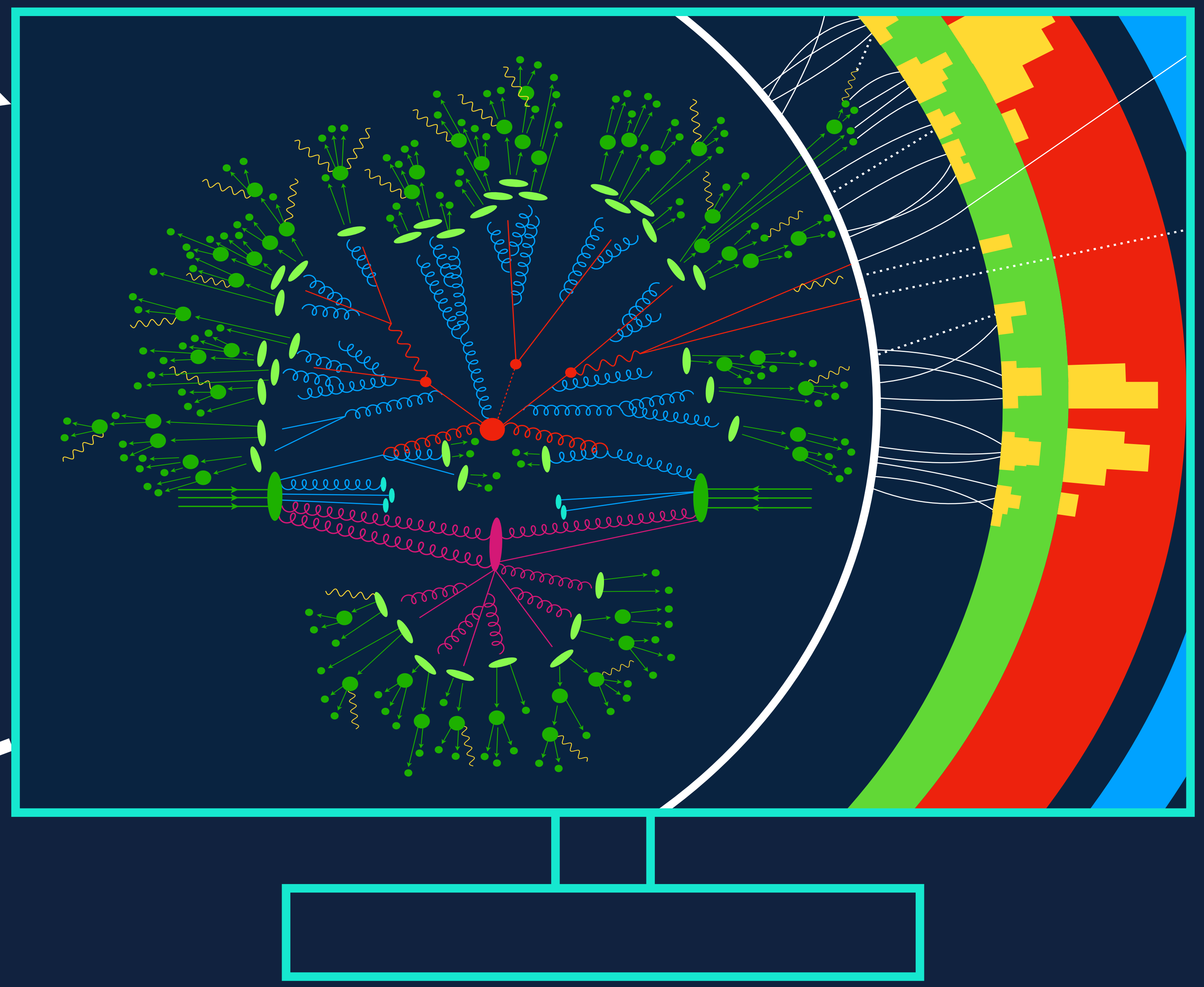
Modern Machine Learning for the LHC Simulation Chain

$$
\begin{aligned}
\mathscr{L} = &-\frac{1}{4}F^{\mu\nu}F_{\mu\nu} \\
&+ i\bar{\psi}\gamma^{\mu}D_{\mu}\psi \\
&+ \bar{\psi}^{i}_{L}y_{ij}\Phi\psi^{j}_{R} + \text{h.c.} \\
&+\ D_{\mu}\Phi\ {}^{2} + V(\Phi) + \text{BSM}
\end{aligned}
$$



UCLouvain

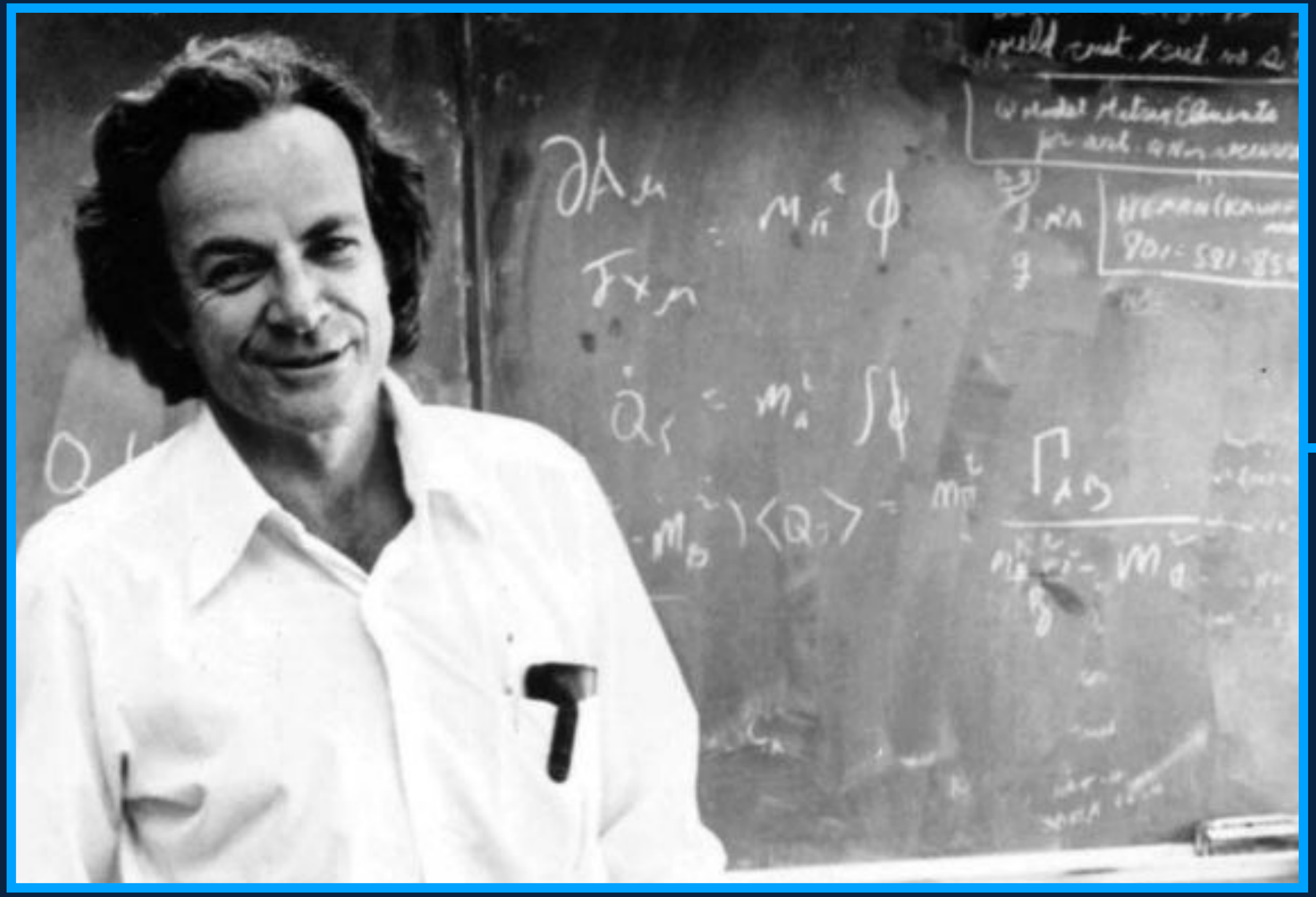
Hammers & Nails — Suisse Edition 2023

Ramon Winterhalder — UCLouvain

Why do we talk about simulations?

A theorist perspective…

Why do we talk about simulations?

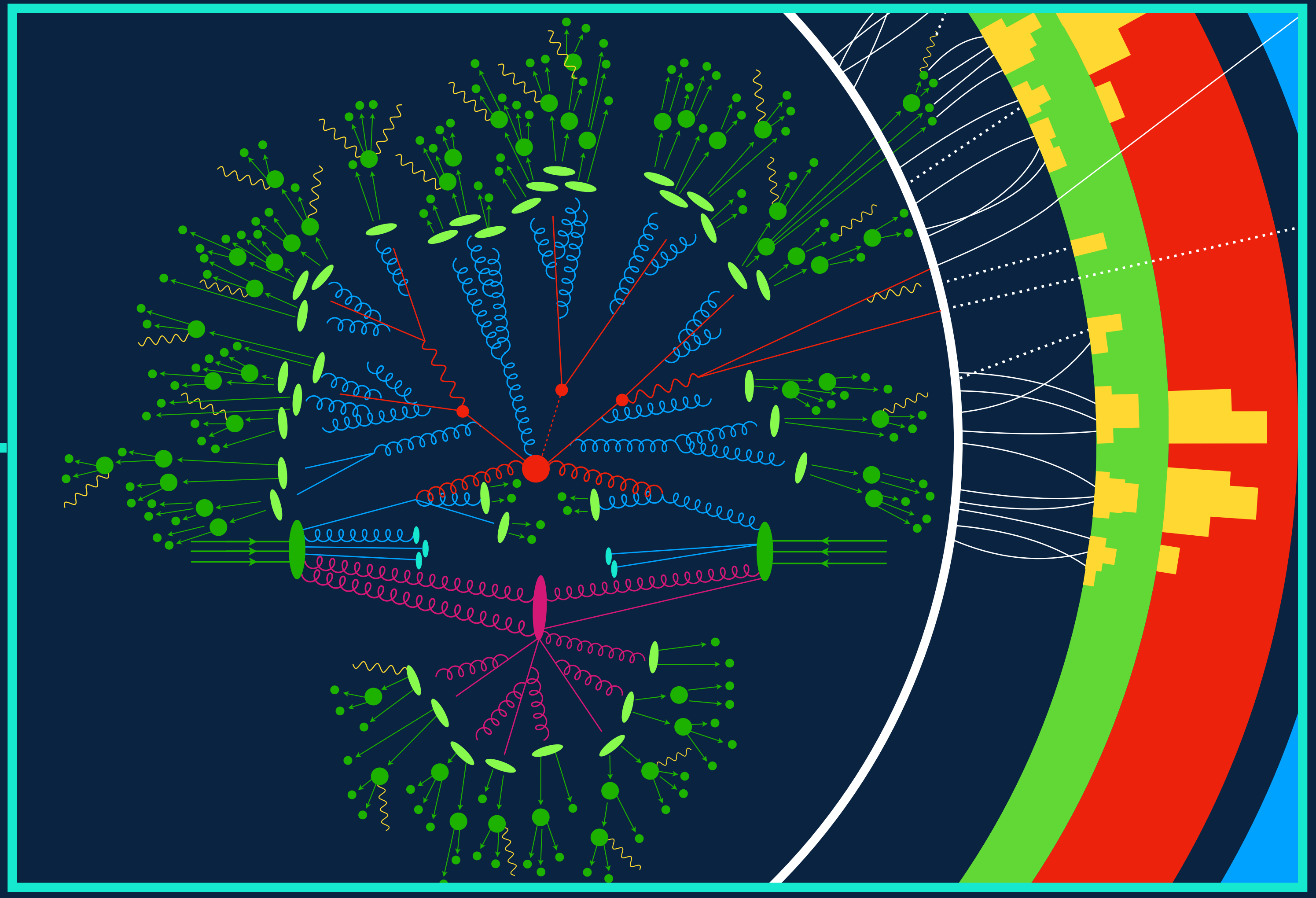


"What I cannot create, I do not understand"

Richard P. Feynman

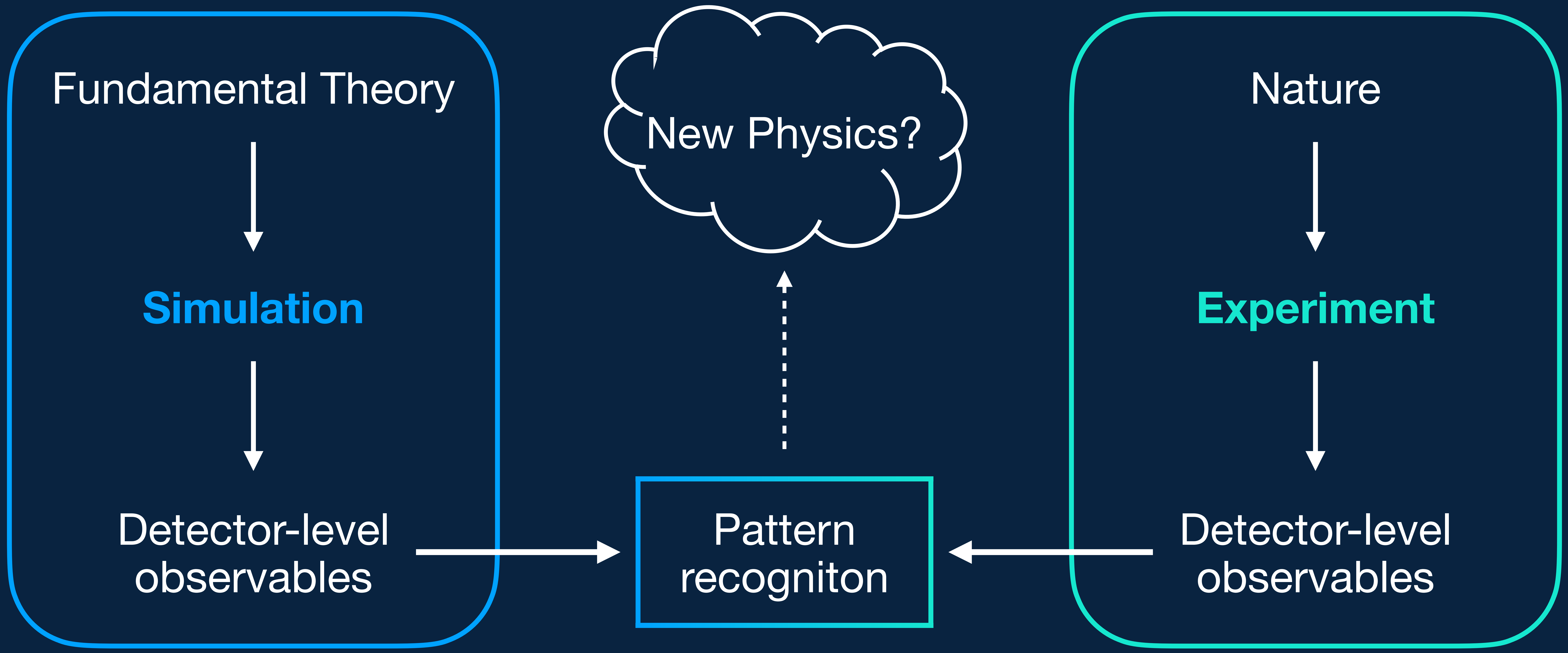
"What I understand, I can create"

LHC Simulations & Generative Modelling

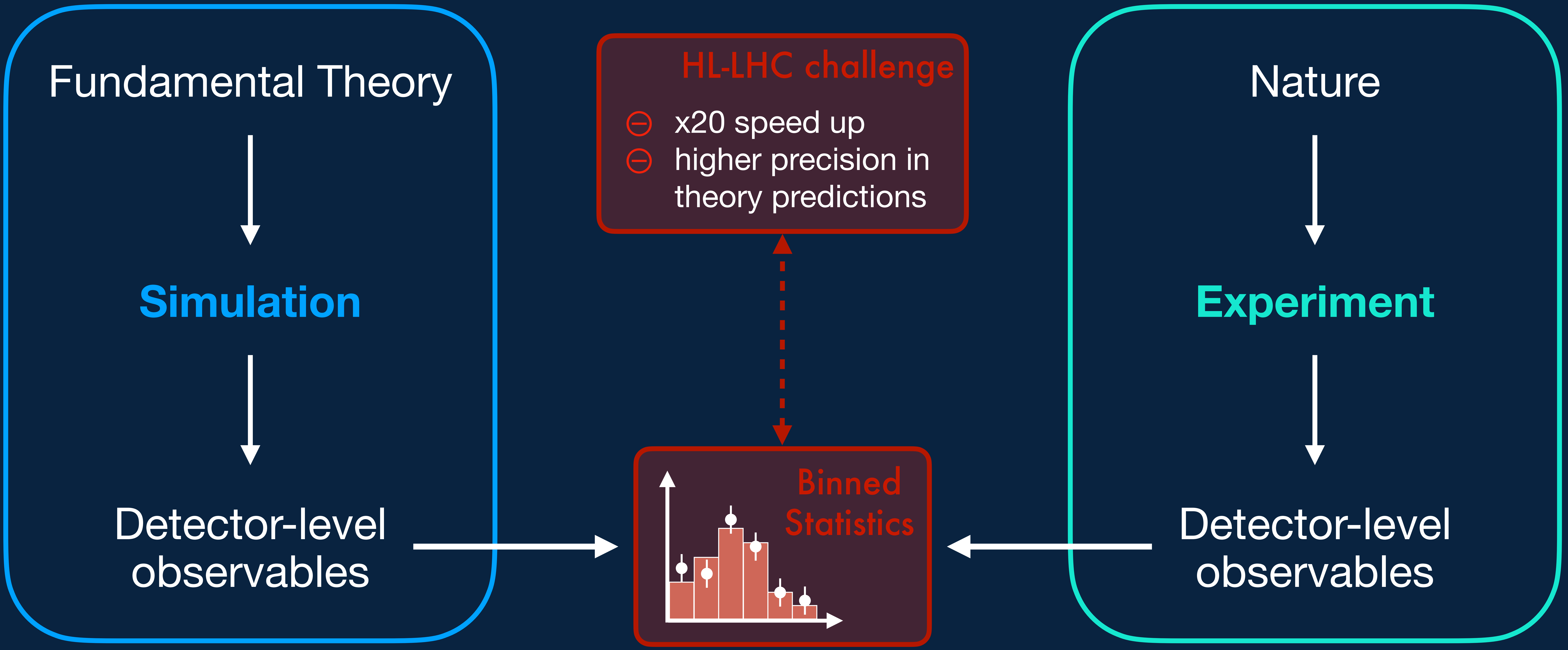


Why do we need them for LHC physics?

LHC analysis (oversimplified)



Problem: Information bottleneck



Solution: LHC analysis + ML

Fundamental Theory

Fast + precise simulations

Simulation

Unfolding

Detector-level observables

HL-LHC chance

- ⊕ x20 speed up
- ⊕ higher precision in theory predictions

Optimal Inference

$$r = \frac{p(x|\theta_0)}{p(x|\theta_1)}$$

$$s = \nabla_\theta \log p(x|\theta)$$

Nature

Experiment

Detector-level observables

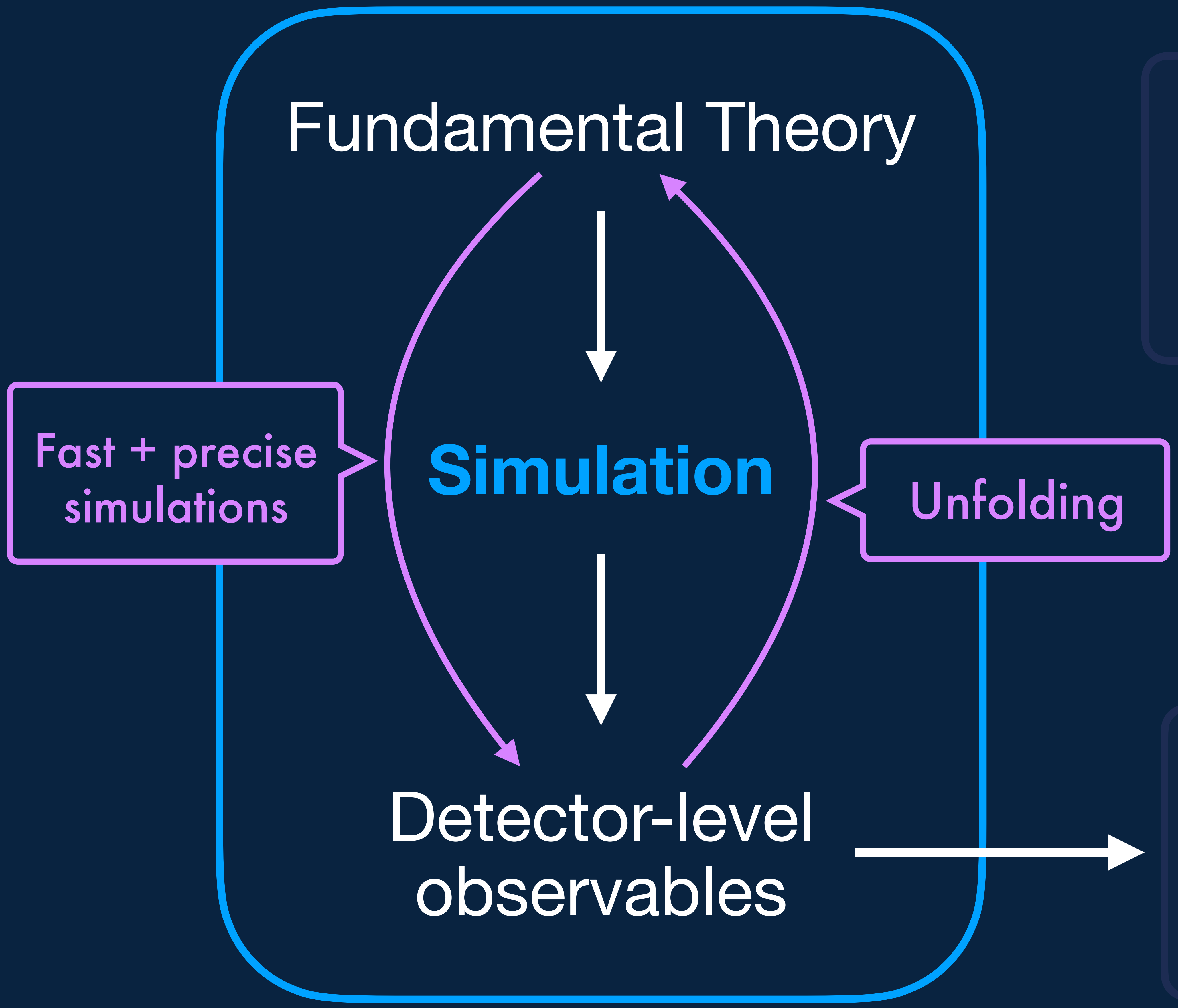
Likelihood-free inference (e.g. MadMiner)
[1506.02169, … 00020, 1805.12244, 1907.10621, 2101.07263, 2210.01680, 2305.10500, 2308.05704,…]

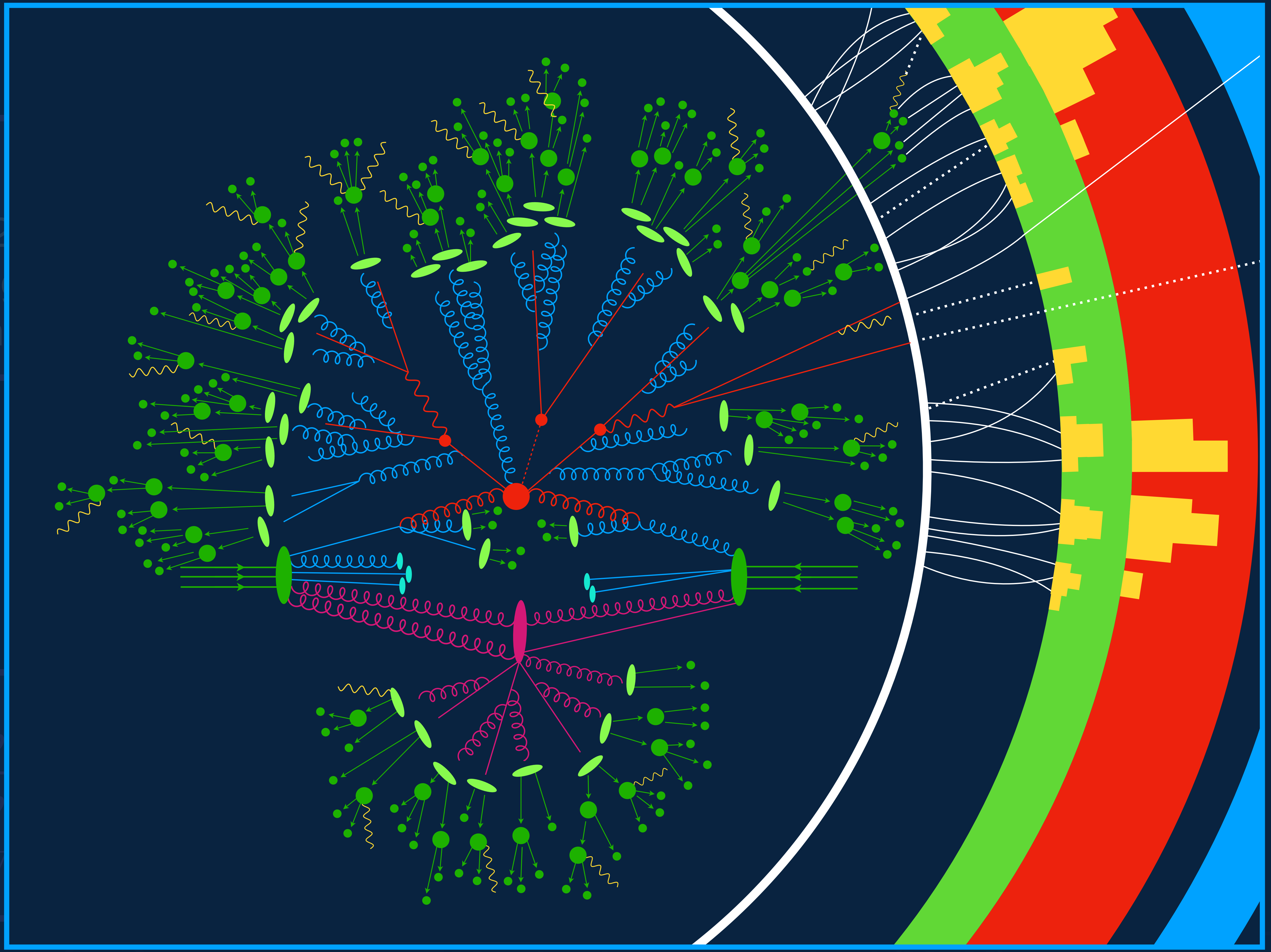
Talk by Jakob Macke

Matrix element method (MEM)
[hep-ex/9808029, hep-ex/0406031, hep-ex/0605118, 1003.1316, 1007.3300, … 1502.02485, 1511.05980, 1511.06170, 1512.03429, 1606.03107, 1710.10699, 1712.03266, 1805.08555, 2008.10949, 2210.00019, 2310.07752,…..]

