Normalizing Flows at the LHC
Preparing for the Future

— ML in HEP, HEPHY/OeAW Vienna —

Claudius Krause

Rutgers, The State University of New Jersey

September 12, 2022

Dr. Claudius Krause

Research Experience

- Effective Field Theories (conceptual & applied)
- Electroweak Phase Transition Beyond the SM
- Machine Learning Applications to simulation & data analyses

CV

- **Dr. rer. nat** (2013–2016) LMU Munich
- **Postdoc** (2016–2018) IFIC Valencia
- **Feodor-Lynen Fellow** (2018–2020) Fermilab
- **Postdoc** (2020–2022) Rutgers University

Talks

- **Seminars** (22) recent: ITP, LBNL, NIKHEF
- **Invited** (26) recent: IAIFI, Snowmass CSS
- **Contributed** (21) recent: MODE, DPF, ML4Jets

Community / Organizer

- "Fast Calorimeter Challenge 2022"
- "Multibosons At The Energy Frontier" workshop at Fermilab
- Pheno-Seminar at Rutgers
- HEP Journal Club at Fermilab

Open Questions in High-Energy Physics.

What's not in the Standard Model

- The Nature of Dark Matter
- Neutrino masses
- The Baryon Asymmetry of the Universe
- Dark Energy / Inflation

Experimental Anomalies

- Flavor Observables
- $(g-2)_\mu$
- Hubble constant H_0

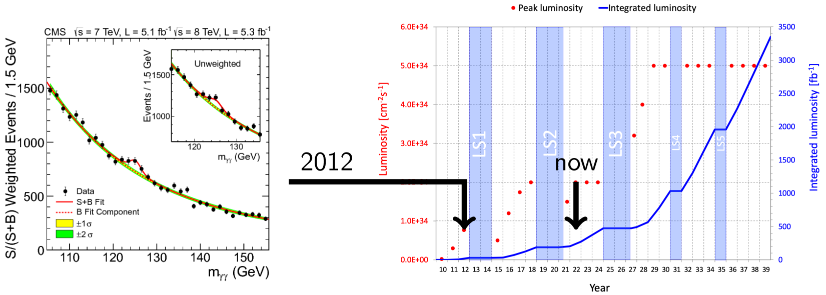
Theoretical Problems

- The Hierarchy Problem
- Origin of Flavor
- Unification of Forces
- Quantum Gravity

Currently explored in Experiments

- Higgs & electroweak Sector of the SM
- Neutrino masses and Hierarchy
- Strong Dynamics
- New Particles & Interactions

We will have a lot more data in the near future.



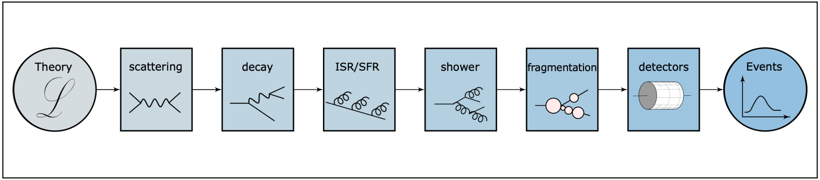
CMS Collaboration [arXiv:1207.7235, Phys.Lett.B]

https://lhc-commissioning.web.cern.ch/schedule/HL-LHC-plots.htm

- We will have 20–25× more data.

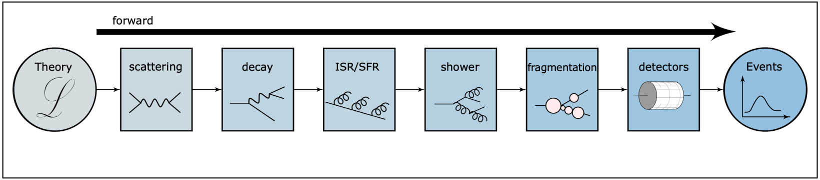
⇒ We want to understand every aspect of it in detail (and find New Physics)!

How do we understand the data based on 1st principles?



Machine Learning and LHC Event Generation, A. Butter, **CK** et al. [2203.07460]

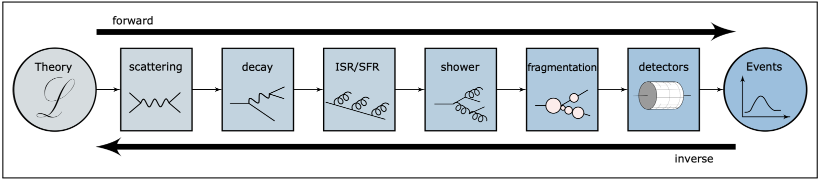
How do we understand the data based on 1st principles?



Machine Learning and LHC Event Generation, A. Butter, **CK** et al. [2203.07460]

1. (A lot of) high-precision simulations.

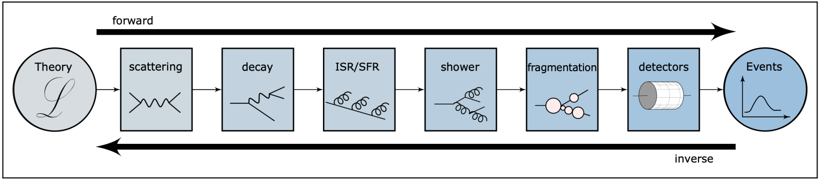
How do we understand the data based on 1st principles?



Machine Learning and LHC Event Generation, A. Butter, **CK** et al. [2203.07460]

1. (A lot of) high-precision simulations.
2. Analyzing high-dimensional data: Simulation-based Inference and data-driven Anomaly Searches.

How do we understand the data based on 1st principles?



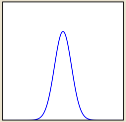
Machine Learning and LHC Event Generation, A. Butter, **CK** et al. [2203.07460]

1. (A lot of) high-precision simulations.
2. Analyzing high-dimensional data: Simulation-based Inference and data-driven Anomaly Searches.

ML has impacted every aspect of the simulation chain, with one class of models being very powerful: **Normalizing Flows**

Normalizing Flows learn a change-of-coordinates efficiently.

"easy" base distribution

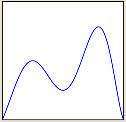


$\Leftrightarrow$

bijective transformation

$\Leftrightarrow$

"target" distribution



$$p(x) = \pi(f(x)) \left| \det \frac{\partial f(x)}{\partial x} \right|$$

density estimation, $p(x)$

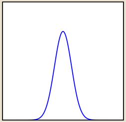
$\longleftarrow$

sample generation

$\longrightarrow$

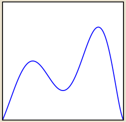
Normalizing Flows learn a change-of-coordinates efficiently.

"easy" base distribution



$\Leftrightarrow$ bijective transformation $\Leftrightarrow$

"target" distribution



$$p(x) = \pi(f(x)) \left| \det \frac{\partial f(x)}{\partial x} \right|$$

density estimation, $p(x)$

$\longleftarrow$

sample generation

$\longrightarrow$

Having access to the log-likelihood (LL) allows several training options:

- $\Rightarrow$ Based on samples: via maximizing LL(samples).
- $\Rightarrow$ Based on target function $f(x)$: via matching $p(x)$ to $f(x)$.

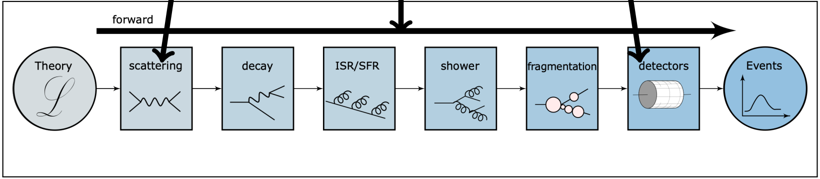
NFs can also be used for inference: learn $p(\text{parameters}|\text{data})$.

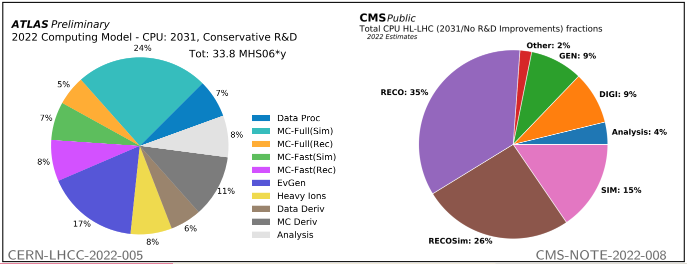
Normalizing Flows attack Bottlenecks in the Analysis Chain

Phase Space Sampling
$\Rightarrow$ sample according to $d\sigma$

Detector Simulation
$\Rightarrow$ sample from $p(\text{showers}|E)$

End-to-End Simulation
$\Rightarrow$ sample from $p(\text{events})$



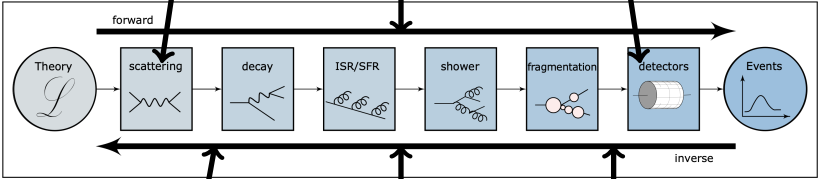


Normalizing Flows increase the Sensitivity in our Analyses

Phase Space Sampling
$\Rightarrow$ sample according to $d\sigma$

Detector Simulation
$\Rightarrow$ sample from $p(\text{showers}|E)$

End-to-End Simulation
$\Rightarrow$ sample from $p(\text{events})$



Bump-Hunt Searches
$\Rightarrow$ use $p(\text{data})$ as bg estimate

Inference
$\Rightarrow$ learn $p(\text{parameters}|\text{data})$

Unfolding
$\Rightarrow$ learn $p(\text{parton}|\textit{event})$

Lattice QCD
$\Rightarrow$ improve MCMC proposals

(My Contributions to) Normalizing Flows at the LHC

Phase Space Sampling

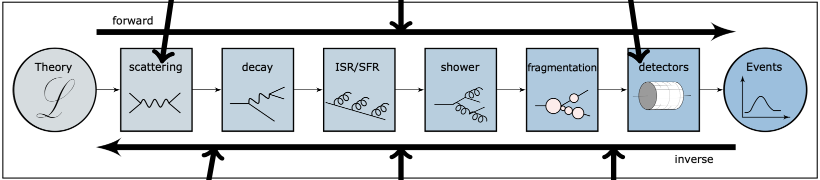
2001.05478, CK: 2001.05486, 2001.10028

Detector Simulation

CK: 2106.05285, 2110.11377

End-to-End Simulation

2110.13632, 2205.01697



Bump-Hunt Searches

2001.04990, CK: 2109.00546, 2203.09470

Inference

2012.09873, 2207.00664

Unfolding

2006.06685

Lattice QCD

1904.12072, 2003.06413

How do Normalizing Flows tame Jacobians?

