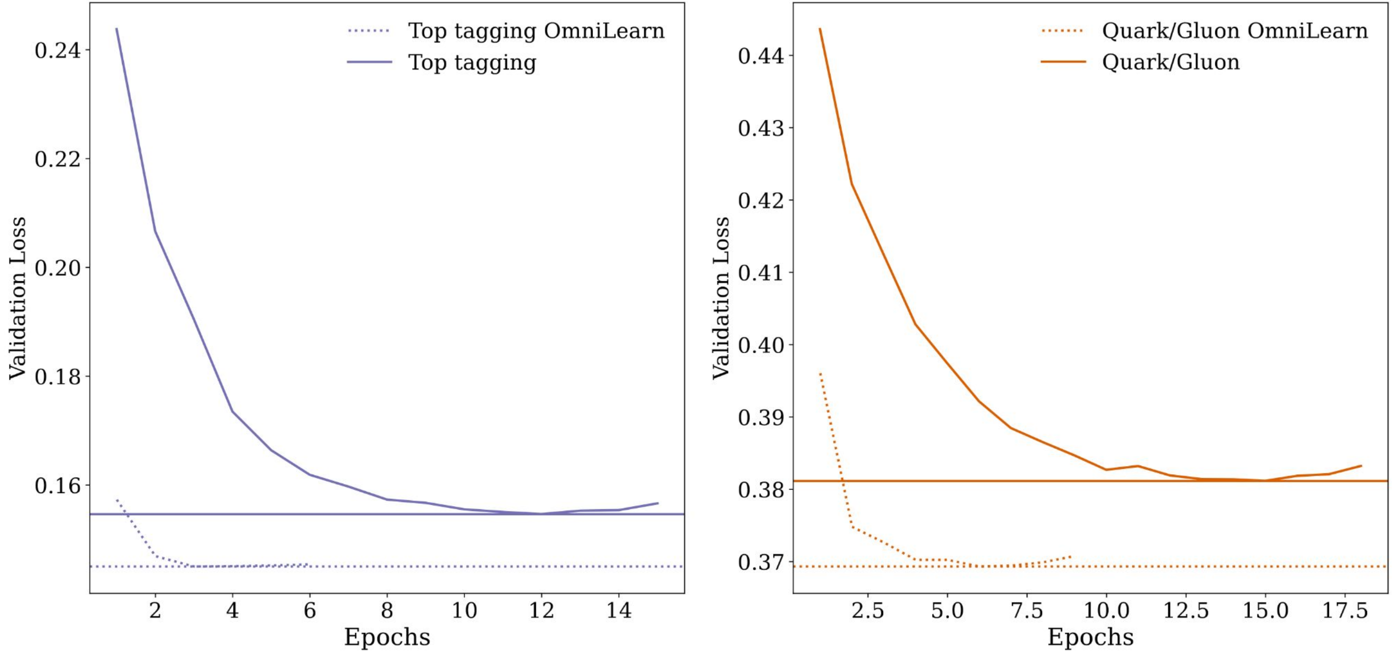
2 different jet categories, AK4 jets simulated in pp collisions with ~~Madgraph +~~ Pythia8 ~~with CMS Delphes detector simulation~~

	Acc	AUC	$1/\epsilon_B$	
			$\epsilon_S = 0.5$	$\epsilon_S = 0.3$
P-CNN [38]	0.827	0.9002	34.7	91.0
PFN [35]	-	0.9005	34.7±0.4	-
ParticleNet [38]	0.840	0.9116	39.8±0.2	98.6±1.3
rPCN [39]	-	0.9081	38.6 ± 0.5	-
ParT [42]	0.840	0.9121	41.3 ± 0.3	101.2 ± 1.1
ParT-f.t. [42]	0.843	0.9151	42.4 ± 0.2	**107.9 ± 0.5**
PET classifier	0.837	0.9110	39.92±0.1	104.9 ± 1.5
OmniLearn	**0.844**	**0.9159**	**43.7±0.3**	**107.7 ± 1.5**

Better than all non-fine-tuned models and similar to PartT performance

Evaluation

Evaluation datasets: **2**

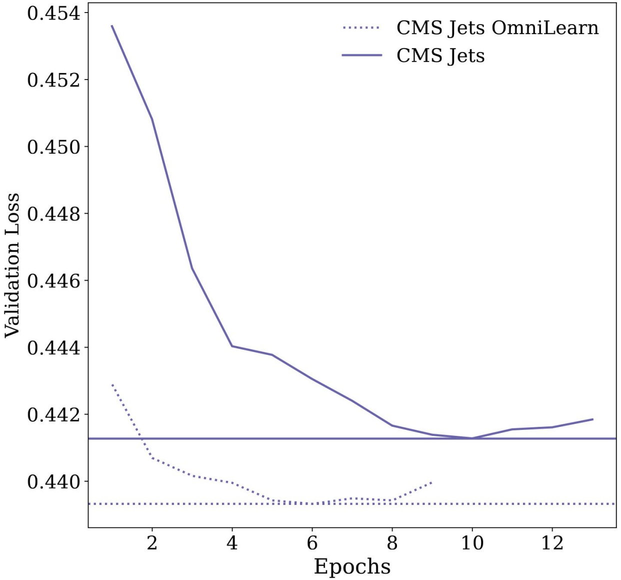


Faster training and better convergence

Evaluation

Evaluation datasets: **3**

2 different jet categories, AK5 jets simulated in pp collisions with Pythia6 with Geant4 Simulation + CMS Particle flow reconstruction

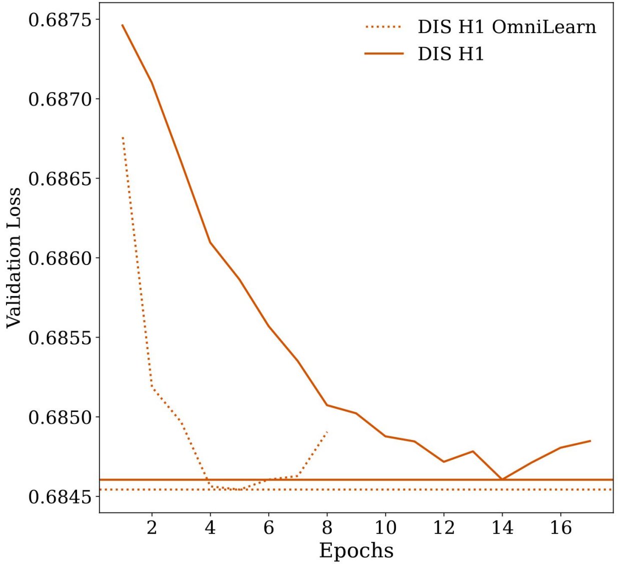


	AUC	Acc	$1/\epsilon_B$	
			$\epsilon_S = 0.5$	$\epsilon_S = 0.8$
PET classifier	0.875	0.796	23.91 ± 0.07	4.770 ± 0.001
OmniLearn	**0.877**	**0.797**	$\mathbf{24.36 \pm 0.01}$	$\mathbf{4.836 \pm 0.004}$

BERKELEY LAB

Evaluation datasets: **4**

2 different jet categories, AK10 jets simulated in ep collisions with Rapgap with Geant3 Simulation + H1 Particle flow reconstruction