Talk by Theo Heimel

Solution: LHC analysis + ML



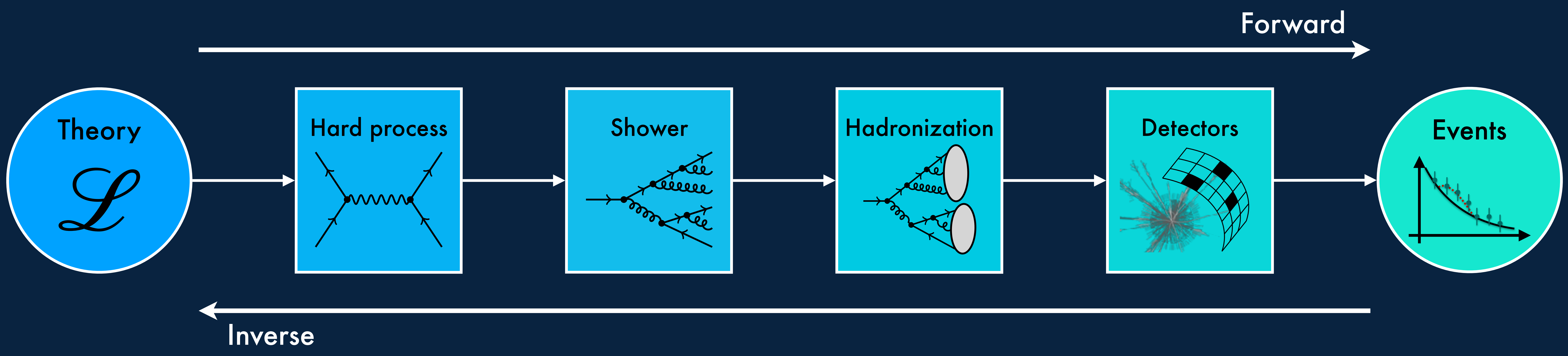


➡ We want to understand **all aspects of data** based on **first principles**!

Understanding LHC data based on 1^{st} principles

What do we need to understand the data?

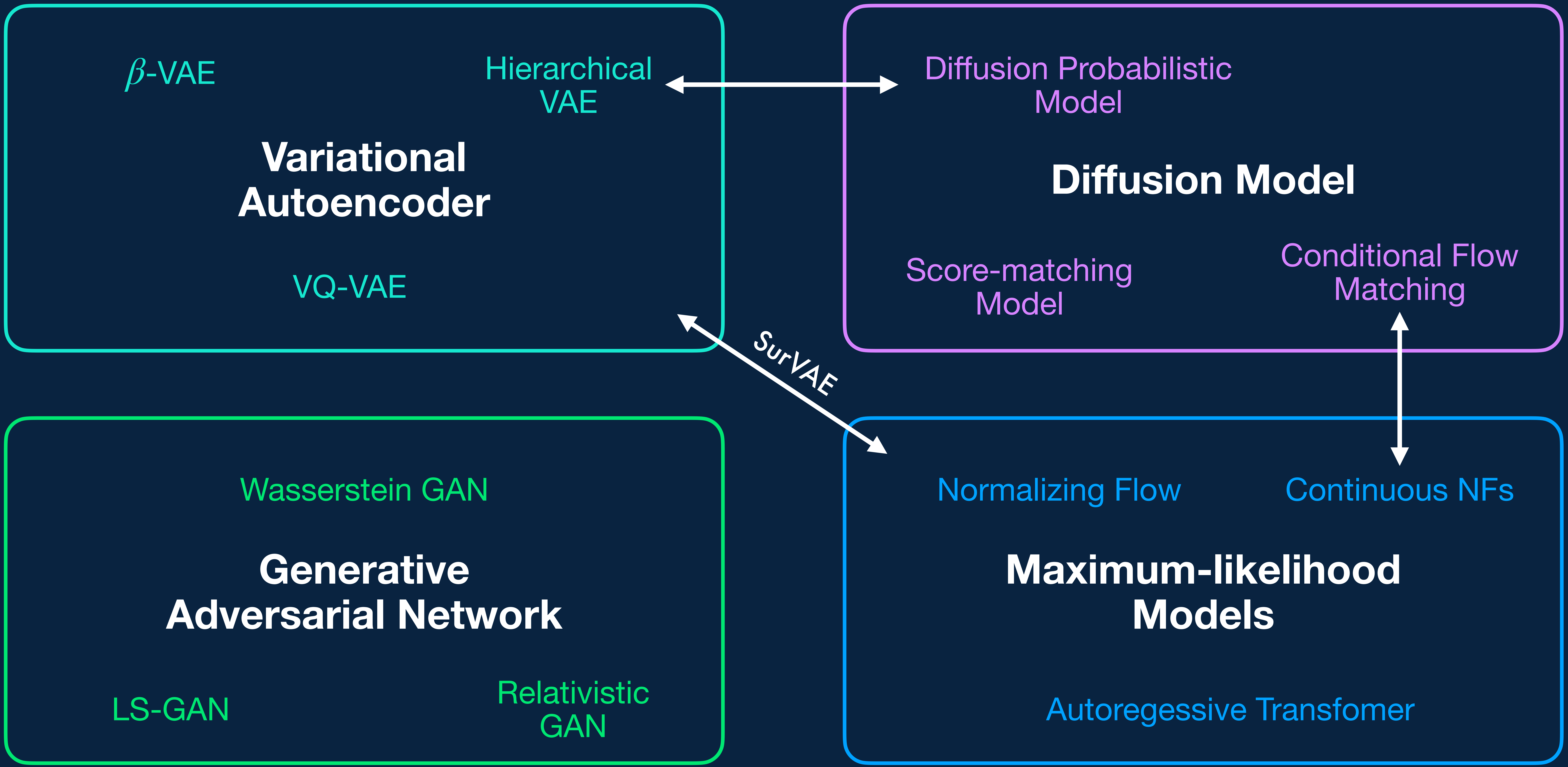
1. Precision simulations (a lot) ← this talk
2. Optimized analyses for high-dimensional data ← Talks by Jakob Macke, Theo Heimel



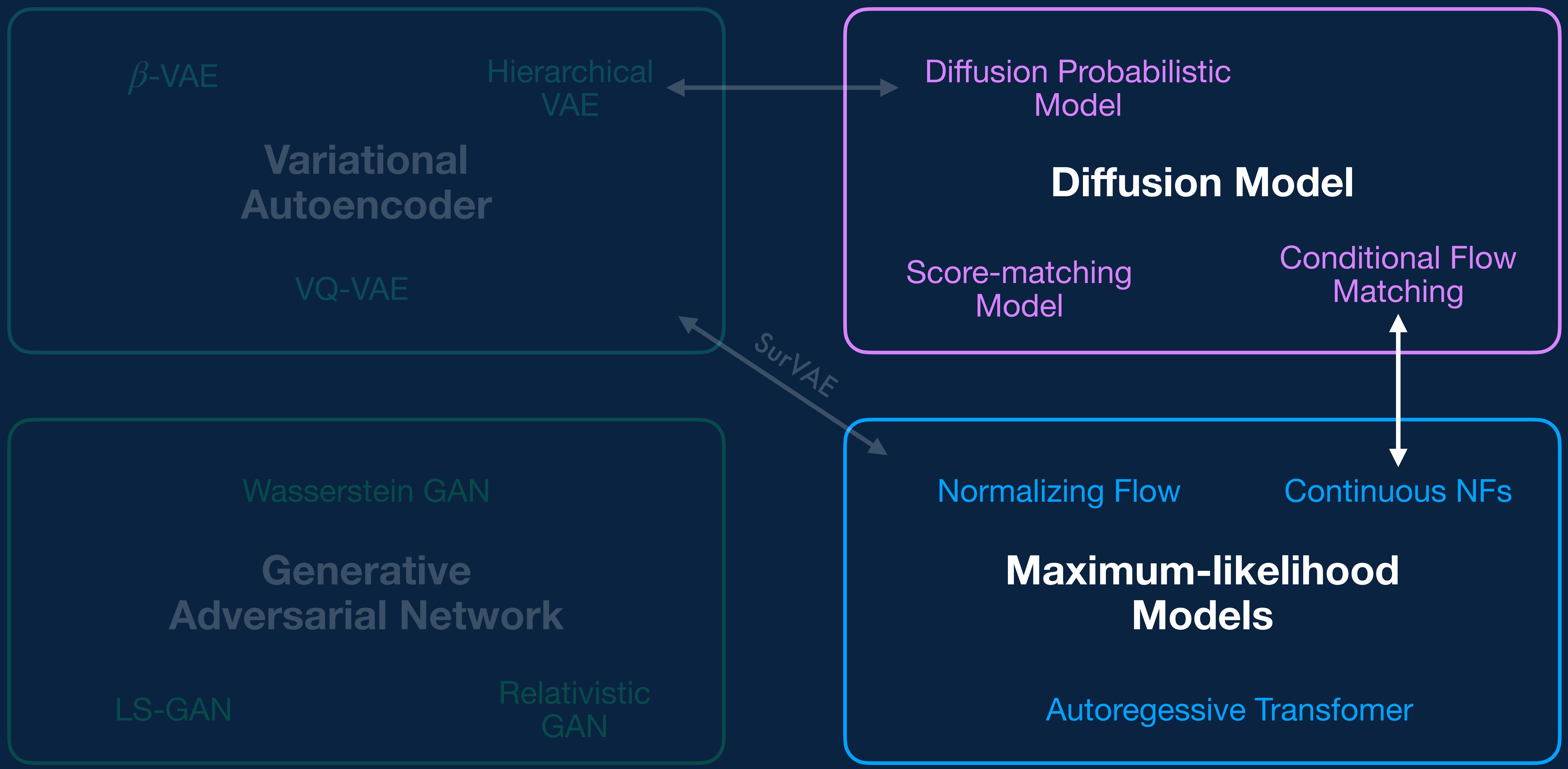
→ **Machine Learning** has significant impact on all aspects

Deep generative models

Deep generative models



Deep generative models



Deep generative models

- ⊕ State-of-the-art in precision
- ⊕ Fast and stable training
- ⊖ Slow likelihood estimation

Application dependent

- ⊕ Fast training and evaluation
- ⊕ Tractable and fast likelihoods
- ⊖ Reduced flexibility and expressivity

Diffusion Probabilistic Model

Diffusion Model

Score-matching Model

Conditional Flow Matching

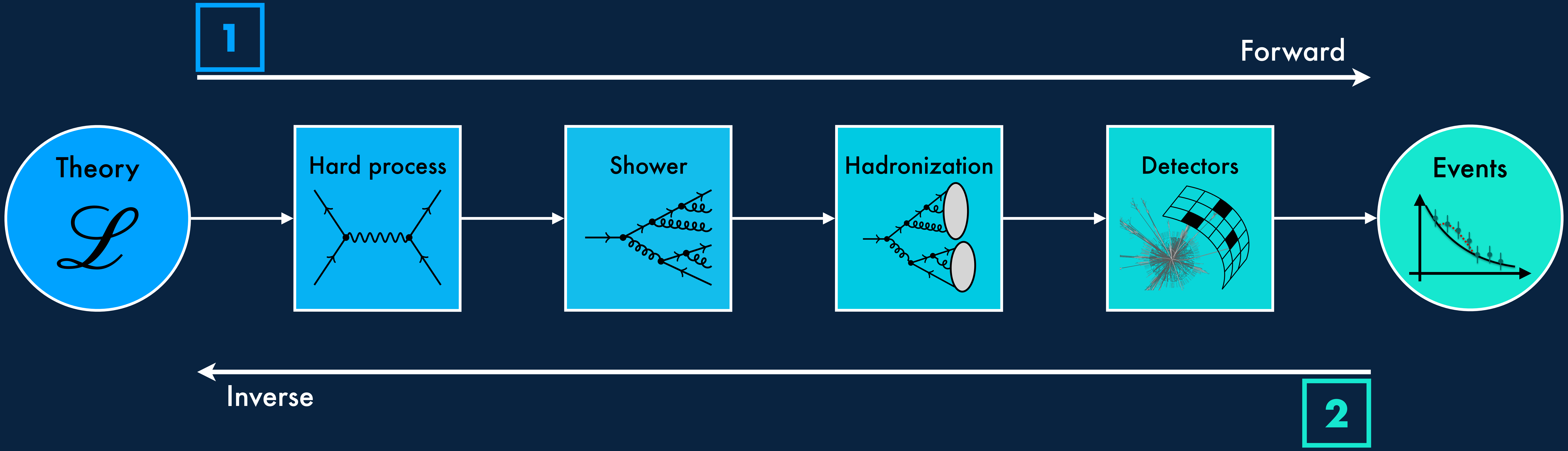
Normalizing Flow

Continuous NFs

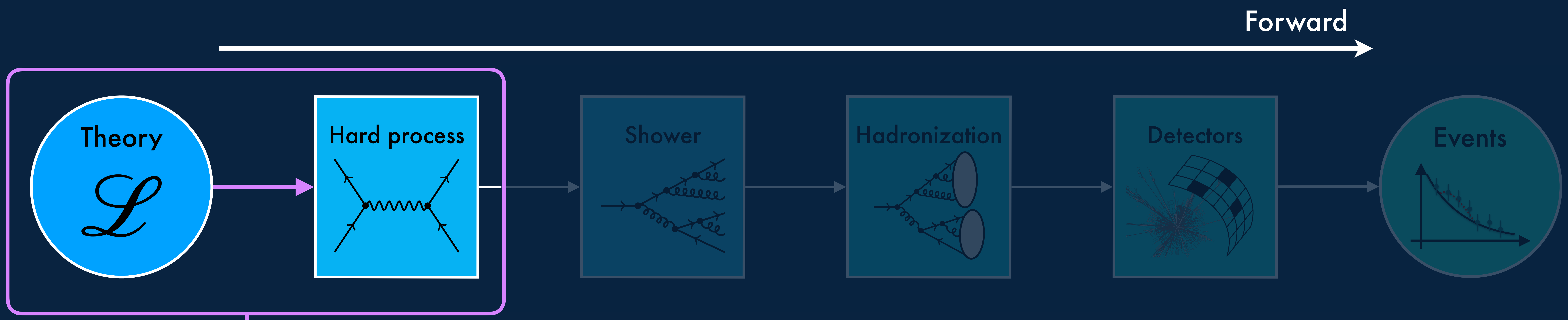
Maximum-likelihood Models

Autoregessive Transfomer

ML aided simulation chain



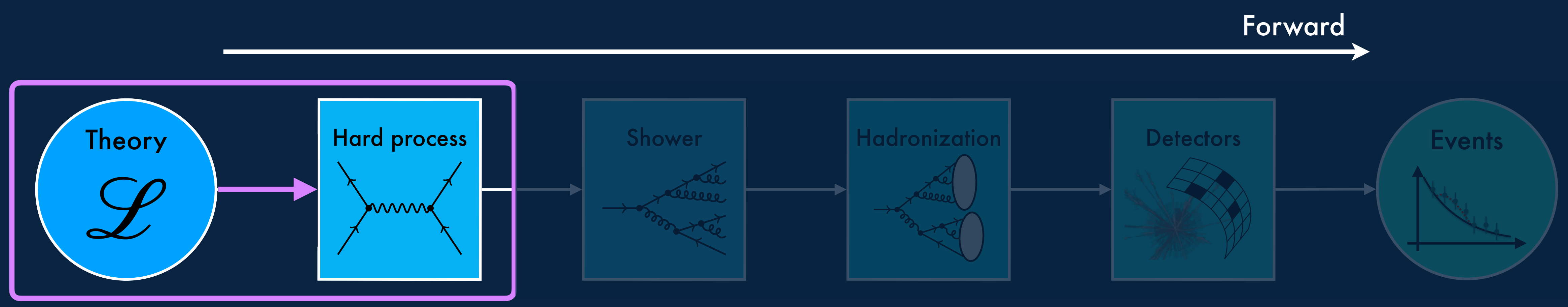
ML for forward simulations



Calculate (differential) cross sections

$$d\sigma \sim \overset{1}{\boxed{\text{pdf}}} \times \overset{2}{\boxed{M(x)}^{2}} \times \overset{3}{\boxed{d\Phi}}$$

ML for forward simulations

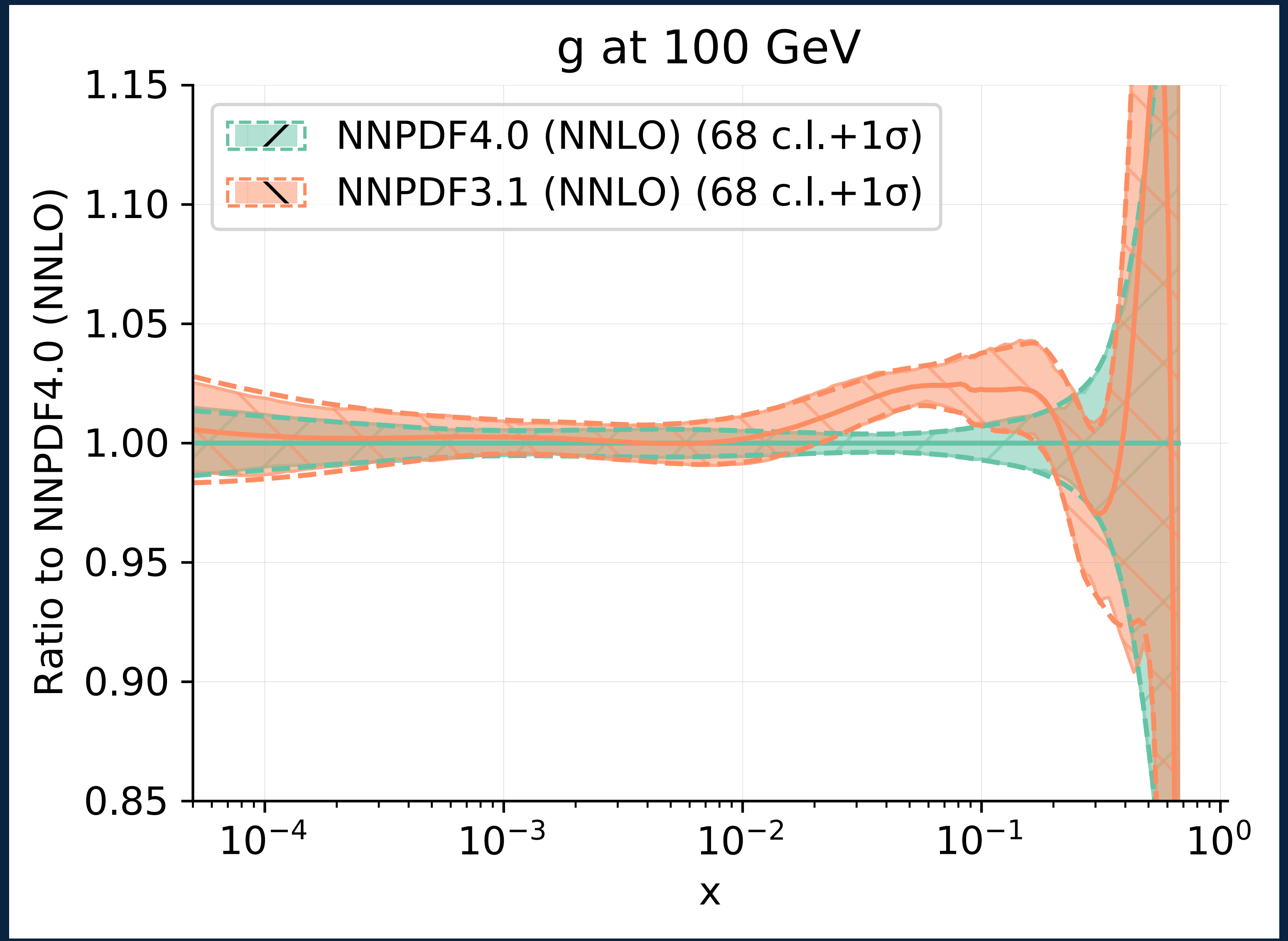


$$d\sigma \sim \boxed{\text{pdf}} \times |M(x)|^2 \times \text{phase space}$$

PDFs: ML reduces uncertainties

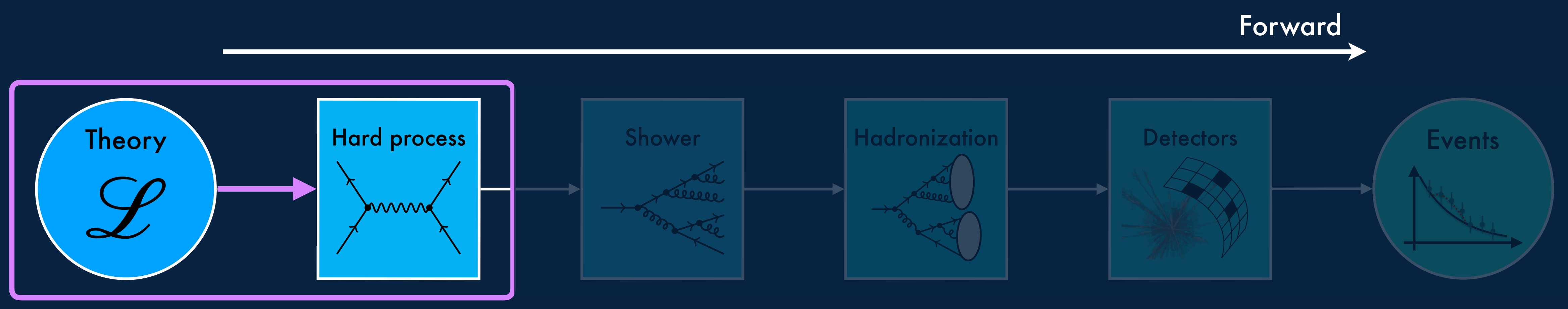
- **NNPDF** uses NN for a long time (no parametric function)
- Modern ML and hyper-opt
  → reduced uncertainties: **3-5% → 1%**
- **GAN**-enhanced PDF compression

NNPDF [2109.02653]



[hep-ph/0204232, 1002.4407, 1410.8849, 1907.05075, 2010.03996, 2012.08221, 2104.04535, 2109.02653, 2109.02671, 2201.07240, 2211.01094, 2212.12569, 2302.08527, 2303.06159, 2307.05967,.....]

ML for forward simulations

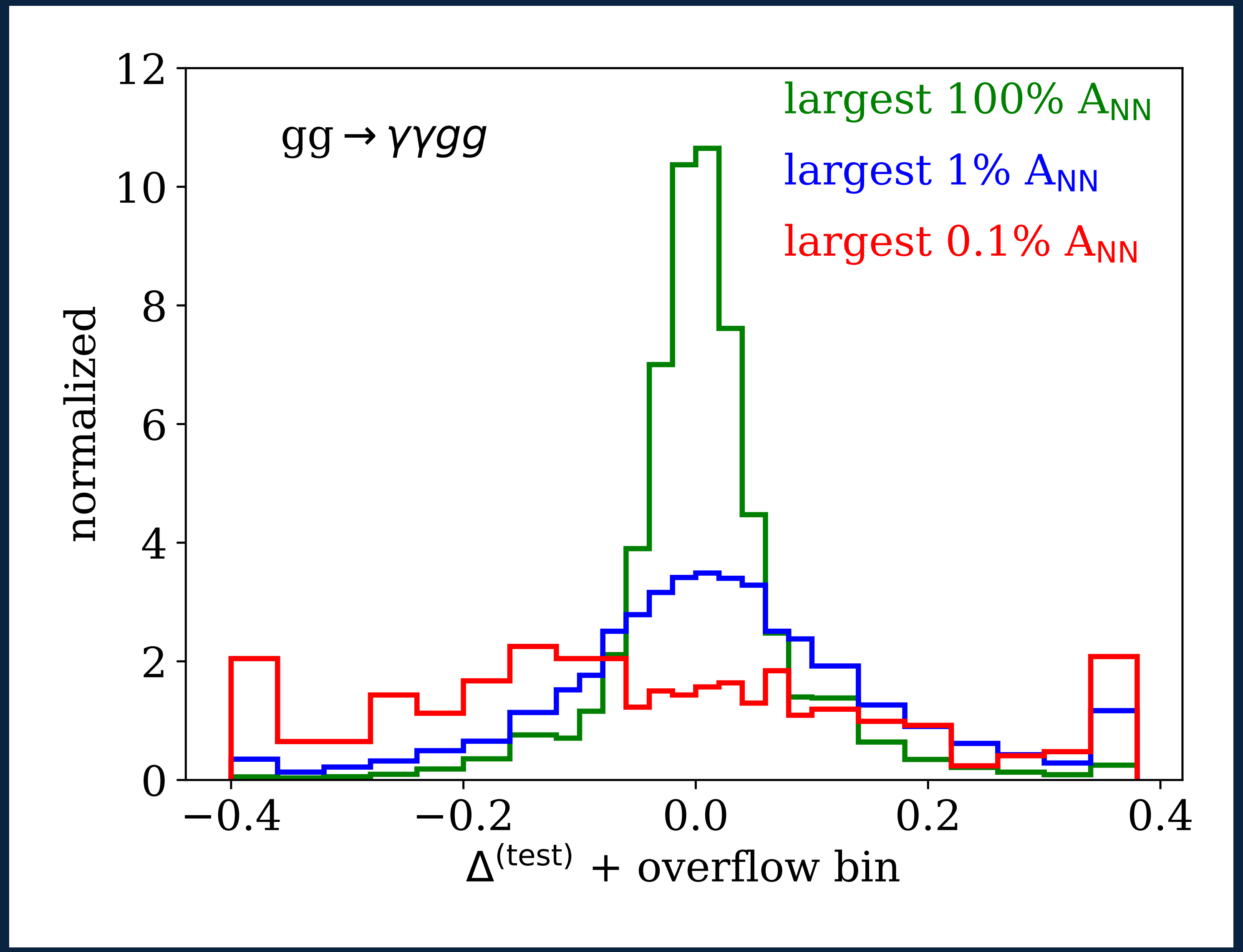


$$d\sigma \sim \text{pdf} \times \boxed{M(x)}^2 \times \text{phase space}$$

Amplitudes: avoid expensive matrix element

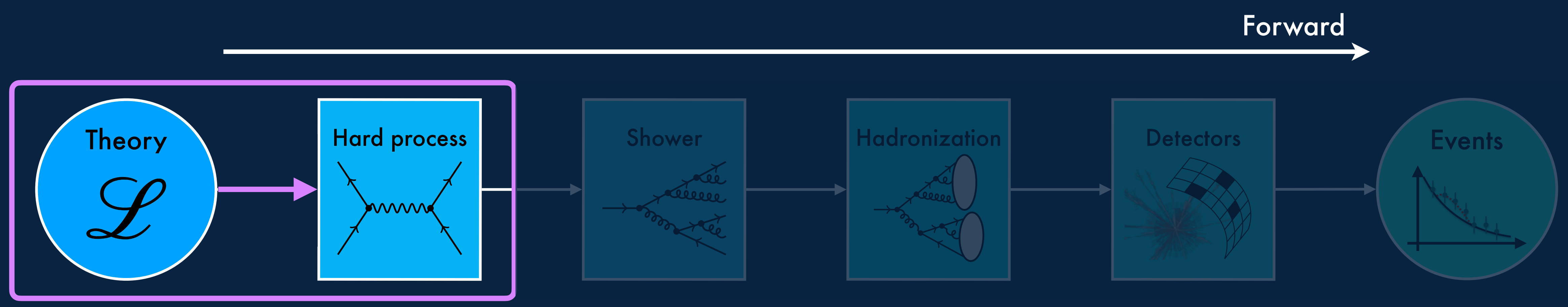
- As "simple" **regression** task
- With uncertainties/boosting using **Bayesian NN**

Badger, Butter, Luchman, Pitz, Plehn [2206.14831]



[1912.11055, 2002.07516, 2006.16273, 2008.10949, 2104.14182, 2105.04898, 2106.09474, 2107.06625, 2109.11964, 2112.09145, 2201.04523, 2206.08901, 2206.04115, 2206.14831, 2301.13562, 2302.04005, 2306.07726,…..]

