ÖAW AUSTRIAN ACADEMY OF SCIENCES

HEPHY INSTITUTE OF HIGH ENERGY PHYSICS

Improving HEP Simulation and Analyses with Invertible Neural Networks

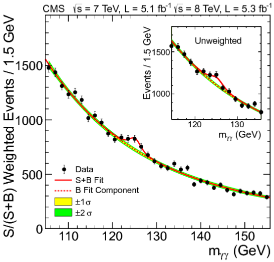
— Seminar at University of Vienna —

Claudius Krause

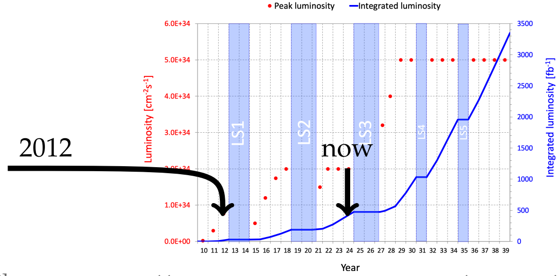
Institute of High Energy Physics (HEPHY), Austrian Academy of Sciences (OeAW)

January 9, 2024

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



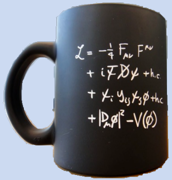
CMS Collaboration [arXiv:1207.7235, Phys.Lett.B]



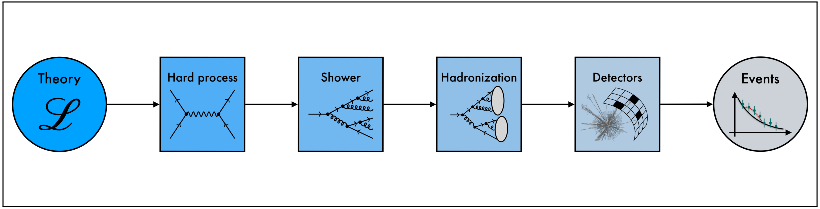
https://lhc-commissioning.web.cern.ch/schedule/HL-LHC-plots.htm

- We will have 20–25× more data.

⇒ We want to understand every aspect of it based on 1st principles! (and find New Physics)

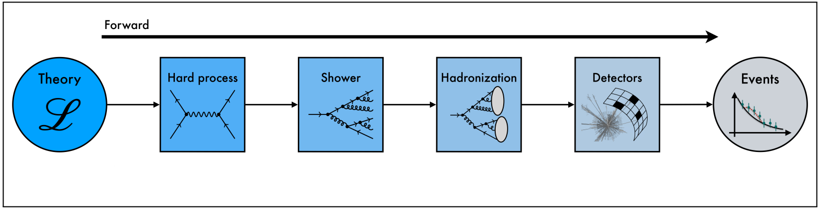


How do we understand the data based on 1st principles?



Machine Learning and LHC Event Generation, A. Butter et al. [2203.07460], R. Winterhalder

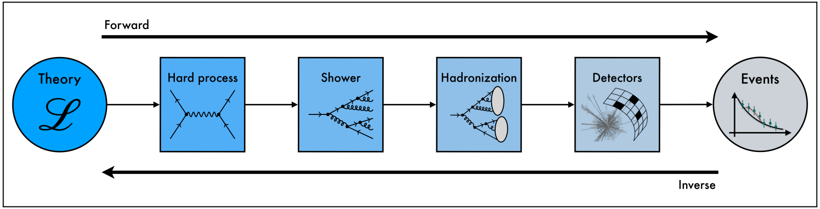
How do we understand the data based on 1st principles?



Machine Learning and LHC Event Generation, A. Butter et al. [2203.07460], R. Winterhalder

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

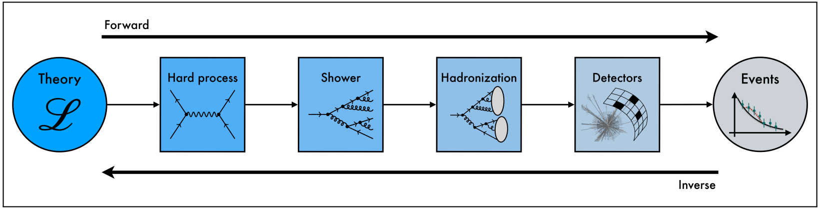
How do we understand the data based on 1st principles?



Machine Learning and LHC Event Generation, A. Butter et al. [2203.07460], R. Winterhalder

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?



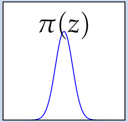
Machine Learning and LHC Event Generation, A. Butter et al. [2203.07460], R. Winterhalder

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

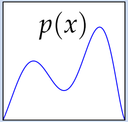


$\Leftrightarrow$

bijective transformation

$\Leftrightarrow$

"target" distribution



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

density estimation, $p(x)$ ←

sample generation →

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 $p(\text{parameters}|\text{data})$.

How do Normalizing Flows tame Jacobians?

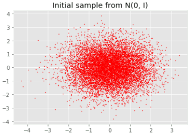
- NFs learn the parameters θ of a series of easy transformations. Dinh et al. [arXiv:1410.8516], Rezende/Mohamed [arXiv:1505.05770]
- Each transformation is 1d & has an analytic Jacobian and inverse.

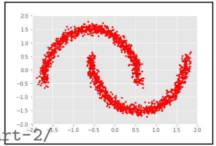
  $\Rightarrow$ We use Rational Quadratic Splines Durkan et al. [arXiv:1906.04032], Gregory/Delbourgo [IMA J. of Num. An., '82]
- Require a triangular Jacobian for faster evaluation.

  $\Rightarrow$ 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.

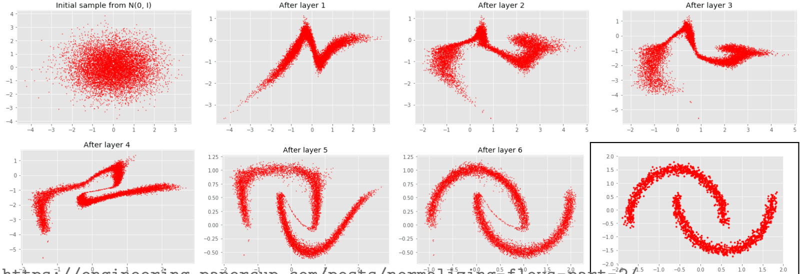




https://engineering.papercup.com/posts/normalizing-flows-part-2/

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 mathematically "exact".

