CaloFlow: Fast and Accurate Generation of Calorimeter Showers with Normalizing Flows

— ML4Jets 2021, Heidelberg —

Claudius Krause

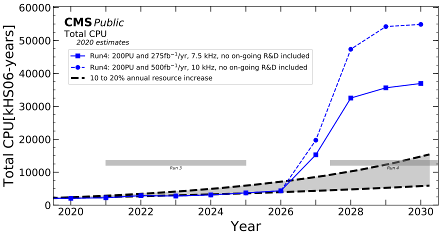
Rutgers, The State University of New Jersey

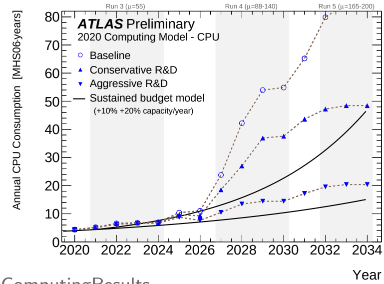
July 7, 2021

RUTGERS
UNIVERSITY | NEW BRUNSWICK

In collaboration with David Shih
arXiv: 2106.05285

Deep Generative Models will be crucial for the LHC.





https://twiki.cern.ch/twiki/bin/view/CMSPublic/CMSOfflineComputingResults
https://twiki.cern.ch/twiki/bin/view/AtlasPublic/ComputingandSoftwarePublicResults

- At the end of LHC Run 3, the computational needs will exceed the available budget.
- A large fraction goes into simulation.

CERN-LHCC-2020-015; LHCC-G-178

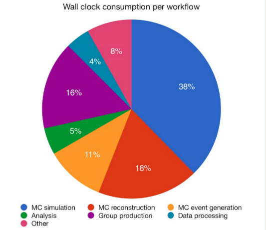
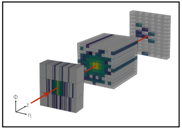
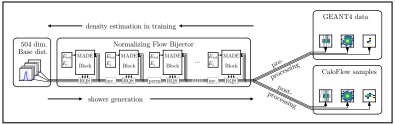


Figure 1: ATLAS CPU hours used by various activities in 2018

CaloFlow: Fast and Accurate Generation of Calorimeter Showers with Normalizing Flows

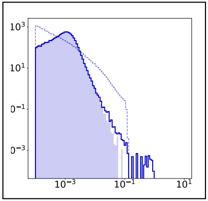


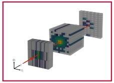
Part I: The Calorimeter Dataset



Part II: Generative Modeling with Normalizing Flows

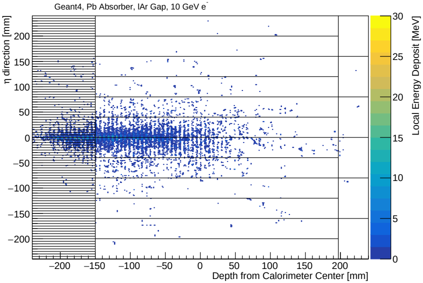
Part III: Performance of CaloFlow

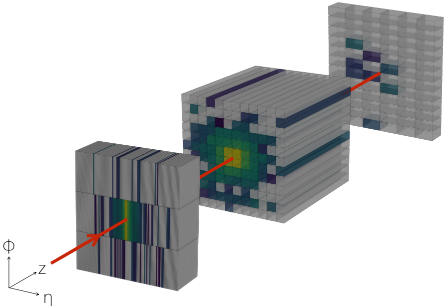




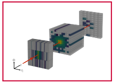
I: We use the same calorimeter geometry and training data as CaloGAN

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



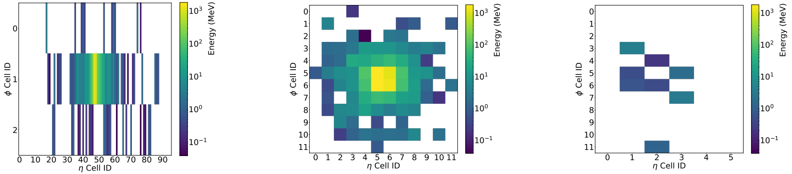


CaloGAN: Paganini, de Oliveira, Nachman [1712.10321, PRD]



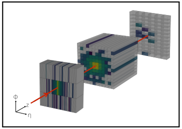
I: We use the same calorimeter geometry and training data as CaloGAN