ML for forward simulations

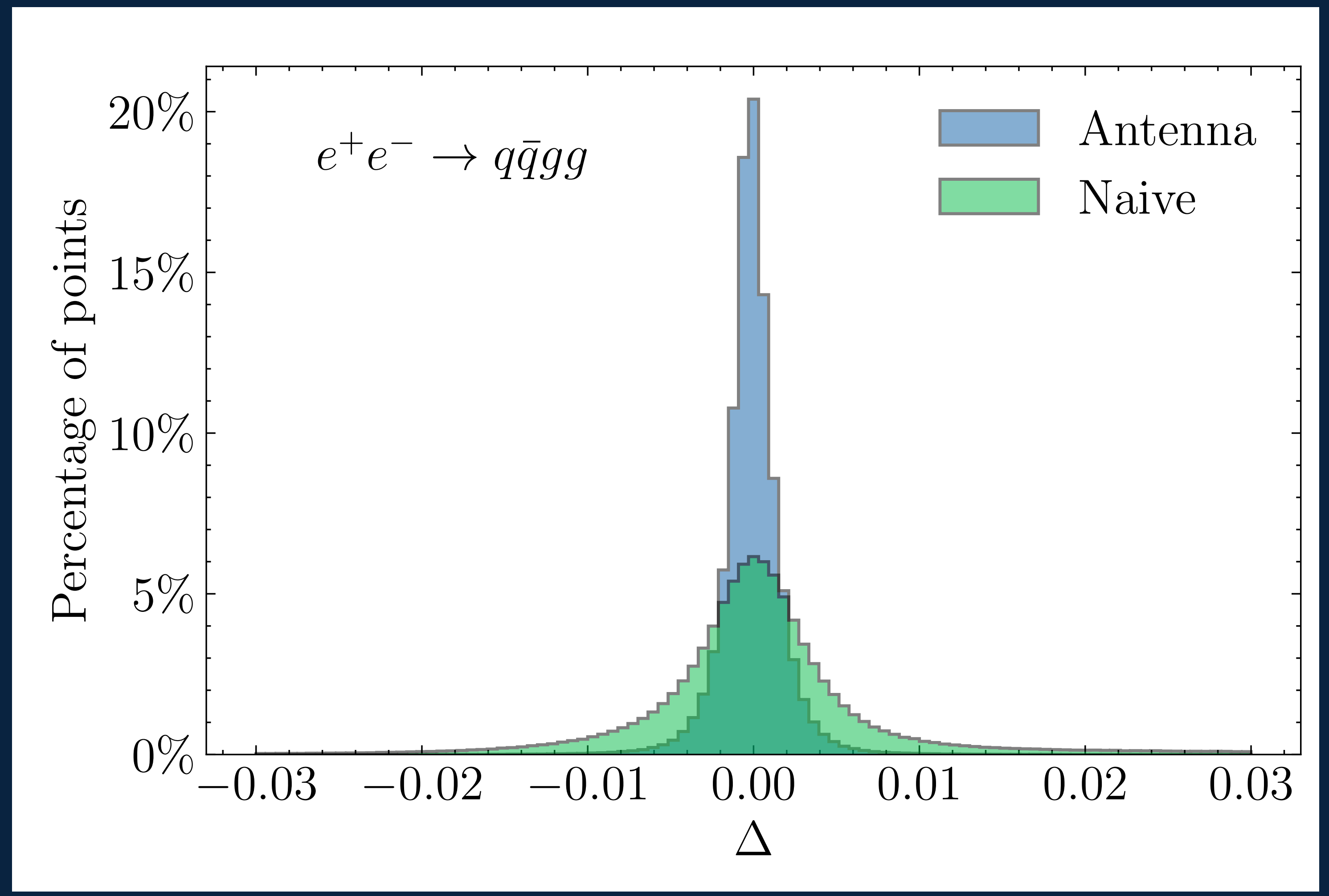


$$d\sigma \sim \text{pdf} \times \boxed{M(x)}^{2} \times \text{phase space}$$

Amplitudes: avoid expensive matrix element

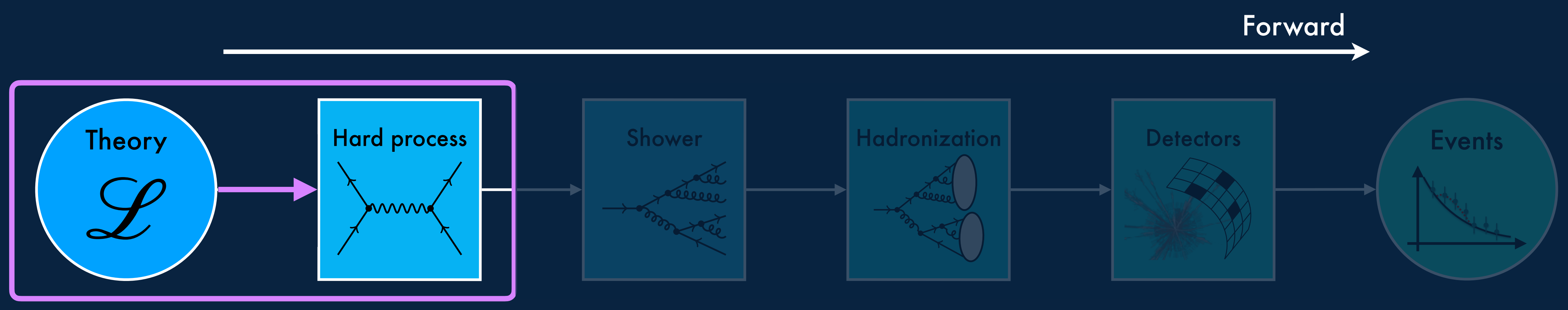
- As "simple" **regression** task
- With uncertainties/boosting using **Bayesian NN**
- Using factorisation ansatz to reach **‰ level** accuracy

Maître, Truong [2302.04005]



[1912.11055, 2002.07516, 2006.16273, 2008.10949, 2104.14182, 2105.04898, 2106.09474, 2107.06625, 2109.11964, 2112.09145, 2201.04523, 2206.08901, 2206.04115, 2206.14831, 2301.13562, 2302.04005, 2306.07726,…..]

ML for forward simulations



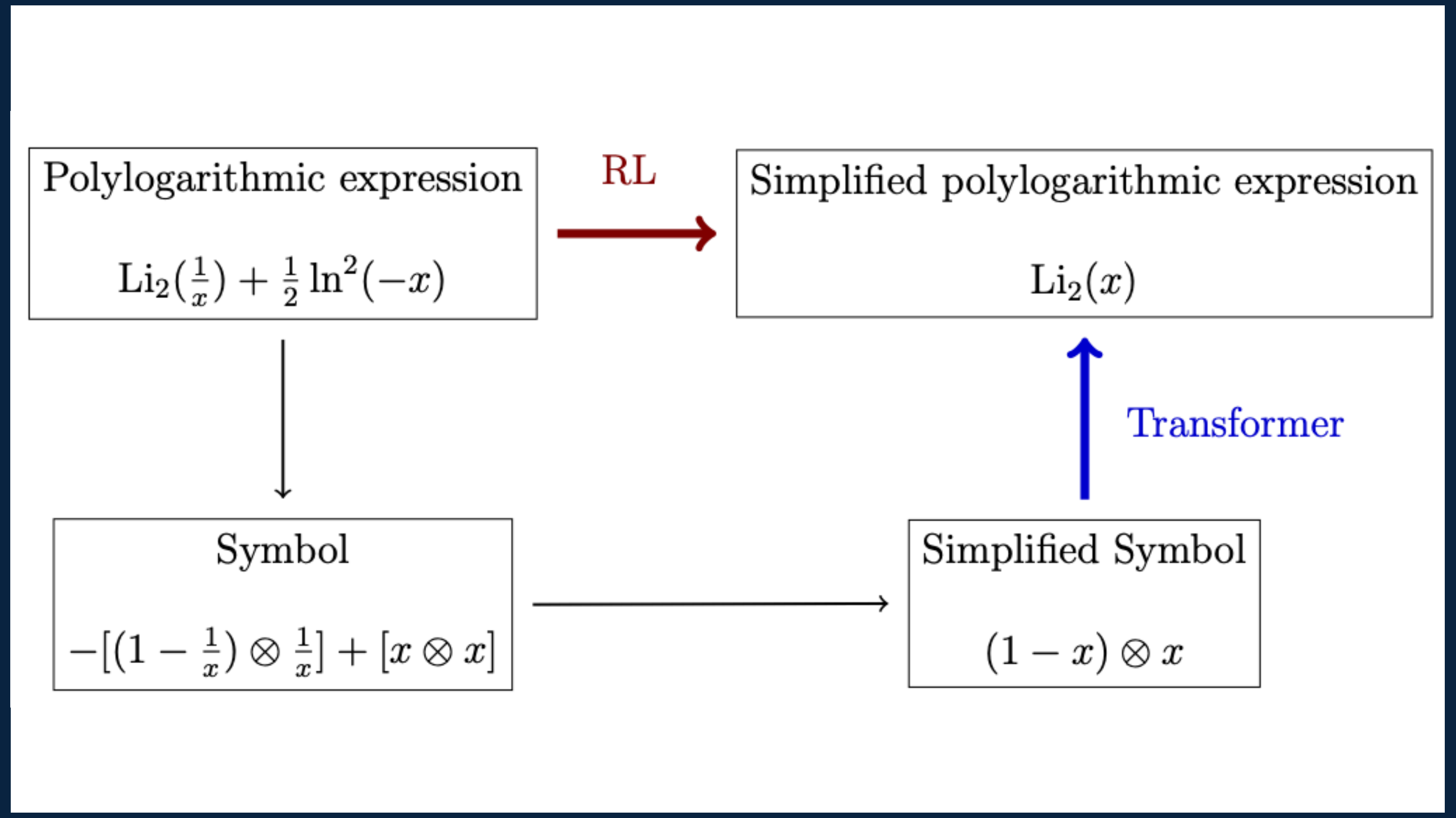
$$\mathrm{d}\sigma \sim \text{pdf} \times \boxed{M(x)}^{2} \times \text{phase space}$$

Amplitudes: avoid expensive matrix element

- As "simple" **regression** task
- With uncertainties/boosting using **Bayesian NN**
- Using factorisation ansatz to reach **‰ level** accuracy
- **RL and/or Transformer** for simplifications of Polylogarithms
- NN-assisted **contour deformation** (Loop integrals)

[1912.11055, 2002.07516, 2006.16273, 2008.10949, 2104.14182, 2105.04898, 2106.09474, 2107.06625, 2109.11964, 2112.09145, 2201.04523, 2206.08901, 2206.04115, 2206.14831, 2301.13562, 2302.04005, 2306.07726,…..]

Dersey, Schwartz, Zhang [2206.04115]



NNContour

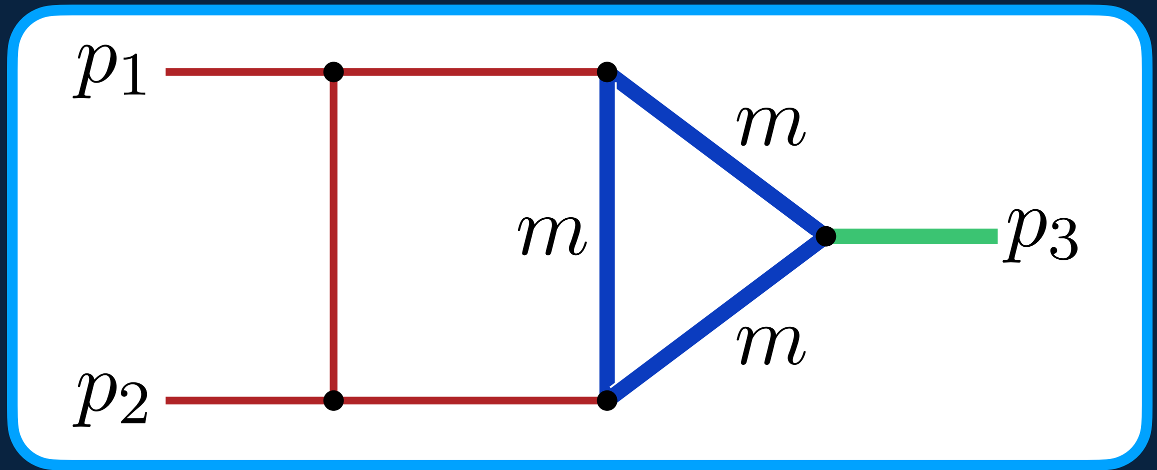
ML for loop integrals

RW, Magerya, Villa, Jones, Kerner, Butter, Heinrich, Plehn [2112.09145]

NNContour — ML for loop integrals

RW et al. [2112.09145]

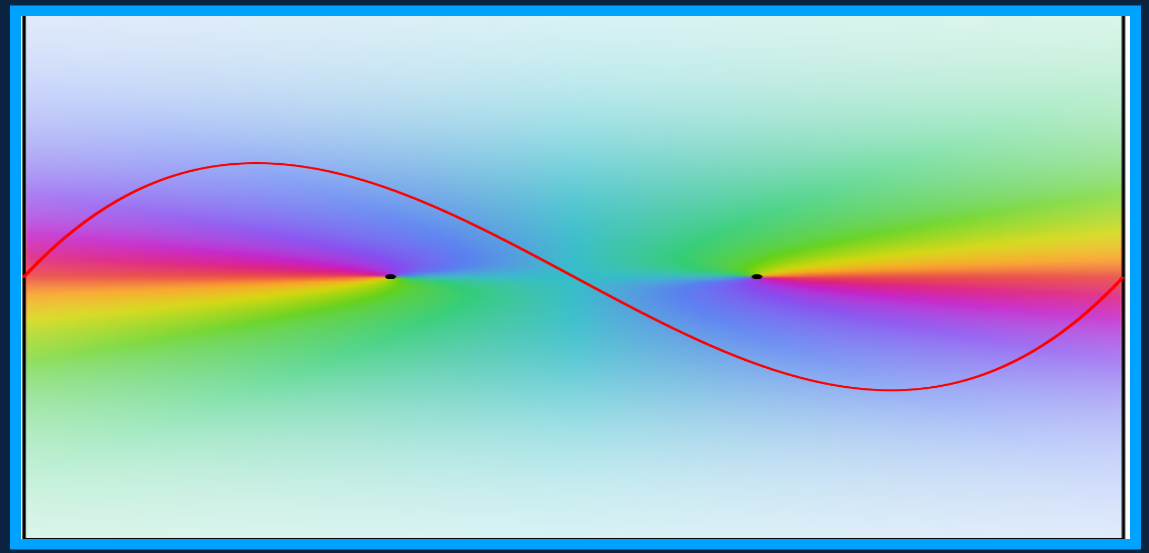
$$G \propto \int_0^1 \prod_{j=1}^{N-1} \mathrm{d}x_j\, x_j^{\nu_j-1} \frac{U^{\nu-(L+1)D/2}}{F^{\nu-LD/2}}$$

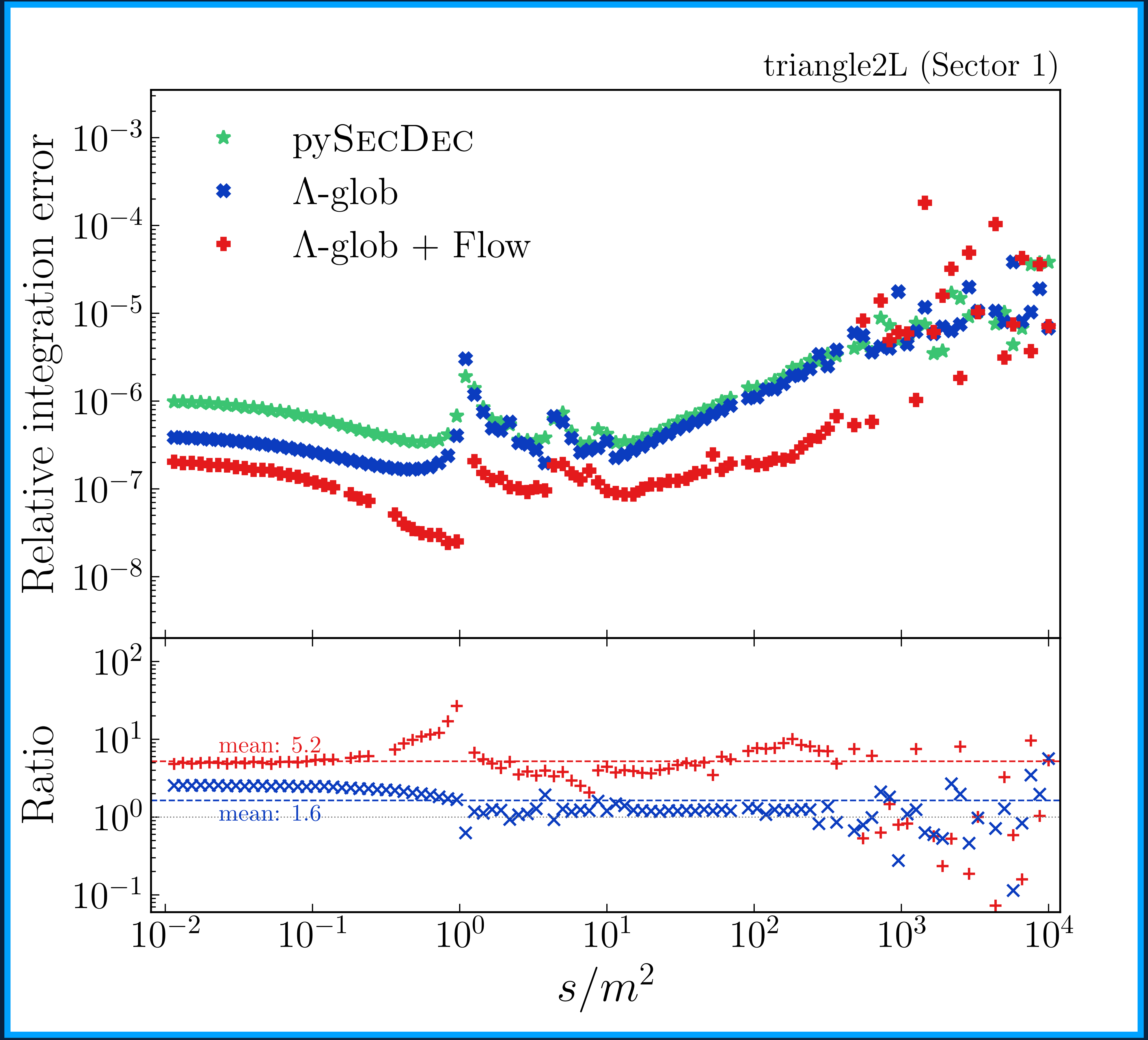


Contains singularities

Cauchy-Theorem → Contour deformation

$$\int_0^1 \prod_{j=1}^{N} \mathrm{d}x_j\, I(\vec{x}) = \int_\gamma \prod_{j=1}^{N} \mathrm{d}z_j\, I(\vec{z})$$

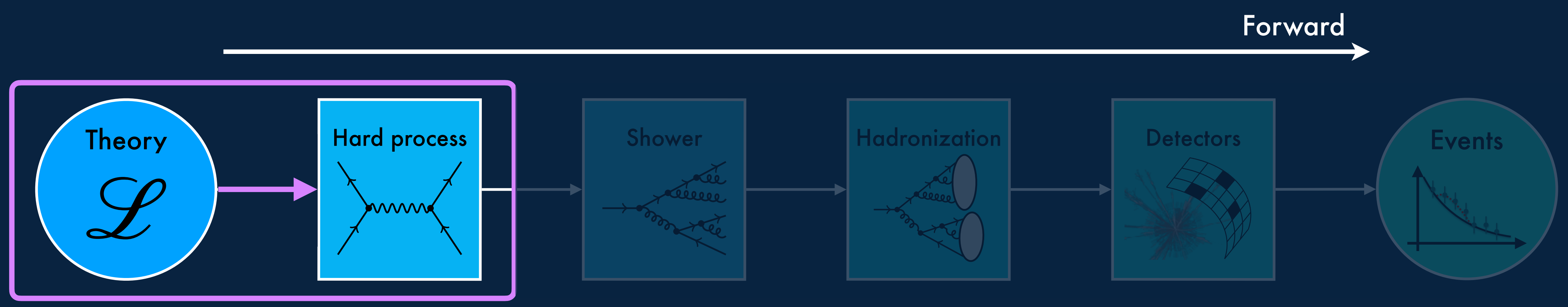




Parametrize with **NF + NN**

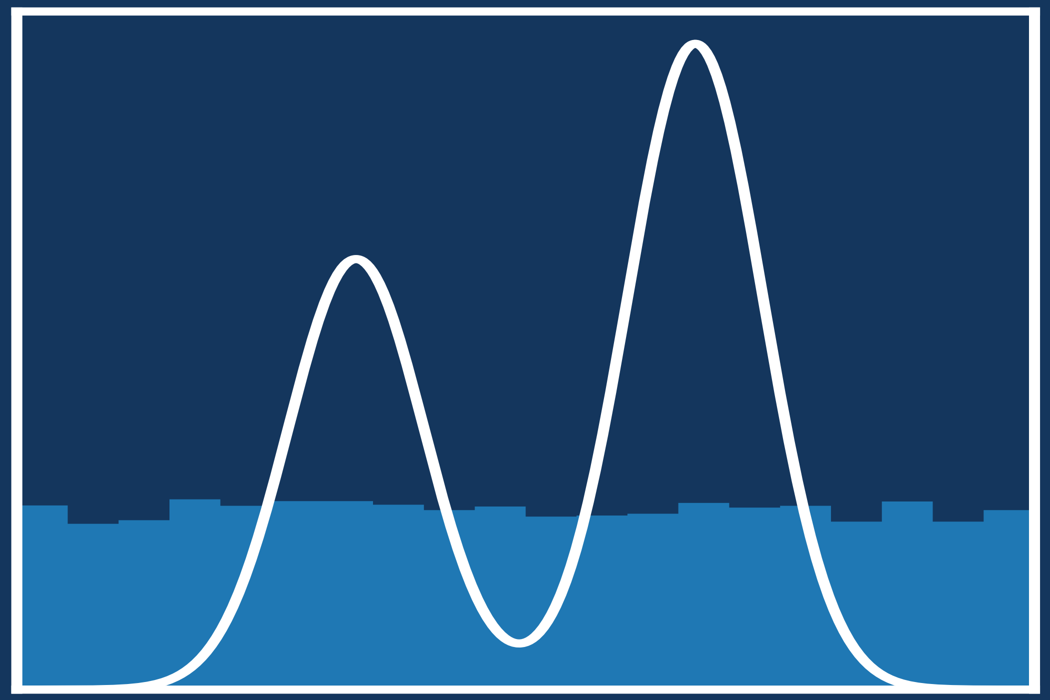
→ Optimal parametrization = small variance