– 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.

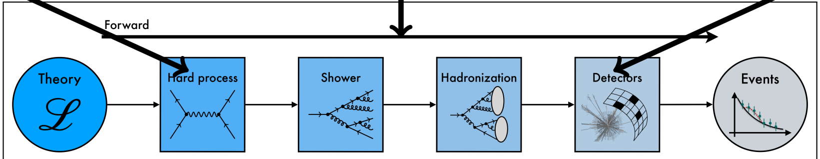
– "Require" a min. number of blocks.

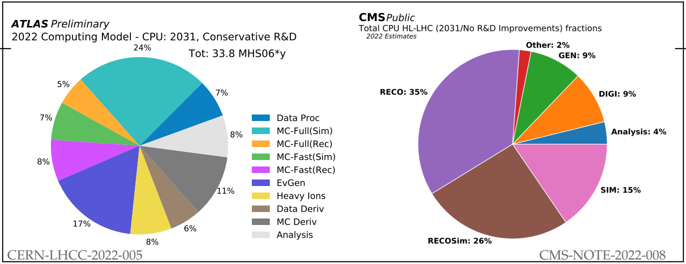
Normalizing Flows attack Bottlenecks in the Analysis Chain

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

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

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





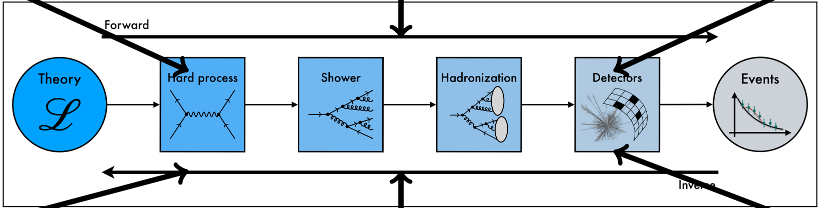
CERN-LHCC-2022-005 CMS-NOTE-2022-008

Normalizing Flows increase the Sensitivity in our Analyses

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

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

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



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

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

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

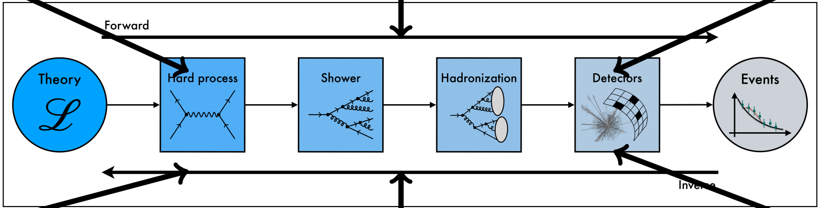
Lattice QCD $\Rightarrow$ improve MCMC proposals

Improving HEP Simulation and Analyses with INNs

1. Phase Space Sampling

End-to-End Simulation

2. Detector Simulation



Inference

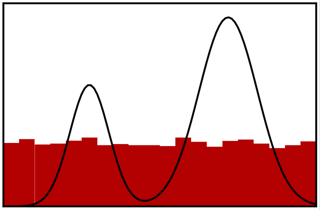
3. Bump-Hunt Searches

Unfolding

Lattice QCD

I: Phase Space integration uses Importance Sampling.

$$I = \int_0^1 f(\vec{x})\ d\vec{x}$$

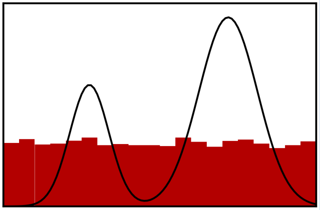


flat sampling:
inefficient.

$$I = \langle f(\vec{x}) \rangle_{x \sim \text{uniform}}$$

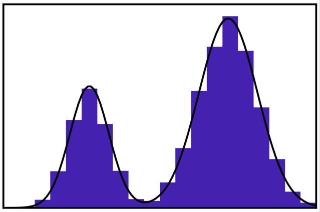
I: Phase Space integration uses Importance Sampling.

$$I = \int_0^1 f(\vec{x})\, d\vec{x}$$



flat sampling:
inefficient.

$$I = \langle f(\vec{x}) \rangle_{x \sim \text{uniform}}$$

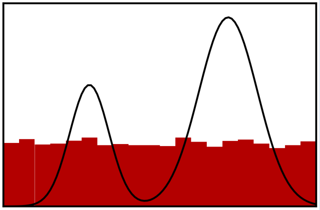


importance sampling:
find g close to f

$$I = \left\langle \frac{f(\vec{x})}{g(\vec{x})} \right\rangle_{x \sim g(x)}$$

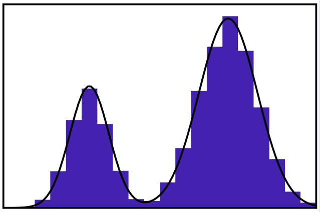
I: Phase Space integration uses Importance Sampling.

$$I = \int_0^1 f(\vec{x})\ d\vec{x}$$



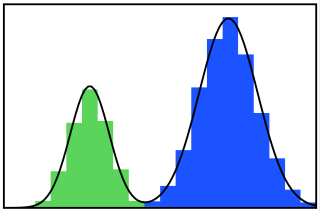
flat sampling: inefficient.

$$I = \langle f(\vec{x}) \rangle_{x\sim\text{uniform}}$$



importance sampling: find g close to f

$$I = \left\langle \frac{f(\vec{x})}{g(\vec{x})} \right\rangle_{x\sim g(x)}$$

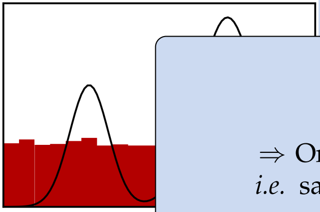


multichannel: one map per channel

$$I = \sum_i \left\langle \alpha_i(x) \frac{f(\vec{x})}{g_i(\vec{x})} \right\rangle_{x\sim g_i(x)}$$

I: Phase Space integration uses Importance Sampling.