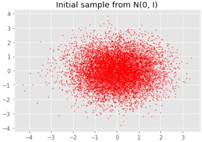
- NFs learn the parameters θ of a series of easy transformations.
  Dinh et al. [arXiv:1410.8516], Rezende/Mohamed [arXiv:1505.05770]

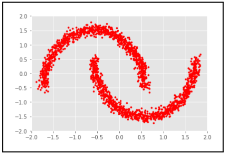
- Each transformation is 1d & has an analytic Jacobian and inverse.
  $\Rightarrow$ We use Rational Quadratic Splines
  Durkan et al. [arXiv:1906.04032], Gregory/Delbourgo [IMA J. of Num. An., '82]

- Require a triangular Jacobian for faster evaluation.
  $\Rightarrow$ The parameters θ depend only on a subset of all other coordinates.

How do Normalizing Flows tame Jacobians?

- NFs learn the parameters θ of a series of easy transformations.
  Dinh et al. [arXiv:1410.8516], Rezende/Mohamed [arXiv:1505.05770]
- Each transformation is 1d & has an analytic Jacobian and inverse.
  $\Rightarrow$ We use Rational Quadratic Splines
  Durkan et al. [arXiv:1906.04032], Gregory/Delbourgo [IMA J. of Num. An., '82]
- Require a triangular Jacobian for faster evaluation.
  $\Rightarrow$ The parameters θ depend only on a subset of all other coordinates.

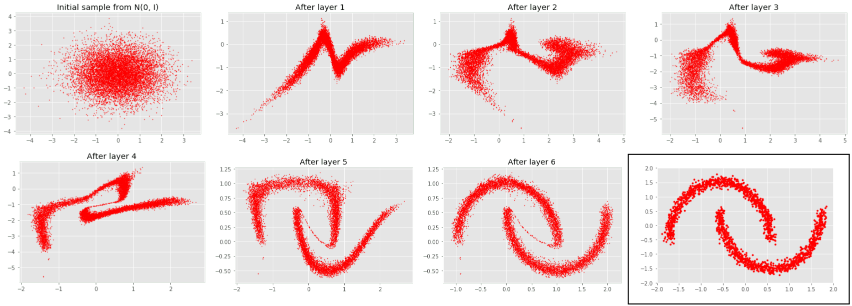




https://engineering.papercup.com/posts/normalizing-flows-part-2/

How do Normalizing Flows tame Jacobians?

- NFs learn the parameters θ of a series of easy transformations.
  Dinh et al. [arXiv:1410.8516], Rezende/Mohamed [arXiv:1505.05770]
- Each transformation is 1d & has an analytic Jacobian and inverse.
  $\Rightarrow$ We use Rational Quadratic Splines
  Durkan et al. [arXiv:1906.04032], Gregory/Delbourgo [IMA J. of Num. An., '82]
- Require a triangular Jacobian for faster evaluation.
  $\Rightarrow$ The parameters θ depend only on a subset of all other coordinates.



https://engineering.papercup.com/posts/normalizing-flows-part-2/

How do Normalizing Flows tame Jacobians?

- NFs learn the parameters θ of a series of easy transformations.
  Dinh et al. [arXiv:1410.8516], Rezende/Mohamed [arXiv:1505.05770]
- Each transformation is 1d & has an analytic Jacobian and inverse.
  $\Rightarrow$ We use Rational Quadratic Splines
  Durkan et al. [arXiv:1906.04032], Gregory/Delbourgo [IMA J. of Num. An., '82]
- Require a triangular Jacobian for faster evaluation.
  $\Rightarrow$ The parameters θ depend only on a subset of all other coordinates.

Autoregressive Blocks (MAF/IAF)

- Coordinates are transformed autoregressivly $\Rightarrow$ $\boxed{\theta_{x_i}(x_{j<i})}$

+ Are very powerful.
− Have a fast and a slow direction.

Bipartite Blocks (Coupling Layers)

- Coordinates are split in 2 sets, transforming each other $\Rightarrow$

$$\boxed{\theta_{x\in A}(x \in B) \quad \& \quad \theta_{x\in B}(x \in A)}$$

+ Are equally fast in both directions.
− Are not as expressive.

(My Contributions to) Normalizing Flows at the LHC

1. Phase Space Sampling

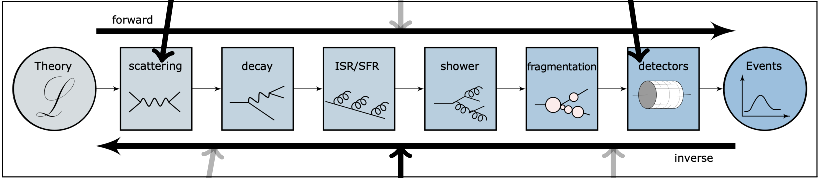
2001.05478, **CK: 2001.05486, 2001.10028**

Detector Simulation

CK: 2106.05285, 2110.11377

End-to-End Simulation

2110.13632, 2205.01697



Bump-Hunt Searches

2001.04990, **CK: 2109.00546,** 2203.09470

Inference

2012.09873, 2207.00664

Unfolding

2006.06685

Lattice QCD

1904.12072, 2003.06413

Phase Space integration uses Importance Sampling.

$$I = \int_0^1 f(\vec{x})\, d\vec{x} \quad \xrightarrow{\text{MC}} \quad \frac{1}{N}\sum_i f(\vec{x}_i) \qquad \vec{x}_i \ldots \text{uniform}, \quad \sigma_{\text{MC}}(I) \sim \frac{1}{\sqrt{N}}$$

$$= \int_0^1 \frac{f(\vec{x})}{q(\vec{x})}\, q(\vec{x}) d\vec{x} \quad \xrightarrow[\text{importance sampling}]{\text{MC}} \quad \frac{1}{N}\sum_i \frac{f(\vec{x}_i)}{q(\vec{x}_i)} \qquad \vec{x}_i \ldots q(\vec{x}),$$

In the limit $q(\vec{x}) \propto f(\vec{x})$, we get $\sigma_{\text{IS}}(I) = 0$

We therefore have to find a $q(\vec{x})$ that approximates the shape of $f(\vec{x})$.

$\Rightarrow$ Once found, we can use it for event generation,
i.e. sampling p_i, ϑ_i, and φ_i according to $d\sigma(p_i, \vartheta_i, \varphi_i)$

Phase Space integration uses Importance Sampling.

$$I = \int_0^1 f(\vec{x})\, d\vec{x} \quad \xrightarrow{\text{MC}} \quad \frac{1}{N}\sum_i f(\vec{x}_i) \qquad \vec{x}_i \ldots \text{uniform}, \quad \sigma_{\text{MC}}(I) \sim \frac{1}{\sqrt{N}}$$

$$= \int_0^1 \frac{f(\vec{x})}{q(\vec{x})}\, q(\vec{x}) d\vec{x} \quad \xrightarrow[\text{importance sampling}]{\text{MC}} \quad \frac{1}{N}\sum_i \frac{f(\vec{x}_i)}{q(\vec{x}_i)} \qquad \vec{x}_i \ldots q(\vec{x}),$$

In the limit $q(\vec{x}) \propto f(\vec{x})$, we get $\sigma_{\text{IS}}(I) = 0$

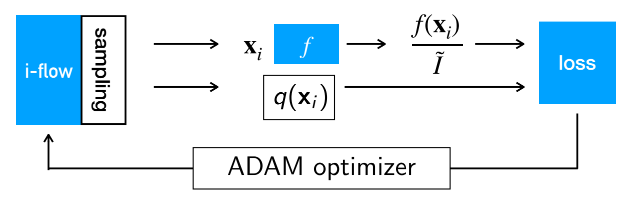
We therefore have to find a $q(\vec{x})$ that approximates the shape of $f(\vec{x})$.

$\Rightarrow$ Once found, we can use it for event generation,
i.e. sampling p_i, ϑ_i, and φ_i according to $d\sigma(p_i, \vartheta_i, \varphi_i)$

We need both samples x and their probability $q(x)$.
$\Rightarrow$ We use a bipartite, coupling-layer-based Flow.

i-flow: Numerical Integration with Normalizing Flows.