ML for forward simulations



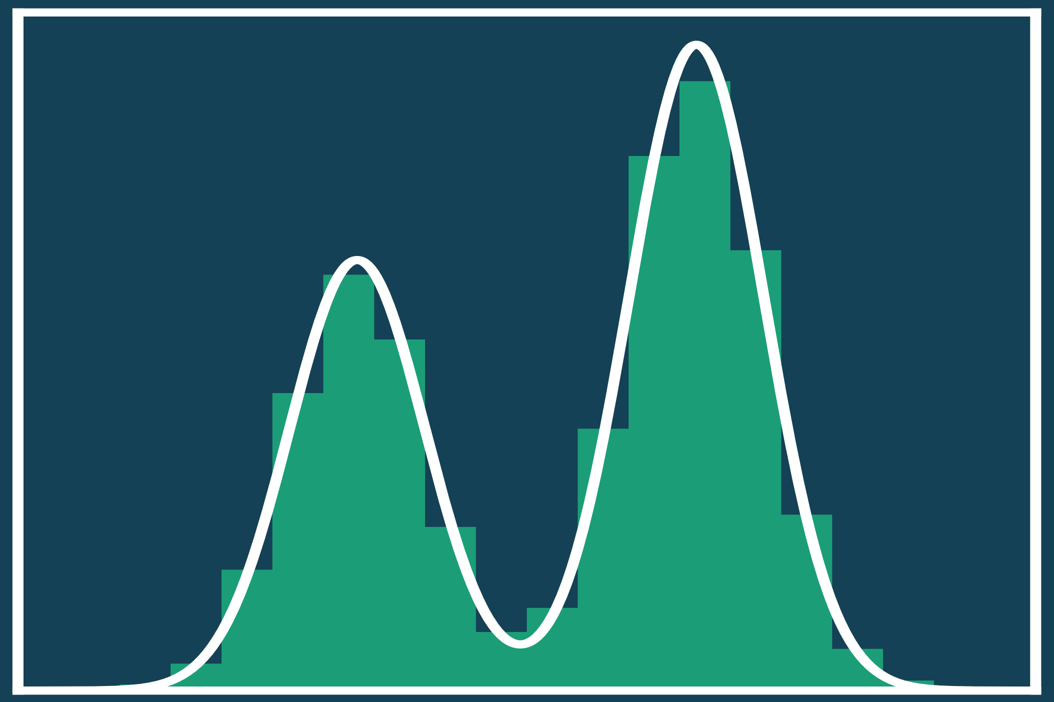
$$d\sigma \sim \text{pdf} \times M(x)^2 \times \boxed{\text{phase space}}$$

Phase space: increase unweighting efficiency



Flat sampling:
inefficient

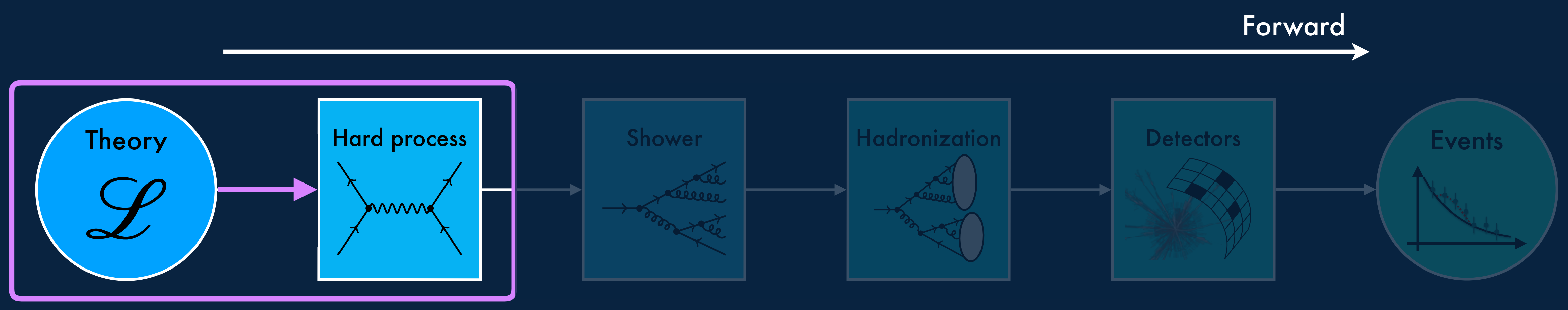
$$I = \langle f(x) \rangle_{x \sim \text{unif}}$$



Importance sampling:
find g close to f

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

ML for forward simulations



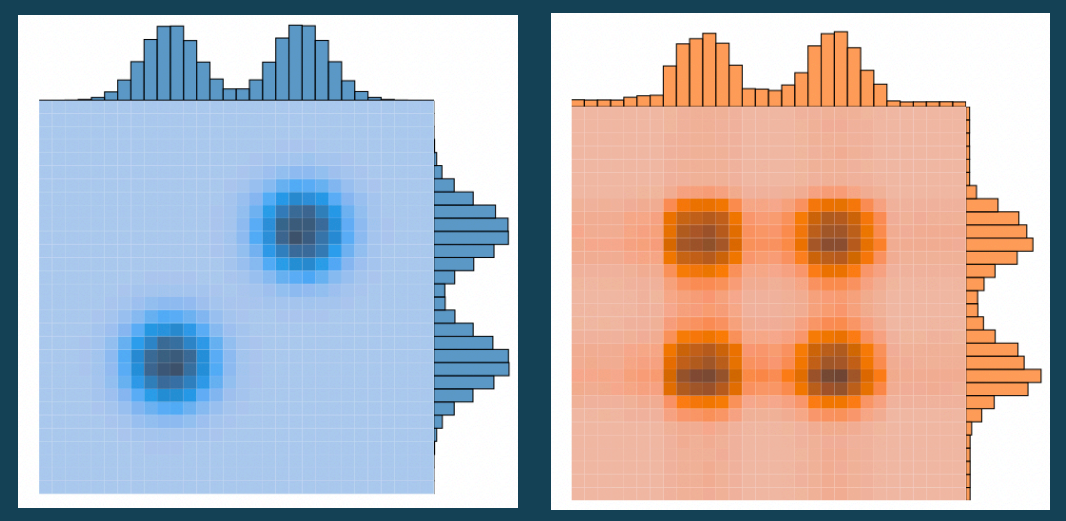
$$d\sigma \sim \text{pdf} \quad \times \quad |M(x)|^2 \quad \times \quad \boxed{\text{phase space}}$$

Phase space: increase unweighting efficiency

- Standard **VEGAS** approach → **fast** but **no correlations**
- Improve with **NN** → **correlations** but **unstable**

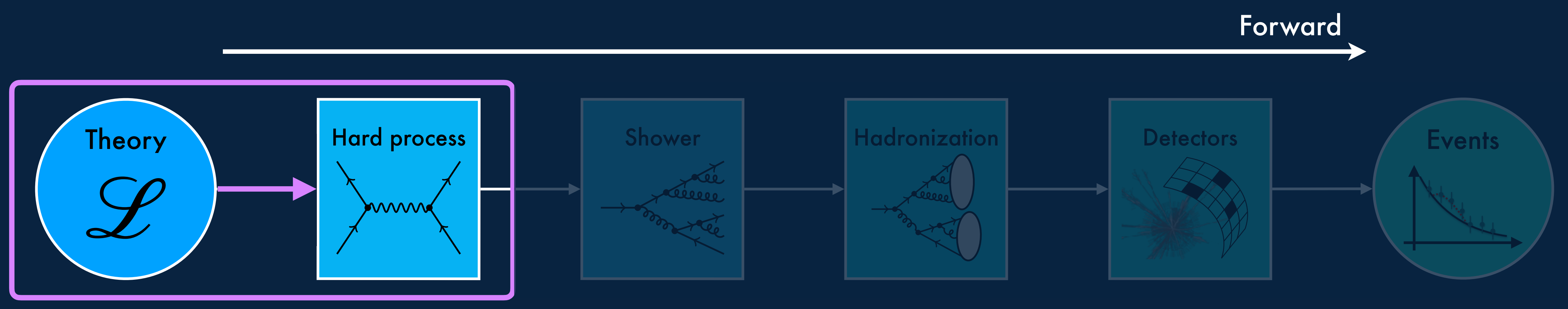
[G. P. Lepage, 1978]

- ⊕ Computationally cheap
- ⊖ High-dim and rich peaking functions → **slow convergence**
- ⊖ Peaks not aligned with grid axes → **phantom peaks**



[1707.00028, 1810.11509, 2001.05478, 2001.05486, 2001.10028, 2005.12719, 2009.07819, 2011.13445, 2112.09145, 2212.06172, 2309.12369,…..]

ML for forward simulations

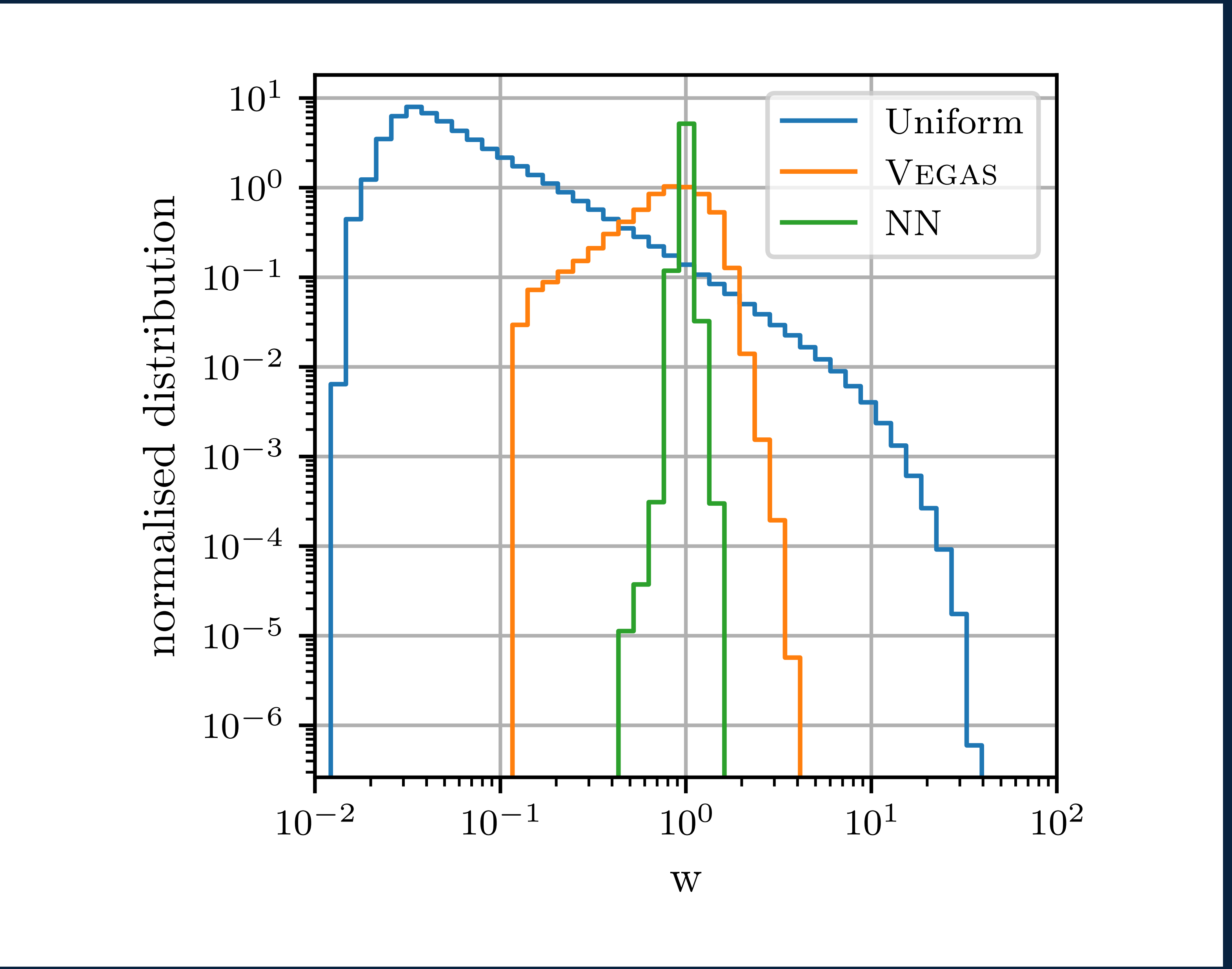


$$d\sigma \sim \text{pdf} \quad \times \quad M(x)^2 \quad \times \quad \boxed{\text{phase space}}$$

Phase space: increase unweighting efficiency

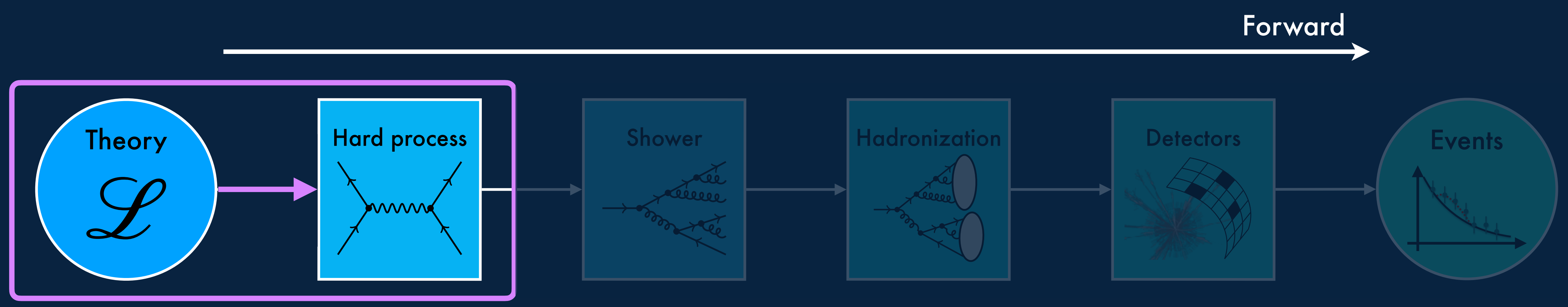
- Standard **VEGAS** approach → **fast** but **no correlations**
- Improve with **NN** → **correlations** but **unstable**
- Use **normalizing flows** → **correlations** and **stable**

Bothmann, Janßen, Knobbe, Schmale, Schumann [2001.05478]



[1707.00028, 1810.11509, 2001.05478, 2001.05486, 2001.10028, 2005.12719, 2009.07819, 2011.13445, 2112.09145, 2212.06172, 2309.12369,.....]

ML for forward simulations



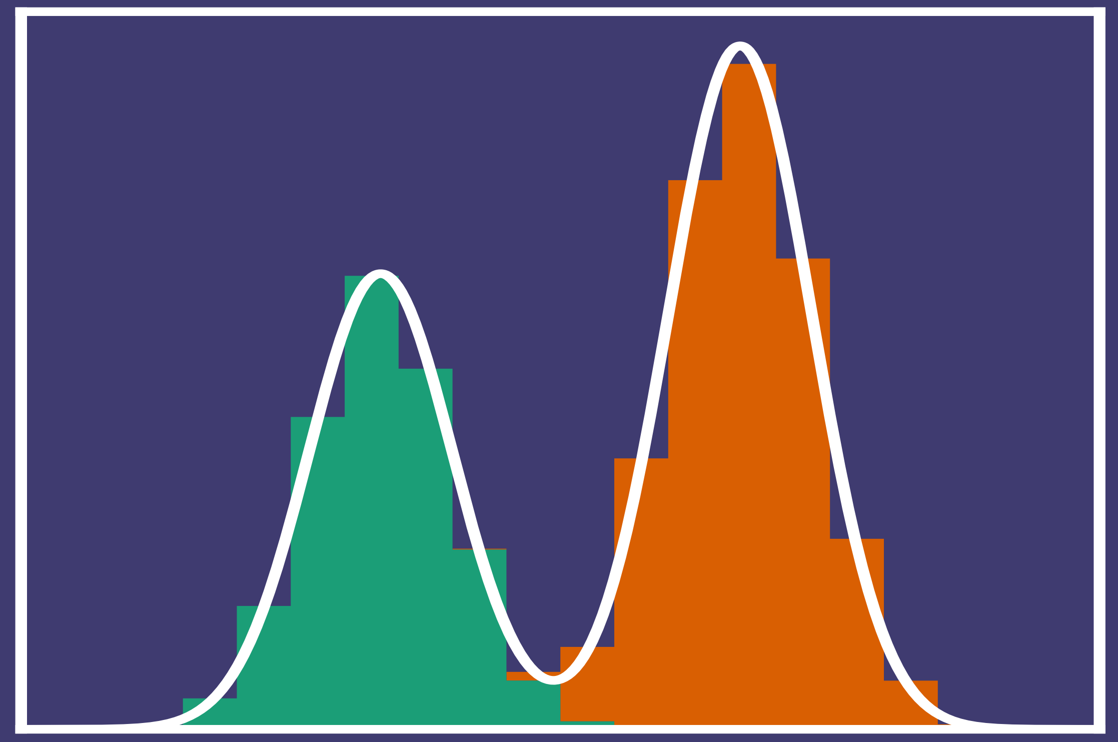
$$d\sigma \sim \text{pdf} \quad \times \quad |M(x)|^2 \quad \times \quad \boxed{\text{phase space}}$$

Phase space: increase unweighting efficiency

- Standard VEGAS approach → **fast** but **no correlations**
- Improve with **NN** → **correlations** but **unstable**
- Use **normalizing flows** → **correlations** and **stable**
- **Multi-channel** approach → split the integral
- Combine all (VEGAS, learned α_i, NF, symmetries,..) → **MadNIS** framework

[1707.00028, 1810.11509, 2001.05478, 2001.05486, 2001.10028, 2005.12719, 2009.07819, 2011.13445, 2112.09145, 2212.06172, 2309.12369,.....]

Kleiss, Pittau [hep-ph/9405257], Maltoni, Stelzer [hep-ph/0208156]



Multi-channel:
one map for each channel

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

MadNIS

Neural Importance Sampling

Heimel, Huetsch, Maltoni, Mattelaer, Plehn, RW [2311.XxxxX]
Heimel, RW, Butter, Isaacson, Krause, Maltoni, Mattelaer, Plehn [2212.06172]

MadNIS — Basic functionality

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

Use physics knowledge to construct channel and mappings

Normalizing flow to refine channel mappings

Fully connected network to refine channel weights

Update simultanously with variance as loss function

MadNIS — Loss function

Minimize total variance

$$\sigma_{\text{tot}}^2 = N \sum_i \frac{\sigma_i^2}{N_i} \qquad \text{with}$$

$$\sigma_i^2 = \left\langle \alpha_i^2(x) \frac{f^2(x)}{g_i^2(x)} \right\rangle_{x\sim g_i(x)} - \left\langle \alpha_i(x) \frac{f(x)}{g_i(x)} \right\rangle_{x\sim g_i(x)}$$

Total variance depends on N_i

$\downarrow$

affects optimal $\alpha_i(x)$

$\downarrow$

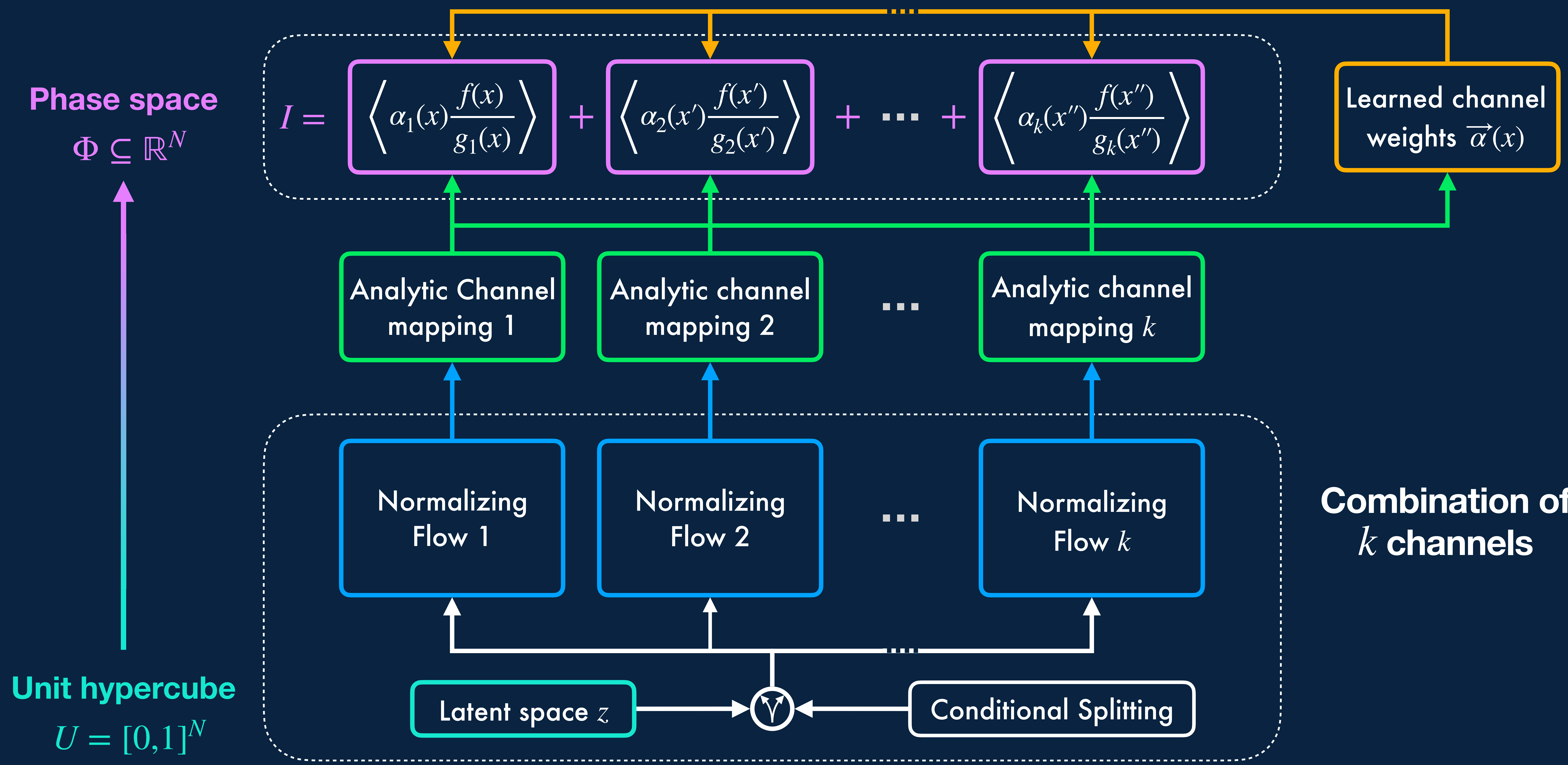
use stratified sampling

$$N_i = N \frac{\sigma_i}{\sum_k \sigma_k}$$

MadNIS loss function

$$\mathscr{L} = \sigma_{\text{tot}}^2 = \sum_{i,k} \sigma_i \, \sigma_k = \left[\sum_i \sigma_i \right]^2$$

MadNIS — Basic functionality



MadNIS — Overview

MadNIS

Basic functionality

- Neural Channel Weights
- Normalizing Flow
- MadGraph matrix elements
- MadEvent channel mappings

Improved multi-channeling

- Stratified sampling/training
- Symmetries between channels
- Channel Dropping
- Partial weight buffering

Improved training

- VEGAS Initialization
- Buffered Training
- Surrogate Integrand

MadNIS — Buffered training

Basic functionality

- Neural Channel Weights
- Normalizing Flow
- MadGraph matrix elements
- MadEvent channel mappings

MadNIS

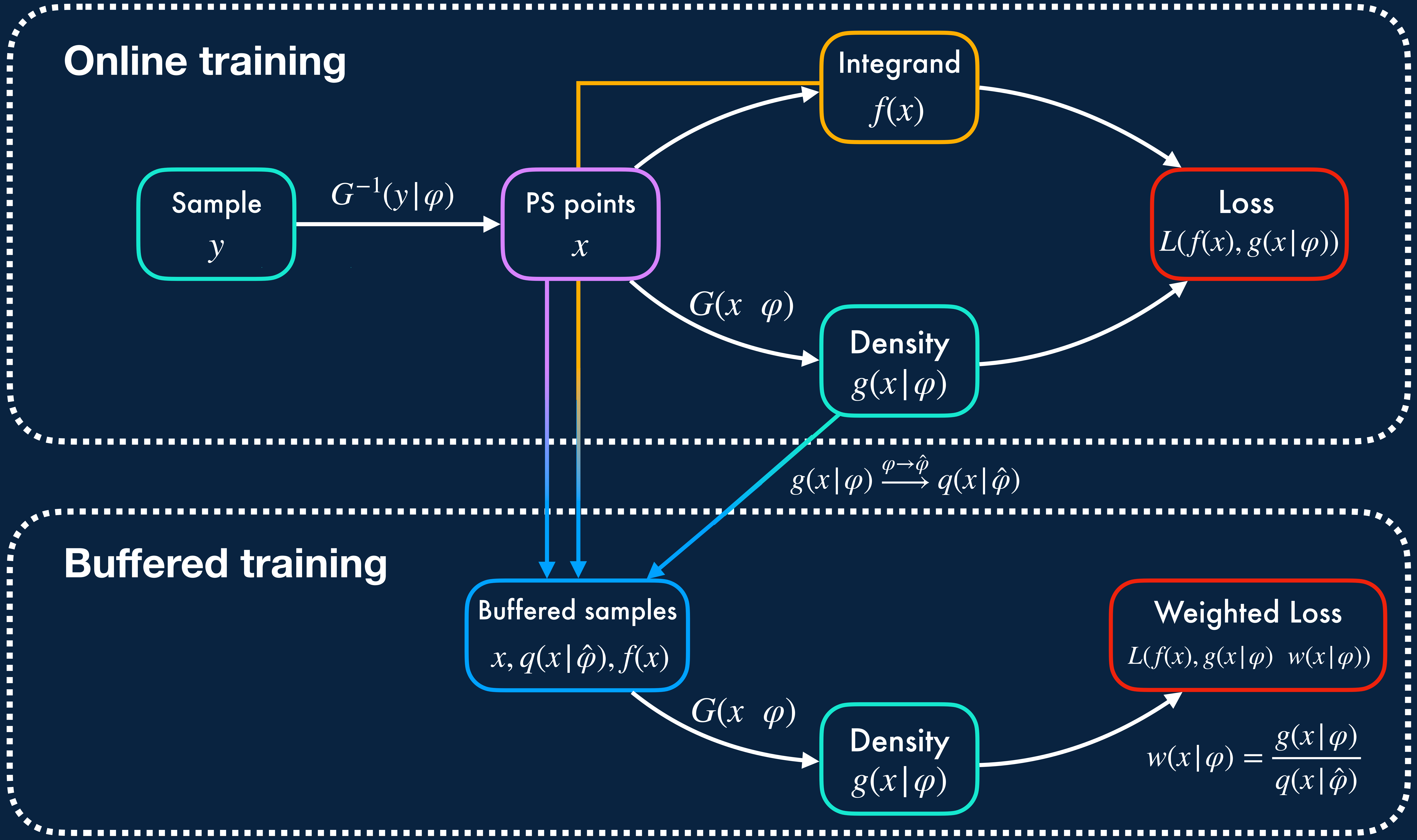
Improved multi-channeling

- Stratified sampling/ training
- Symmetries between channels
- Channel Dropping
- Partial weight buffering