- We use the Geant4 simulated data of CaloGAN available at
  doi: 10.17632/pvn3xc3wy5.1
- These are showers of e^+, γ, and π^+ (100k each)
- All are centered and perpendicular
- E_{tot} is uniform in $[1, 100]$ GeV and given in addition to the energy deposits per voxel:

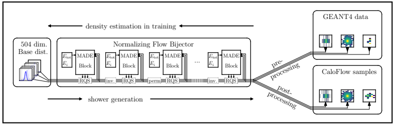


CaloGAN: Paganini, de Oliveira, Nachman [1712.10321, PRD]

CaloFlow: Fast and Accurate Generation of Calorimeter Showers with Normalizing Flows

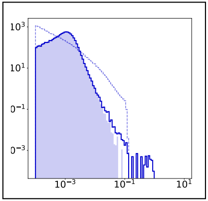


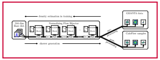
Part I: The Calorimeter Dataset



Part II: Generative Modeling with Normalizing Flows

Part III: Performance of CaloFlow





II: Normalizing Flows learn a change-of-coordinates efficiently.

"easy" base distribution ⇔ bijective trans-formation ⇔ "target" distribution, data

← density estimation, $p(x)$

sample generation →

Normalizing Flows . . . Dinh et al. [arXiv:1410.8516], Rezende/Mohamed [arXiv:1505.05770], Review: Papamakarios et al. [arXiv:1912.02762]

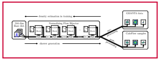
- . . . learn the parameters of a series of easy transformations.
- Each transformation is bijective and has an analytic Jacobian and inverse.

⇒ We use a piecewise Rational Quadratic Spline. Durkan et al. [arXiv:1906.04032]

- An autoregressive architecture ensures a triangular Jacobian.

⇒ We use a Masked Autoregressive Flow (MAF) architecture.

Germain et al. [arXiv:1502.03509], Papamakarios et al. [arXiv:1705.07057]



II: Normalizing Flows resolve a few challenges of Deep Generative Models.

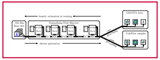
General challenges of deep generative models:

- ⇒ By which metric can we monitor the quality of the generator?
- ⇒ Energy conservation and other constraints on samples
- ⇒ Sparse data and “sharp edges”
- ⇒ Faster sampling vs. (longer) training times

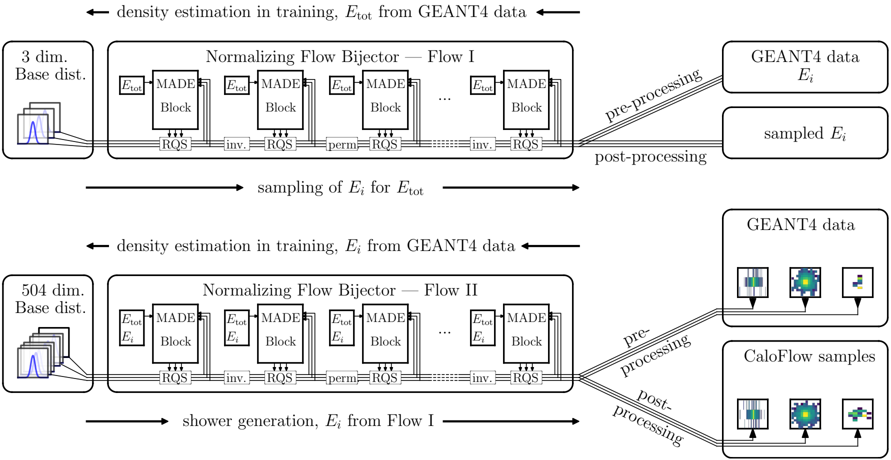
Normalizing Flows:

- ✓ learn $p(x)$ explicitly
- ✓ training is more stable
- ✓ model selection is straightforward
- ✓ no mode collapse and artefacts in samples

- × sparse data is hard to model
- × MAF can be trained with $\log p(x)$, but samples slow



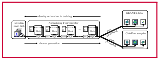
II: CaloFlow uses a 2-step approach.