How it works:



i-flow: C. Gao, J. Isaacson, **CK** [arXiv:2001.05486, ML:ST]
gitlab.com/i-flow/i-flow

Statistical Divergences are used as loss functions:

- Kullback-Leibler (KL) divergence:
  $D_{KL} = \int p(x) \log \frac{p(x)}{q(x)} dx \quad \approx \quad \frac{1}{N} \sum \frac{p(x_i)}{q(x_i)} \log \frac{p(x_i)}{q(x_i)}, \qquad x_i \ldots q(x)$

Sherpa needs a high-dimensional integrator.

Sherpa is a Monte Carlo event generator for the **S**imulation of **H**igh-**E**nergy **R**eactions of **PA**rticles. We use Sherpa to

- compute the matrix element of the process.
- map the unit-hypercube of our integration domain to momenta and angles. To improve efficiency, Sherpa uses a recursive multichannel algorithm.

$$\boxed{\Rightarrow n_{dim} = \underbrace{3n_{final} - 4}_{\text{kinematics}} + \underbrace{n_{final} - 1}_{\text{multichannel}}}$$

https://sherpa.hepforge.org/

Sherpa needs a high-dimensional integrator.

Sherpa is a Monte Carlo event generator for the **S**imulation of **H**igh-**E**nergy **R**eactions of **PA**rticles. We use Sherpa to

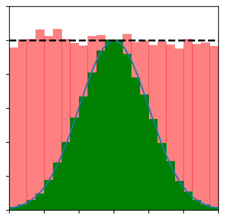
- compute the matrix element of the process.
- map the unit-hypercube of our integration domain to momenta and angles. To improve efficiency, Sherpa uses a recursive multichannel algorithm.

$$\boxed{\Rightarrow n_{dim} = \underbrace{3n_{final} - 4}_{\text{kinematics}} + \underbrace{n_{final} - 1}_{\text{multichannel}}}$$

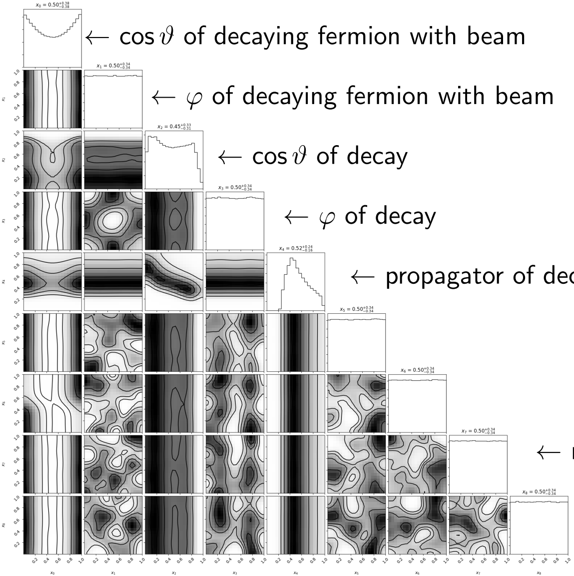
https://sherpa.hepforge.org/

Figure of merit: Unweighting efficiency

- Unweighting: we need to accept/reject each event with probability $\frac{f(x_i)}{\max f(x)}$. The kept events are unweighted and reproduce the shape of $f(x)$.
- The unweighting efficiency is the fraction of events that "survives" this procedure.



An easy example: $e^+e^- \to 3j$.



$\leftarrow \cos\vartheta$ of decaying fermion with beam

$\leftarrow \varphi$ of decaying fermion with beam

Target distribution

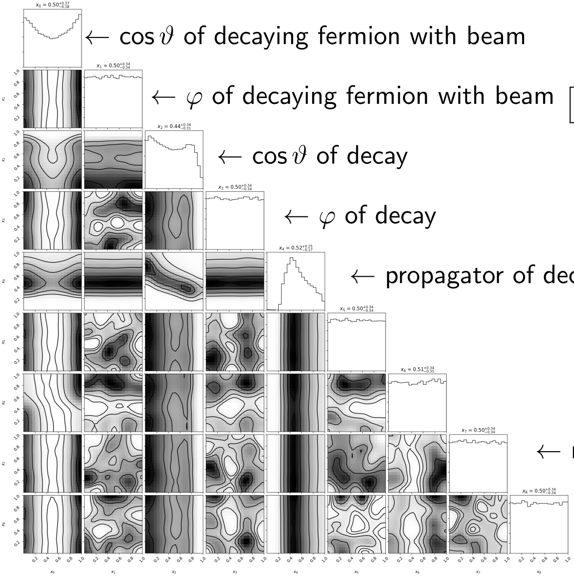
$\leftarrow \cos\vartheta$ of decay

$\leftarrow \varphi$ of decay

$\leftarrow$ propagator of decaying fermion

$\leftarrow$ multichannel

An easy example: $e^+e^- \to 3j$.



$\leftarrow \cos\vartheta$ of decaying fermion with beam

$\leftarrow \varphi$ of decaying fermion with beam

Learned distribution

$\leftarrow \cos\vartheta$ of decay

$\leftarrow \varphi$ of decay

$\leftarrow$ propagator of decaying fermion

$\leftarrow$ multichannel

High Multiplicities are difficult to learn in this setup.

unweighting efficiency $\langle w \rangle / w_{\max}$		LO QCD $n=0$	$n=1$	$n=2$	$n=3$
$W^+ + n$ jets	Sherpa	$2.8 \cdot 10^{-1}$	$3.8 \cdot 10^{-2}$	$7.5 \cdot 10^{-3}$	$1.5 \cdot 10^{-3}$
	`i-flow`	$6.1 \cdot 10^{-1}$	$1.2 \cdot 10^{-1}$	$1.0 \cdot 10^{-2}$	$1.8 \cdot 10^{-3}$
	Gain	**2.2**	**3.3**	**1.4**	**1.2**
$W^- + n$ jets	Sherpa	$2.9 \cdot 10^{-1}$	$4.0 \cdot 10^{-2}$	$7.7 \cdot 10^{-3}$	$2.0 \cdot 10^{-3}$
	`i-flow`	$7.0 \cdot 10^{-1}$	$1.5 \cdot 10^{-1}$	$1.1 \cdot 10^{-2}$	$2.2 \cdot 10^{-3}$
	Gain	**2.4**	**3.3**	**1.4**	**1.1**
$Z + n$ jets	Sherpa	$3.1 \cdot 10^{-1}$	$3.6 \cdot 10^{-2}$	$1.5 \cdot 10^{-2}$	$4.7 \cdot 10^{-3}$
	`i-flow`	$3.8 \cdot 10^{-1}$	$1.0 \cdot 10^{-1}$	$1.4 \cdot 10^{-2}$	$2.4 \cdot 10^{-3}$
	Gain	**1.2**	**2.9**	**0.91**	**0.51**

C. Gao, S. Höche, J. Isaacson, **CK**, H. Schulz [arXiv:2001.10028, PRD]

High Multiplicities are difficult to learn in this setup.

unweighting efficiency $\langle w \rangle / w_{\max}$		LO QCD $n=0$	$n=1$	$n=2$	$n=3$
$W^+ + n$ jets	Sherpa	$2.8 \cdot 10^{-1}$	$3.8 \cdot 10^{-2}$	$7.5 \cdot 10^{-3}$	$1.5 \cdot 10^{-3}$
	`i-flow`	$6.1 \cdot 10^{-1}$	$1.2 \cdot 10^{-1}$	$1.0 \cdot 10^{-2}$	$1.8 \cdot 10^{-3}$
	Gain	**2.2**	**3.3**	**1.4**	**1.2**
$W^- + n$ jets	Sherpa	$2.9 \cdot 10^{-1}$	$4.0 \cdot 10^{-2}$	$7.7 \cdot 10^{-3}$	$2.0 \cdot 10^{-3}$
	`i-flow`	$7.0 \cdot 10^{-1}$	$1.5 \cdot 10^{-1}$	$1.1 \cdot 10^{-2}$	$2.2 \cdot 10^{-3}$
	Gain	**2.4**	**3.3**	**1.4**	**1.1**
$Z + n$ jets	Sherpa	$3.1 \cdot 10^{-1}$	$3.6 \cdot 10^{-2}$	$1.5 \cdot 10^{-2}$	$4.7 \cdot 10^{-3}$
	`i-flow`	$3.8 \cdot 10^{-1}$	$1.0 \cdot 10^{-1}$	$1.4 \cdot 10^{-2}$	$2.4 \cdot 10^{-3}$
	Gain	**1.2**	**2.9**	**0.91**	**0.51**

C. Gao, S. Höche, J. Isaacson, **CK**, H. Schulz [arXiv:2001.10028, PRD]

Improvements:

- make channel number a conditional variable and learn it separately.
- re-use matrix elements multiple times.
- introduce learnable soft permutations, use VEGAS for base dist.