Improved training

- VEGAS Initialization
- **Buffered Training**
- Surrogate Integrand

MadNIS — Buffered training



MadNIS — Buffered training

Training algorithm

generate new samples, train on them, save samples

↓

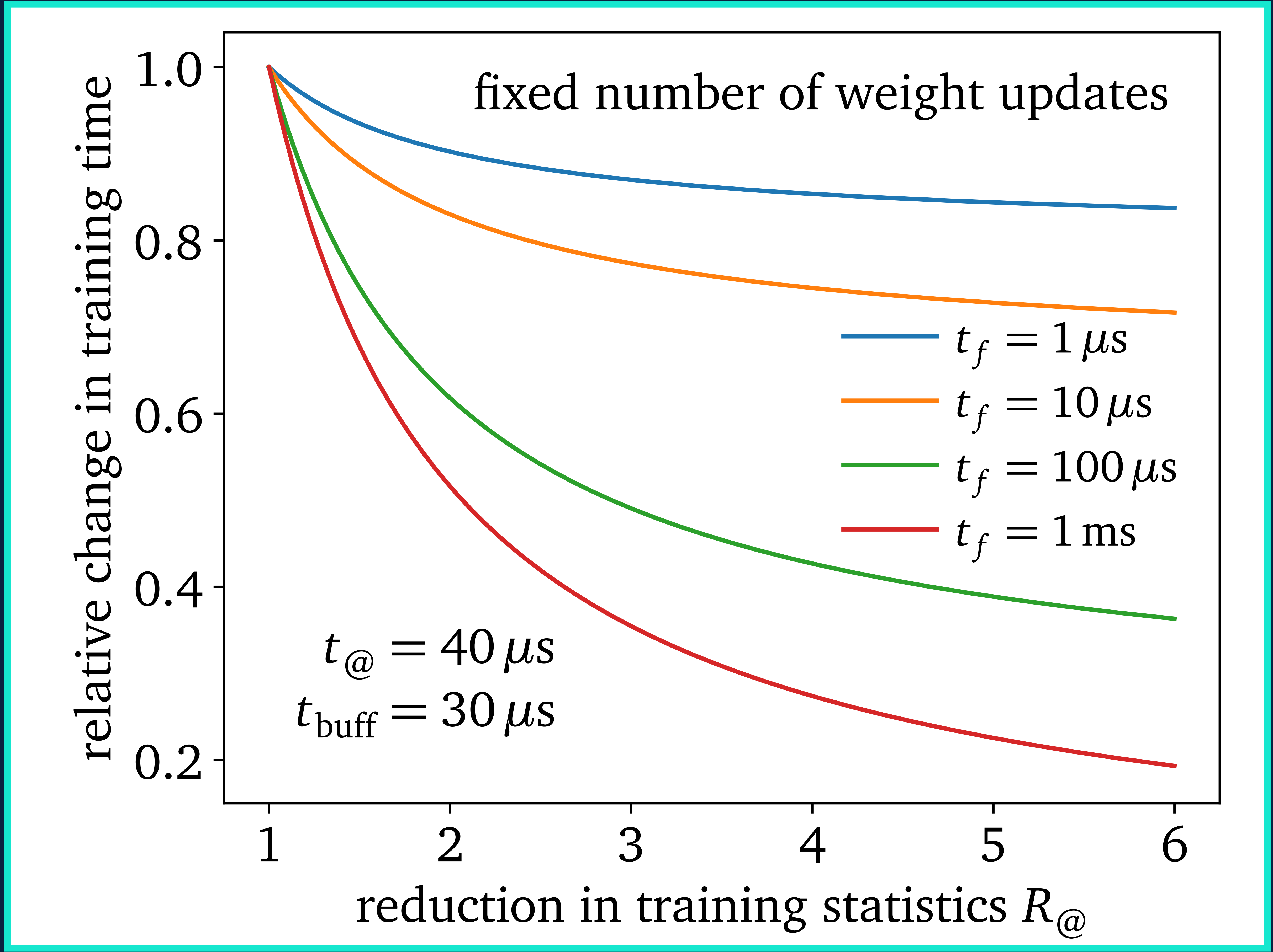
train on saved samples n times

↓

repeat

Reduction in training statistics by

$$R_@ = n + 1$$



MadNIS — VEGAS initialization

Basic functionality

- Neural Channel Weights
- Normalizing Flow
- MadGraph matrix elements
- MadEvent channel mappings

MadNIS

Improved multi-channeling

- Stratified sampling/ training
- Symmetries between channels
- Channel Dropping
- Partial weight buffering

Improved training

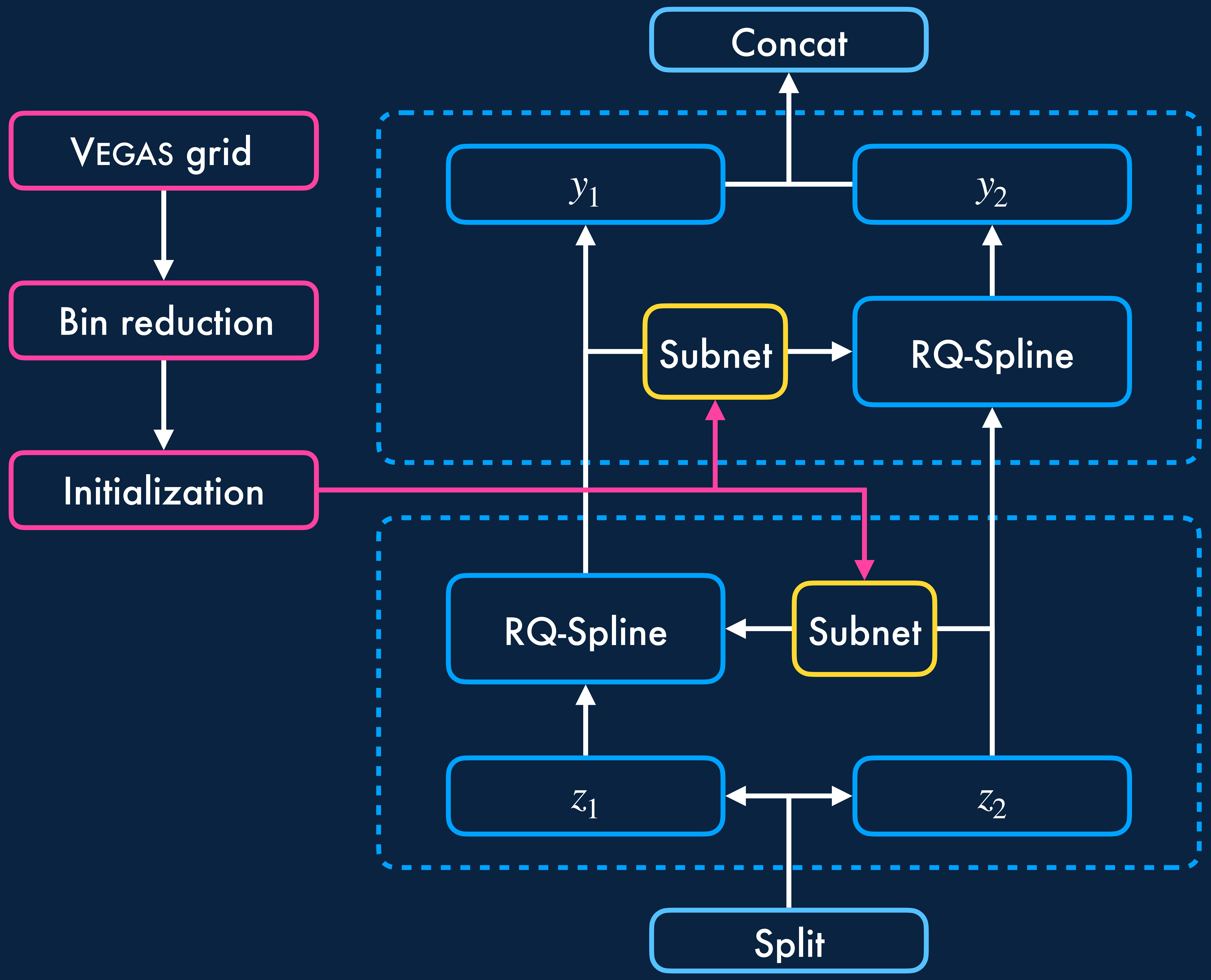
- **VEGAS Initialization**
- Buffered Training
- Surrogate Integrand

MadNIS — VEGAS initialization

	VEGAS	Flow
Training	Fast	Slow
Correlations	No	Yes

Combine advantages:

Pre-trained VEGAS grid as starting point for flow training



MadNIS — Improved multi-channeling

MadNIS

Basic functionality

- Neural Channel Weights
- Normalizing Flow
- MadGraph matrix elements
- MadEvent channel mappings

Improved multi-channeling

- Stratified sampling/ training
- Symmetries between channels
- Channel Dropping
- Partial weight buffering

Improved training

- VEGAS Initialization
- Buffered Training
- Surrogate Integrand

MadNIS — Improved multi-channeling

Use symmetries

Groups of channels only **differ by permutations** of final state momenta

↓

use **common** flow

combine in loss function

Stratified training

Channels have different **contributions** to the total variance

↓

more samples for channels with **higher variance** during training

Channel dropping

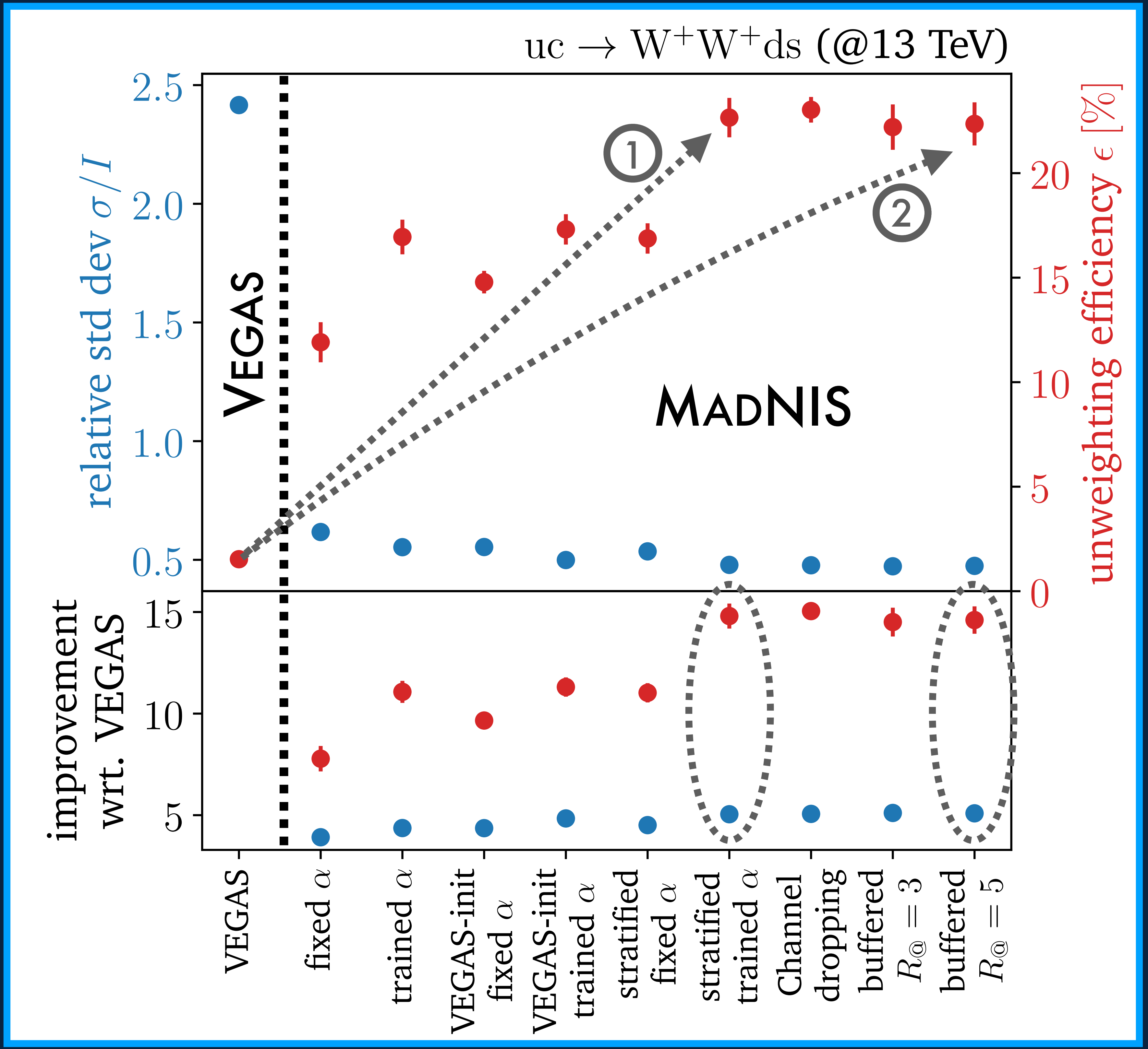
MadNIS often **reduces contribution** of some channels to total integral

↓

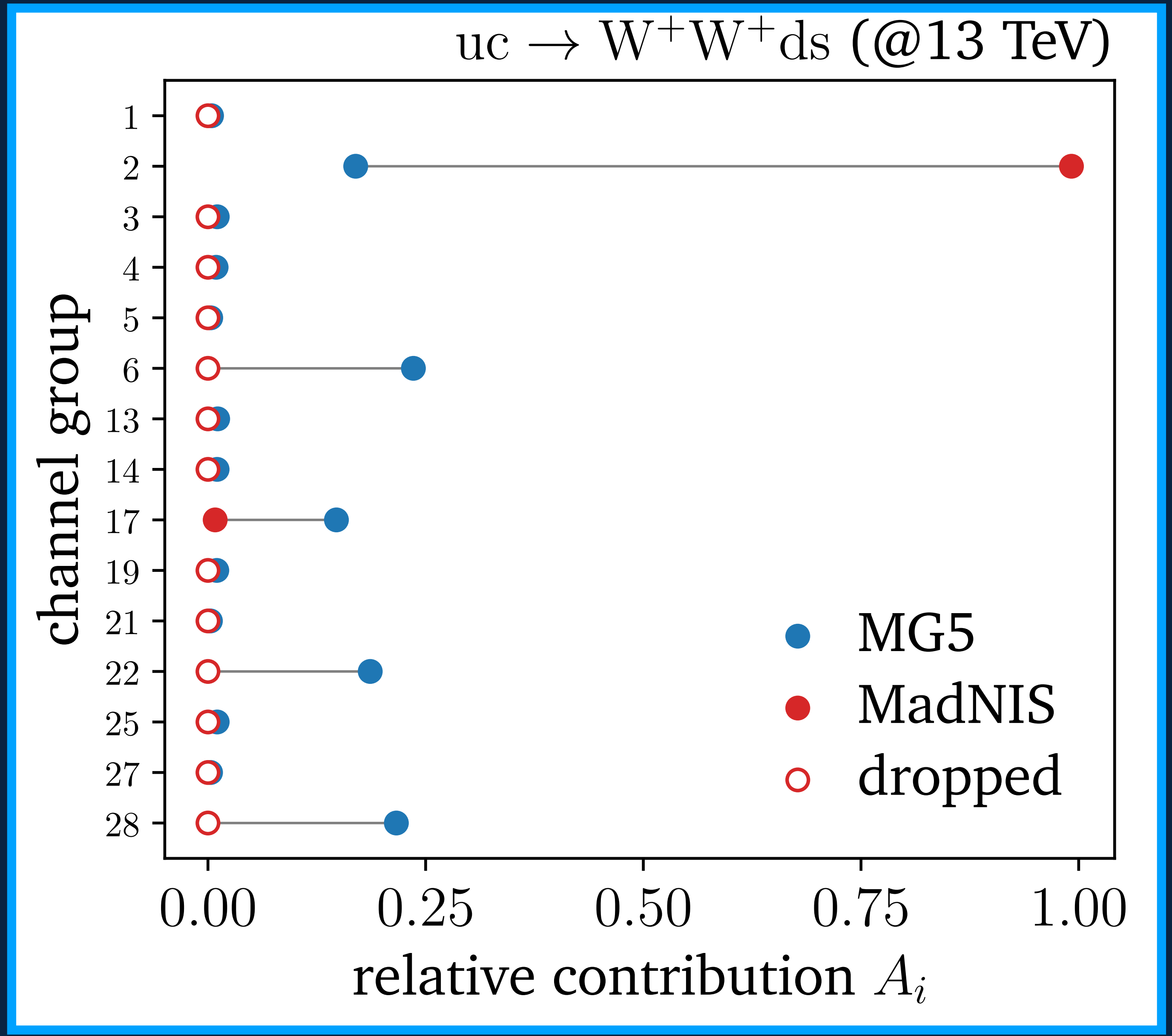
remove these channels from the **training** completely

Reduced complexity
Improved stability

MadNIS — Results



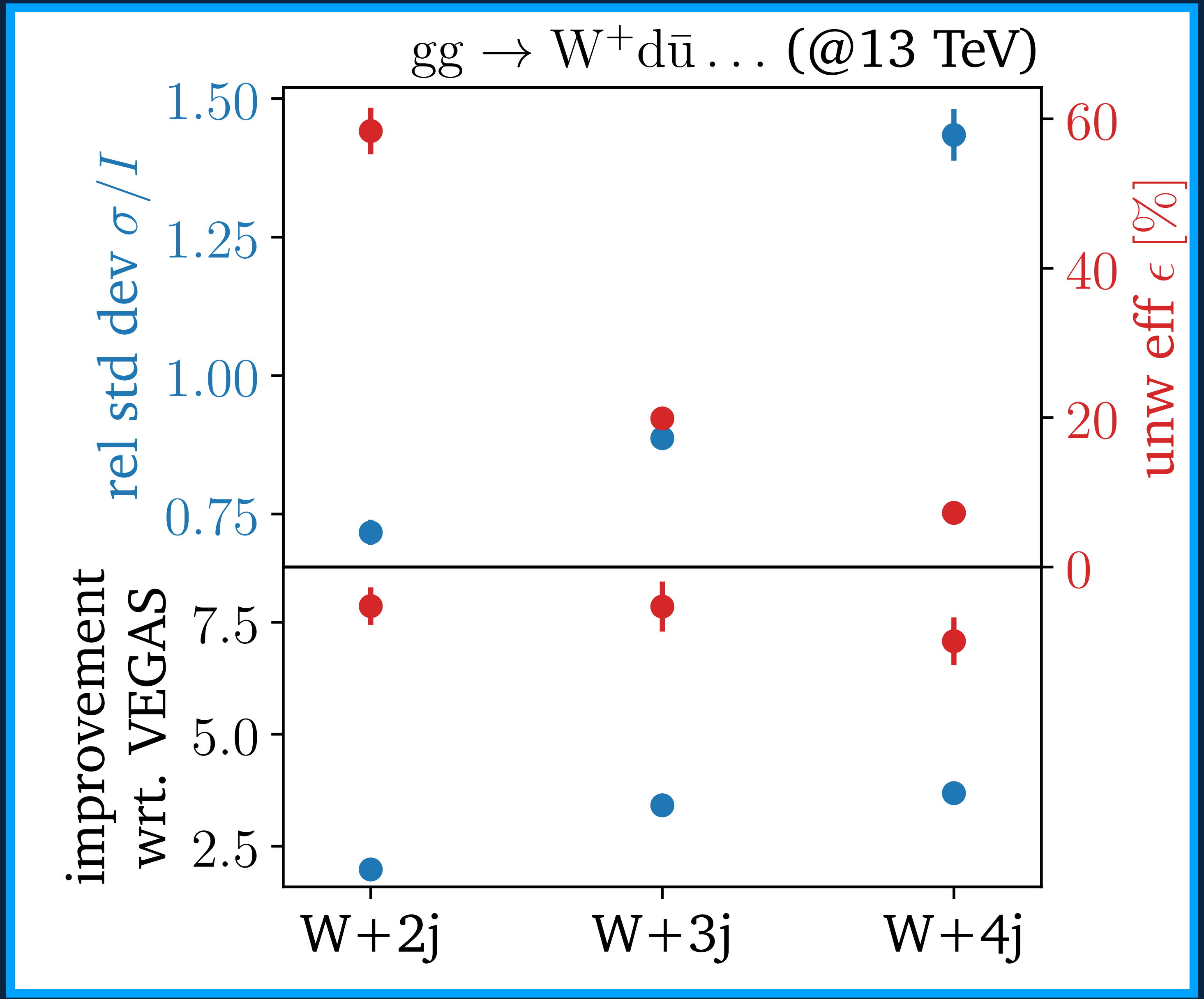
1. excellent results with all improvements
2. same performance with buffered training



many channels **are zero**

↓

droppig channels



$gg \to W^+ d\bar{u} g g$

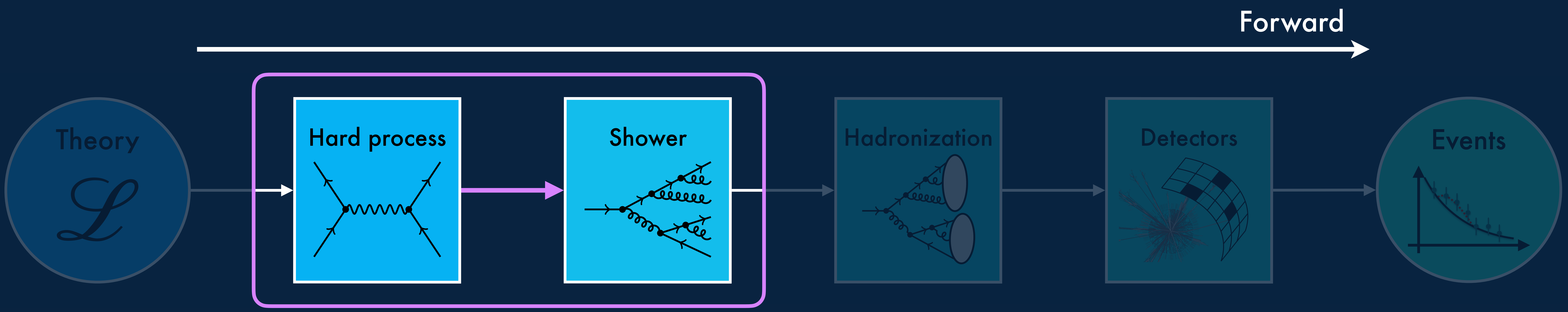
384 channels, 108 symm.

$\eta = 7.5\,\% = 5.9\,\eta_{\text{VEGAS}}$

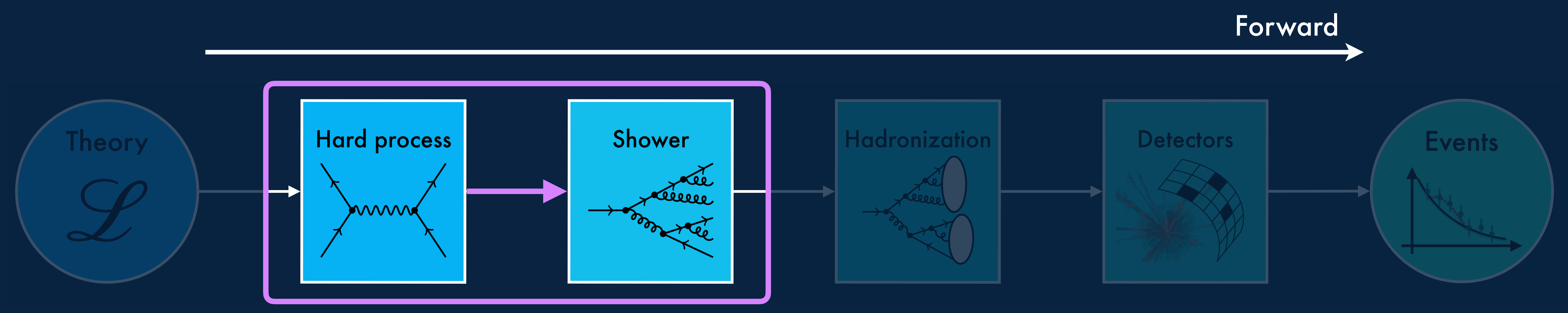
↓

scales well with high multiplicity and many channels

ML for forward simulations



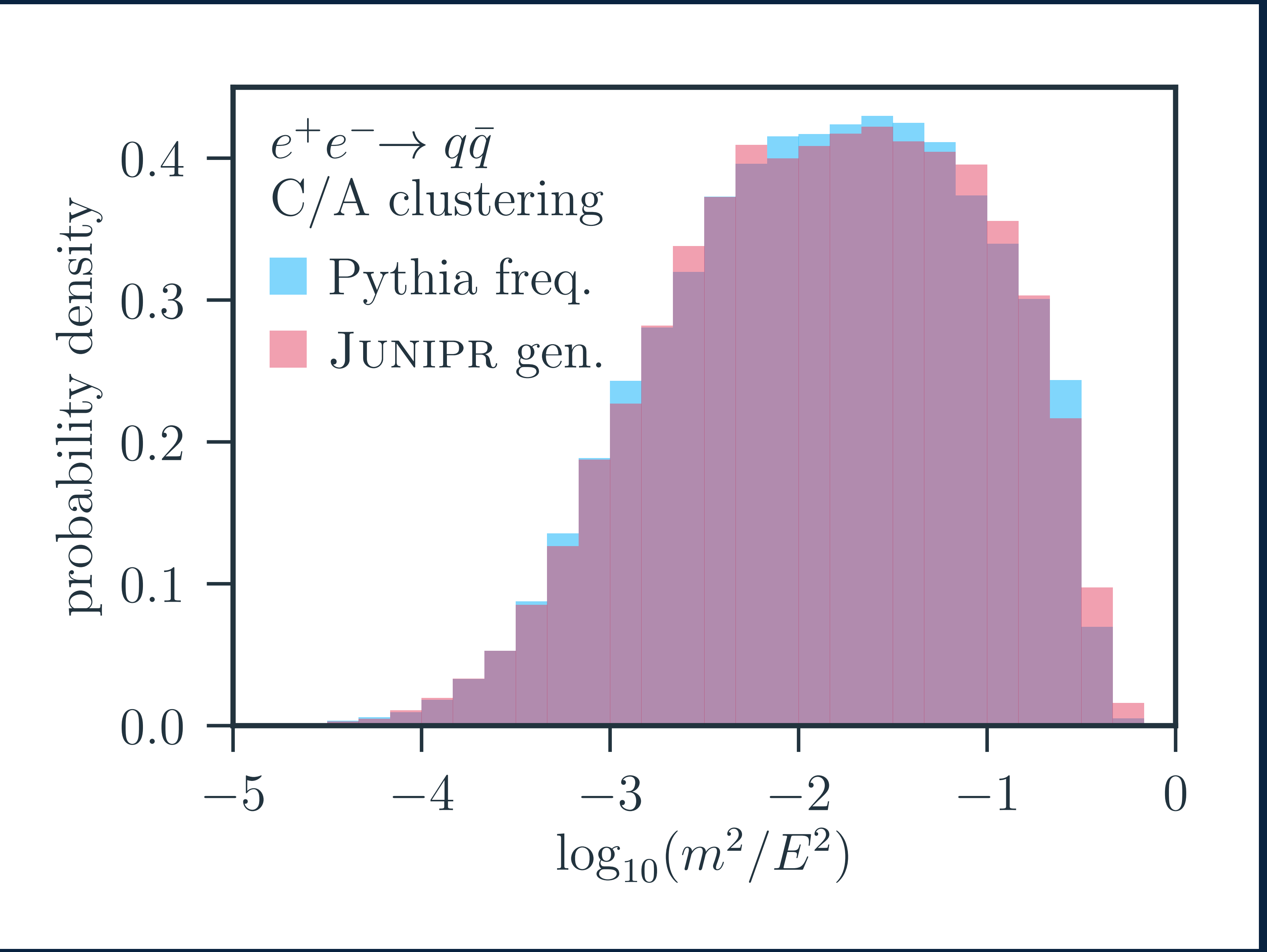
ML for forward simulations



Parton shower: improve over semi-classical approach

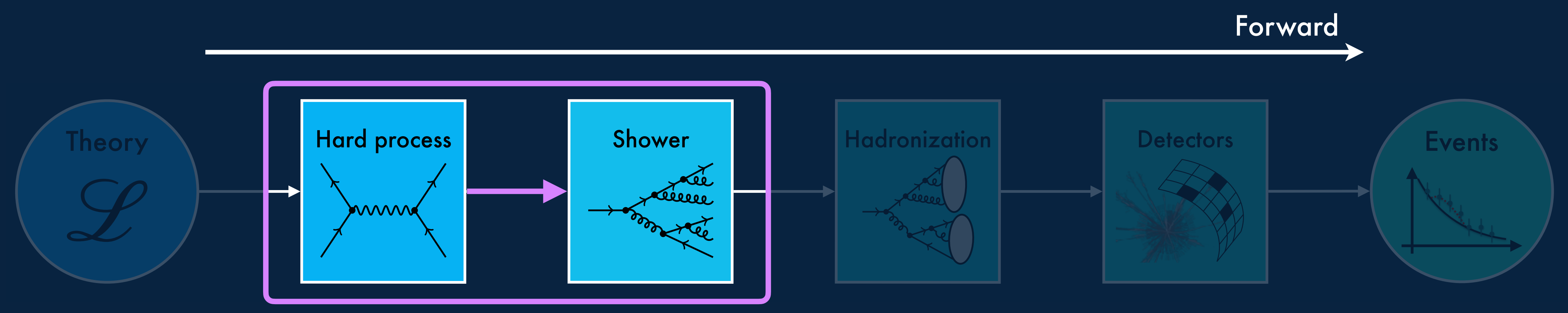
Andreassen, Feige, Frye, Schwartz [1804.09720]

- Splittings are iterative
  → can be learned with RNN (**JUNIPR**)



[1701.05927, 1703.06114, 1804.09720, 1807.03685, 1808.07802, 1810.05165, 1906.10137, 2009.04842, 2012.06582, 2012.09873, 2106.11535, 2109.15197, 2111.12849, 2211.06406, 2211.10295, 2212.08751, 2301.08128, 2303.05376, 2304.01266, 2307.06836, 2310.00049,….]