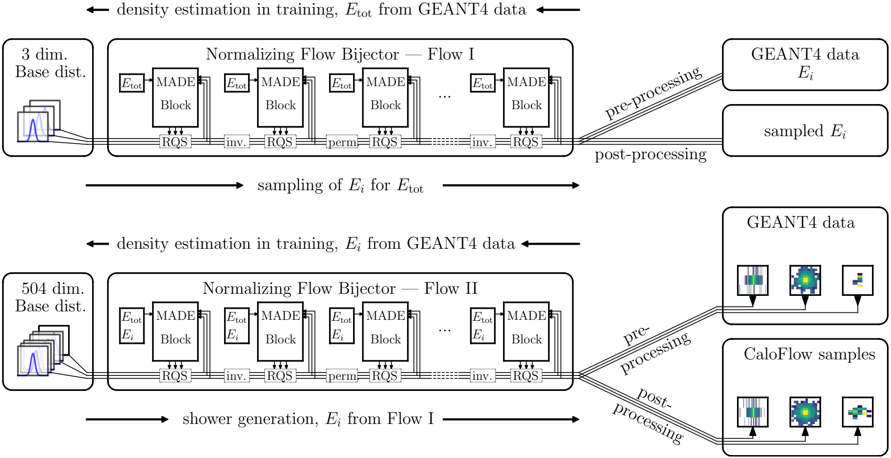
Data processing Flow I

"←" map E_i to $[0,1]$ "→" invert logit

"←" work in logit space "→" map back to E_i



II: CaloFlow uses a 2-step approach.

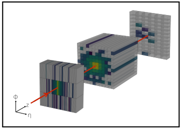


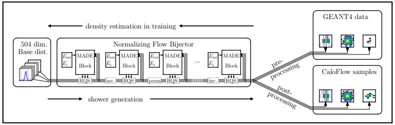
Data processing Flow II

"←"	add noise	"→"	invert logit
"←"	normalize layers to 1	"→"	renormalize to E_i
"←"	work in logit space	"→"	apply threshold

CaloFlow: Fast and Accurate Generation of Calorimeter Showers with Normalizing Flows

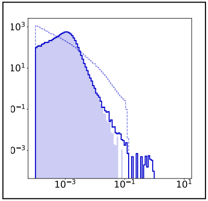
Part I: The Calorimeter Dataset

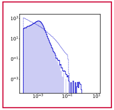




Part II: Generative Modeling with Normalizing Flows

Part III: Performance of CaloFlow



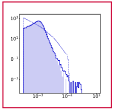


III: A classifier is the “ultimate metric”.

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

AUC / JSD		DNN		CNN	
		GEANT4 vs. CALOGAN	GEANT4 vs. CALOFLOW	GEANT4 vs. CALOGAN	GEANT4 vs. CALOFLOW
e^+	unnorm.	0.999(0) / 0.961(3)	0.607(21) / 0.027(19)	0.945(0) / 0.584(1)	0.509(1) / 0.002(0)
	norm.	1.000(0) / 0.989(1)	0.726(5) / 0.124(5)	0.999(0) / 0.957(3)	0.688(54) / 0.095(61)
γ	unnorm.	1.000(0) / 0.986(2)	0.537(11) / 0.004(2)	0.969(1) / 0.681(3)	0.516(1) / 0.001(0)
	norm.	1.000(0) / 0.995(1)	0.698(1) / 0.072(2)	1.000(0) / 0.994(1)	0.651(30) / 0.058(25)
π^+	unnorm.	0.999(0) / 0.957(3)	0.643(2) / 0.051(1)	0.983(3) / 0.765(23)	0.554(1) / 0.009(0)
	norm.	1.000(0) / 0.994(1)	0.758(6) / 0.105(13)	1.000(0) / 0.996(1)	0.813(9) / 0.244(16)

AUC ($\in [0.5, 1]$): Area Under the ROC Curve
JSD ($\in [0, 1]$): Jensen-Shannon divergence based on the binary cross entropy

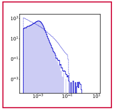


III: CaloFlow has moderate speed.