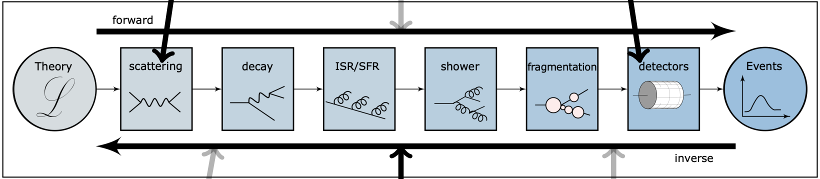
A. Butter, T. Heimel, J. Isaacson, **CK**, F. Maltoni, O. Mattelaer, T. Plehn, R. Winterhalder [in preparation]

(My Contributions to) Normalizing Flows at the LHC

Phase Space Sampling
2001.05478, CK: **2001.05486**, **2001.10028**

2. Detector Simulation
CK: **2106.05285**, **2110.11377**

End-to-End Simulation
2110.13632, 2205.01697



Bump-Hunt Searches
2001.04990, CK: **2109.00546**, 2203.09470

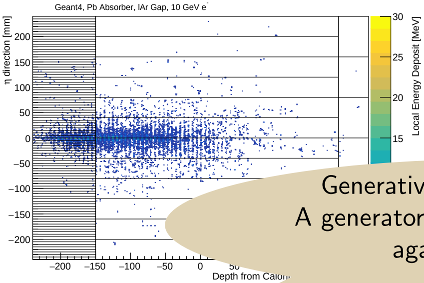
Inference
2012.09873, 2207.00664

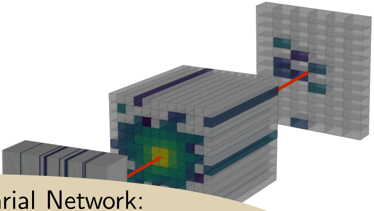
Unfolding
2006.06685

Lattice QCD
1904.12072, 2003.06413

We use the same calorimeter geometry as CaloGAN

- We consider a toy calorimeter inspired by the ATLAS ECal: flat alternating layers of lead and LAr
- They form three instrumented layers of dimension 3×96, 12×12, and 12×6



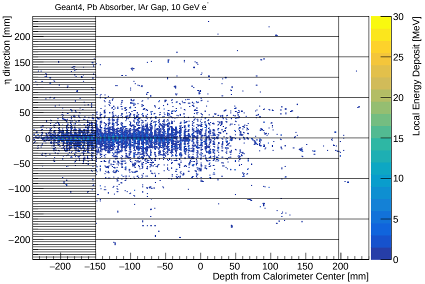


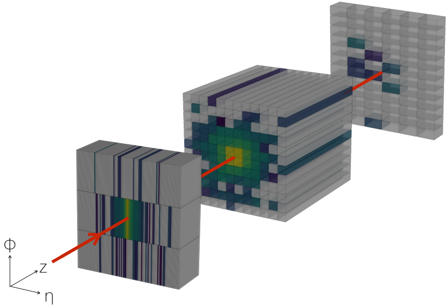
Generative Adversarial Network:
A generator and a critic play a game against each other.

CaloGAN: Paganini, de Oliveira, Nachman [1705.02355, PRL; 1712.10321, PRD]

We use the same calorimeter geometry as CaloGAN

- We consider a toy calorimeter inspired by the ATLAS ECal: flat alternating layers of lead and LAr
- They form three instrumented layers of dimension 3×96, 12×12, and 12×6

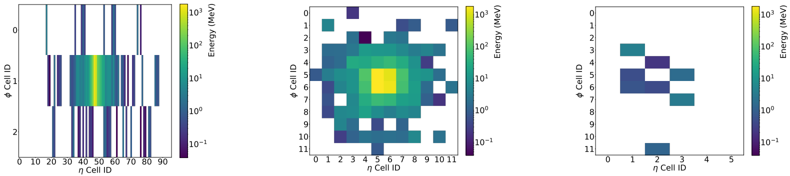




CaloGAN: Paganini, de Oliveira, Nachman [1705.02355, PRL; 1712.10321, PRD]

We use the same calorimeter geometry as CaloGAN

- The Geant4 configuration of CaloGAN is available at
  https://github.com/hep-lbdl/CaloGAN
- We produce our own dataset: available at [DOI: 10.5281/zenodo.5904188]
- Showers of e^+, γ, and π^+ (100k each)
- All are centered and perpendicular
- E_{inc} is uniform in $[1, 100]$ GeV and given in addition to the energy deposits per voxel:



CaloGAN: Paganini, de Oliveira, Nachman [1705.02355, PRL; 1712.10321, PRD]

CaloFlow uses a 2-step approach to learn $p(\vec{\mathcal{I}}|E_{\rm inc})$.

Flow I

- learns $p_1(E_0, E_1, E_2|E_{\rm inc})$
- is optimized using the log-likelihood.

Flow II

- learns $p_2(\hat{\vec{\mathcal{I}}}|E_0, E_1, E_2, E_{\rm inc})$ of normalized showers
- in CaloFlow v1 (2106.05285 — called "teacher"):
  - Masked Autoregressive Flow trained with log-likelihood
  - Slow in sampling ($\approx 500\times$ slower than CaloGAN)
- in CaloFlow v2 (2110.11377 — called "student"):
  - Inverse Autoregressive Flow trained with Probability Density Distillation from teacher (log-likelihood prohibitive)

    van den Oord et al. [1711.10433]

    i.e. matching IAF parameters to frozen MAF
  - Fast in sampling ($\approx 500\times$ faster than CaloFlow v1)

A Classifier provides the “ultimate metric”.

According to the Neyman-Pearson Lemma we have:

- The likelihood ratio is the most powerful test statistic to distinguish the two samples.
- A powerful classifier trained to distinguish the samples should therefore learn (something monotonically related to) this.
- If this classifier is confused, we conclude $p_{\text{GEANT4}}(x) = p_{\text{generated}}(x)$

$\Rightarrow$ This captures the full 504-dim. space.

? But why wasn't this used before?

$\Rightarrow$ Previous deep generative models were separable to almost 100%!

DCTRGAN: Diefenbacher et al. [2009.03796, JINST]

CaloFlow passes the "ultimate metric" test.

According to the Neyman-Pearson Lemma we have:
$p_{\text{Geant4}}(x) = p_{\text{generated}}(x)$ if a classifier cannot distinguish data from generated samples.

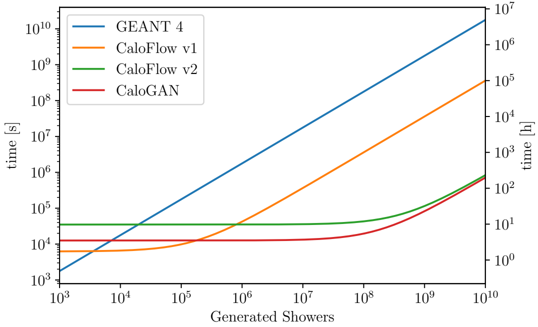
AUC		DNN based classifier		
		Geant4 vs. CaloGAN	Geant4 vs. (teacher) CaloFlow v1	Geant4 vs. (student) CaloFlow v2
e^+	unnorm.	1.000(0)	0.859(10)	0.786(7)
	norm.	1.000(0)	0.870(2)	0.824(4)
γ	unnorm.	1.000(0)	0.756(48)	0.758(14)
	norm.	1.000(0)	0.796(2)	0.760(3)
π^+	unnorm.	1.000(0)	0.649(3)	0.729(2)
	norm.	1.000(0)	0.755(3)	0.807(1)

AUC $(\in [0.5, 1])$: Area Under the ROC Curve, smaller is better, i.e. more confused

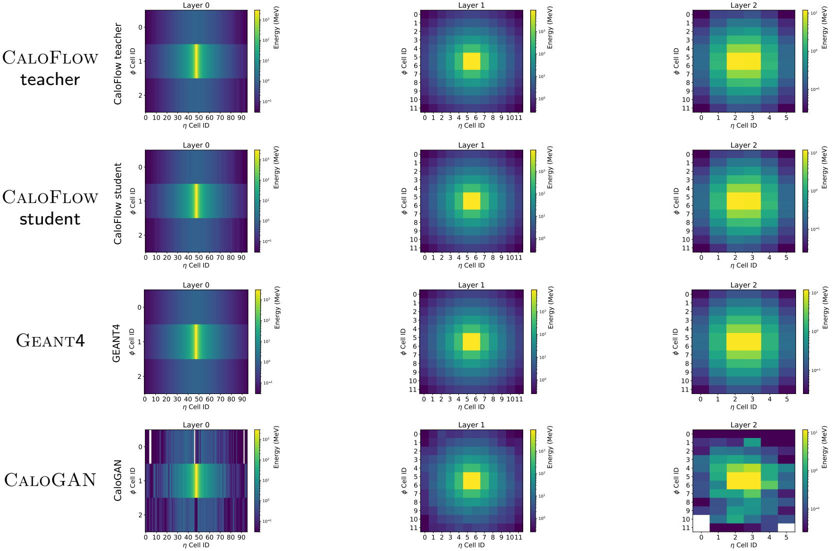
Sampling Speed: The Student beats the Teacher!