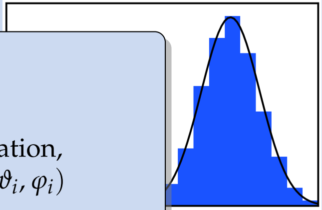
$$I = \int_0^1 f(\vec{x})\ d\vec{x}$$



flat sa…
ineffi…

$$I = \langle f(\vec{x}) \rangle_{x \sim \text{uniform}}$$

$$I = \left\langle \frac{f(\vec{x})}{g(\vec{x})} \right\rangle_{x \sim g(x)}$$



…nel: one
…channel

$$I = \sum_i \left\langle \alpha_i(x) \frac{f(\vec{x})}{g_i(\vec{x})} \right\rangle_{x \sim g_i(x)}$$

We therefore have to find a $g(\vec{x})$
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 $g(x)$.
$\Rightarrow$ We use a bipartite, coupling-layer-based flow.

I: MadNIS — Neural Importance Sampling

$$I = \sum_i \left\langle \alpha_i(x) \frac{f(\vec{x})}{g_i(\vec{x})} \right\rangle_{x \sim g_i(x)}$$

Construct channels and mapping using physics knowledge.

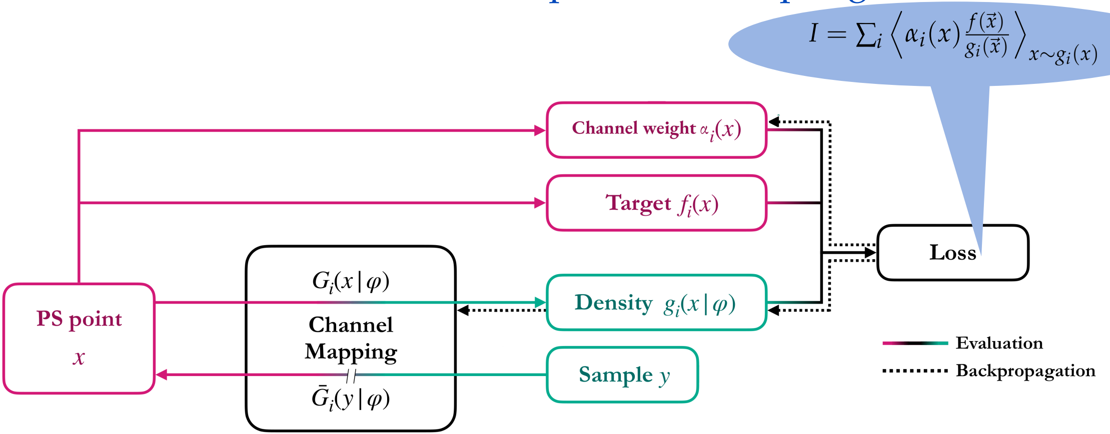
Normalizing Flow to refine channel mapping.

Fully connected network to refine channel weights.

Optimize simultaneously with variance as loss.

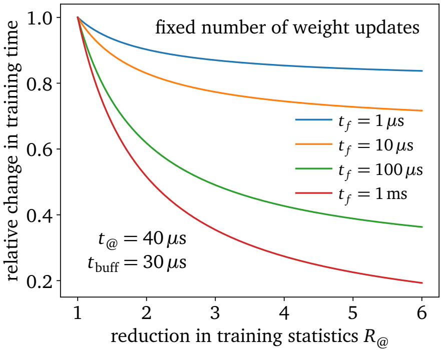
A. Butter, T. Heimel, J. Isaacson, CK, F. Maltoni, O. Mattelaer, T. Plehn, R. Winterhalder [2212.06172, SciPost]

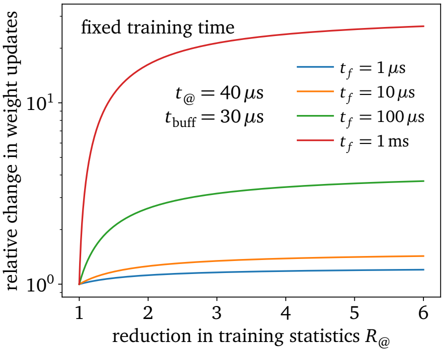
I: MadNIS — Neural Importance Sampling



A. Butter, T. Heimel, J. Isaacson, CK, F. Maltoni, O. Mattelaer, T. Plehn, R. Winterhalder [2212.06172, SciPost]

I: MadNIS re-uses expensive matrix elements

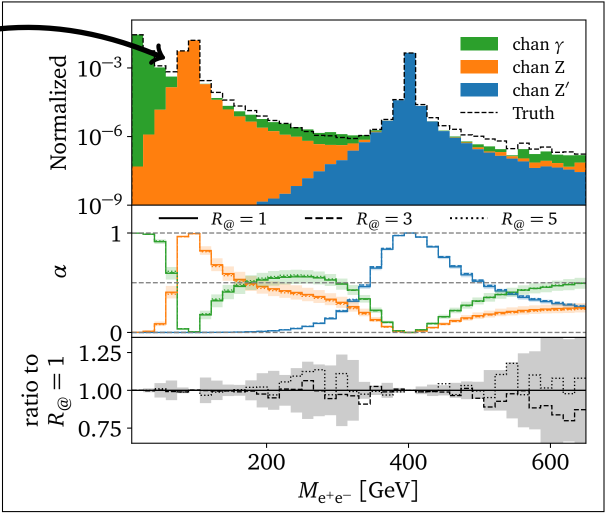




A. Butter, T. Heimel, J. Isaacson, CK, F. Maltoni, O. Mattelaer, T. Plehn, R. Winterhalder [2212.06172, SciPost]

I: MadNIS — Results for Drell-Yan + Z'

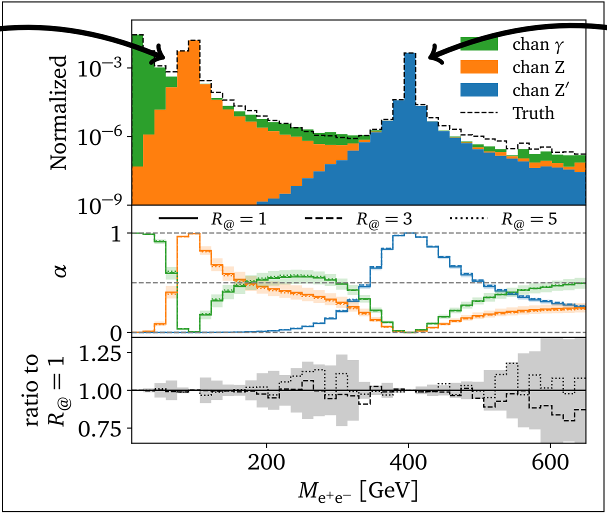
Learned distribution matches truth.