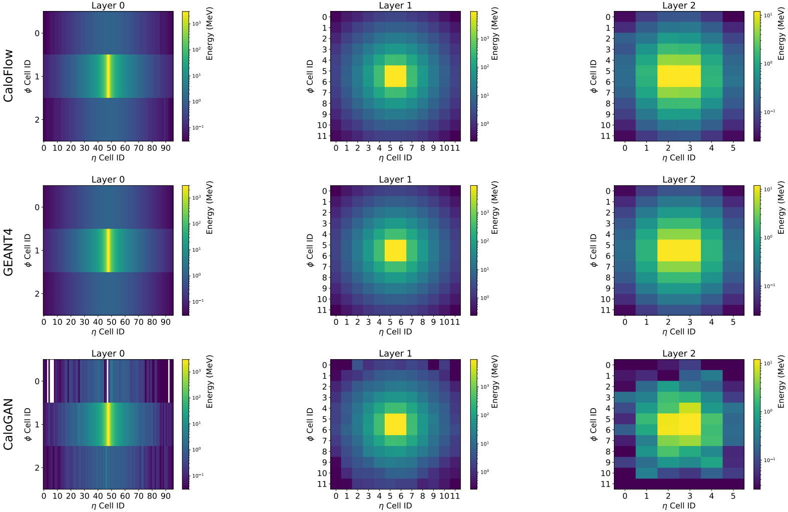
	CaloFlow*	CaloGAN*		Geant4†
training	22+82 min	210 min		0 min
generation	time per shower			
batch size		batch size req.	100k req.	
10	835 ms	455 ms	2.2 ms	1772 ms
100	96.1 ms	45.5 ms	0.3 ms	1772 ms
1000	41.4 ms	4.6 ms	0.08 ms	1772 ms
5000	36.2 ms	1.0 ms	0.07 ms	1772 ms
10000	36.2 ms	0.5 ms	0.07 ms	1772 ms

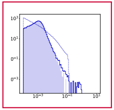
*: on our Titan V GPU

†: on the CPU of CaloGAN: Paganini, de Oliveira, Nachman [1712.10321, PRD]