	CaloFlow*		CaloGAN*	Geant4†
	teacher	student		
training	22+82 min	+ 480 min	210 min	0 min
generation time per shower	36.2 ms	**0.08 ms**	**0.07 ms**	1772 ms

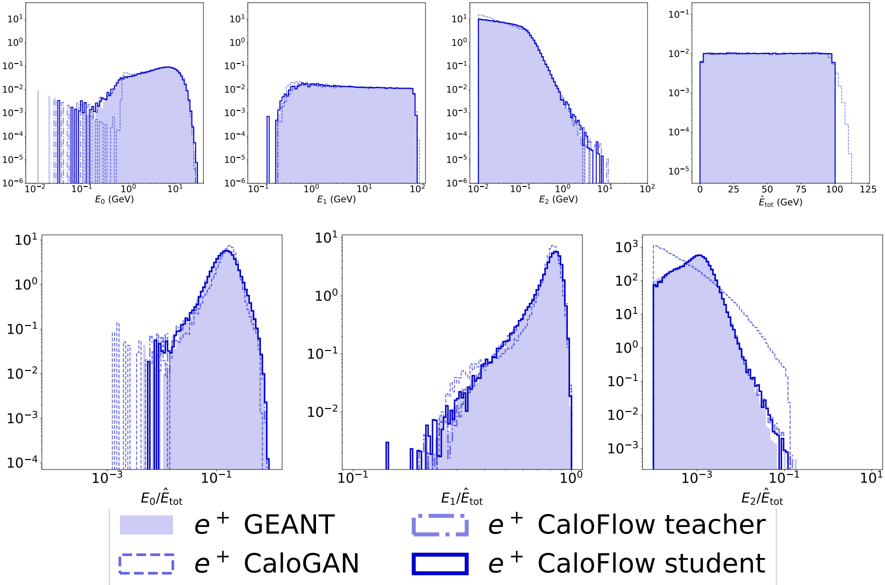
*: on our Titan V GPU, †: on the CPU of CaloGAN: Paganini, de Oliveira, Nachman [1712.10321, PRD]



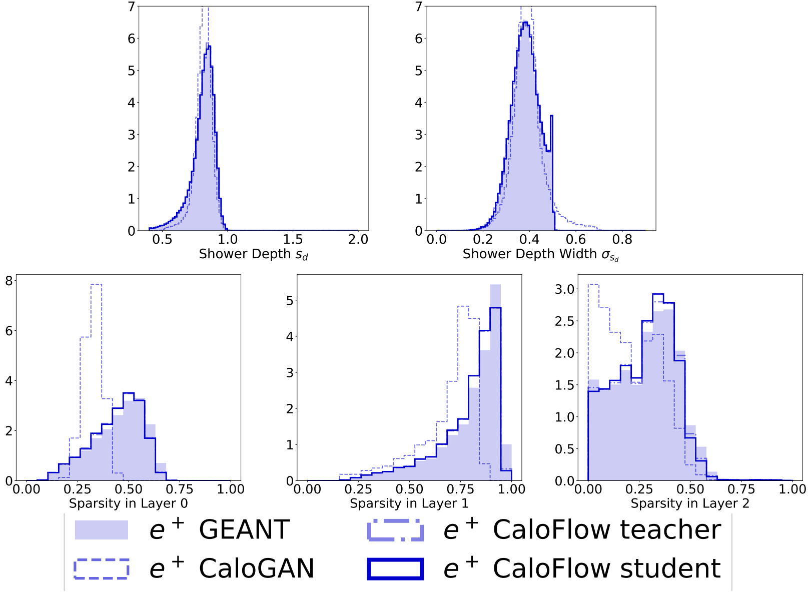
CaloFlow: Comparing Shower Averages: e^+



CaloFlow: histograms: e^+



CaloFlow: histograms: e^+



Going the next step: towards deployment in FastSimulation

We have a rapidly evolving field: need a survey of current approaches on a common dataset!

$\Rightarrow$ Fast Calorimeter Challenge 2022 https://calochallenge.github.io/homepage/

Michele Faucci Giannelli, Gregor Kasieczka, **CK**, Ben Nachman, Dalila Salamani, David Shih, and Anna Zaborowska

- Dataset 1: AtlFast3 trainig data (γ: 368, π: 533 voxels)
  [2109.02551, Comput.Softw.Big Sci.]
- Dataset 2: simulated detector (e^-: 6480 voxels)
- Dataset 3: simulated detector (e^-: 40500 voxels)

Submissions will be presented at ML4Jets in November.

Going the next step: towards deployment in FastSimulation

We have a rapidly evolving field: need a survey of current approaches on a common dataset!

⇒ Fast Calorimeter Challenge 2022 https://calochallenge.github.io/homepage/

Michele Faucci Giannelli, Gregor Kasieczka, **CK**, Ben Nachman, Dalila Salamani, David Shih, and Anna Zaborowska

- Dataset 1: AtlFast3 trainig data (γ: 368, π: 533 voxels)
  [2109.02551, Comput.Softw.Big Sci.] CK et al. [in preparation]
- Dataset 2: simulated detector (e^-: 6480 voxels)
  CK et al. [in preparation]
- Dataset 3: simulated detector (e^-: 40500 voxels)
  CK et al. [in preparation]

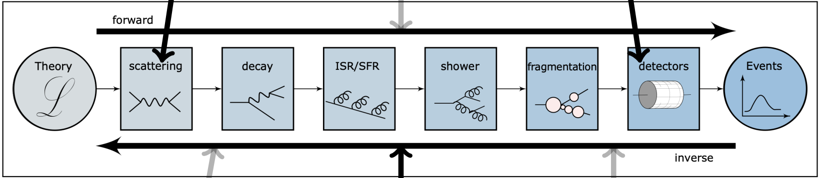
Submissions will be presented at ML4Jets in November.

(My Contributions to) Normalizing Flows at the LHC

Phase Space Sampling
2001.05478, **CK: 2001.05486, 2001.10028**

Detector Simulation
CK: 2106.05285, 2110.11377

End-to-End Simulation
2110.13632, 2205.01697



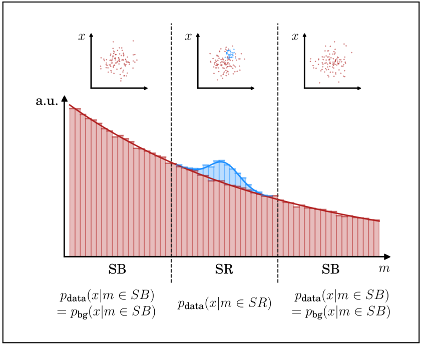
3. Bump-Hunt Searches
2001.04990, **CK: 2109.00546,** 2203.09470

Inference
2012.09873, 2207.00664

Unfolding
2006.06685

Lattice QCD
1904.12072, 2003.06413

Bump Hunts have few model assumptions.



Assumptions

- signal is localized in m
- background in m is smooth
- $\exists$ additional discriminating features x

Select events with

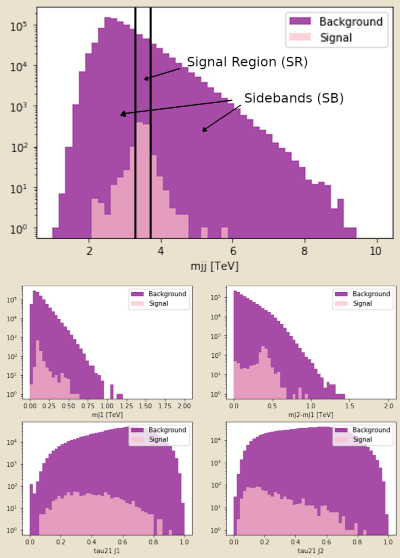
$$\Rightarrow \frac{p_{\text{data}}}{p_{\text{background}}} \sim \frac{p_{\text{signal}}}{p_{\text{background}}}$$

Bump Hunts have few model assumptions.

LHC Olympics R&D dataset:

- 1,000,000 QCD dijet events
- 1,000 signal events
  $W' \to X(\to qq)Y(\to qq)$
- $m_{W'} = 3.5\text{TeV}$,
  $m_X = 500\text{GeV}, \quad m_Y = 100\text{GeV}$
- In SR, $3.3\text{TeV} < m_{JJ} < 3.7\text{TeV}$:
  - 121,352 bg events
  - 772 sg events
- $S/\sqrt{B} = 2.2$

LHCO: G. Kasieczka et al. [2101.08320]



Simulation-based approaches are model-dependent.

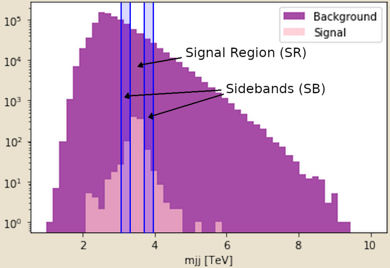
Simulation-based approaches:

- fully supervised:
  train classifier on simulated signal and background
  - depends on quality of simulation
  - high signal model dependence
  - provides upper limit on all approaches
- idealized anomaly detector:
  train classifier on data and simulated background
  - depends on quality of simulation
  - still background model dependent
  - provides upper limit on data-driven anomaly detection

Data-driven approaches are background model-independent.

Classification without Labels (CWoLa) Hunting:

- assume $p_{\text{bg, SR}}(m_{JJ}, x) = p_{\text{data, SB}}(m_{JJ}, x)$
- train classifier between data (SR) and data (SB)
- not robust against correlations

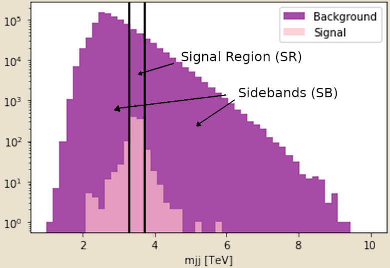


E.M. Metodiev, B. Nachman, J. Thaler, [1708.02949 JHEP]
J.H. Collins, K. Howe, B. Nachman, [1805.02664 PRL, 1902.02634 PRD]

Data-driven approaches are background model-independent.

Anomaly Detection with Density Estimation (ANODE):

- train "outer" density estimator $p_{\text{data}}(x|m_{JJ} \in SB)$
- train "inner" density estimator $p_{\text{data}}(x|m_{JJ} \in SR)$
- compute $\frac{p_{\text{inner}}(x|m_{JJ})}{p_{\text{outer}}(x|m_{JJ})}$ for $m_{JJ} \in SR$



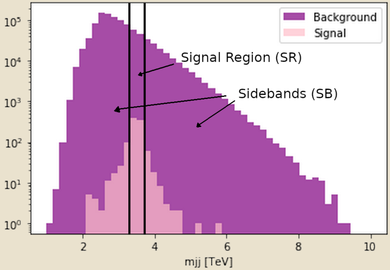
- robust against correlations, but harder learning task.