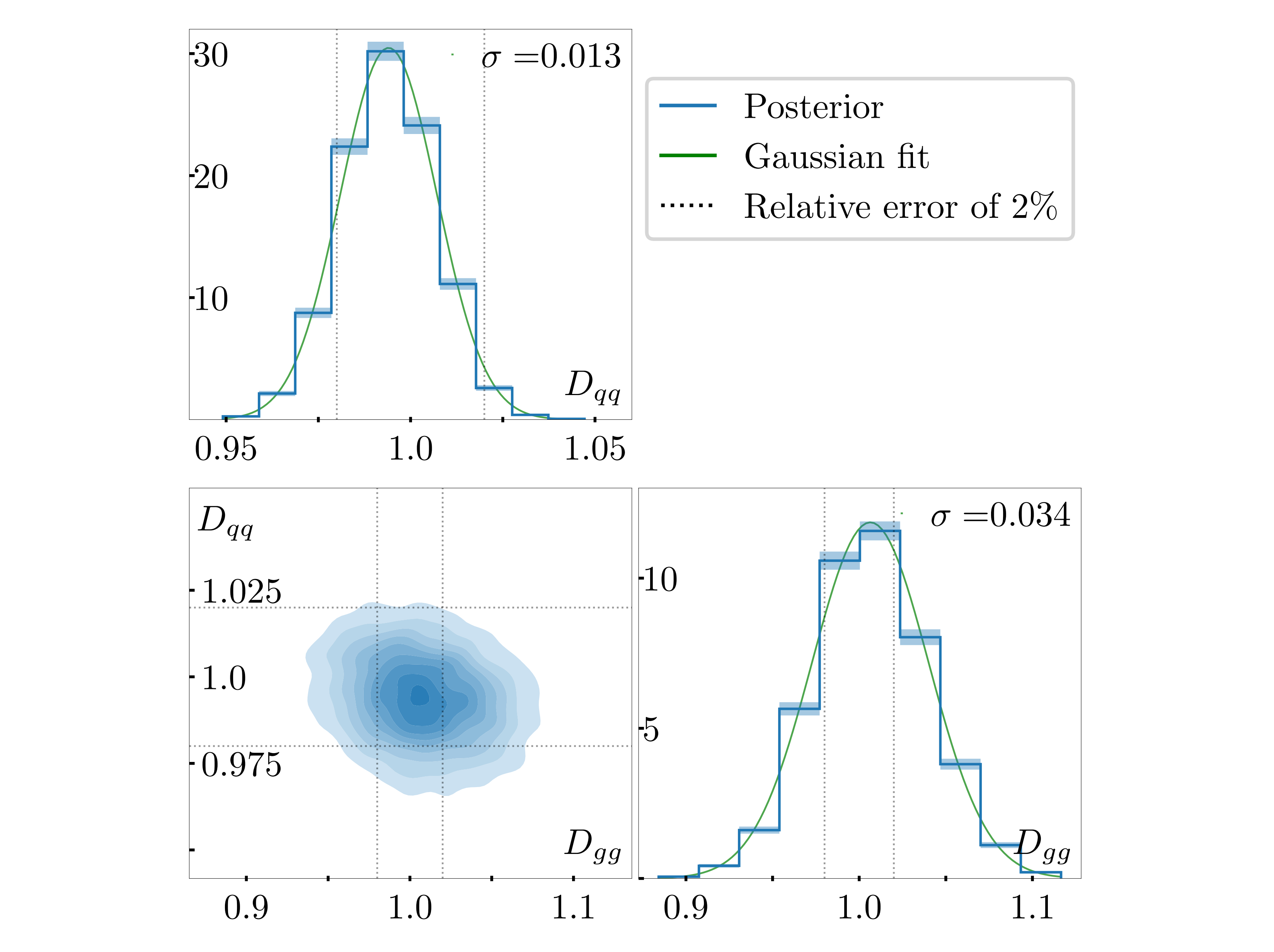
ML for forward simulations



Parton shower: improve over semi-classical approach

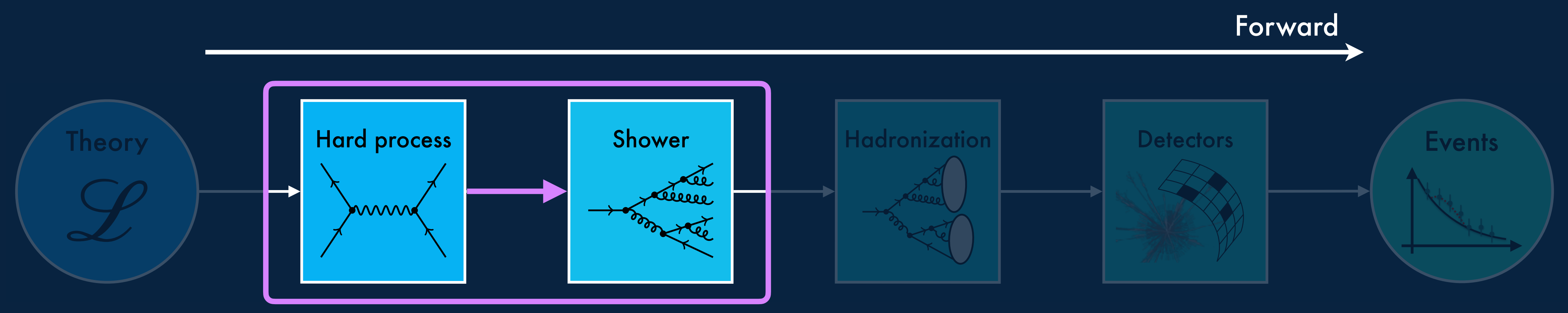
Bieringer, Butter, Heimel, Höche, Radev, Köthe, Plehn [2012.09873]

- Splittings are iterative
  → can be learned with RNN (**JUNIPR**)
- Using **ML-based inference** to improve splitting kernels



[1701.05927, 1703.06114, 1804.09720, 1807.03685, 1808.07802, 1810.05165, 1906.10137, 2009.04842, 2012.06582, 2012.09873, 2106.11535, 2109.15197, 2111.12849, 2211.06406, 2211.10295, 2212.08751, 2301.08128, 2303.05376, 2304.01266, 2307.06836, 2310.00049,….]

ML for forward simulations

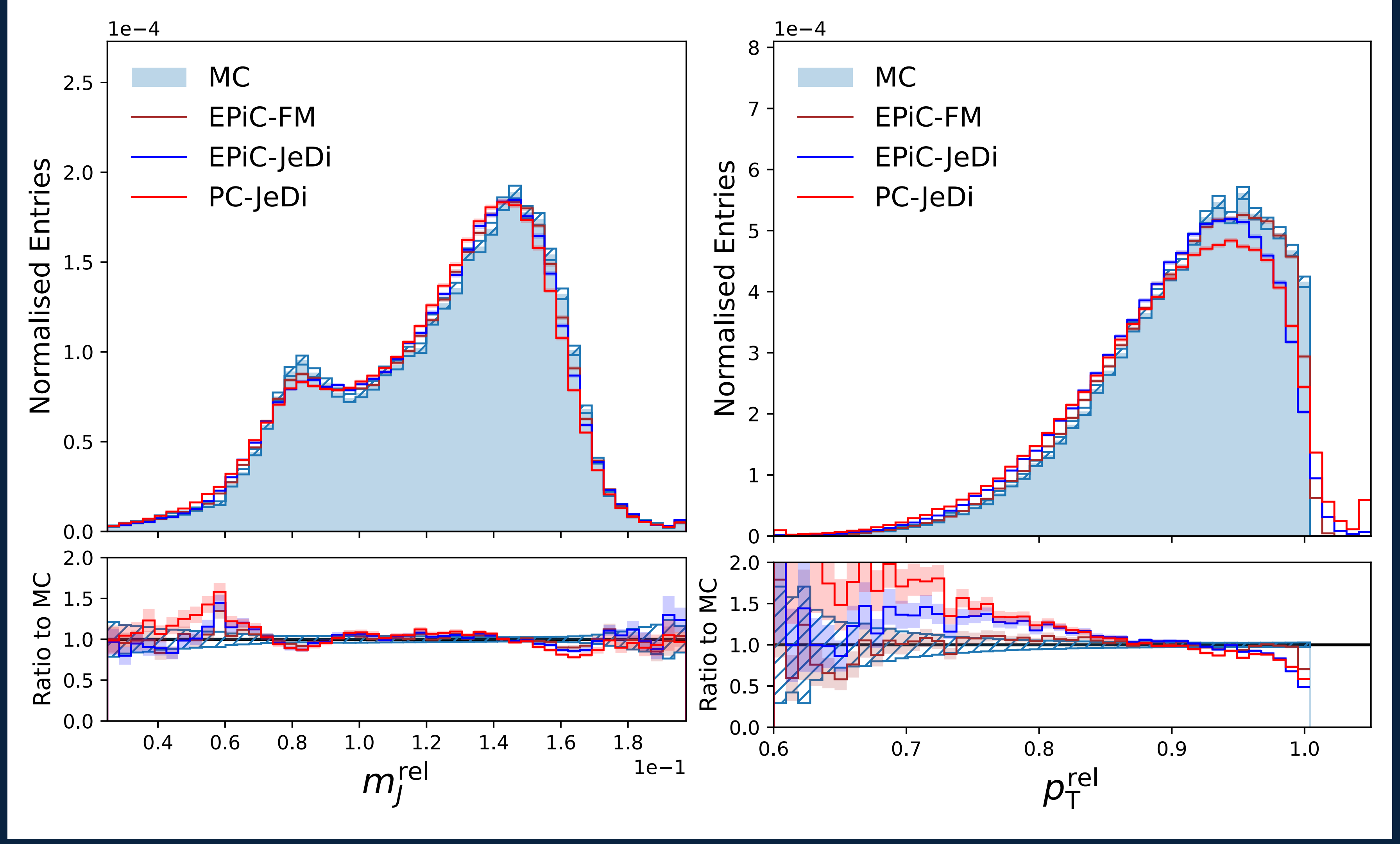


Parton shower: improve over semi-classical approach

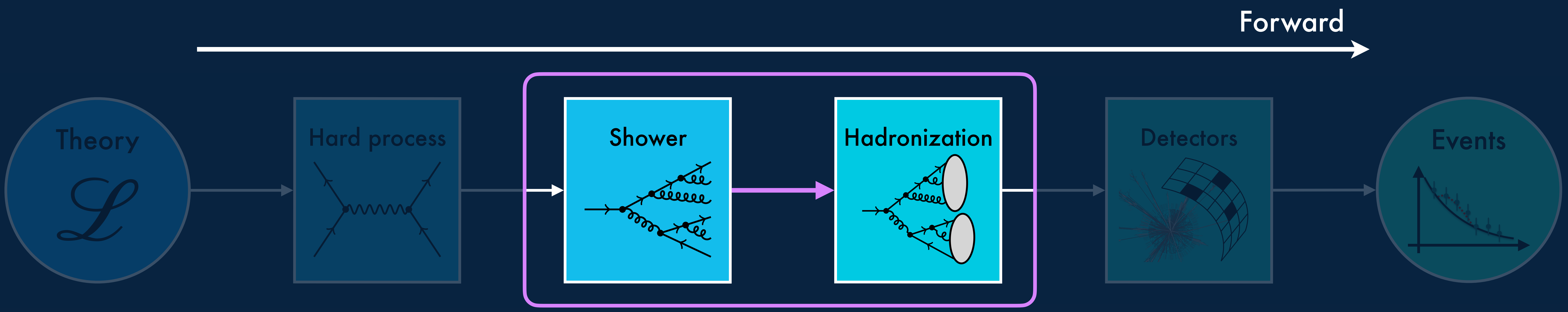
- Splittings are iterative
  → can be learned with RNN (**JUNIPR**)
- Using **ML-based inference** to improve splitting kernels
- End-to-end generation with **particle clouds**
  → SOTA based on **diffusion models**

[1701.05927, 1703.06114, 1804.09720, 1807.03685, 1808.07802, 1810.05165, 1906.10137, 2009.04842, 2012.06582, 2012.09873, 2106.11535, 2109.15197, 2111.12849, 2211.06406, 2211.10295, 2212.08751, 2301.08128, 2303.05376, 2304.01266, 2307.06836, 2310.00049,….]

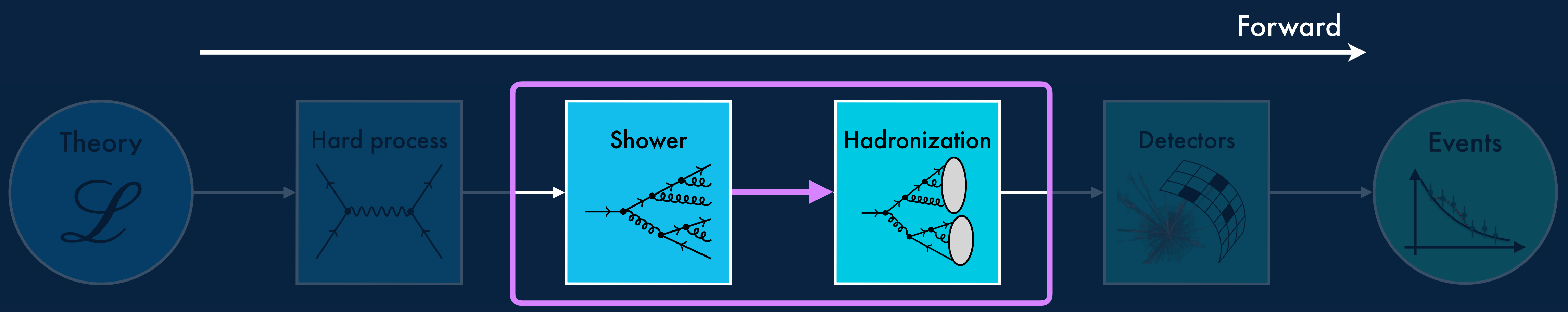
Buhmann et al. [2310.00049]



ML for forward simulations



ML for forward simulations



Fragmentation: remove modelling bias

- Same technique as for PDFs

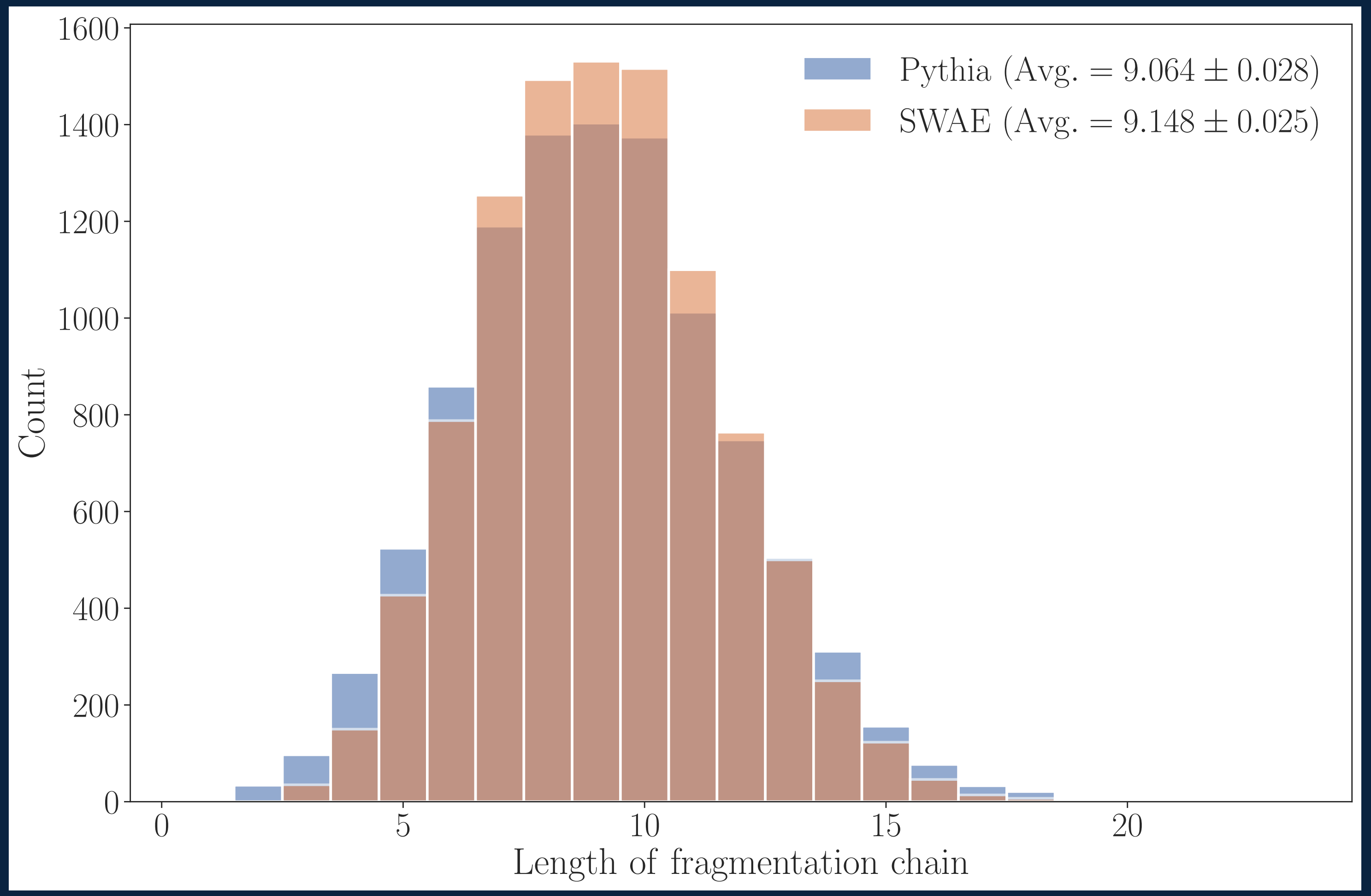
[1706.07049, 1807.03310, 2105.08725, 2202.10779…..]

Hadronization: better model non-perturbative effects

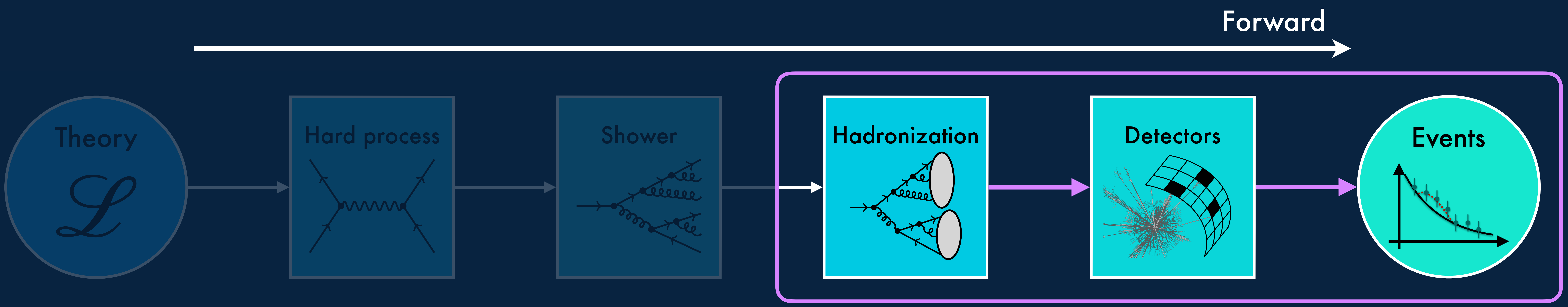
- Improve existing **clustering or Lund string** model
- **Generative ML** for more generic approach

[2203.04983, 2203.12660, 2305.17169,…..]