Heimel, CK et al. [2212.06172, SciPost]

I: MadNIS — Results for Drell-Yan + Z'

Learned distribution matches truth.

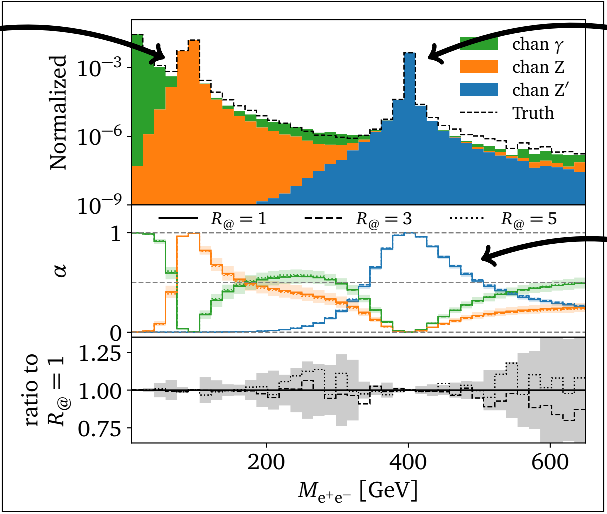


Peaks are learned by different channels.

Heimel, CK et al. [2212.06172, SciPost]

I: MadNIS — Results for Drell-Yan + Z'

Learned distribution matches truth.

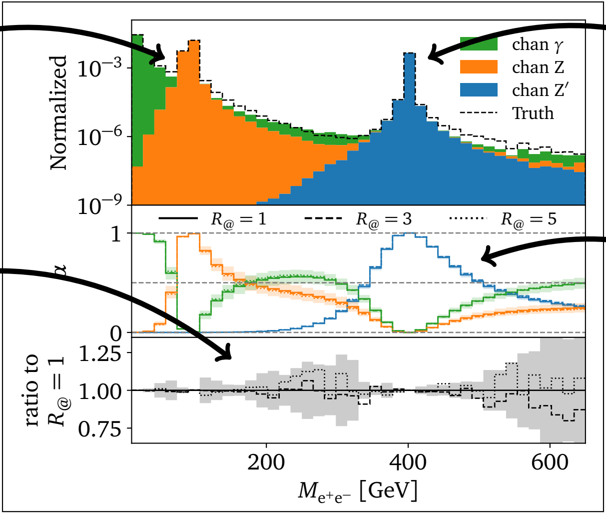


Peaks are learned by different channels.

Channel weights are learned by the network

Heimel, CK et al. [2212.06172, SciPost]

I: MadNIS — Results for Drell-Yan + Z'



Learned distribution matches truth.

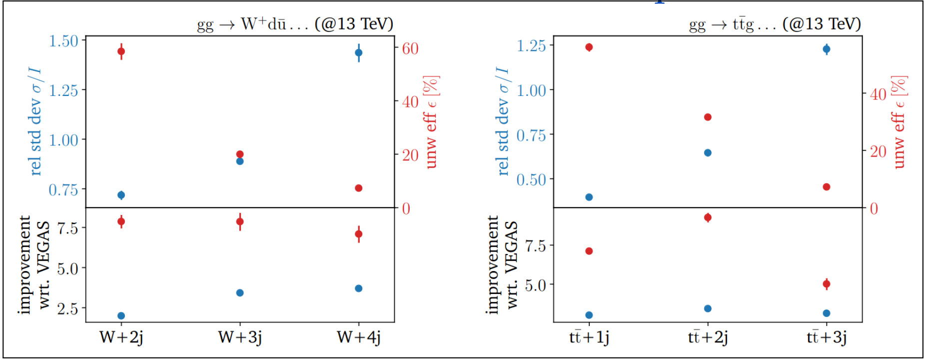
Re-uses samples to make training faster.

Peaks are learned by different channels.

Channel weights are learned by the network

Heimel, CK et al. [2212.06172, SciPost]

I: The MadNIS reloaded — more processes



- VEGAS initialization
- channel dropping
- stratified training
- buffered training

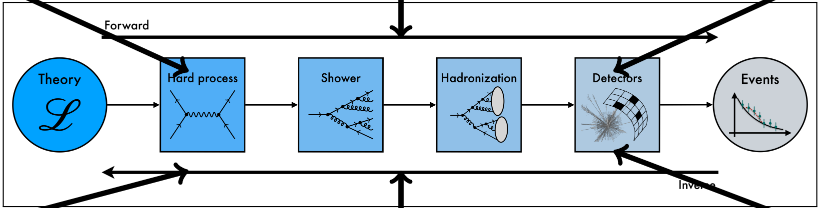
T. Heimel, N. Huetsch, F. Maltoni, O. Mattelaer, T. Plehn, R. Winterhalder [2311.01548]