B. Nachman, D. Shih, [2001.04990, PRD]

Data-driven approaches are background model-independent.

Classifying Anomalies THrough Outer Density Estimation (CATHODE):

- train "outer" density estimator $p_{\text{data}}(x|m_{JJ} \in SB)$
- sample "artificial" events from $p_{\text{outer}}(x|m_{JJ} \in SR)$
- can also oversample
- train a classifier on these samples vs data



$\Rightarrow$ combines the best of CWoLa-Hunting and ANODE!

A. Hallin, J. Isaacson, G. Kasieczka, **CK**, B. Nachman, T. Quadfasel, M. Schlaffer, D. Shih, M. Sommerhalder [2109.00546, PRD]

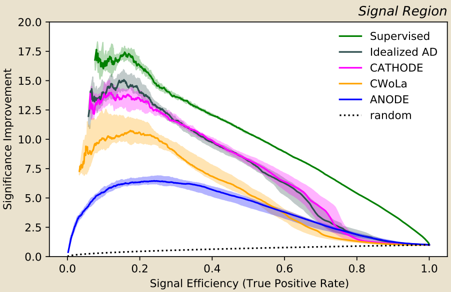
CATHODE outperforms other anomaly detectors.

Results:

- showing $\mathrm{SIC} = \mathrm{TPR}/\sqrt{\mathrm{FPR}}$
- CATHODE approaches idealized AD
- outperforms ANODE (only 1 density estimator)
- outperforms CWoLa (robust against correlations)

A. Hallin, **CK** et al. [2109.00546, PRD]

Significance Improvement Characteristic



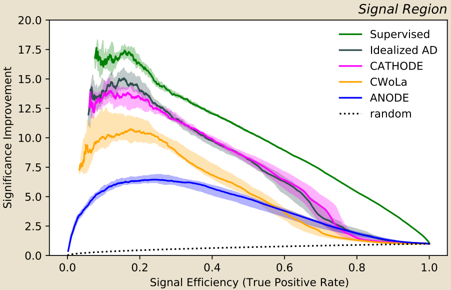
CATHODE outperforms other anomaly detectors.

Results:

- showing $\text{SIC} = \text{TPR}/\sqrt{\text{FPR}}$
- CATHODE approaches idealized AD
- outperforms ANODE (only 1 density estimator)
- outperforms CWoLa (robust against correlations)

A. Hallin, **CK** et al. [2109.00546, PRD]

Significance Improvement Characteristic



⇒ These strategies are now being explored in ATLAS and CMS.

ATLAS [2005.02983, PRL]

Normalizing Flows at the LHC
Preparing for the Future

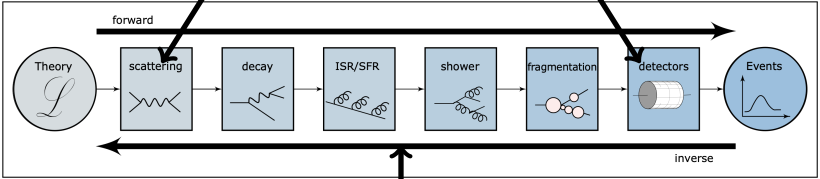
- We expect 25$\times$ more LHC data in the future.
- Understanding everything based on 1st principles suffers from computational bottlenecks that can be tackled with ML, and especially Normalizing Flows.

Normalizing Flows at the LHC
Preparing for the Future

- We expect 25× more LHC data in the future.
- Understanding everything based on 1st principles suffers from computational bottlenecks that can be tackled with ML, and especially Normalizing Flows.

1. Phase Space Sampling

2. Detector Simulation

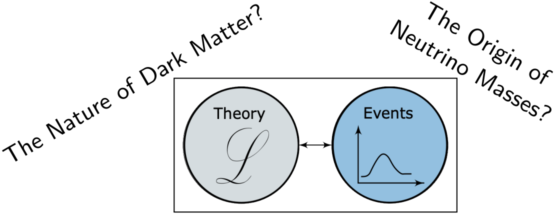


3. Bump-Hunt Searches

Normalizing Flows at the LHC
Preparing for the Future

⇒ With more efficiency and more sensitivity, we will be able to use the LHC to its full potential.

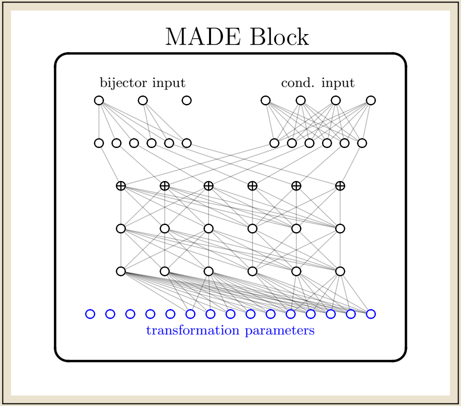
⇒ Normalizing Flows — density estimators and generative models — will help with this endeavor.



The Baryon Asymmetry
of the Universe?

Backup

Taming Jacobians 1: with Autoregressive Blocks



$$\boxed{\theta_{x_i}(x_{j<i})}$$

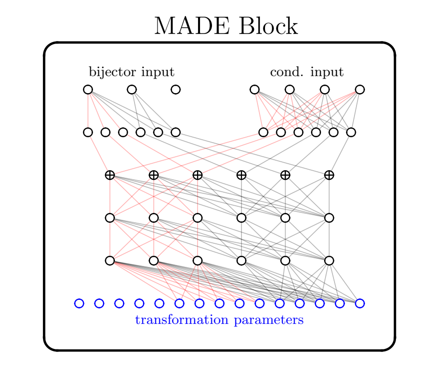
Implementation via masking:

- a single "forward" pass gives all $\theta_{x_i}(x_{i-1}\ldots x_1)$.
  $\Rightarrow$ very fast
- its "inverse" needs to loop through all dimensions.
  $\Rightarrow$ very slow

Germain/Gregor/Murray/Larochelle [arXiv:1502.03509]

- Masked Autoregressive Flow (MAF) is slow in sampling and fast in inference.
  Papamakarios et al. [arXiv:1705.07057]
- Inverse Autoregressive Flow (IAF) is fast in sampling and slow in inference.
  Kingma et al. [arXiv:1606.04934]

Taming Jacobians 1: with Autoregressive Blocks



$\boxed{\theta_{x_i}(x_{j<i})}$

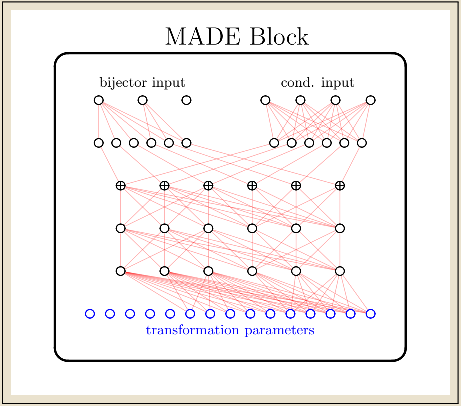
Implementation via masking:

- a single "forward" pass gives all $\theta_{x_i}(x_{i-1}\dots x_1)$.
  $\Rightarrow$ very fast
- its "inverse" needs to loop through all dimensions.
  $\Rightarrow$ very slow

Germain/Gregor/Murray/Larochelle [arXiv:1502.03509]

- Masked Autoregressive Flow (MAF) is slow in sampling and fast in inference.
  Papamakarios et al. [arXiv:1705.07057]
- Inverse Autoregressive Flow (IAF) is fast in sampling and slow in inference.
  Kingma et al. [arXiv:1606.04934]

Taming Jacobians 1: with Autoregressive Blocks



$$\boxed{\theta_{x_i}(x_{j<i})}$$

Implementation via masking:

- a single "forward" pass gives all $\theta_{x_i}(x_{i-1} \dots x_1)$.
  $\Rightarrow$ very fast
- its "inverse" needs to loop through all dimensions.
  $\Rightarrow$ very slow

Germain/Gregor/Murray/Larochelle [arXiv:1502.03509]

- Masked Autoregressive Flow (MAF) is slow in sampling and fast in inference.
  Papamakarios et al. [arXiv:1705.07057]
- Inverse Autoregressive Flow (IAF) is fast in sampling and slow in inference.
  Kingma et al. [arXiv:1606.04934]

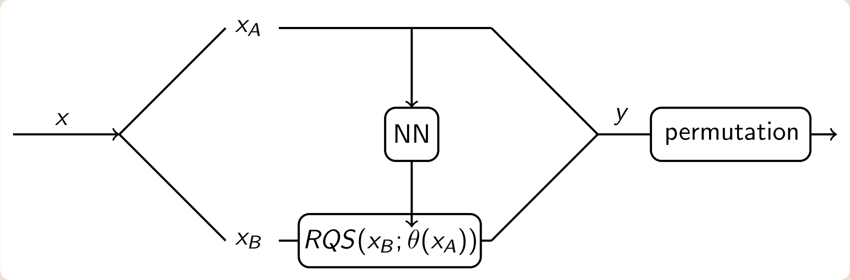
Taming Jacobians 2: with Bipartite Blocks

$$\boxed{\theta_{x\in A}(x \in B) \qquad \& \qquad \theta_{x\in B}(x \in A)}$$

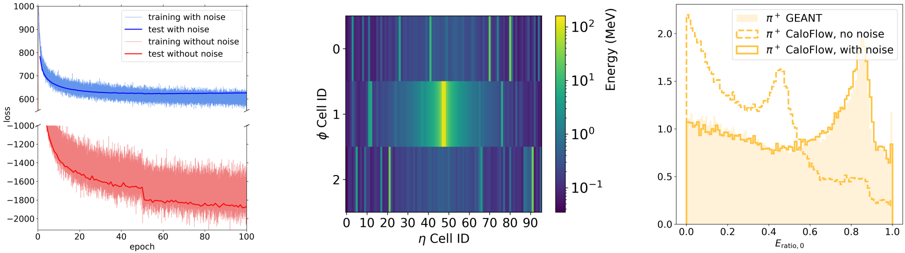
- Coordinates are split in 2 sets, transforming each other.