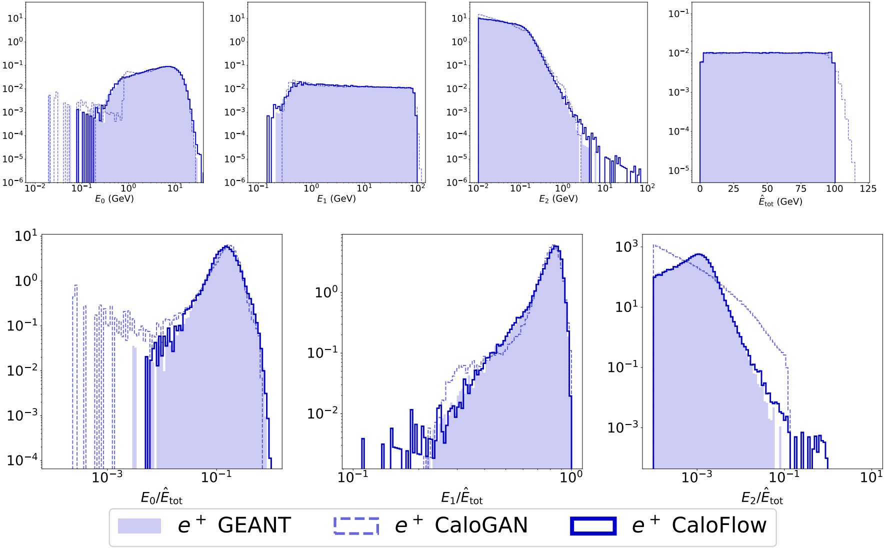


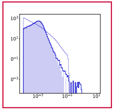
III: Comparing Shower Averages: e^+



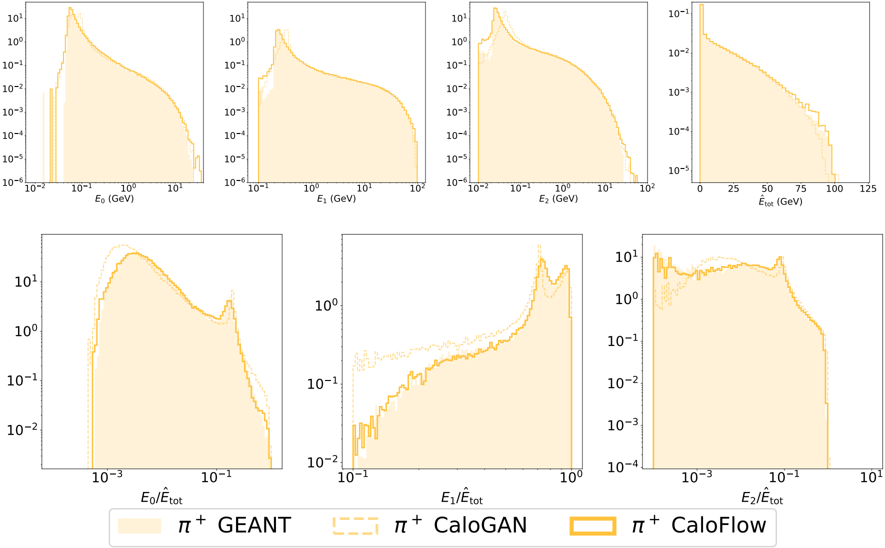


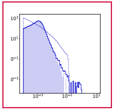
III: Flow I histograms: e^+



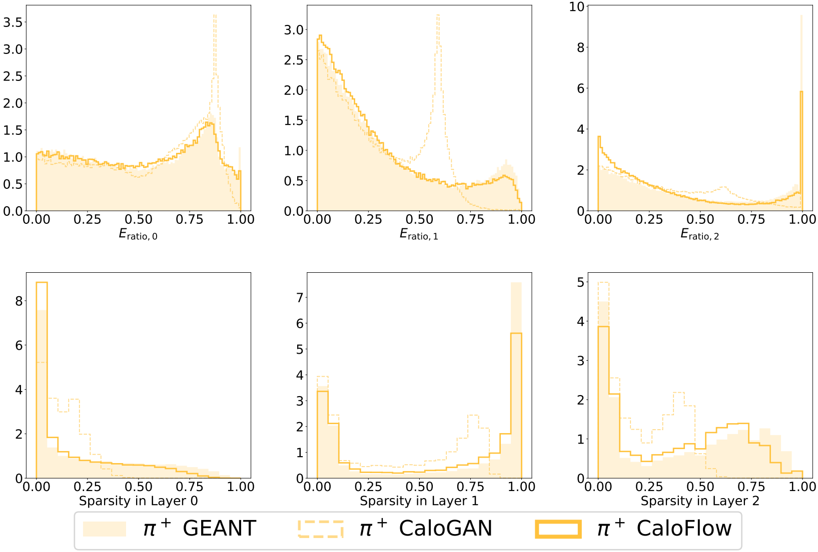


III: Flow I histograms: π^+





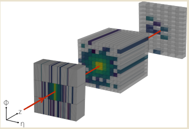
III: Flow II histograms: π^+



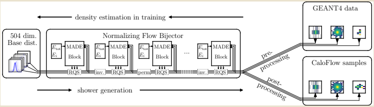
CaloFlow: Fast and Accurate Generation of Calorimeter Showers with Normalizing Flows

- We use the same calorimeter and training data as the original CaloGAN.
- These are 504-dim. showers of e^+, γ, and π^+

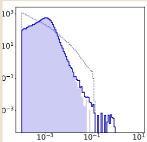
⇒ First time application of Normalizing Flows!



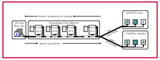
- Having $\log p(x)$ allows stable training and straightforward model selection.
- We use a 2-step setup to ensure energy conservation.



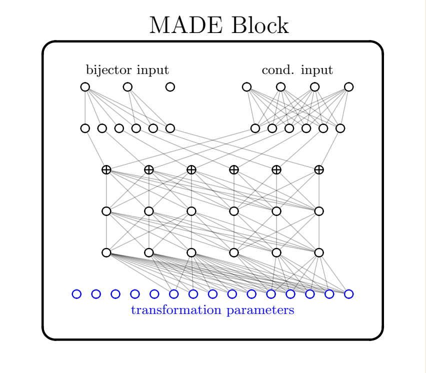
- I argued that a classifier provides the "ultimate test" of a generative model.
- I showed how CaloFlow passes this stringent test, along with more qualitative comparisons (histograms, images).



Backup



II: Ensuring the Autoregressive Property.

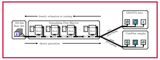


Implementation via masking:

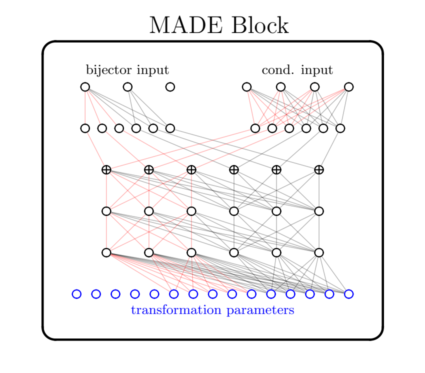
- a single "forward" pass gives the full output of all $p(x_i|x_{i-1}\dots x_1)$.
  $\Rightarrow$ very fast
- the "inverse" needs to loop through all dimensions and gets a single $p(x_i|x_{i-1}\dots x_1)$ each time.
  $\Rightarrow$ very slow

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

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



II: Ensuring the Autoregressive Property.