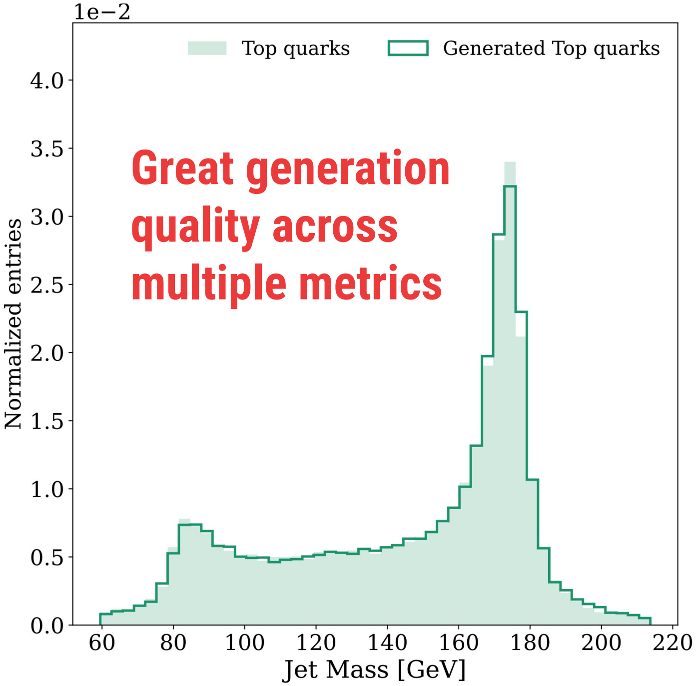


	AUC	Acc	$1/\epsilon_B$	
			$\epsilon_S = 0.1$	$\epsilon_S = 0.5$
PET classifier	0.5691	0.547	17.73±0.04	2.467±0.002
OmniLearn	0.5695	0.547	17.78±0.06	2.470±0.003

Jet Generation

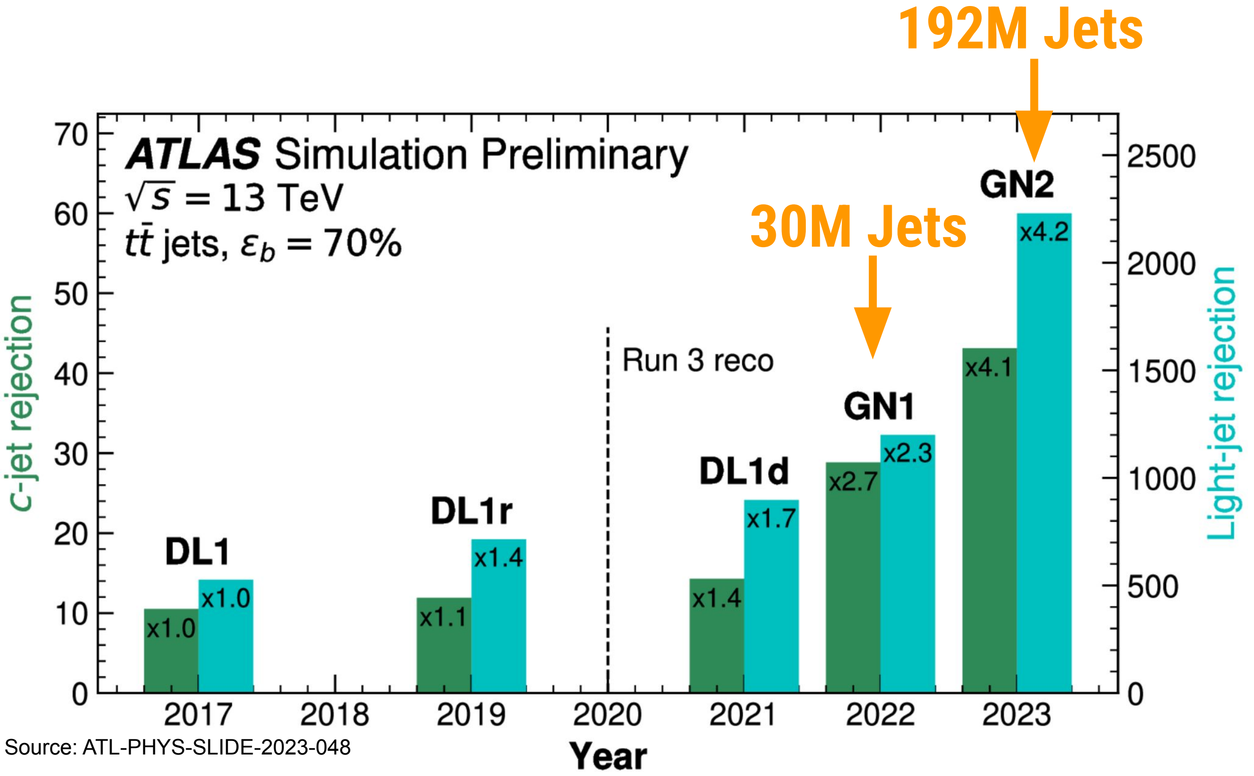
Evaluation datasets: **6**

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



“Application Highlight

BERKELEY LAB



Source: ATL-PHYS-SLIDE-2023-048

Pushing classification performance requires lots of data!

FastSim to FullSim

Evaluation datasets: **7**

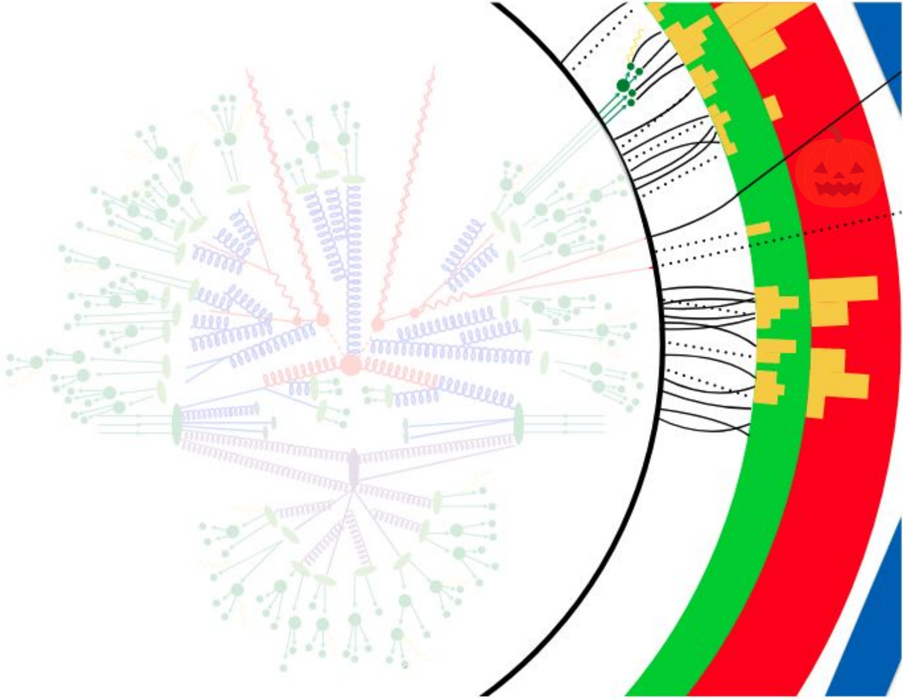
OmniLearn is trained on cheap Delphes simulations. Can we fine-tune to Run 2 **ATLAS** Full simulation + Reconstruction?