+ Forward and inverse pass are equally fast.
– Not as powerful as autoregressive blocks.

Dinh et al. [arXiv:1410.8516]

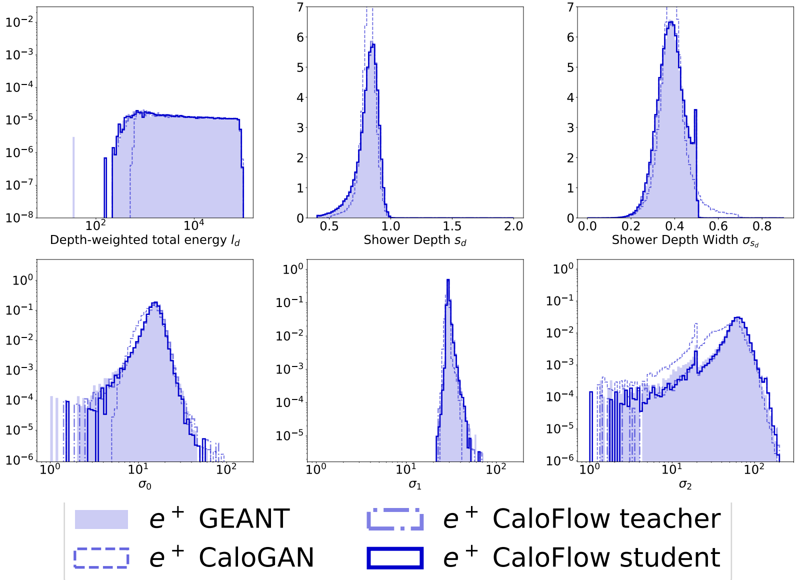


Adding Noise is important for the sampling quality.

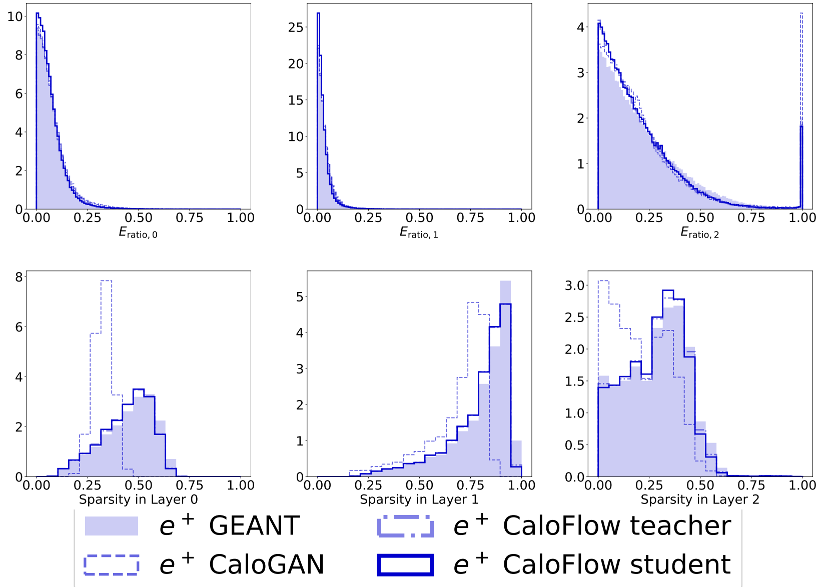


- The log-likelihood is less noisy, but smaller. Yet, the quality of the samples is much better!
- This is due to a "wider" mapping of space and less overfitting.

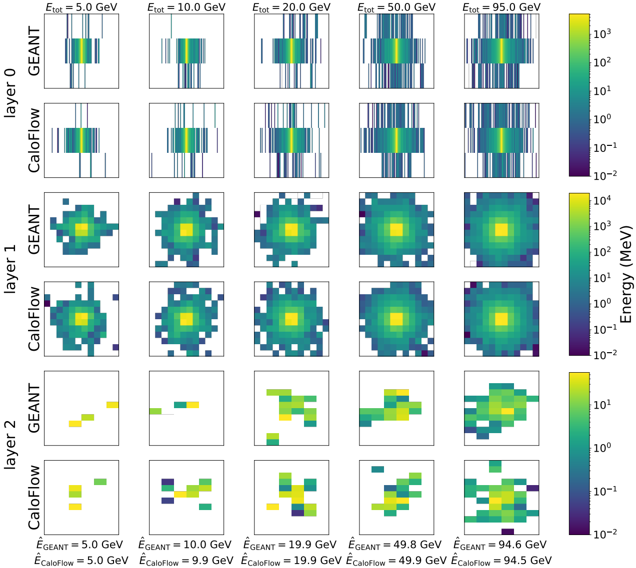
CaloFlow: Flow I+II histograms: e^+



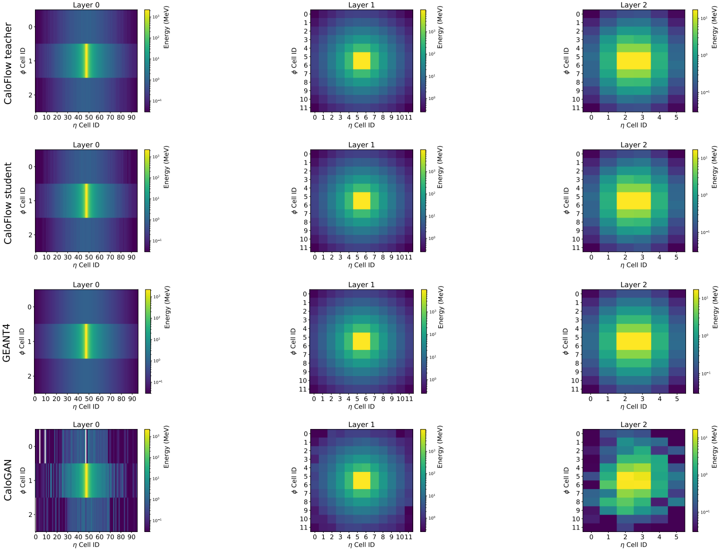
CaloFlow: Flow II histograms: e^+



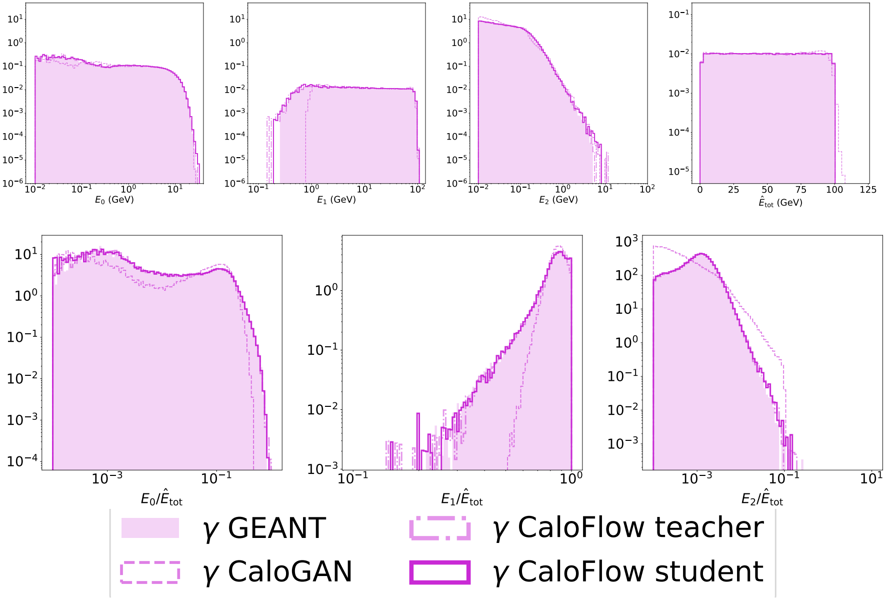
Nearest Neighbors: e^+ (student)