Improving HEP Simulation and Analyses with INNs

1. Phase Space Sampling

End-to-End Simulation

2. Detector Simulation



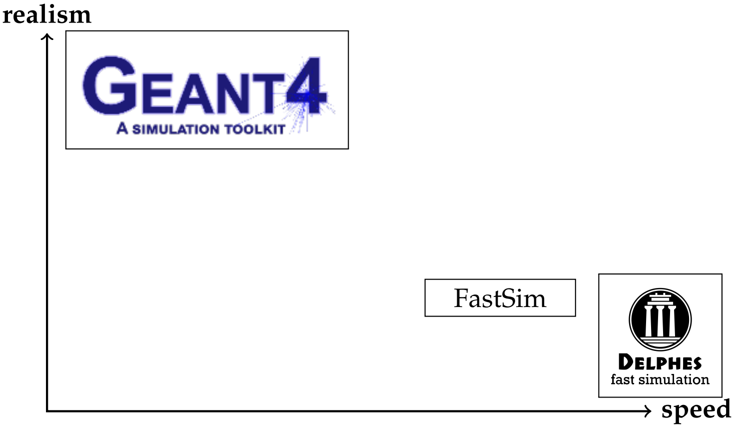
Inference

3. Bump-Hunt Searches

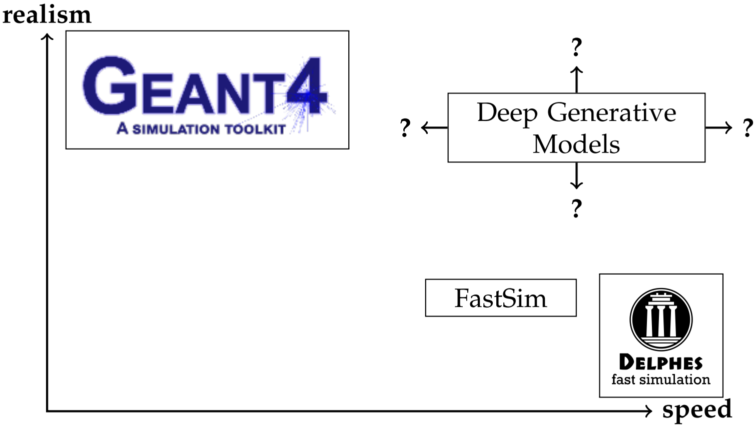
Unfolding

Lattice QCD

II: Detector simulation is computationally expensive.

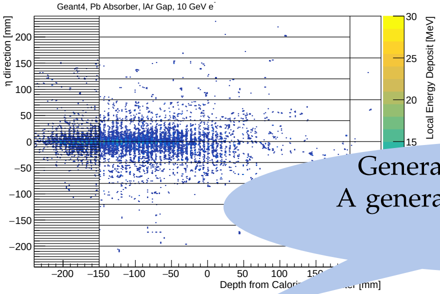


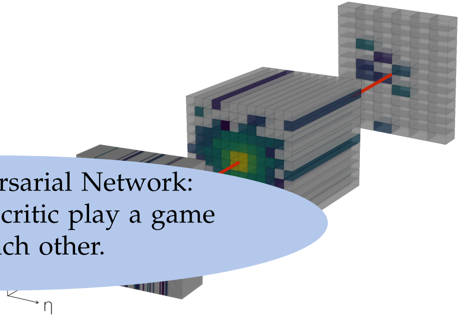
II: Detector simulation is computationally expensive.



II: CaloFlow uses 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



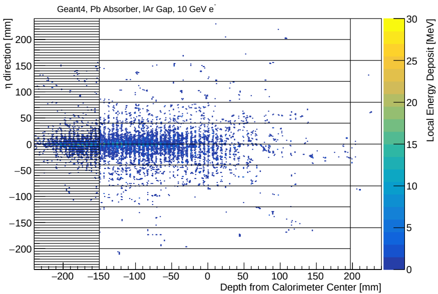


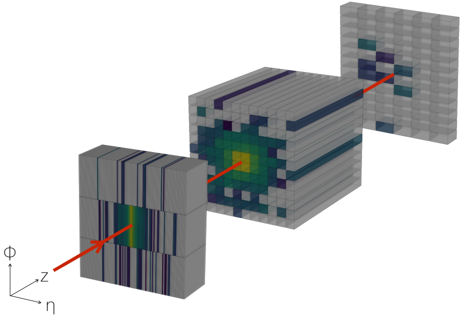
Generative Adversarial Network:
A generator and a critic play a game against each other.

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

II: CaloFlow uses 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

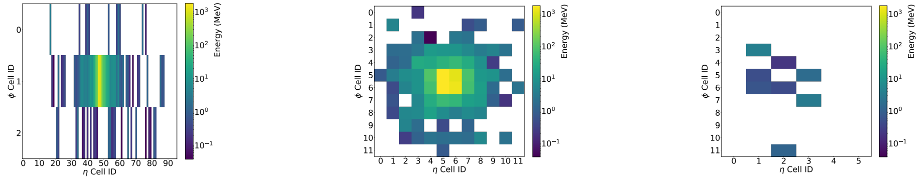




CaloGAN: Paganini, de Oliveira, Nachman [1705.02355, PRL; 1712.10321, PRD]

II: CaloFlow uses 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} 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]

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

Flow I learns $p_1(E_0, E_1, E_2|E_{\text{inc}})$
⇒ is a Masked Autoregressive Flow, optimized using the log-likelihood.

Flow II learns $p_2(\hat{\vec{\mathcal{I}}}|E_0, E_1, E_2, E_{\text{inc}})$ 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), i.e. matching IAF parameters to frozen MAF van den Oord et al.[1711.10433]

  ⇒ Fast in sampling ($\approx 500\times$ faster than CaloFlow v1)

II: 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 ⇒ $p_{\text{GEANT4}}(x) = p_{\text{generated}}(x)$
- Even if not, the classifier extracts a lot of useful information. R. Das, CK, et al. [2305.16774]

⇒ This captures the full phase space incl. correlations.

? But why wasn't this used before?

DCTRGAN: Diefenbacher et al. [2009.03796, JINST]

⇒ Previous deep generative models were separable to almost 100%!

II: 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.

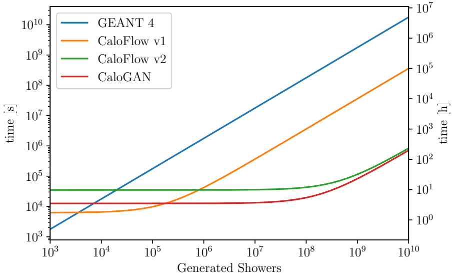
AUC		Geant4 vs. CaloGAN	Geant4 vs. (teacher) CaloFlow v1	Geant4 vs. (student) CaloFlow v2
e^+	low-level	1.000(0)	0.870(2)	0.824(4)
	high-level	1.000(0)	0.795(1)	0.762(3)
γ	low-level	1.000(0)	0.796(2)	0.760(3)
	high-level	1.000(0)	0.727(2)	0.739(2)
π^+	low-level	1.000(0)	0.755(3)	0.807(1)
	high-level	1.000(0)	0.888(1)	0.893(2)

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

II: Sampling Speed: The Student beats the Teacher!