- Matches SOTA with **10%** of the data
- Improves on SOTA if all events are used

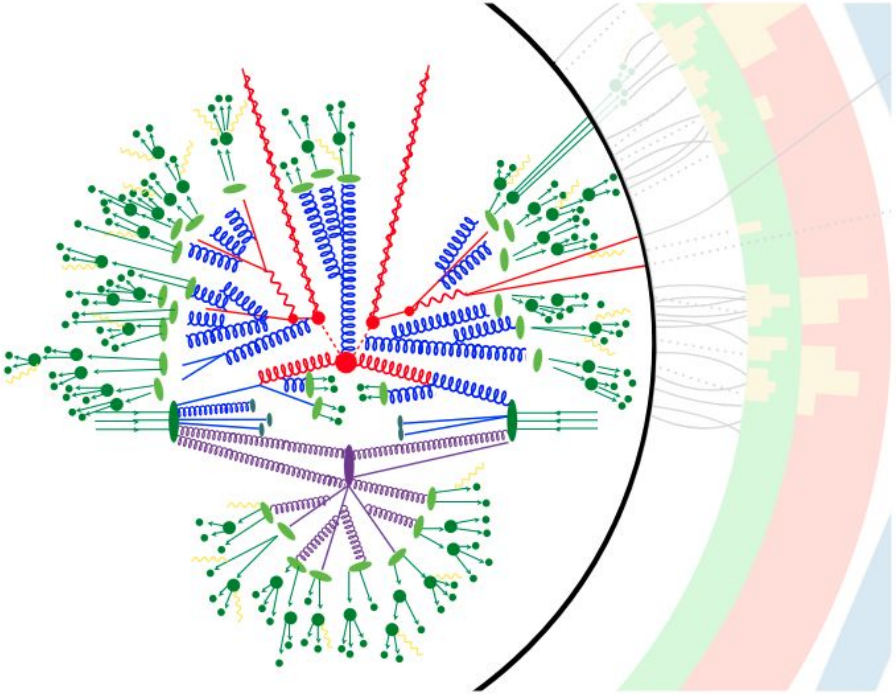
	AUC	Acc	$1/\epsilon_B$	
			$\epsilon_S = 0.5$	$\epsilon_S = 0.8$
ResNet 50	0.885	0.803	21.4	5.13
EFN	0.901	0.819	26.6	6.12
hlDNN	0.938	0.863	51.5	10.5
DNN	0.942	0.868	67.7	12.0
PFN	0.954	0.882	108.0	15.9
ParticleNet	0.961	0.894	153.7	20.4
PET classifier (4M)	0.959	0.890	146.5	19.4
OmniLearn (4M)	0.961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

Unfolding

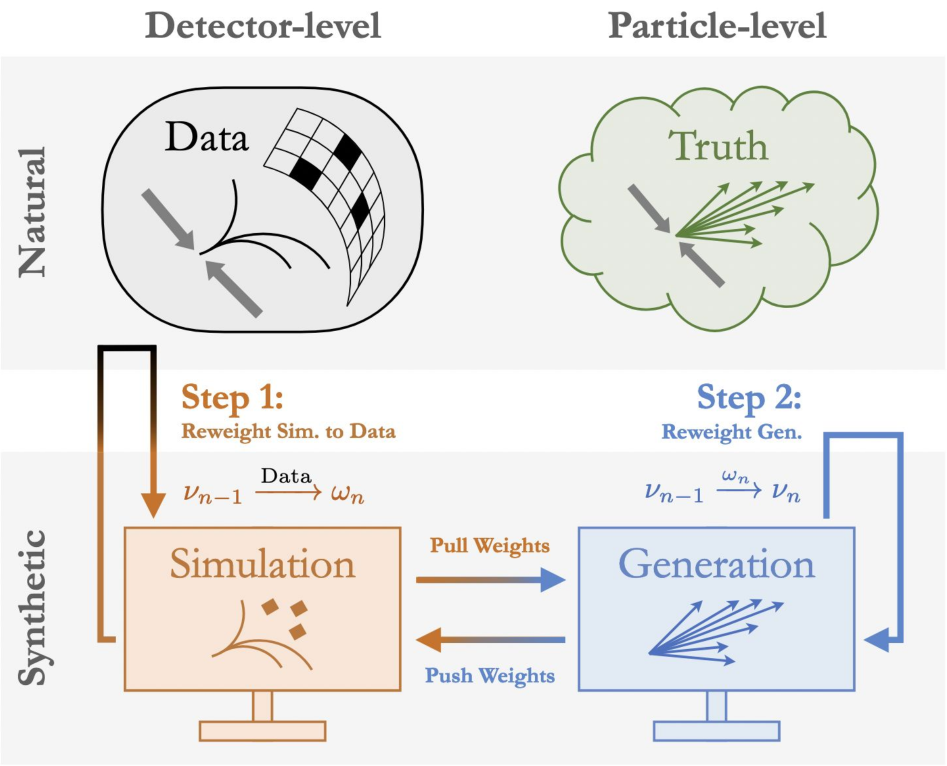
What we measure



What we want



OmniFold



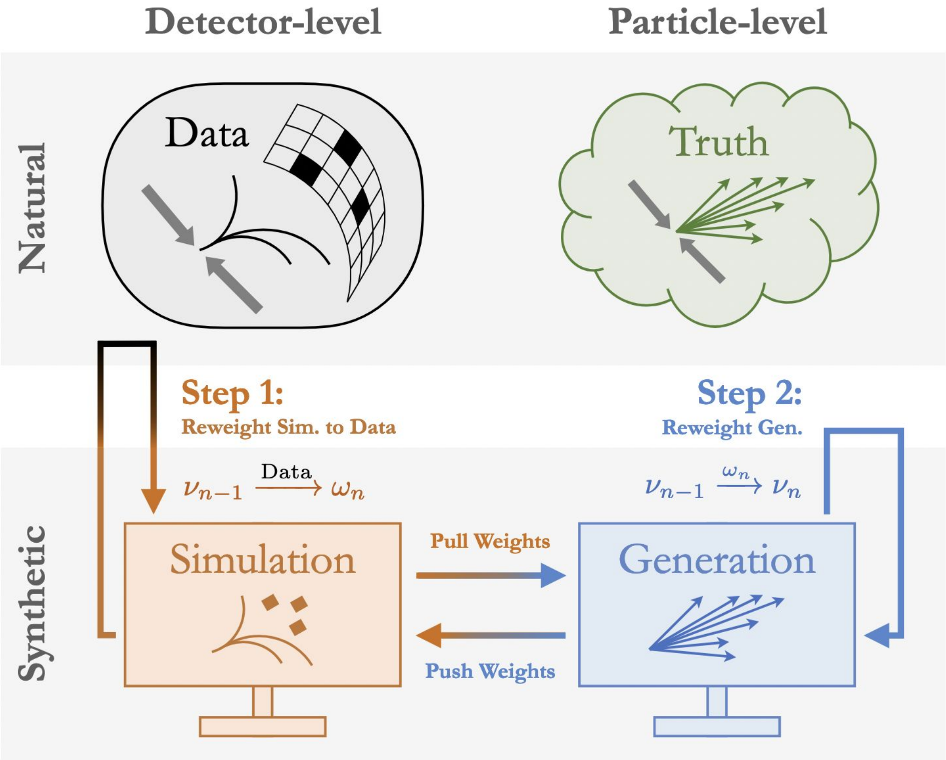
2-step iterative process

- **Step 1**: Reweight simulations to look like data
- **Step 2**: Convert learned weights into functions of particle level objects

Learn a reweighting function between data and simulation

Source: Andreassen et al. PRL 124, 182001 (2020)

OmniFold



2-step iterative process

- **Step 1**: Reweight simulations to look like data
- **Step 2**: Convert learned weights into functions of particle level objects

Classification!