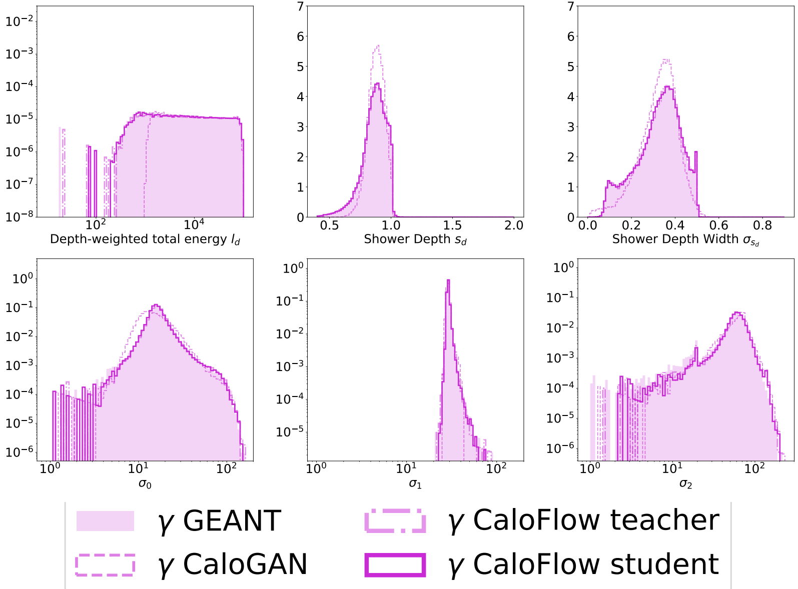
Comparing Shower Averages: γ



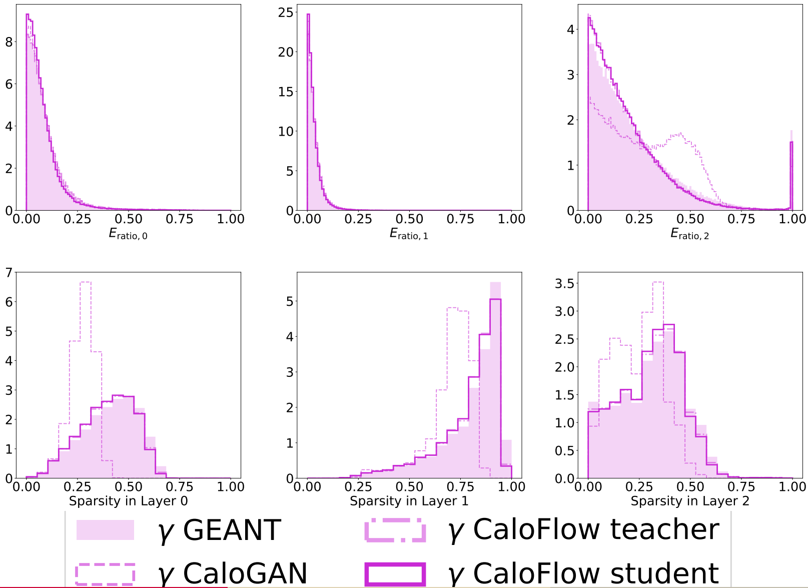
Flow I histograms: γ



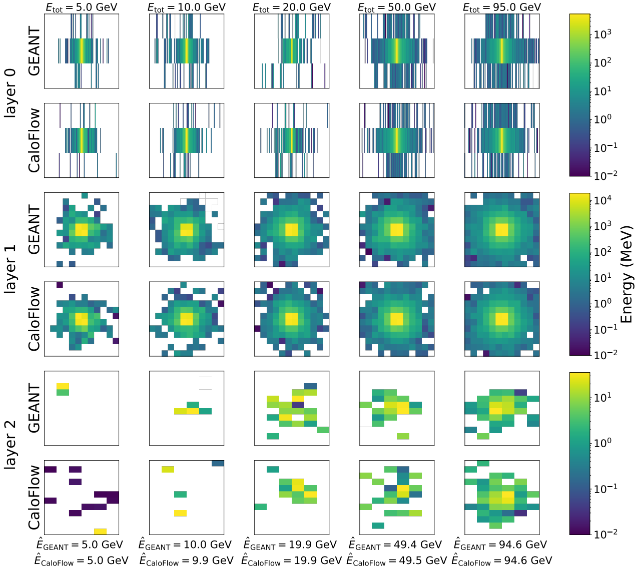
Flow I+II histograms: γ



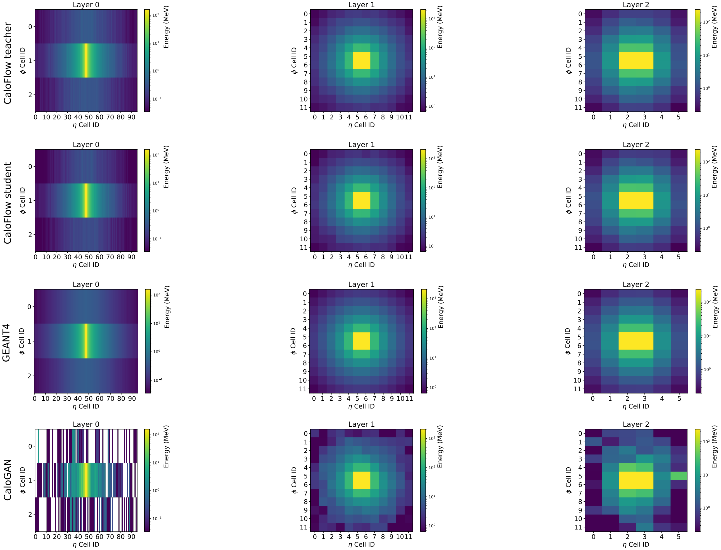
Flow II histograms: γ



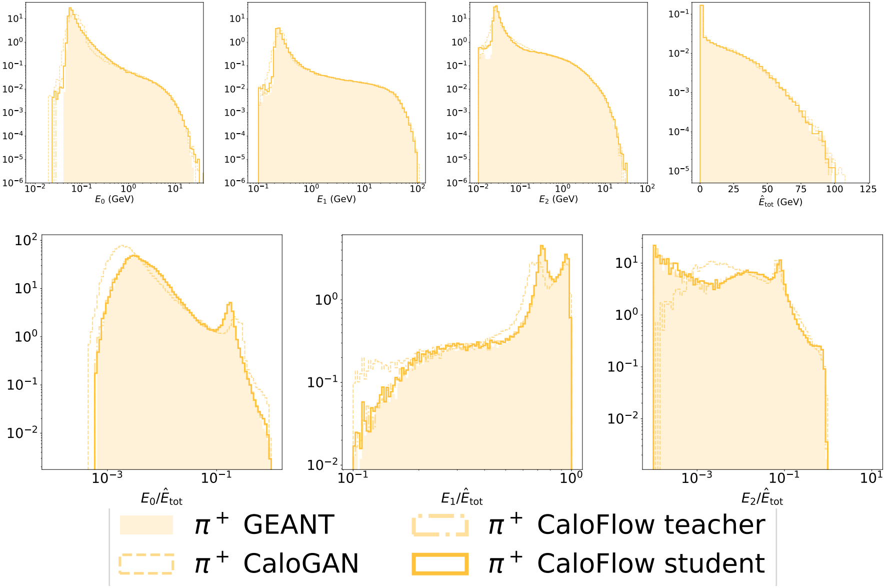
Nearest Neighbors: γ (student)



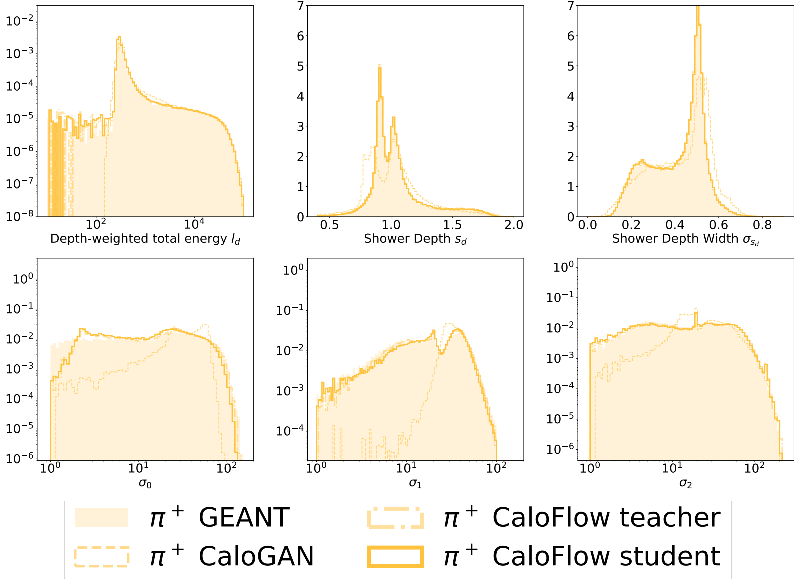
Comparing Shower Averages: π^+



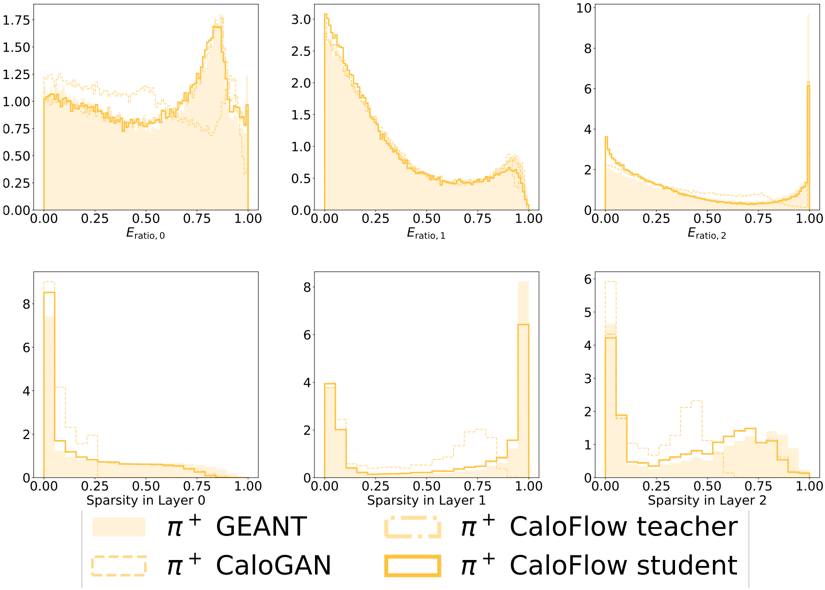
CaloFlow: Flow I histograms: π^+



CaloFlow: Flow I+II histograms: π^+



CaloFlow: Flow II histograms: π^+



CaloFlow: Nearest Neighbors: π^+ (student)