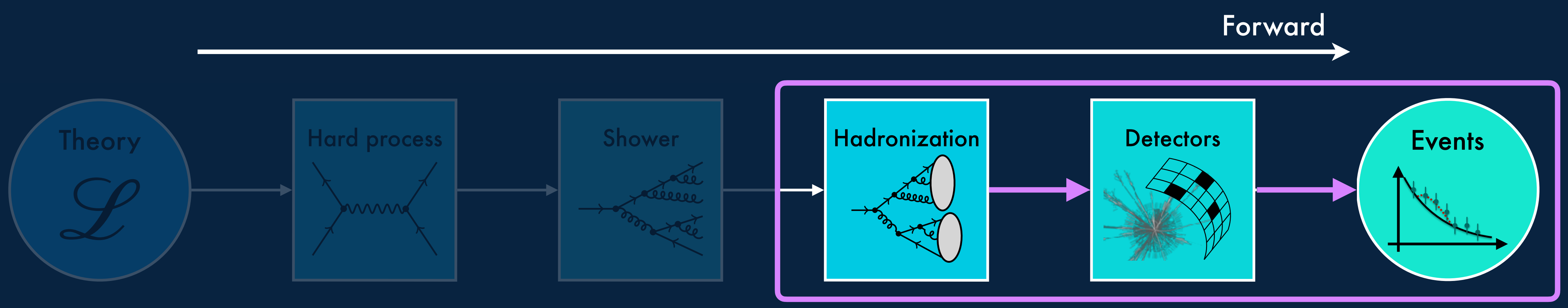
Ilten, Menzo, Youssef, Zupan [2203.04983]



ML for forward simulations



ML for forward simulations



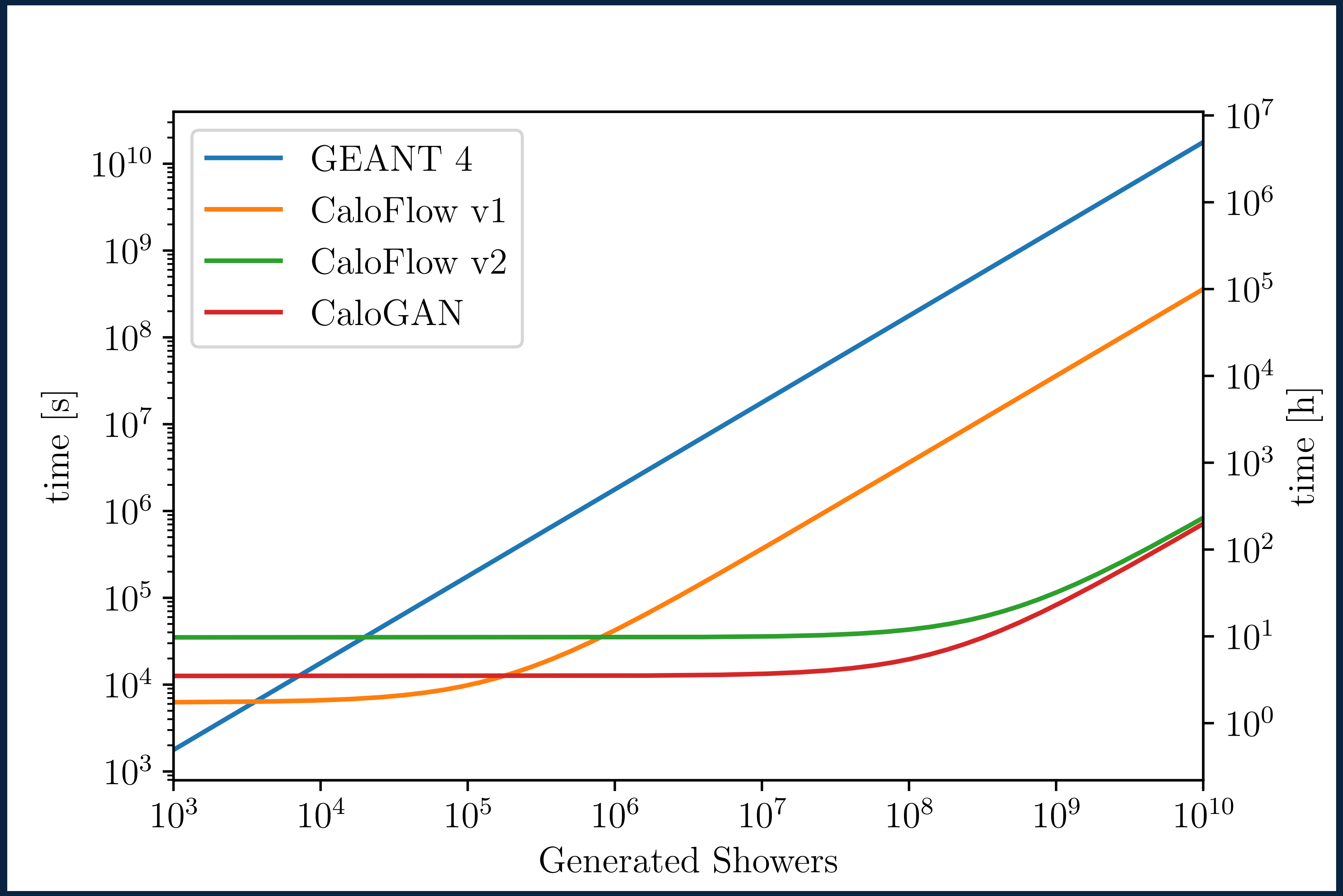
Detector simulation: speed-up GEANT4

- Up to **~10^4 faster**
- More ideas developed in **CaloChallenge 2022**

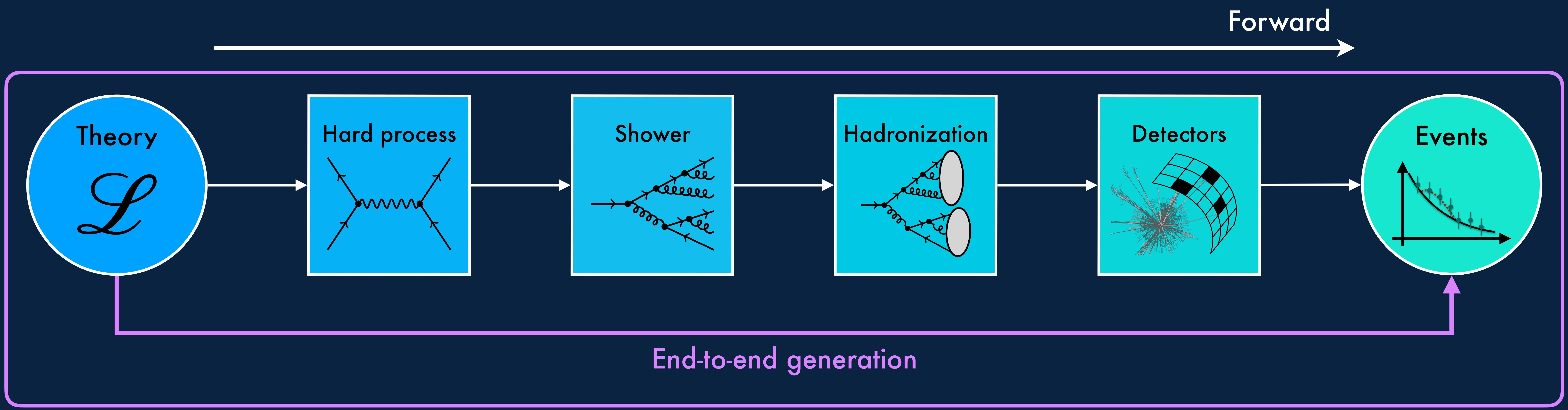
https://calochallenge.github.io/homepage/

[1705.02355, 1711.08813, 1712.10321, 1802.03325, 1807.01954, 1912.06794, 2102.12491, 2106.05285, 2109.02551, 2110.11377, 2206.11898, 2211.15380, 2302.11594, 2305.04847, 2305.11934, 2305.15254, 2307.04780, 2308.03876, 2308.11700, 2309.06515, …]

Krause, Shih [2110.11377]

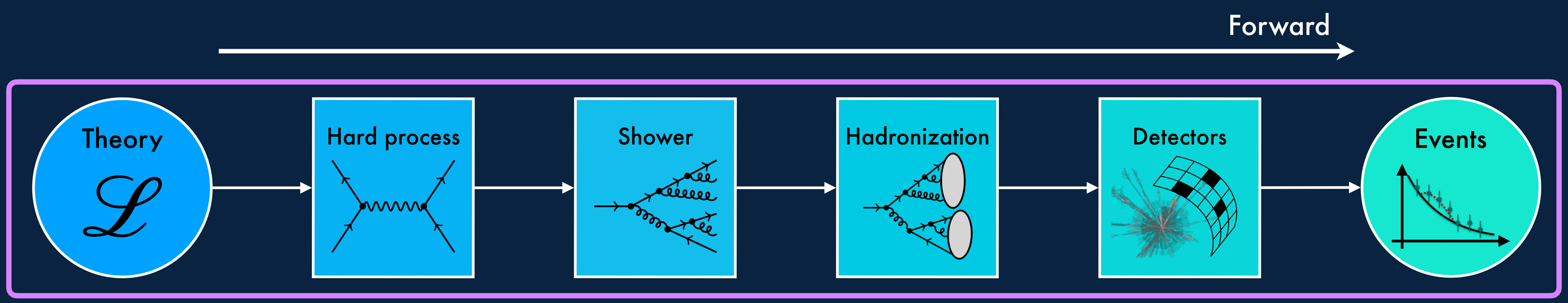


ML for forward simulations



more in talk by Lukas Heinrich

ML for forward simulations

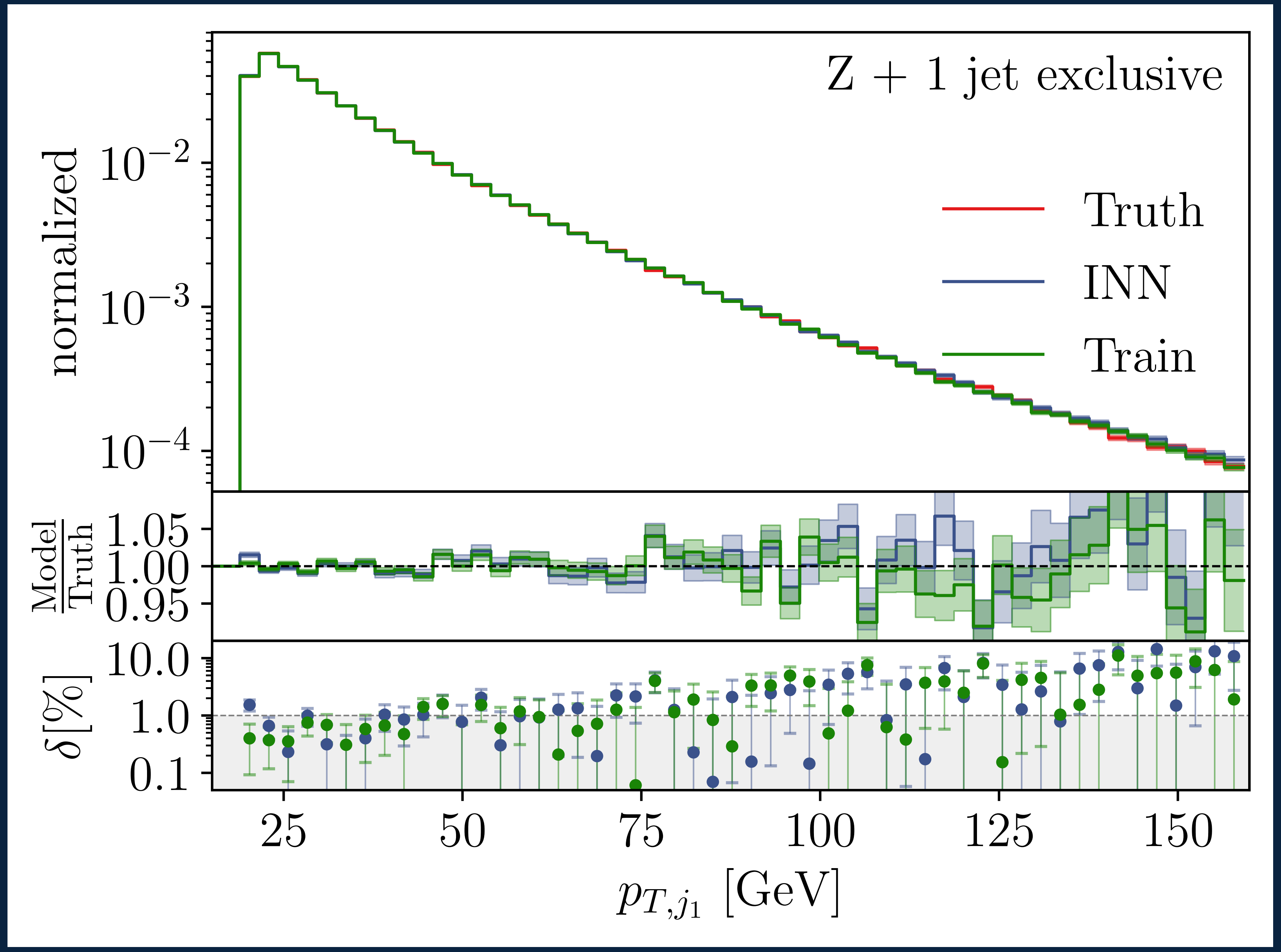


End-to-end generators learn multiple steps at once

Precision generation

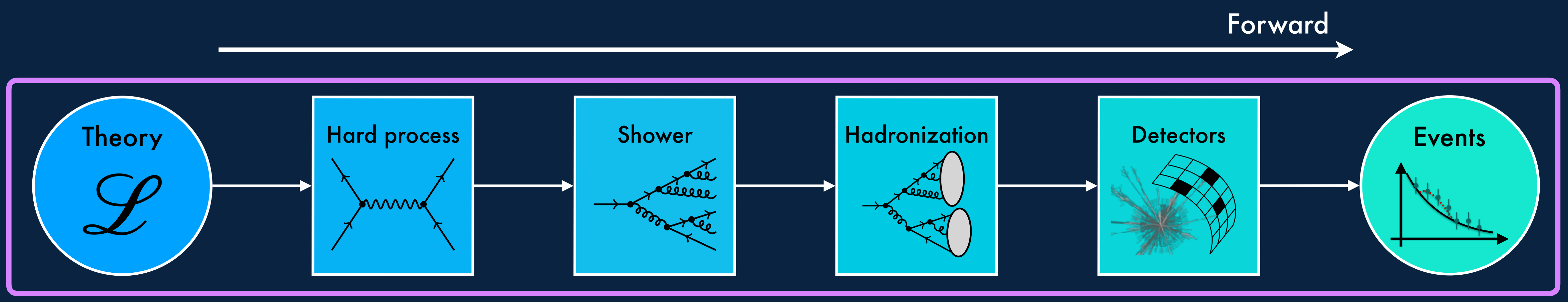
- First attempts based on **GANs and VAEs**
- Improved speed and efficiency with **Flows**

Butter, Heimel, Hummerich, Krebs, Plehn, Rousselot, Vent [2110.13632]



[1901.00875, 1901.05282, 1903.02433, 1907.03764, 1912.02748, 2001.11103, 2011.13445, 2101.08944, 2110.13632, 2211.13630, 2303.05376, 2305.07696, 2305.10475, 2305.16774, 2307.06836 .....]

ML for forward simulations

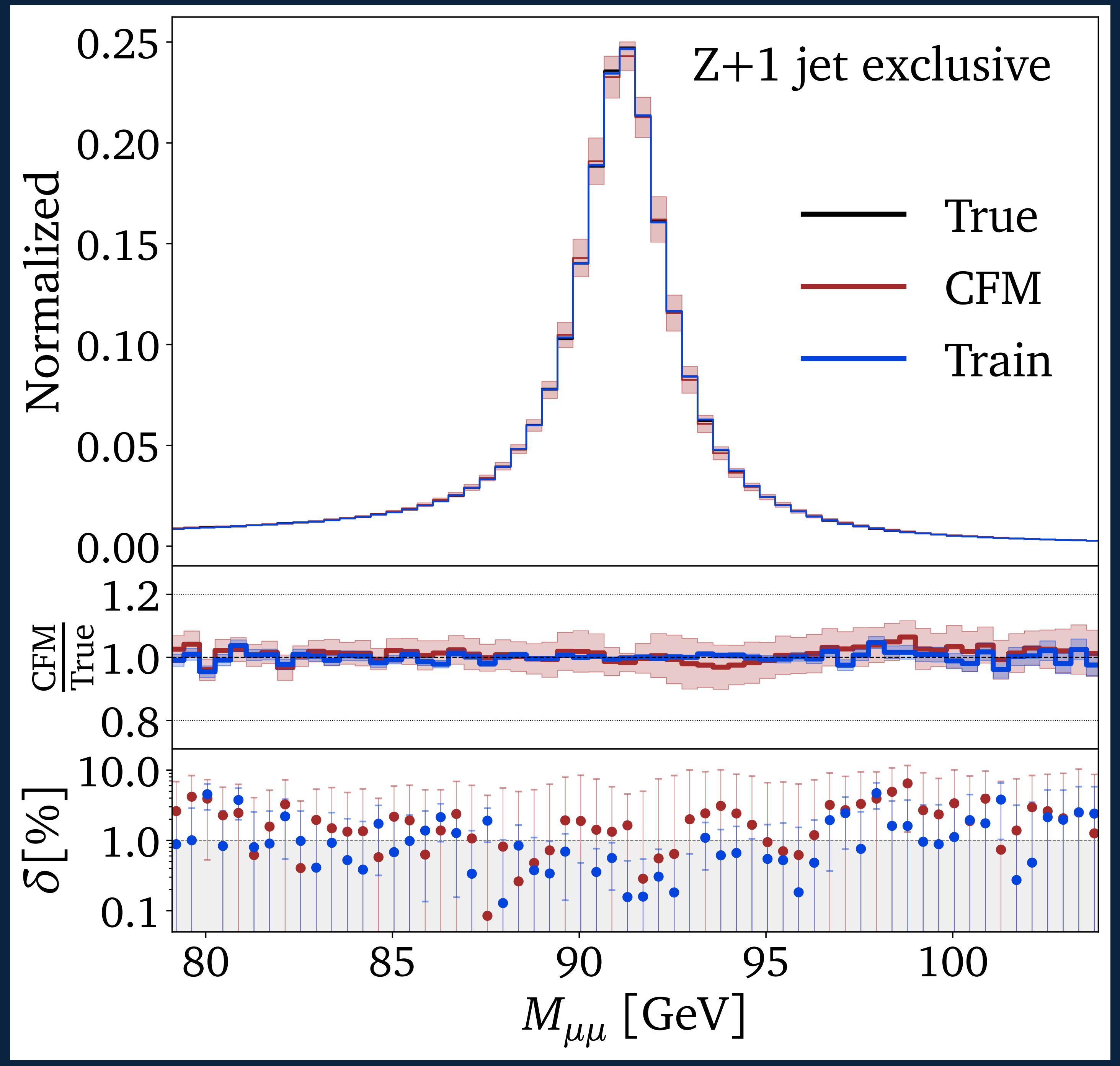


End-to-end generators learn multiple steps at once

Precision generation

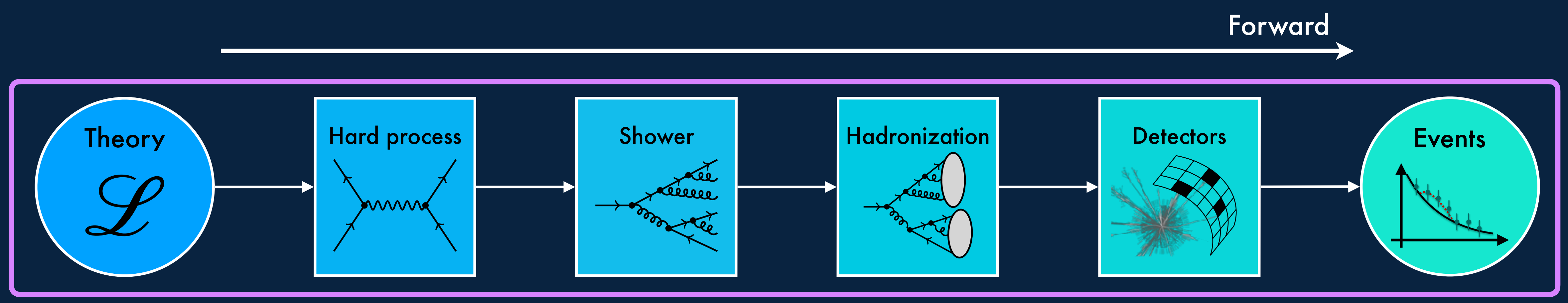
- First attempts based on **GANs and VAEs**
- Improved speed and efficiency with **Flows**
- High precision with **Diffusion** and **Transformer** models

Butter, Huetsch, Schweitzer, Plehn, Sorrenson, Spinner [2110.11377]



[1901.00875, 1901.05282, 1903.02433, 1907.03764, 1912.02748, 2001.11103, 2011.13445, 2101.08944, 2110.13632, 2211.13630, 2303.05376, 2305.07696, 2305.10475, 2305.16774, 2307.06836 .....]

ML for forward simulations



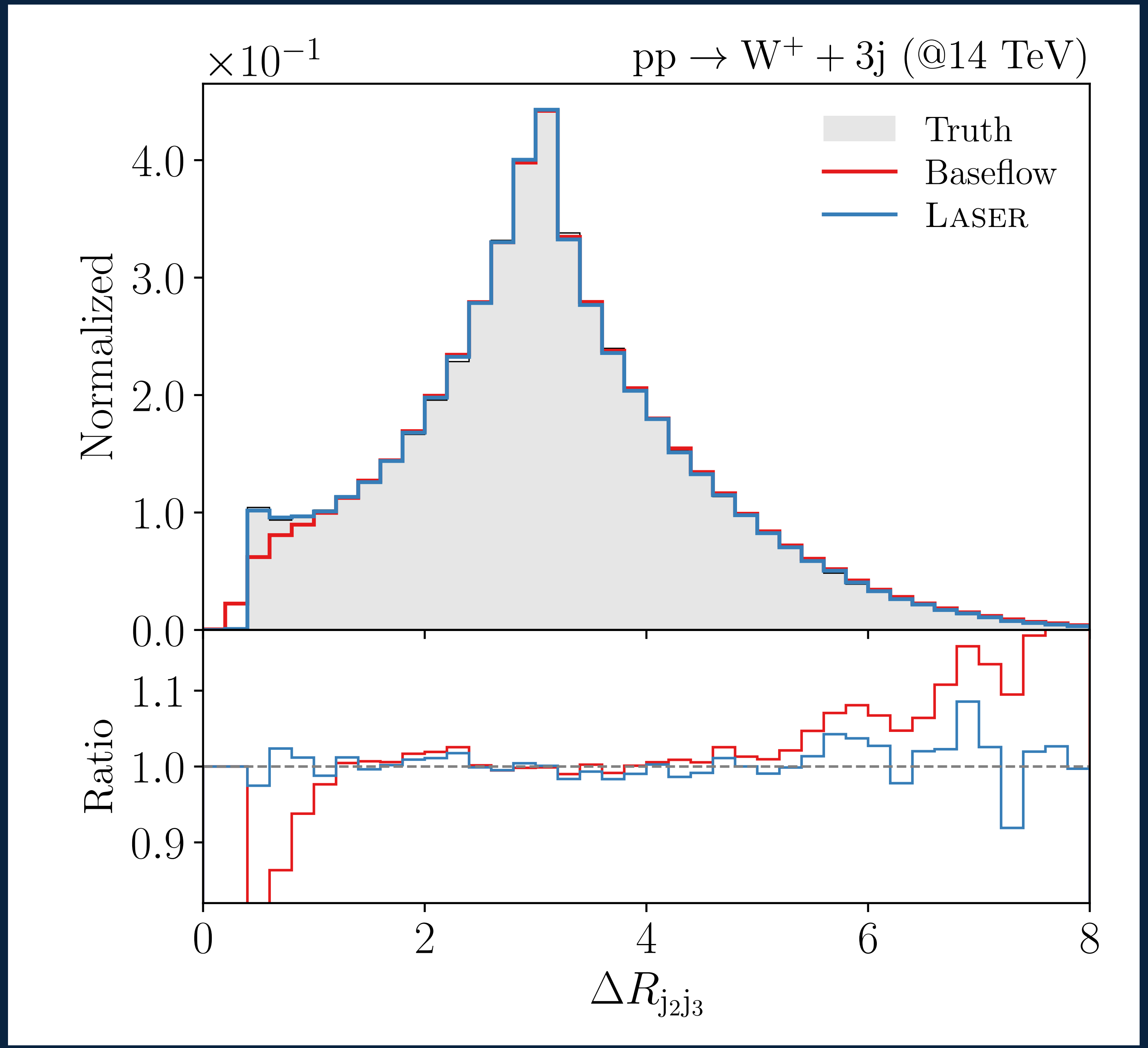
End-to-end generators learn multiple steps at once

Precision generation

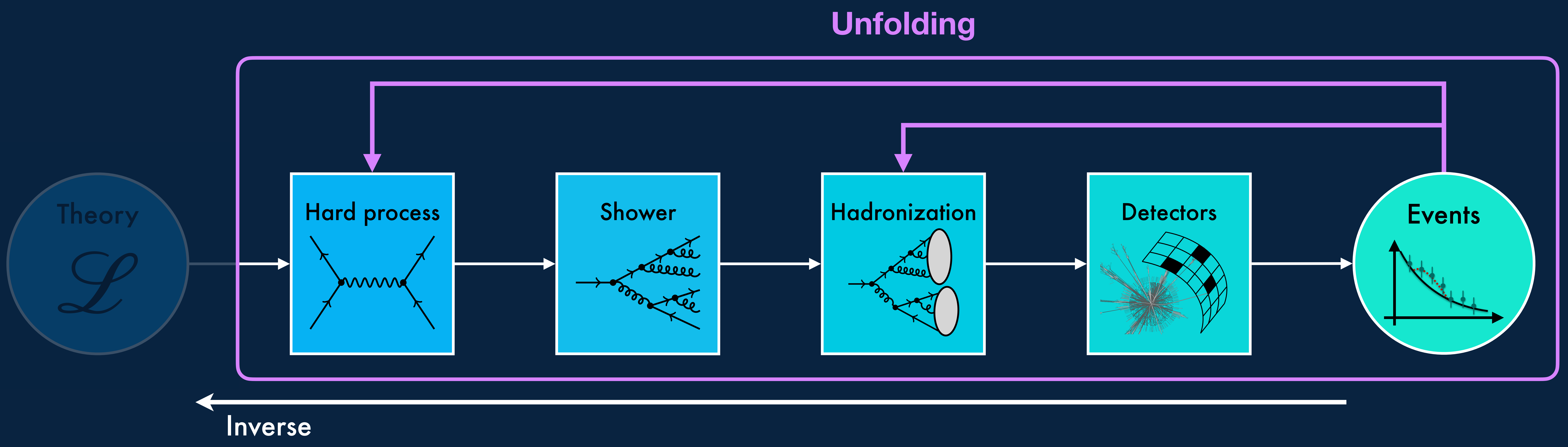
- First attempts based on **GANs and VAEs**
- Improved speed and efficiency with **Flows**
- High precision with **Diffusion** and **Transformer** models
- **Bayesian NN** and **classifiers** for full control

[1901.00875, 1901.05282, 1903.02433, 1907.03764, 1912.02748, 2001.11103, 2011.13445, 2101.08944, 2110.13632, 2211.13630, 2303.05376, 2305.07696, 2305.10475, 2305.16774, 2307.06836 .....]