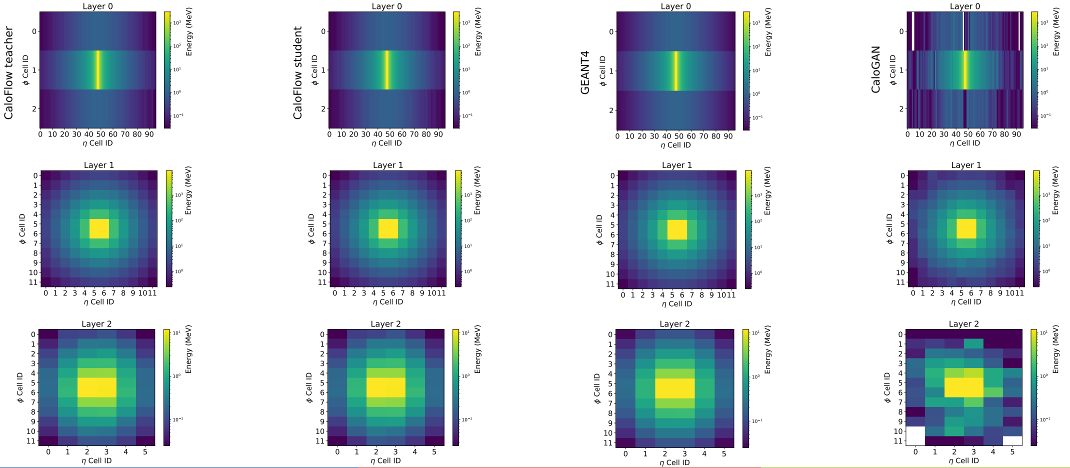
	CaloFlow* teacher	CaloFlow* student	CaloGAN*	Geant4†
training	22+82 min	+ 480 min	210 min	0 min
generation time per shower	36.2 ms	**0.08 ms**	**0.07 ms**	1772 ms

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

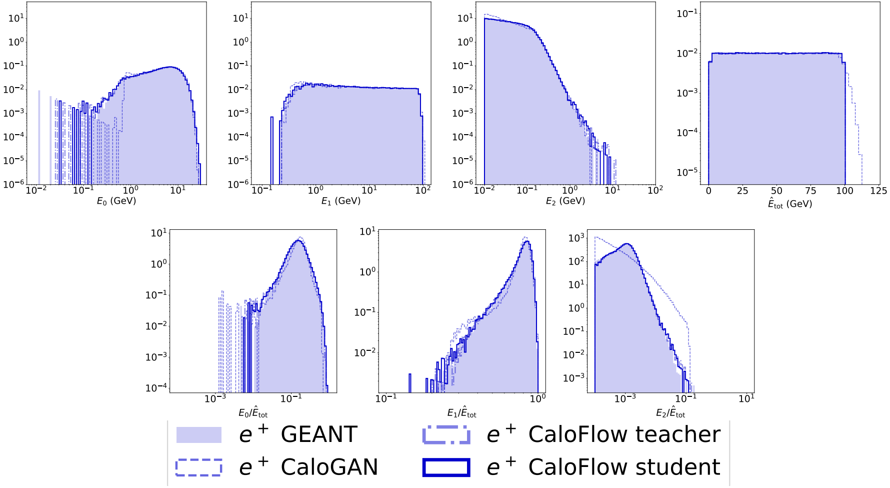


II: CaloFlow: Comparing Shower Averages: e^+.

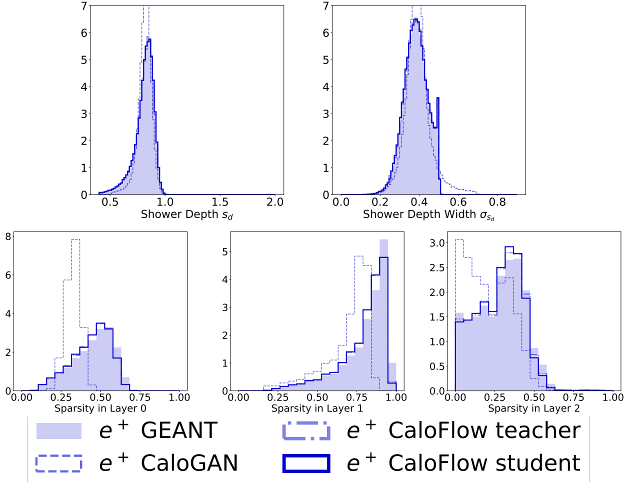
CaloFlow teacher | CaloFlow student | Geant4 | CaloGAN



II: CaloFlow: histograms: e^+.



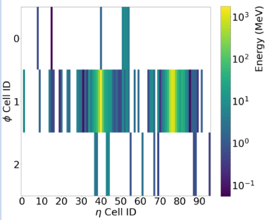
II: CaloFlow: histograms: e^+.



II: What else can we do with the likelihood?

Anomaly Detection.

- Find anomalous showers,
  e.g. coming from multiple photons.
- Works "broader" than dedicated classifiers.



CK, Nachman, Pang, Shih, Zhu [2312.11618]

Inference.

- Find which E_{inc} maximizes $p(\text{shower}|E_{inc})$.
- Is prior independent.

Du, **CK**, Nachman, Pang, Shih [in prep.]

II: Going the next step: towards deployment in FastSimulation.

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

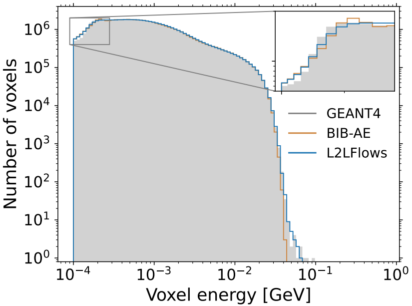
- Dataset 1: AtlFast3 trainig data (γ: 368, π: 533 voxels)
  [2109.02551, Comput.Softw.Big Sci.] CaloFlow works: CK/Pang/Shih [2210.14245]
- Dataset 2: simulated detector (e^-: 6480 voxels) ⇒ need new ideas!
- Dataset 3: simulated detector (e^-: 40500 voxels) ⇒ need new ideas!

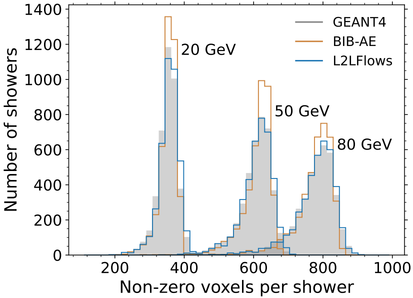
Submissions were presented at a workshop in Rome and at ML4Jets-22 / ML4Jets-23.

II: Larger datasets require new ideas — L2LFlows.

L2LFlows: Learn shower shapes one at a time, leveraging how the shower develops.