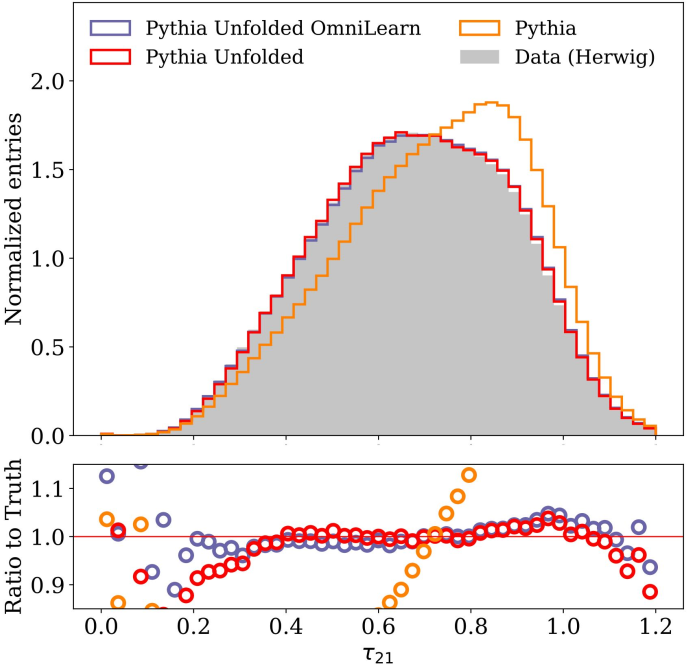
Source: Andreassen et al. PRL 124, 182001 (2020)

Unfolding

Evaluation datasets: **8**

Unbinned Unfolding using the OmniFold workflow. More **precise** than traditional unfolding and more **efficient** than previous ML models. Ex: H1 trained thousands of networks

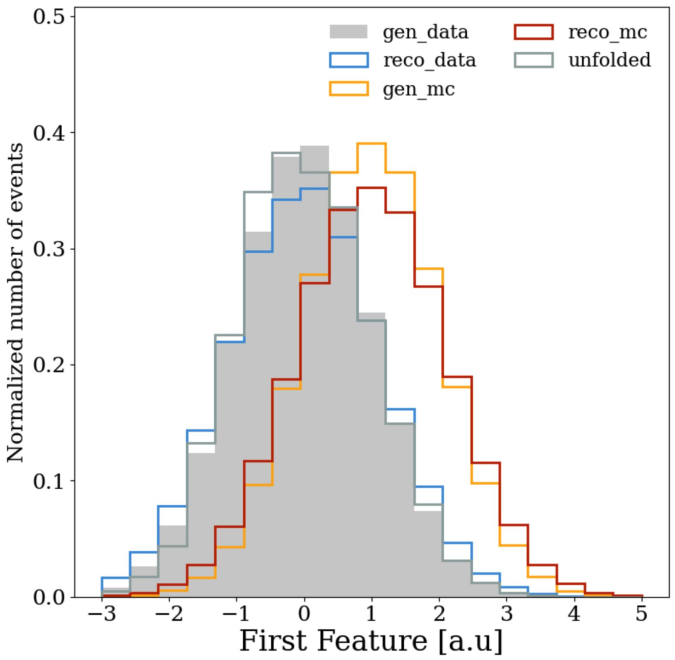
Metric	MultiFold	UniFold	IBU	OmniFold		
				DeepSets	PET classifier	OmniLearn
Jet mass	3.80	8.82	9.31	**2.77**	**2.8±0.9**	**2.6±0.8**
N	0.89	1.46	1.51	**0.33**	0.50±0.15	**0.34±0.1**
Jet Width	0.09	0.15	0.11	0.10	0.09±0.02	**0.07±0.01**
$\log \rho$	0.37	0.59	0.71	0.35	0.23±0.07	**0.14±0.03**
τ_{21}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



See also: H1 Collaboration, PLB 844 (2023) 138101

OmniFold

- OmniFold is also now available on pip:
  - **pip install omnifold**
  - GitHub repository: https://github.com/ViniciusMikuni/omnifold
- **RooUnfold** implementation coming soon!

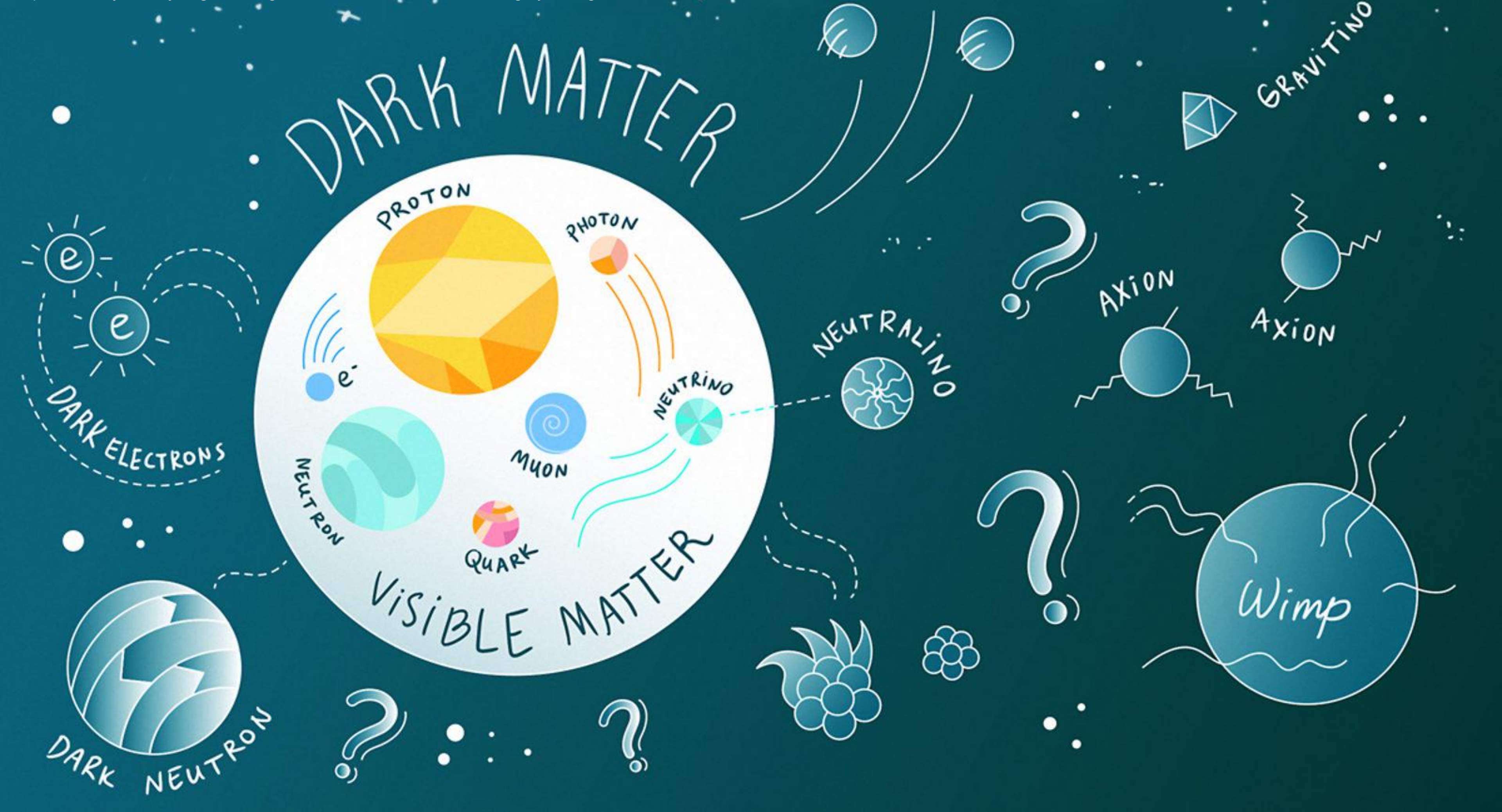


Let's now unfold!