Nachman, RW [2305.07696]



ML for inverse simulations



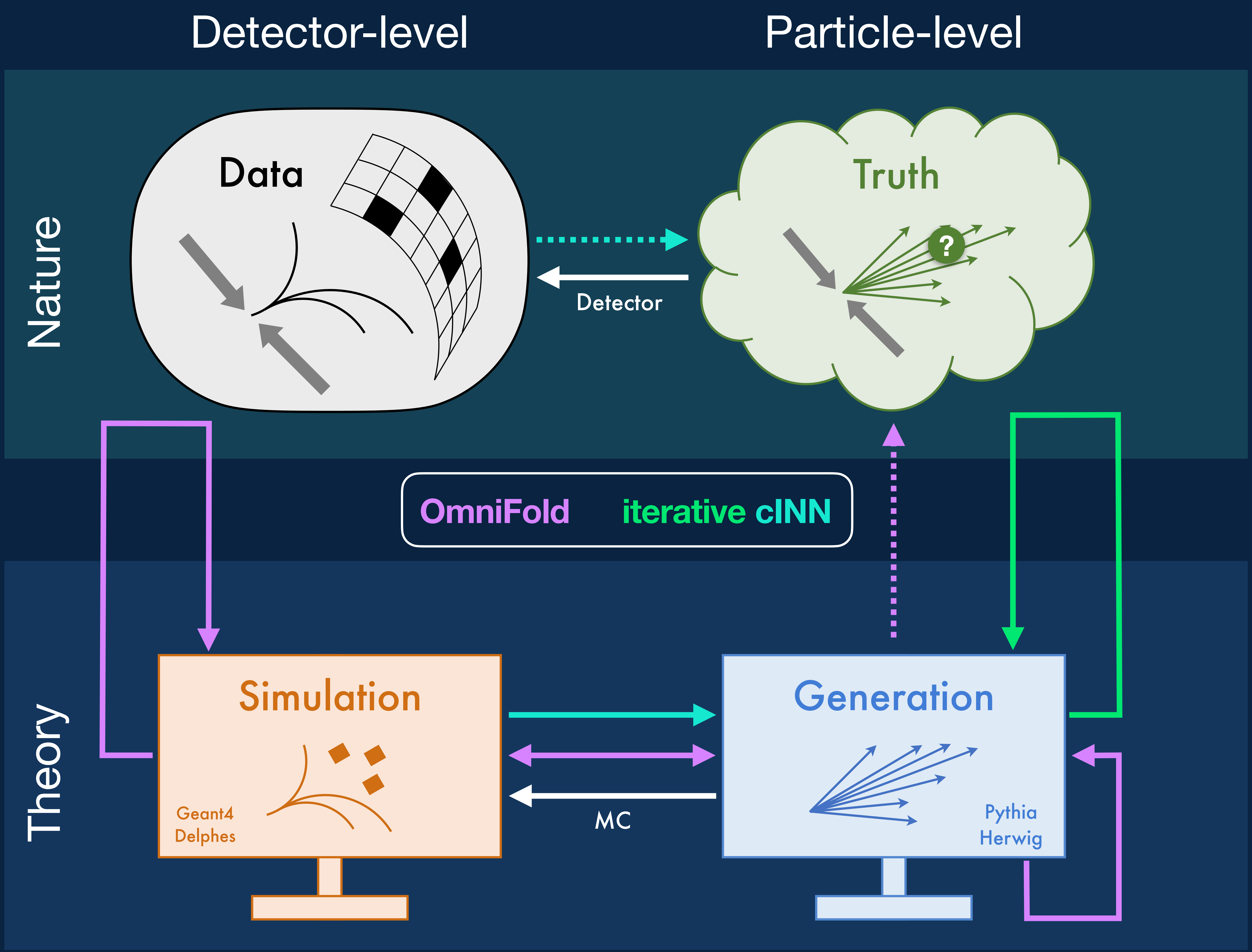
Unfolding

Classifier-based ⟷ **Density-based**

reweighting

generative

Unfolding — Basic concept



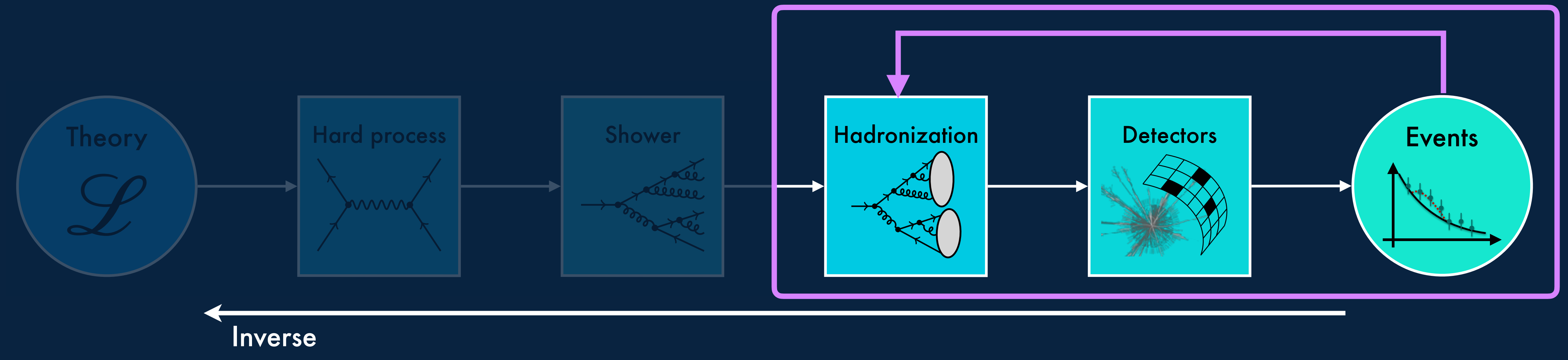
Requirements:
- High dimensional
- Unbinned
- Statistically well defined

Omnifold [1911.09107]
Andreassen, Komiske, Metodiev, Nachman

cINN [2006.06685]
Bellagente, Butter, Kasieczka, Plehn, Rousselot, RW, Ardizzone, Köthe

IcINN [2212.08674]
Backes, Butter, Dunford, Malaescu

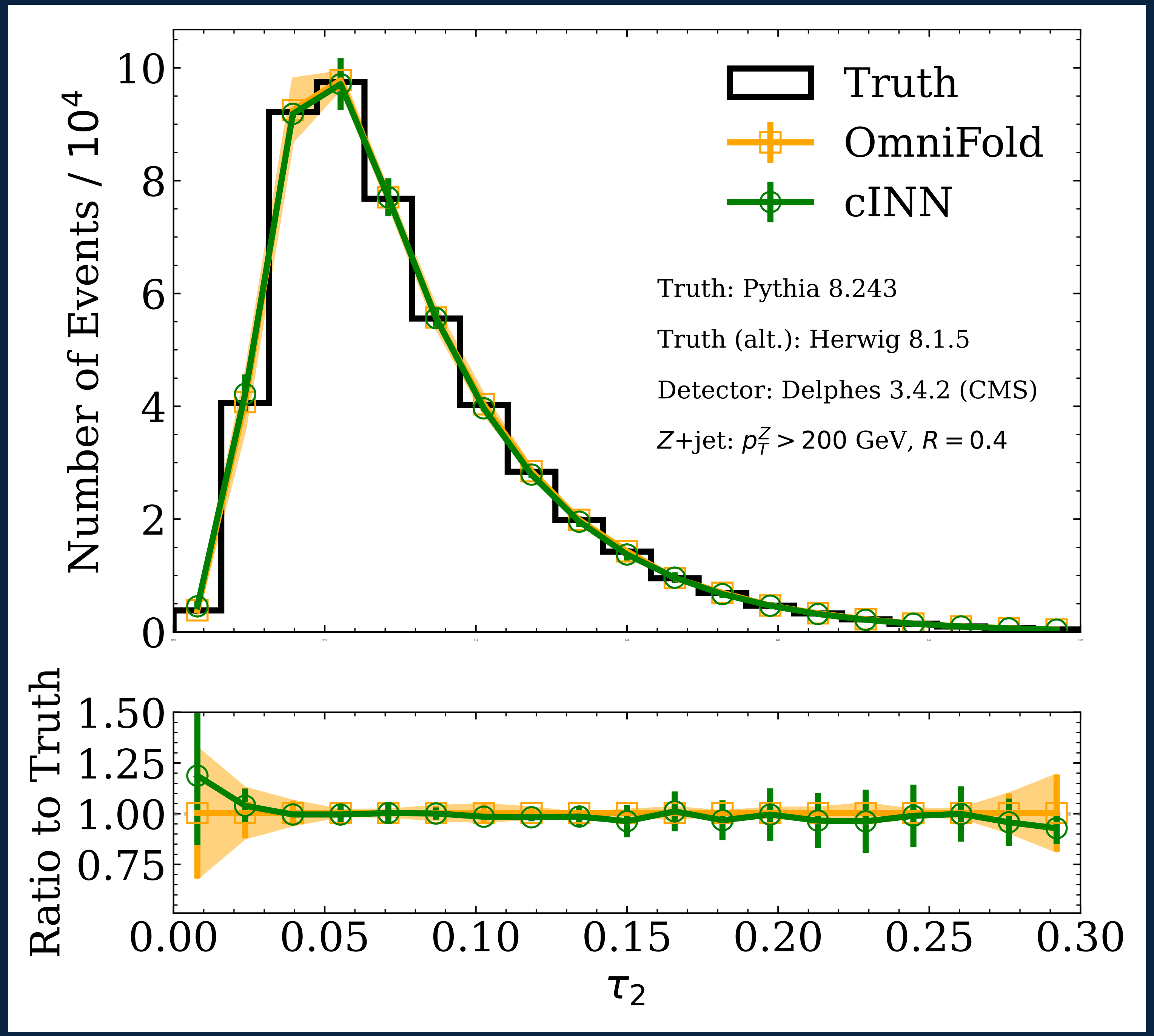
ML for inverse simulations



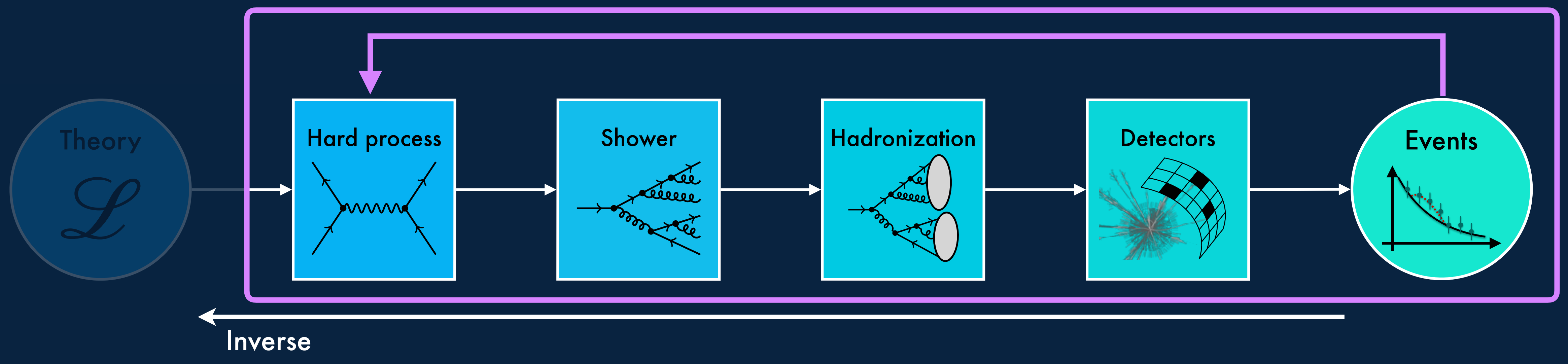
Unfolding of detector effects

- Must be high-dimensional, unbinned, and statistically well-defined
- **Classifier-based** MC reweighting
  [1911.09107, 2105.04448, 2105.09923, 2109.13243 2302.05390]
- **Density-based** generative unfolding
  [1806.00433, 1912.00477, 2006.06685, 2101.08944, 2109.13243, 2207.00664, 2212.08674, 2305.10399, 2307.02405, 2308.00027, 2308.12351, 2310.17037]

Arratia *et al.* [2109.13243]



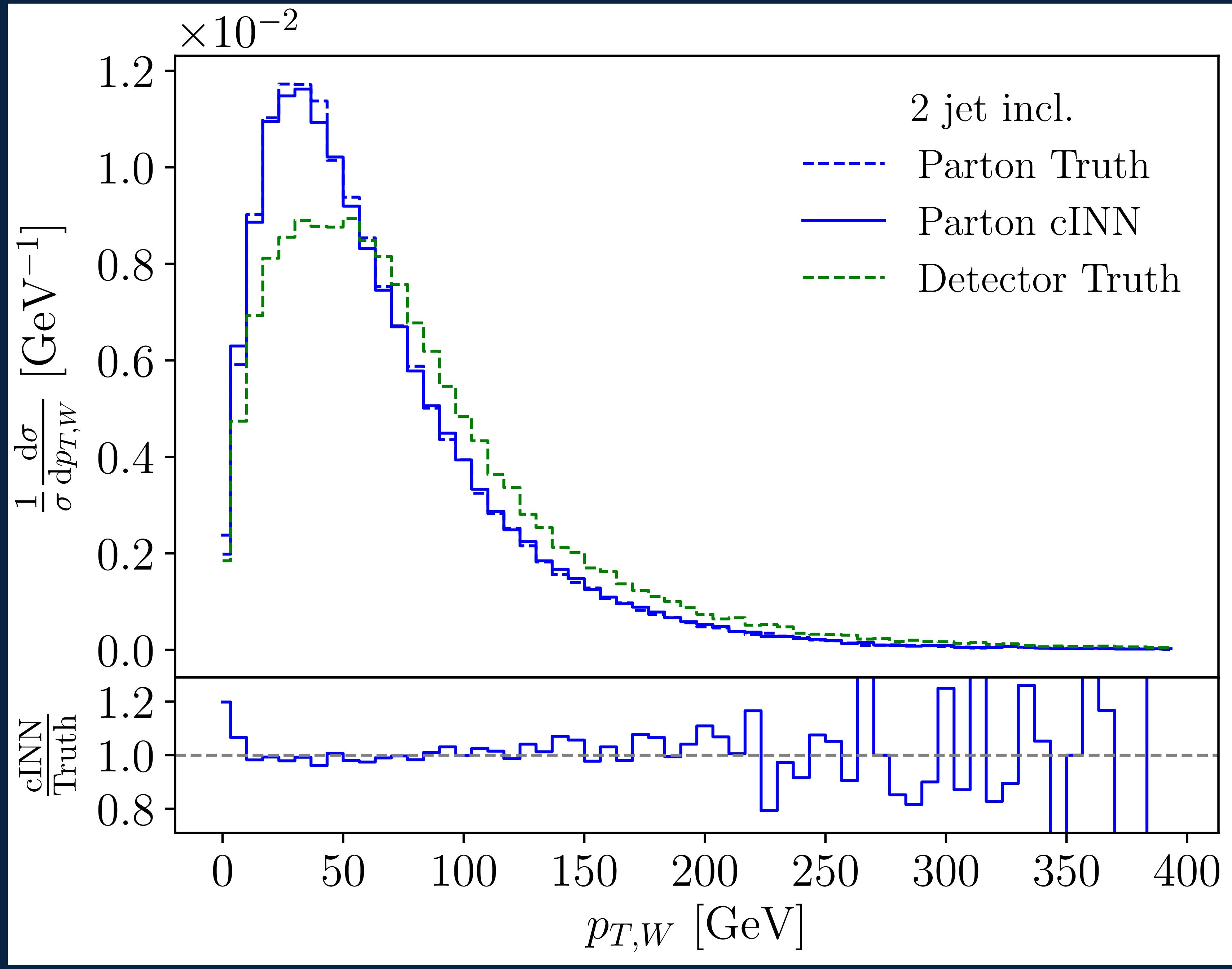
ML for inverse simulations



Inverting to parton level

Bellagente, Butter, Kasieczka, Plehn,
Rousselot, RW, Ardizzone, Köthe [2109.13243]

- Inversion of **QCD radiation** and heavy particle (t,W,Z,h) decays
- Use same techniques as before (**cINNs, Classifiers** + others)



[1912.00477, 2006.06685, 2101.08944, 2207.00664, 2210.00019, 2307.02405, 2308.00027, 2310.07752]

Summary and Outlook

Take-home message

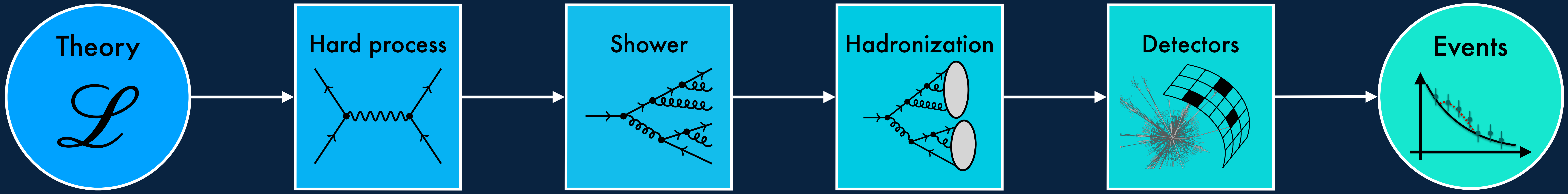
- ML beneficial in **every step** in the simulation chain
- We find both **proof-of-concepts** as well as established use cases **(→ MadNIS)**
- Interesting **interplay** between **HEP** and **ML**

  → HEP simulations provide **~infinite data** for ML

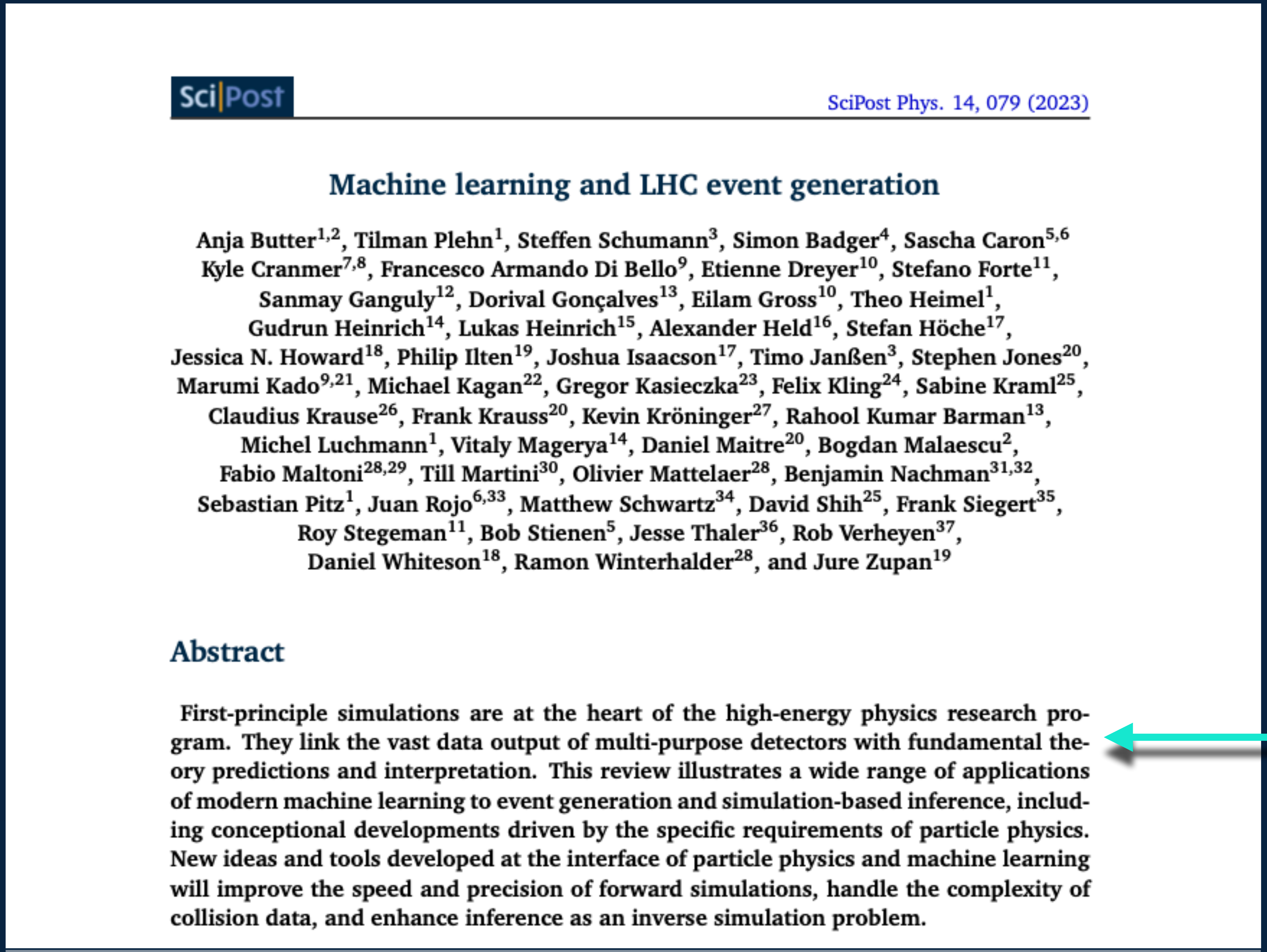
  → HEP requirements (**precision, symmertries,…**) **different** than industry applications

Future tasks

- **Full integration** of ML-based simulations into standard tools → **MadGraph,….**
- Make everything run on the **GPU and differentiable** (MadJax - Heinrich et al. [2203.00057])
- Foster deeper collaboration between **theory, experiment,** and **ML** community



Summary and Outlook



SciPost — SciPost Phys. 14, 079 (2023)

Machine learning and LHC event generation

Anja Butter[1,2], Tilman Plehn[1], Steffen Schumann[3], Simon Badger[4], Sascha Caron[5,6] Kyle Cranmer[7,8], Francesco Armando Di Bello[9], Etienne Dreyer[10], Stefano Forte[11], Sanmay Ganguly[12], Dorival Gonçalves[13], Eilam Gross[10], Theo Heimel[1], Gudrun Heinrich[14], Lukas Heinrich[15], Alexander Held[16], Stefan Höche[17], Jessica N. Howard[18], Philip Ilten[19], Joshua Isaacson[17], Timo Janßen[3], Stephen Jones[20], Marumi Kado[9,21], Michael Kagan[22], Gregor Kasieczka[23], Felix Kling[24], Sabine Kraml[25], Claudius Krause[26], Frank Krauss[20], Kevin Kröninger[27], Rahool Kumar Barman[13], Michel Luchmann[1], Vitaly Magerya[14], Daniel Maitre[20], Bogdan Malaescu[2], Fabio Maltoni[28,29], Till Martini[30], Olivier Mattelaer[28], Benjamin Nachman[31,32], Sebastian Pitz[1], Juan Rojo[6,33], Matthew Schwartz[34], David Shih[25], Frank Siegert[35], Roy Stegeman[11], Bob Stienen[5], Jesse Thaler[36], Rob Verheyen[37], Daniel Whiteson[18], Ramon Winterhalder[28], and Jure Zupan[19]

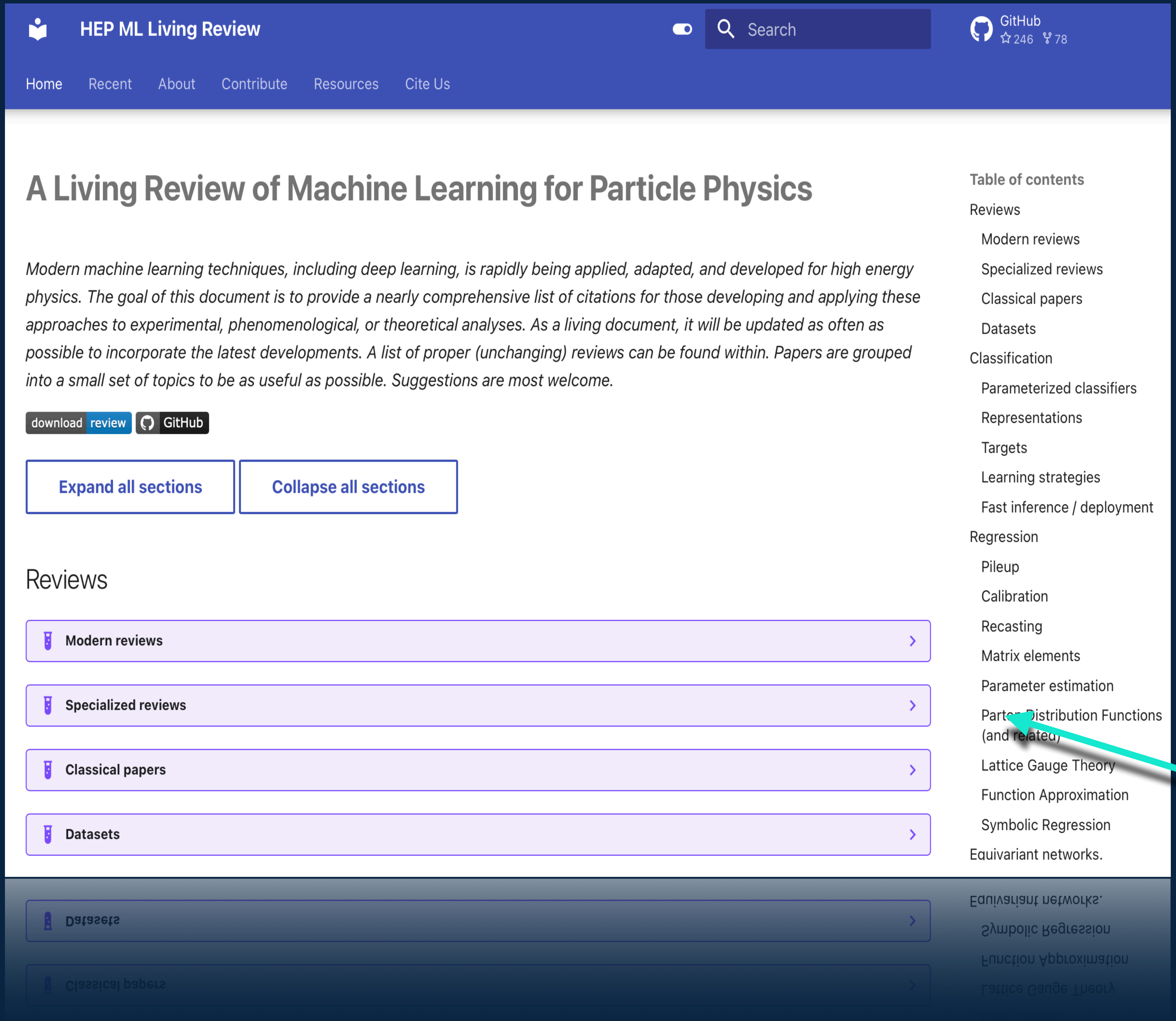
Abstract

First-principle simulations are at the heart of the high-energy physics research program. They link the vast data output of multi-purpose detectors with fundamental theory predictions and interpretation. This review illustrates a wide range of applications of modern machine learning to event generation and simulation-based inference, including conceptional developments driven by the specific requirements of particle physics. New ideas and tools developed at the interface of particle physics and machine learning will improve the speed and precision of forward simulations, handle the complexity of collision data, and enhance inference as an inverse simulation problem.

Future tasks

- **Full integration** of ML-based simulations into standard tools → **MadGraph,….**
- Make everything run on the **GPU and differentiable** (MadJax - Heinrich et al. [2203.00057])
- Foster deeper collaboration between **theory, experiment,** and **ML** community
- More details in our **Snowmass report**

Summary and Outlook



Future tasks

- **Full integration** of ML-based simulations into standard tools → **MadGraph,….**
- Make everything run on the **GPU and differentiable** (MadJax - Heinrich et al. [2203.00057])
- Foster deeper collaboration between **theory, experiment,** and **ML** community
- More details in our **Snowmass report**
- Stay tuned for many other **ML4HEP applications**

HEPML