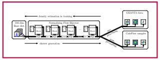


Implementation via masking:

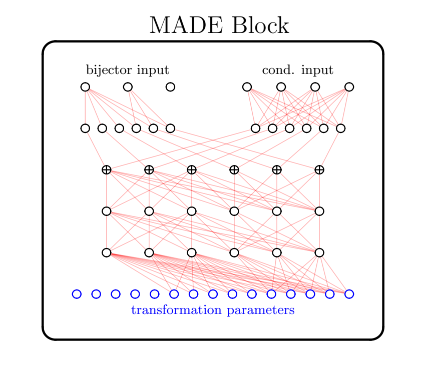
- a single "forward" pass gives the full output of all $p(x_i|x_{i-1} \dots x_1)$.
  $\Rightarrow$ very fast
- the "inverse" needs to loop through all dimensions and gets a single $p(x_i|x_{i-1} \dots x_1)$ each time.
  $\Rightarrow$ very slow

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

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



II: Ensuring the Autoregressive Property.

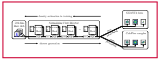


Implementation via masking:

- a single "forward" pass gives the full output of all $p(x_i|x_{i-1}\dots x_1)$.
  $\Rightarrow$ very fast
- the "inverse" needs to loop through all dimensions and gets a single $p(x_i|x_{i-1}\dots x_1)$ each time.
  $\Rightarrow$ very slow

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

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



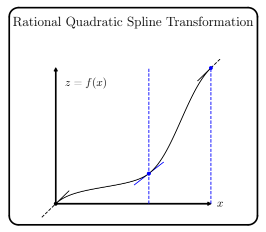
II: The Coupling Function defines the bijection.

The coupling function (transformation)

- must be invertible and expressive
- is chosen to factorize:
  $\vec{C}(\vec{x}; \vec{p}) = (C_1(x_1; p_1), C_2(x_2; p_2), \ldots, C_n(x_n; p_n))^T$,
  where $\vec{x}$ are the coordinates to be transformed and $\vec{p}$ the parameters of the transformation.

rational quadratic spline coupling function: Durkan et al. [arXiv:1906.04032]

Gregory/Delbourgo [IMA Journal of Numerical Analysis, '82]

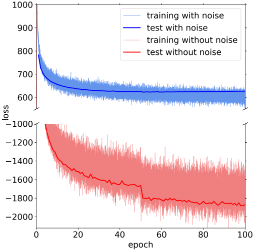


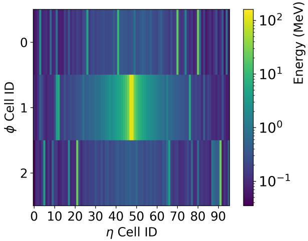
$$C = \frac{a_2\alpha^2 + a_1\alpha + a_0}{b_2\alpha^2 + b_1\alpha + b_0}$$

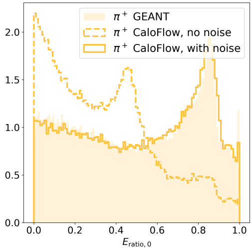
- still rather easy
- more flexible

The NN predicts the bin widths, heights, and derivatives that go in $a_i \& b_i$.

Adding Noise is important for the sampling quality.