In [10]:
```
omnifold = MultiFold(
    "Gaussian_test",
    model1,
    model2,
    data,
    mc,
    batch_size = 1024,
    niter = 5,  #Number of Iterations
    epochs=100,
    weights_folder = 'weights',
    verbose = True,
)
```

In [11]:
```
omnifold.Preprocessing()
omnifold.Unfold()
```

https://www.symmetrymagazine.org/article/december-2013/four-things-you-might-not-know-about-dark-matter

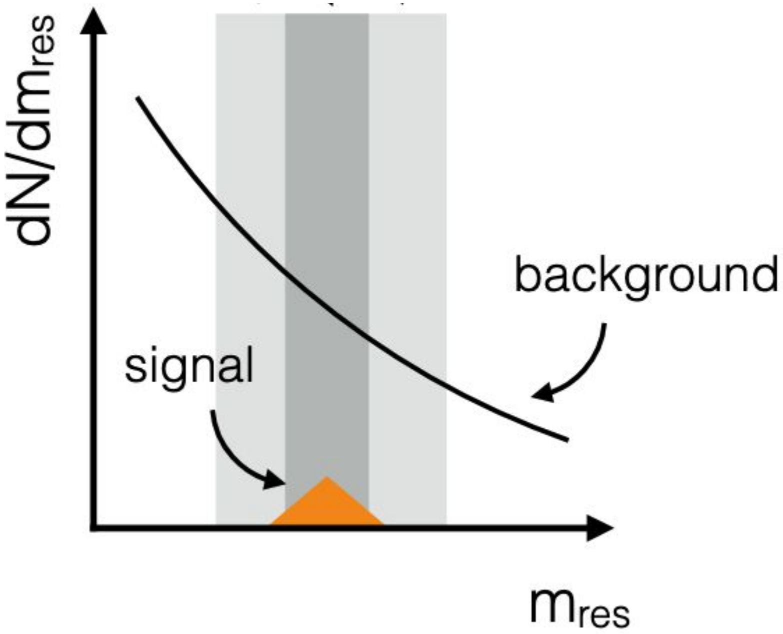


Anomaly Detection

Evaluation datasets: **9**

Bump-hunting using ML:

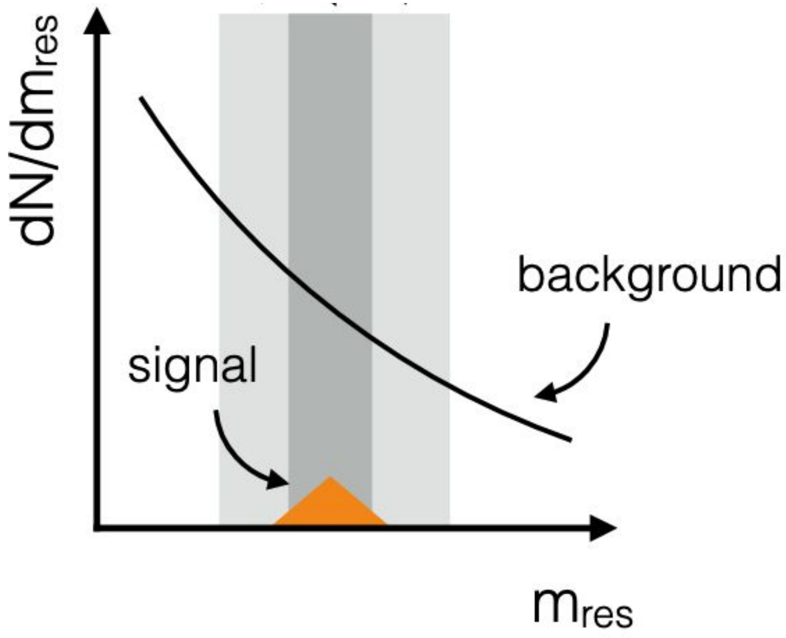
- Use the background in the sideband to estimate the background in the signal region
- Compare the estimated background with the data



Evaluation datasets: **9**

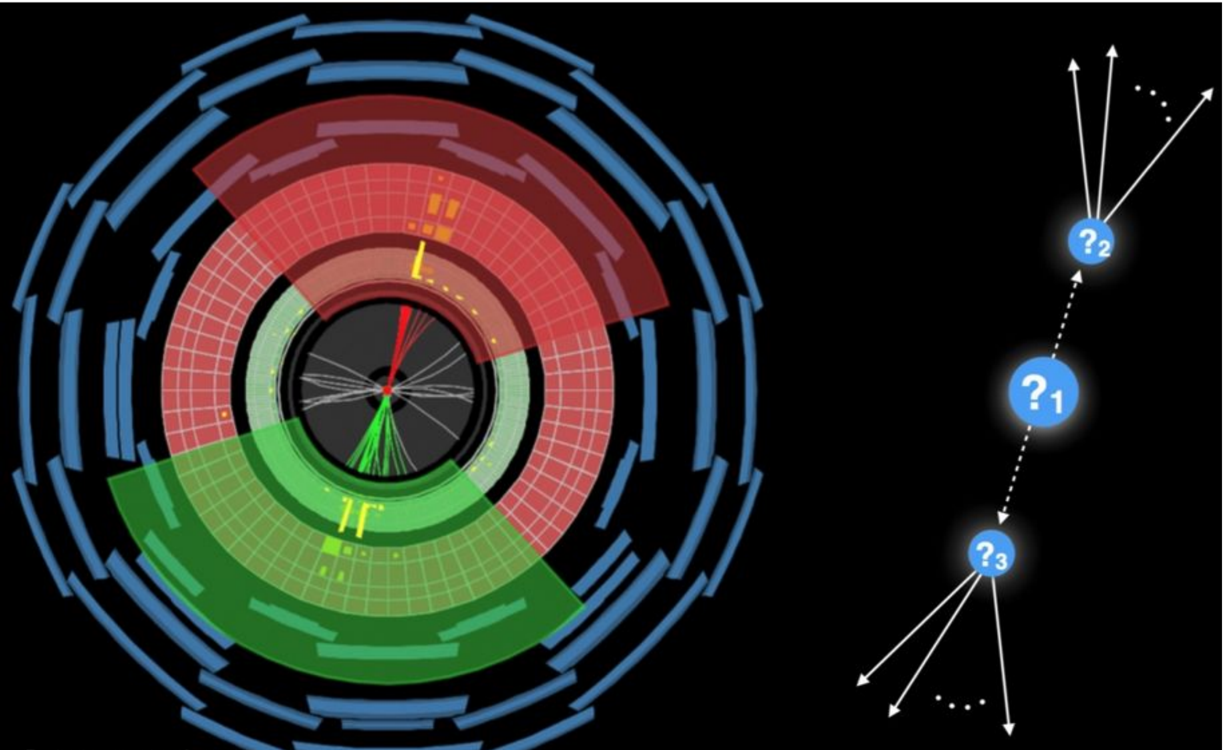
Bump-hunting using ML:

- **Generative Model**
- **Classifier**



LHCO dataset

Evaluation datasets: **9**



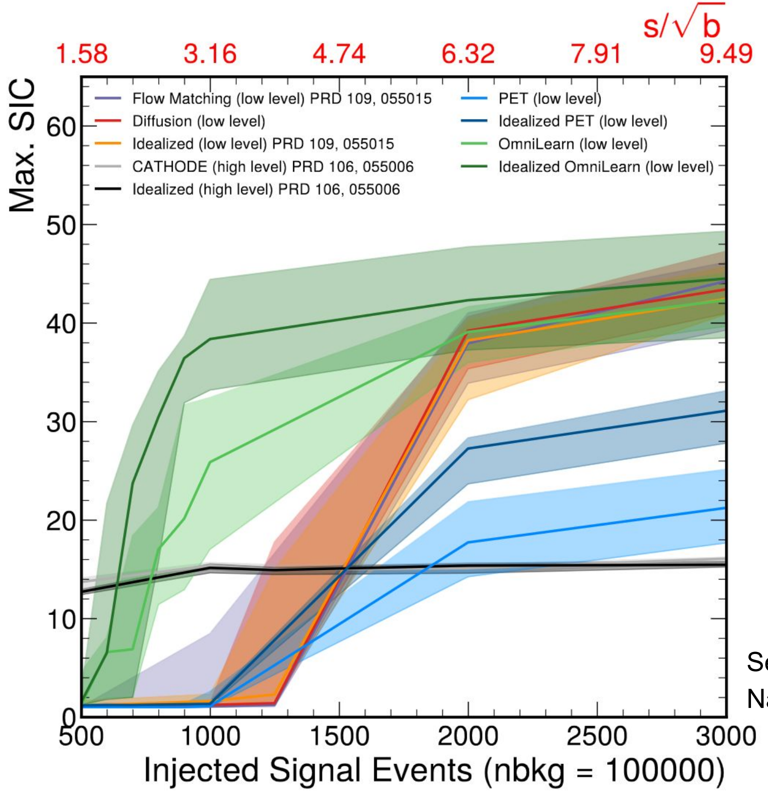
Detector cross-section image credit: ATLAS