1. learns $p_1(E_1, E_2, E_3, \dots, E_{45}|E_{\text{inc}})$ → how energy is distributed among layers.
2. learns $p_i(\hat{\mathcal{I}}_i|E_i, E_{\text{inc}}, E_{i-k}, \hat{\mathcal{I}}_{i-k})$ → how the shower in the layer i looks like.



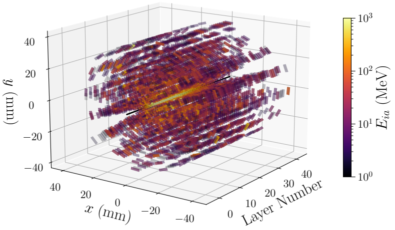


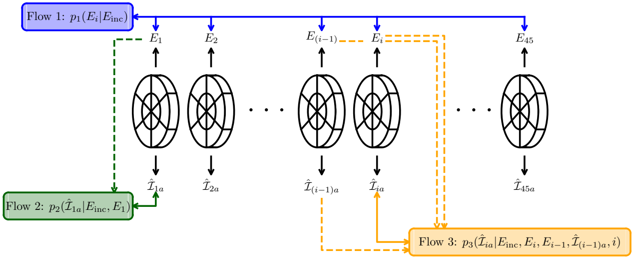
Classifier AUCs: L2LFlows: 0.8518(42) BIB-AE: 0.9947(25)

II: Larger datasets require new ideas — iCaloFlow.

iCaloFlow: Split learning $p(\vec{\mathcal{I}}|E_{\text{inc}})$ into 3 steps, leveraging the detector geometry.

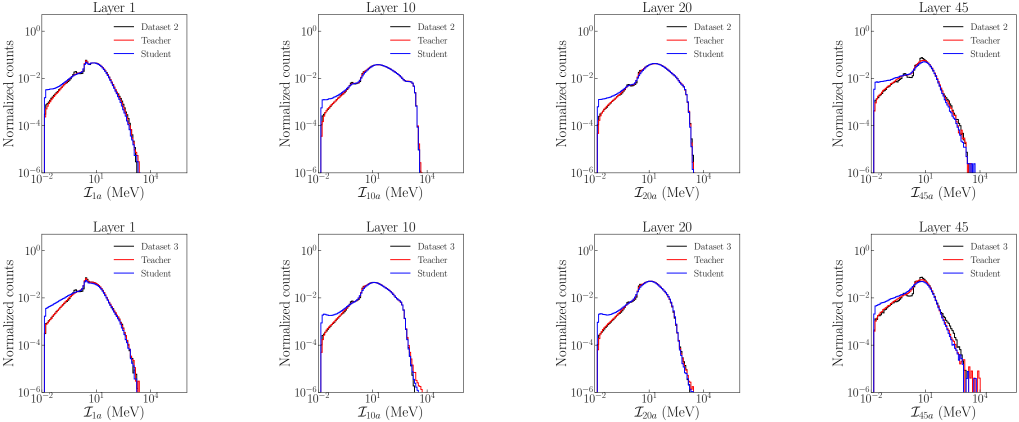
1. learns $p_1(E_1, E_2, E_3, \ldots, E_{45}|E_{\text{inc}})$ → how energy is distributed among layers.
2. learns $p_2(\mathcal{I}_1|E_1, E_{\text{inc}})$ → how the shower in the first layer looks like.
3. learns $p_3(\mathcal{I}_n|\mathcal{I}_{n-1}, n, E_n, E_{n-1}, E_{\text{inc}})$ → how the shower in layer n looks like, given layer $n-1$





M. Buckley, CK, I. Pang, D. Shih [2305.11934]

II: iCaloFlow: shows promising results.

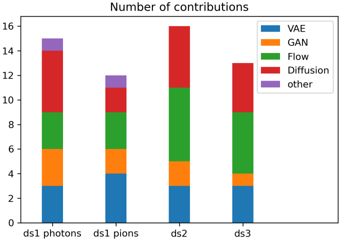


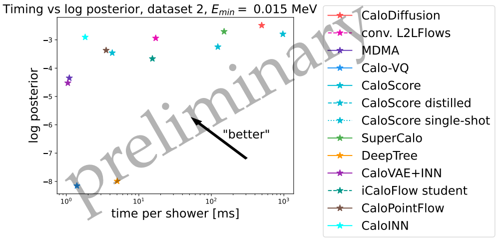
Classifier AUCs:				
	dataset 2, low:	0.797(5)	dataset 2, high:	0.798(3)
	dataset 3, low:	0.911(3)	dataset 3, high:	0.941(1)

II: The Fast Calorimeter Simulation Challenge 2022.

Final write-up is currently being prepared. It compares:

- high-level features (observables)
- low-level features (voxels) via classifiers
- time and memory usage
- ...





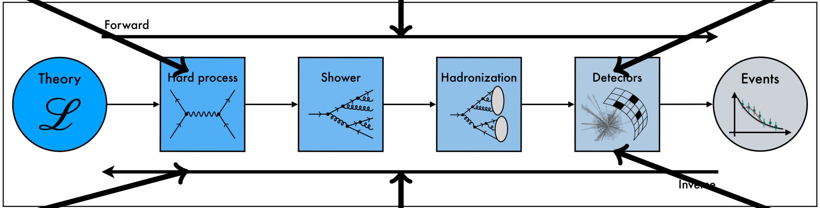
see C.Krause at ML4Jets 2023

Improving HEP Simulation and Analyses with INNs

1. Phase Space Sampling

End-to-End Simulation

2. Detector Simulation



Inference

3. **Bump-Hunt Searches**

Unfolding

Lattice QCD

III: How to look for New Physics at the LHC with few assumptions

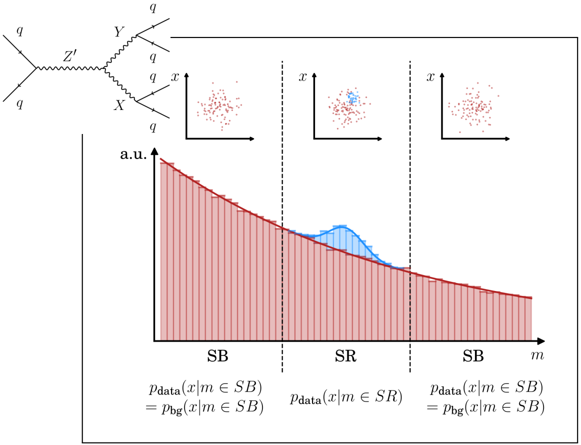
Assumptions in Bump Hunts:

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

Select events with

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

1. Scan Signal Region (SR) across m
2. Perform background fit and obtain p-value for bump.

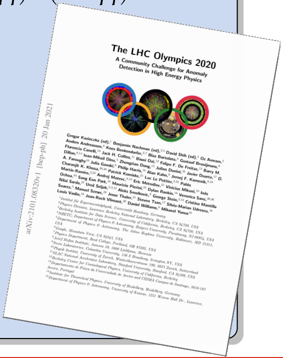


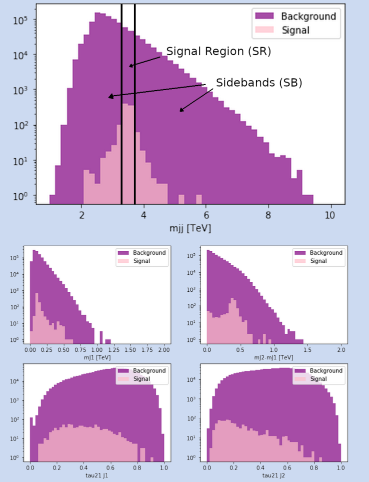
III: The LHC-Olympics looked at di-jet Resonances.

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]

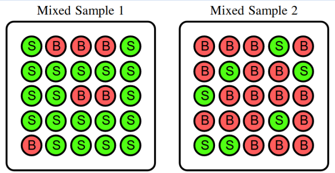

The LHC Olympics 2020
A Community Challenge for Anomaly Detection in High Energy Physics
arXiv:2101.08320v1 [hep-ph] 20 Jan 2021



III: We can get the likelihood ratio using ML: Classifiers.

According to the Neyman-Pearson Lemma we have:

- The likelihood ratio is the most powerful test statistic to distinguish two samples.
- A powerful classifier trained to distinguish the samples should therefore learn (something monotonically related to) this.



- Classification without Labels (CWoLa) learns from mixed samples.
- An optimal classifier is also optimal for distinguishing S from B.

E.M. Metodiev, B. Nachman, J. Thaler, [1708.02949 JHEP]

III: 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

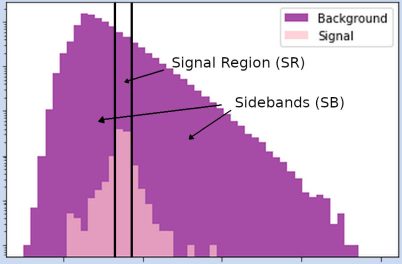
- idealized anomaly detector:

  train classifier on data and simulated background

  - depends on quality of simulation
  - still background model dependent
  - provides upper limit on data-driven anomaly detection

III: Data-driven approaches are background model-independent.

Anomaly Detection with Density Estimation (ANODE):



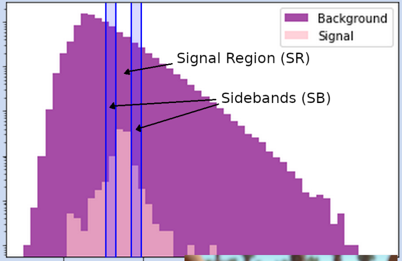
- train "outer" density estimator $p_{\text{data}}(x|m_{JJ} \in SB)$
- train "inner" density estimator $p_{\text{data}}(x|m_{JJ} \in SR)$
- compute $\frac{p_{\text{inner}}(x|m_{JJ})}{p_{\text{outer}}(x|m_{JJ})}$ for $m_{JJ} \in SR$
- robust against correlations, but harder learning task.

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

III: Data-driven approaches are background model-independent.

Classification without Labels (CWoLa) Hunting:

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



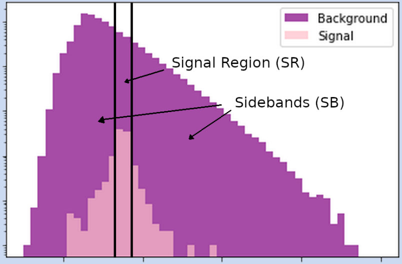
E.M. Metodiev, B. Nachman, J. Thaler, [1708.02949 JHEP]
J.H. Collins, K. Howe, B. Nachman, [1805.02664 PRL, 1902.02634 PRD]

"Coala Hunting" via midjourney.com ⇒

III: 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



⇒ 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]

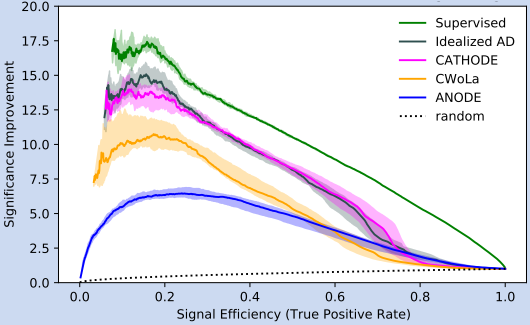
III: CATHODE outperforms other anomaly detectors.

Results:

- CATHODE approaches idealized AD
- outperforms ANODE (only 1 density estimator)
- outperforms CWoLa (robust against correlations)
- benefits from oversampling

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

Significance Improvement Characteristic $= \mathrm{TPR}/\sqrt{\mathrm{FPR}}$



⇒ These strategies are now being explored in ATLAS and CMS. ATLAS [2005.02983, PRL]

Improving HEP Simulation and Analyses with Invertible Neural Networks

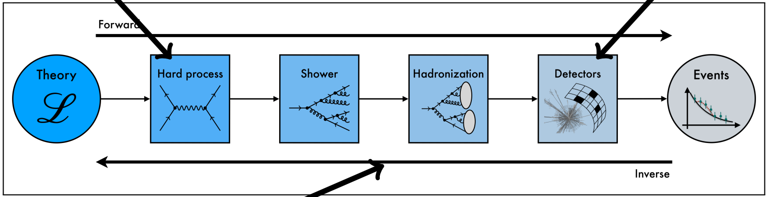
- We expect 20$\times$ more LHC data in the future.
- Understanding everything based on 1st principles suffers from computational bottlenecks that can be tackled with ML, and especially Normalizing Flows.

Improving HEP Simulation and Analyses with Invertible Neural Networks

- We expect 20× 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

⇒`https://iml-wg.github.io/HEPML-LivingReview/`