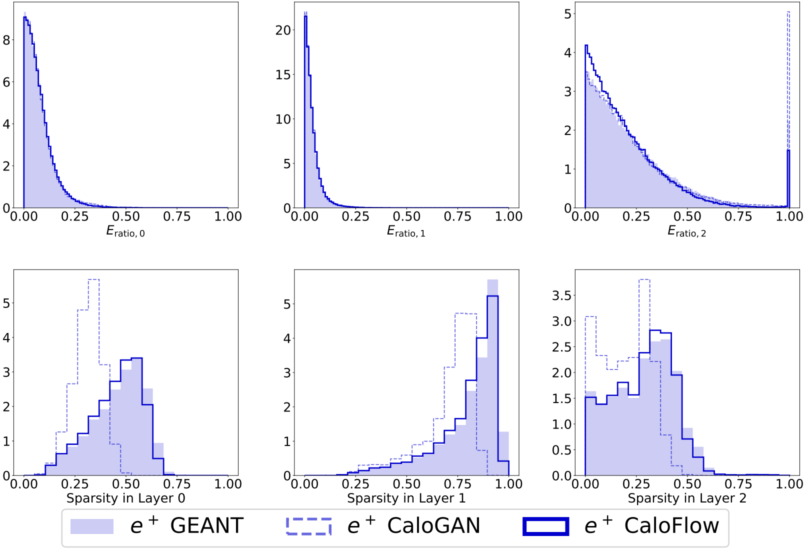




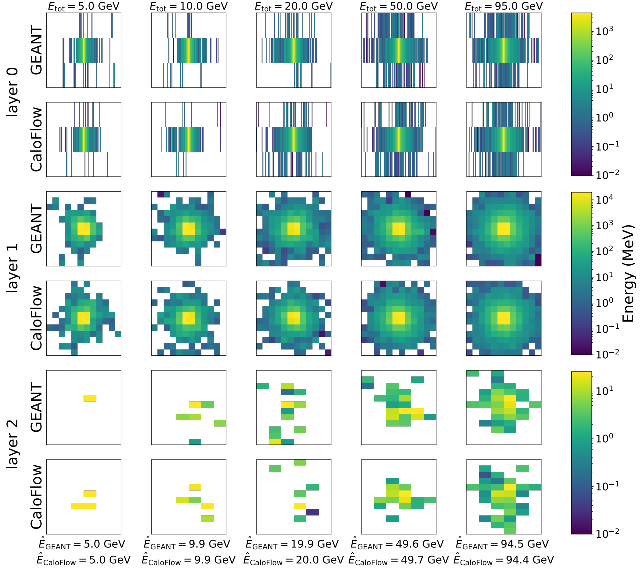


- The log-likelihood is less noisy, but smaller. Yet, the quality of the samples is much better!
- This is due to a "wider" mapping of space and less overfitting.