LHCO R&D dataset

- Resonant **dijet** final state: A->B(qq)C(qq) with m_A, m_B , m_C = 3.5, 0.5, 0.1 TeV

Anomaly Detection

Evaluation datasets: **9**



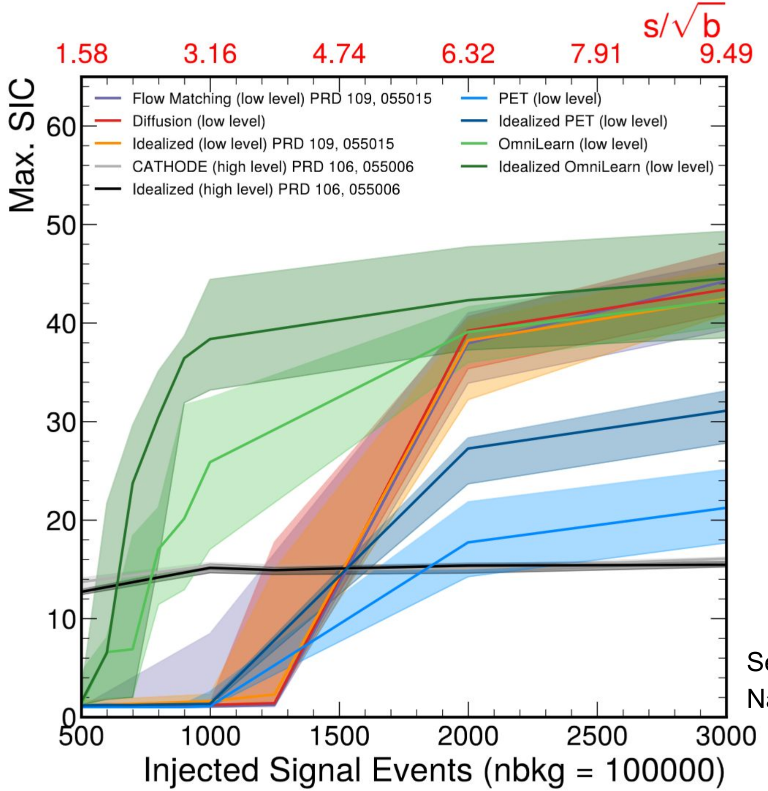
- **Generate** the full dijet system: 2*279*3 = **1674** numbers to generate
- **Classify** data from background

SIC = Significance Improvement Curve (TPR/sqrt(FPR) vs TPR) "By how much can I improve the significance of a particular signal given an initial significance."

See also: E. Buhmann, C. Ewen, G. Kasieczka, **V. Mikuni,** B. Nachman, and D. Shih, Phys. Rev. D 109, 055015

Anomaly Detection

Evaluation datasets: **9**



- **Generate** the full dijet system: 2*279*3 = **1674** numbers to generate
- **Classify** data from background

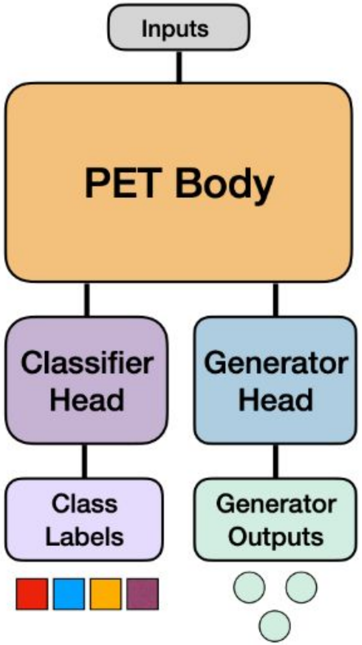
Previous results were limited by the amount of data in the SR: Only sensitive to NP when **S/B > 3% ~ 4σ**

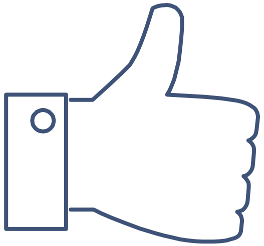
OmniLearn founds the NP with **S/B = 0.7% ~ 2σ**

See also: E. Buhmann, C. Ewen, G. Kasieczka, **V. Mikuni,** B. Nachman, and D. Shih, Phys. Rev. D 109, 055015

Conclusion

- **OmniLearn**: learn a general representation of jets
- Evaluation across **9 different downstream datasets**
- Evaluate the performance on **jet tagging**, **jet generation**, **unfolding**, and **anomaly detection**
- OmniLearn improves upon SOTA or/and converges quicker than models trained from scratch
- Magnify the statistical power of the data: **Not only Big Data benefits from AI**
- **Try it out yourself**: https://github.com/ViniciusMikuni/OmniLearn/ and check out the paper: arXiv:2404.16091





THANKS!

Any questions?

Backup

Comparison Between Models

BERKELEY LAB

Language inspired models

- Data are tokenized
- Unsupervised and general pre-training
- Big models often required

OmniLearn

- Data are continuous
- HEP has one of the best simulators across all sciences: supervised pre-training
- Medium models that can fit on standard GPUs are still useful

Input Dropout

Not all datasets contain the same information:

- Let the model learn with and without some features
- **Feature Dropout**: With fixed probability, set some of the input features to 0

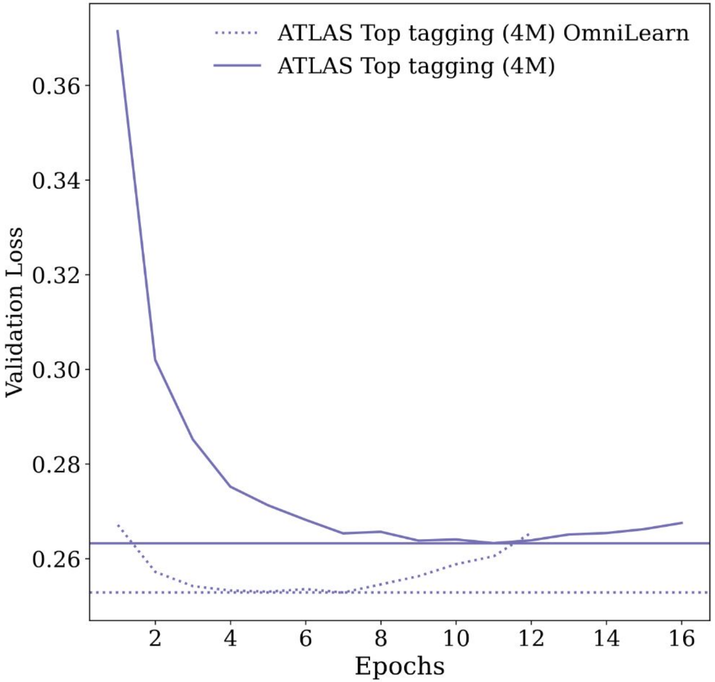
$$f_1, f_2, f_3, f_4, \boxed{f_5, f_6, f_7, f_8}$$