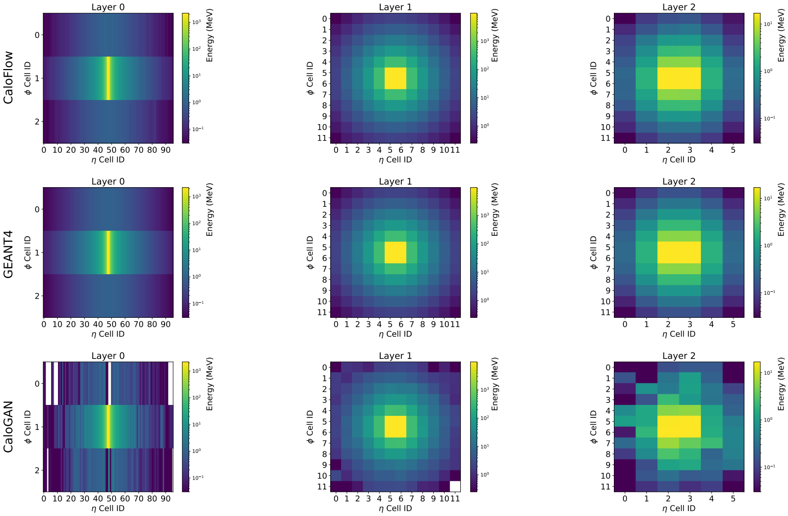
Flow II histograms: e^+



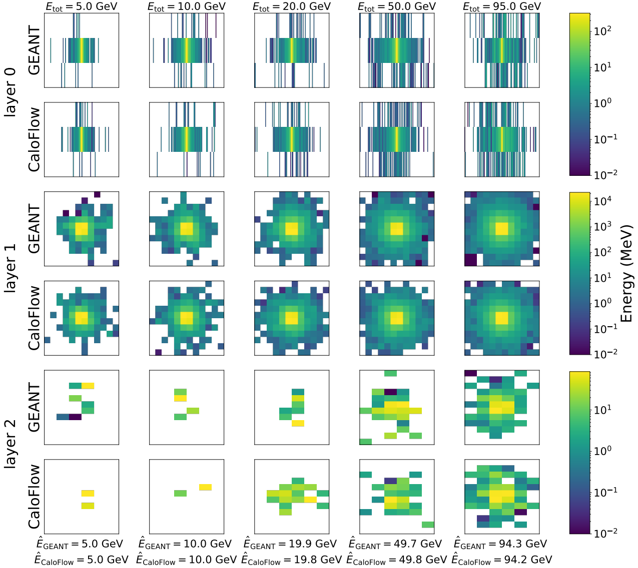
Nearest Neighbors: e^+



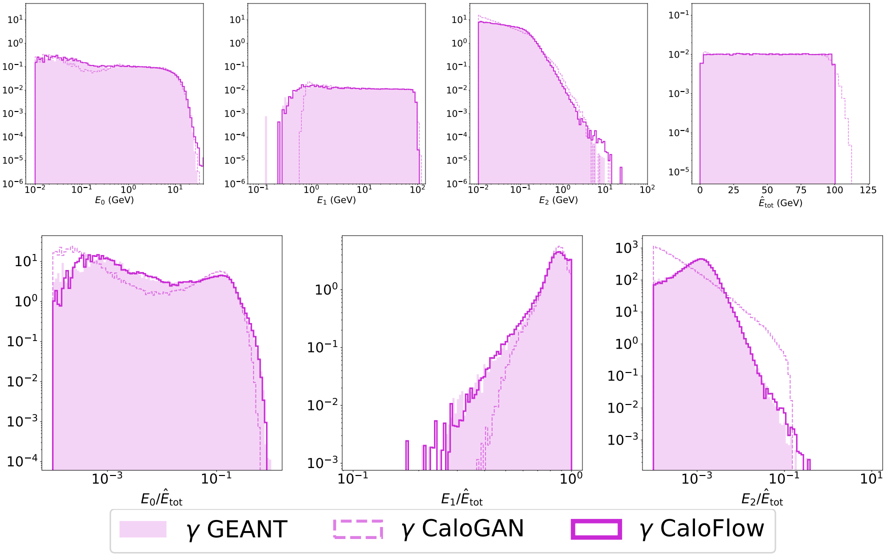
Comparing Shower Averages: γ



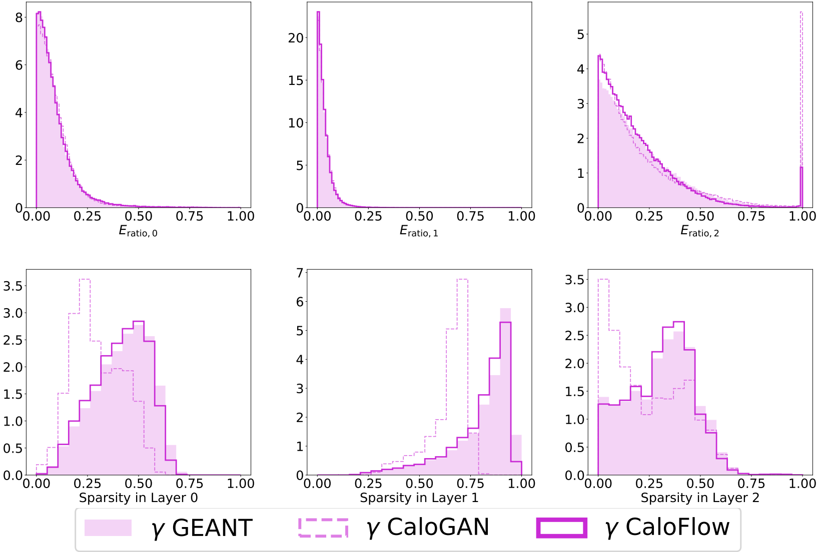
Nearest Neighbors: γ



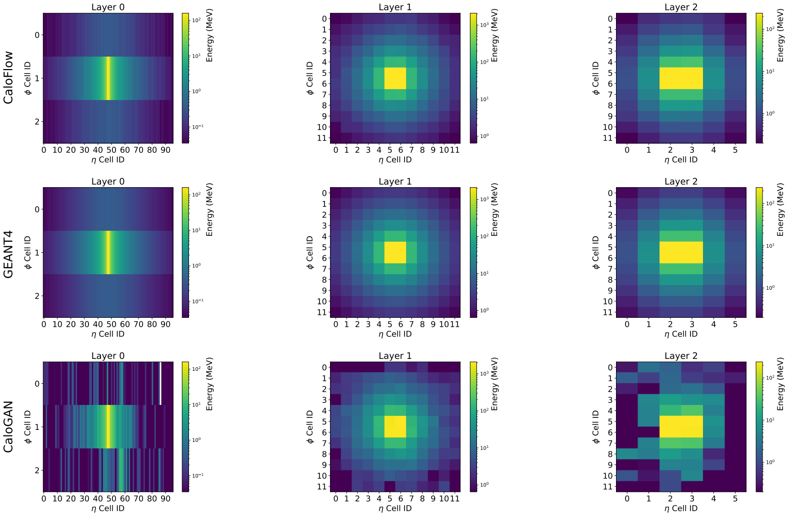
Flow I histograms: γ



Flow II histograms: γ



Comparing Shower Averages: π^+



Nearest Neighbors: π^+