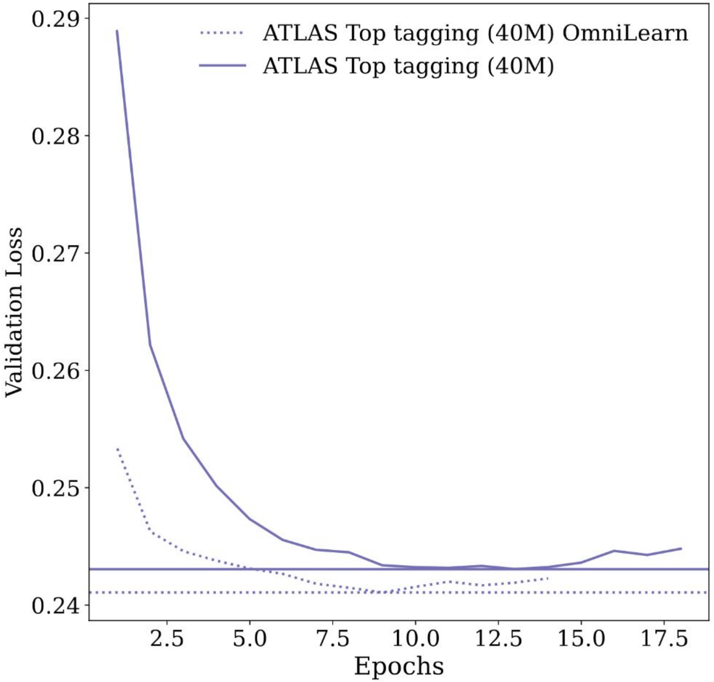
$$\boxed{f_5, f_6, f_7, f_8} \quad \mathbf{p = 0.9}$$

$$\boxed{0,0,0,0} \quad \mathbf{p = 0.1}$$

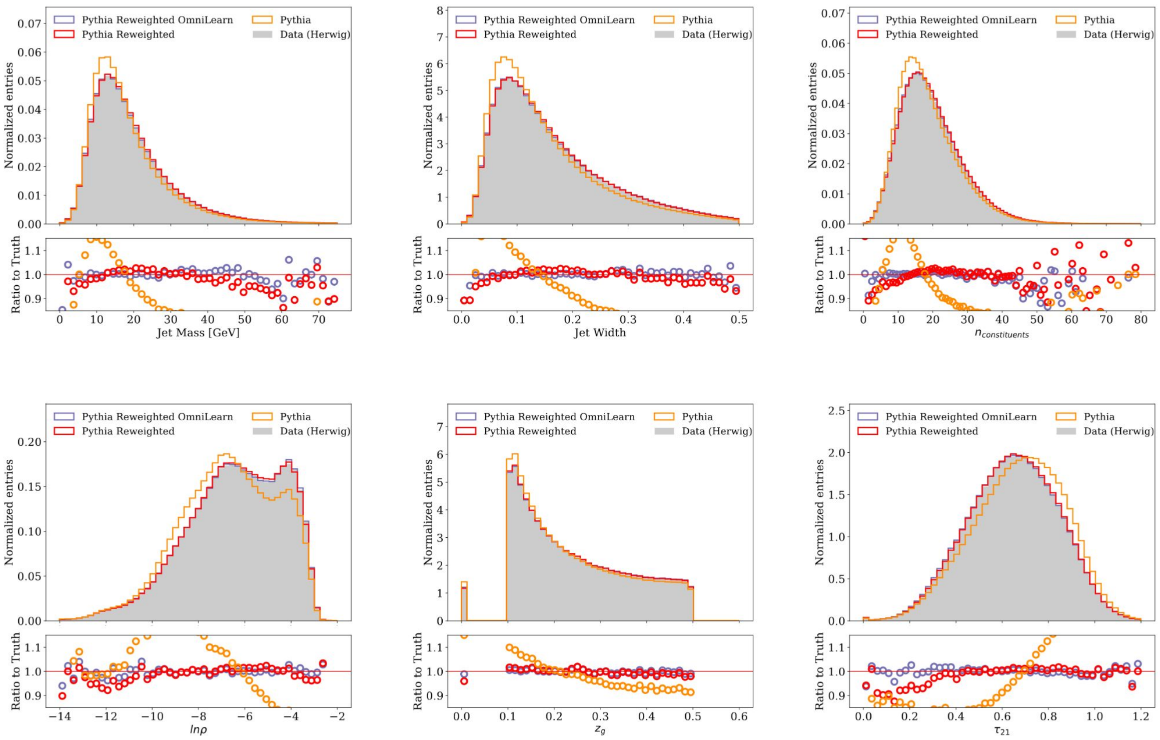
More details at: https://arxiv.org/abs/2404.16091

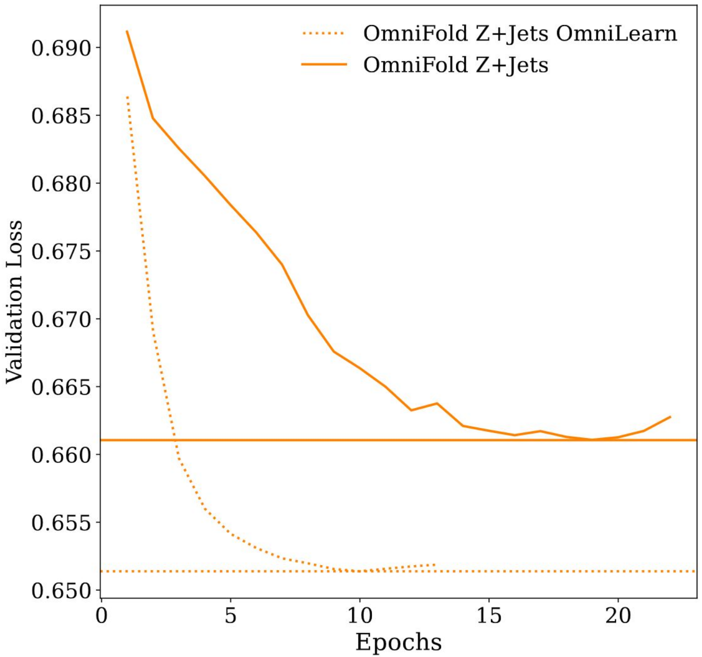
ATLAS Loss Curves



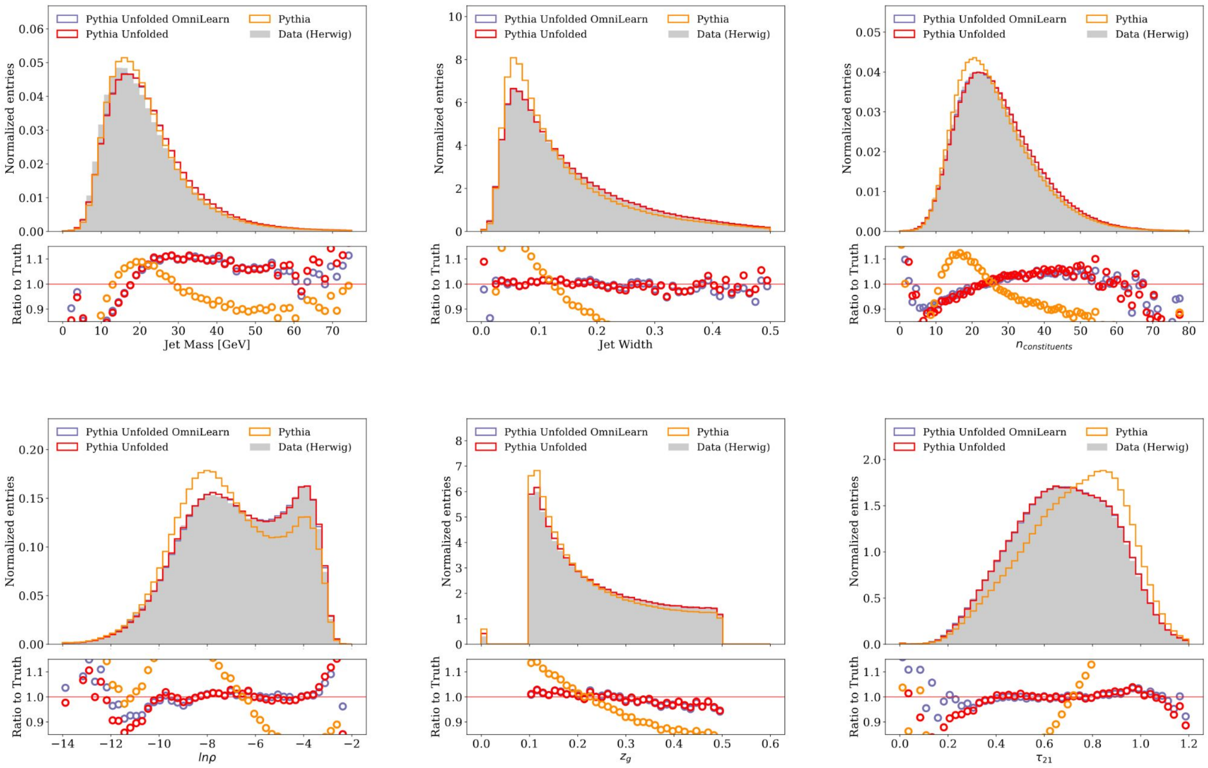


OmniLearn for reweighting

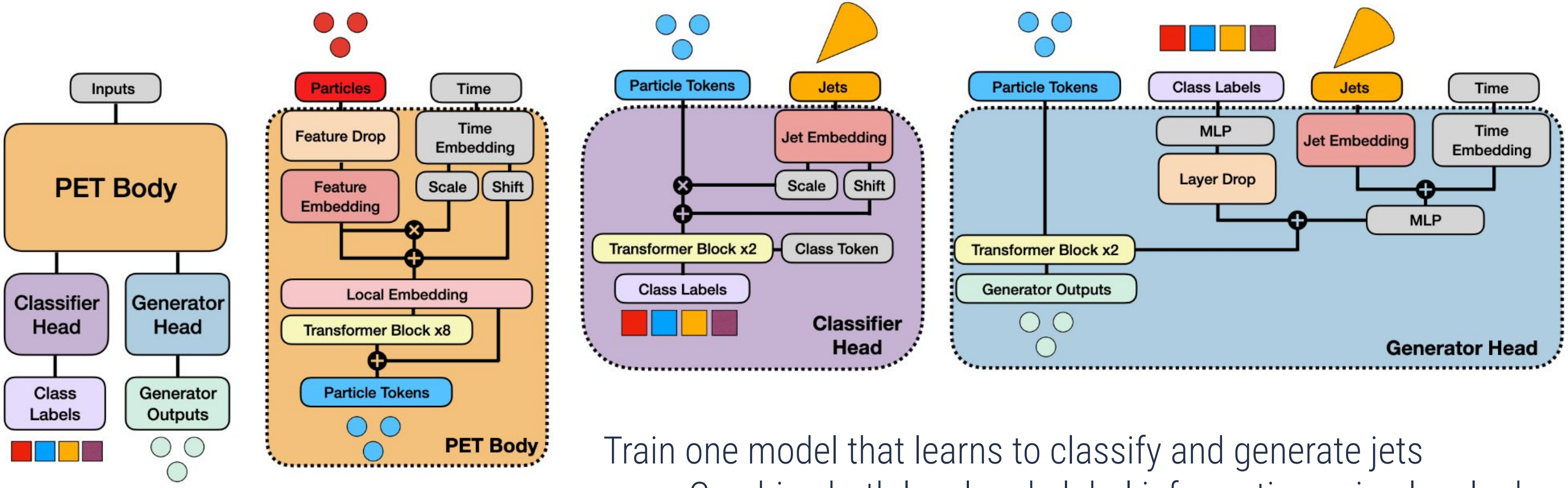




OmniLearn for Unfolding



PET

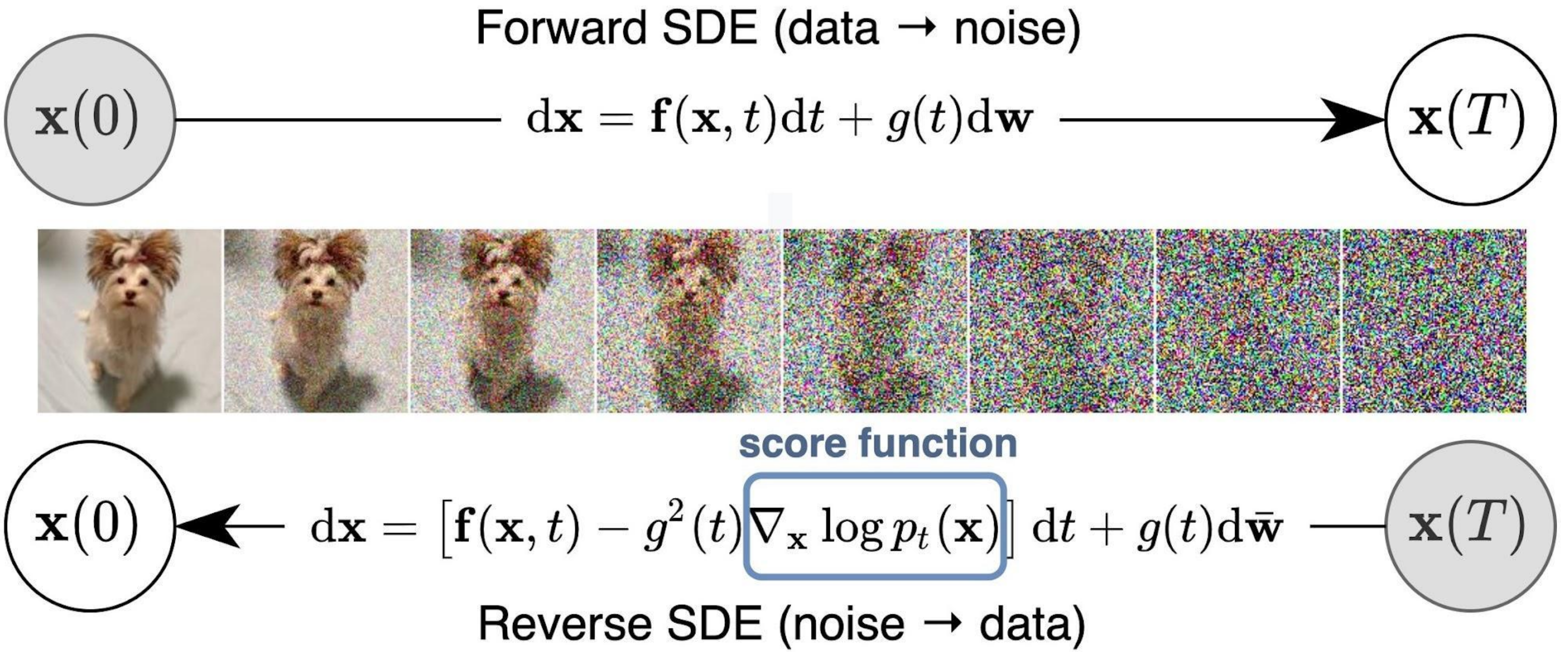


Train one model that learns to classify and generate jets

- Combine both local and global information using local edges and a transformer: **P**oint-**E**dge **T**ransformer

More details at: https://arxiv.org/abs/2404.16091

Diffusion Generative Models



Source: https://yang-song.net/blog/2021/score/

Loss function

BERKELEY LAB

$$\mathcal{L} = \mathcal{L}_{\text{class}} + \mathcal{L}_{\text{gen}} + \mathcal{L}_{\text{class smear}}$$
$$= \text{CE}(y, y_{\text{pred}}) + \|\mathbf{v} - \mathbf{v}_{\text{pred}}\|^2 + \alpha^2 \text{CE}(y, \hat{y}_{\text{pred}})$$

Straightforward loss function:

- **Cross entropy** for each class
- Perturbed data prediction from the **diffusion loss**
- Classification over perturbed inputs: **data augmentation**!

More details at: https://arxiv.org/abs/2404.16091

Diffusion 101

Diffusion models are the go to for data generation

- Simple training: take data **x**, perturb with a Gaussian of mean μ and std σ
- **x' = μ*x + σ*ε, ε~N(0,1)**
- Ask the network to predict the noise injected
- **L = ||D(x') - ε||^2**