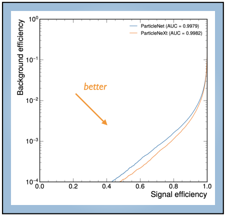
Machine Learning for LHC Theory

Particle Physics Seminars at BNL

Anja Butter

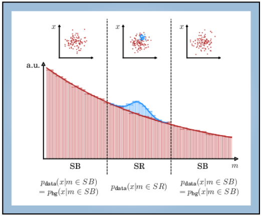
ITP, Universität Heidelberg / LPNHE, Paris

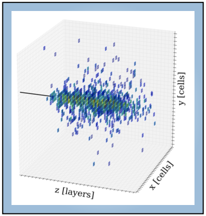
Performance boosts and new developments for many applications



←Top tagging

Anomaly detection →





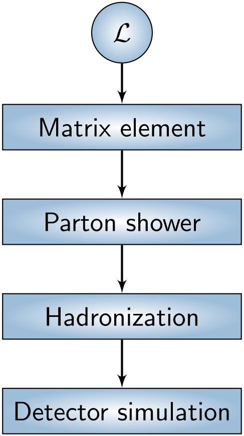
←Detector simulation

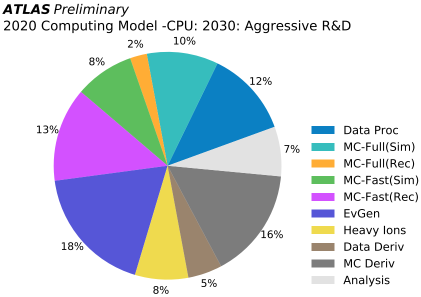
What about ML4Theory?

Better predictions?
→ ML for precision simulations

Better understanding?
→ Turn data into theory?

Precision simulations with limited resources





Speed = Precision

Boosting standard event generation...

1. Generate phase space points

2. Calculate event weight

$$w_{event} = \boxed{f(x_1, Q^2) f(x_2, Q^2)} \times \boxed{\mathcal{M}(x_1, x_2, p_1, \dots p_n)} \times \boxed{J(p_i(r))^{-1}}$$

3. Unweighting
$\rightarrow$ optimal for $w \approx 1$

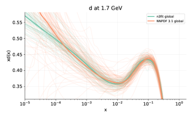
Boosting standard event generation...

$$w_{event} = \underbrace{f(x_1, Q^2) f(x_2, Q^2)}_{\text{PDF}} \times \overbrace{\mathcal{M}(x_1, x_2, p_1, \ldots p_n)}^{\text{Matrix element}} \times \underbrace{J(p_i(r))^{-1}}_{\text{Phase space mapping}}$$

Boosting standard event generation...

Matrix element

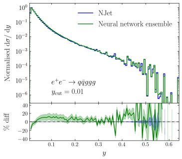
$$w_{event} = \boxed{f(x_1, Q^2) f(x_2, Q^2)} \times \boxed{\mathcal{M}(x_1, x_2, p_1, \ldots p_n)} \times \boxed{J(p_i(r))^{-1}}$$



- NNPDF since 2002(!)
- S. Carrazza, J. Cruz-Martinez [1907.05075]

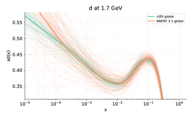
Phase space mapping

Boosting standard event generation...



- Amplitude estimation
- S. Badger, J. Bullock [2002.07516]
- J. Bendavid [1707.00028]

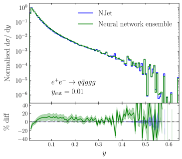
$$w_{event} = f(x_1, Q^2)f(x_2, Q^2) \times \mathcal{M}(x_1, x_2, p_1, \ldots p_n) \times J(p_i(r))^{-1}$$



- NNPDF since 2002(!)
- S. Carrazza, J. Cruz-Martinez [1907.05075]

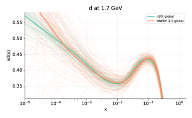
Phase space mapping

Boosting standard event generation...

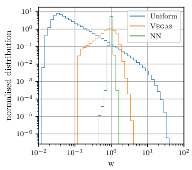


- Amplitude estimation
- S. Badger, J. Bullock [2002.07516]
- J. Bendavid [1707.00028]

$$w_{event} = f(x_1, Q^2) f(x_2, Q^2) \times \mathcal{M}(x_1, x_2, p_1, \ldots p_n) \times J(p_i(r))^{-1}$$



- NNPDF since 2002(!)
- S. Carrazza, J. Cruz-Martinez [1907.05075]



- Learn phase space mapping ($\to w \approx 1$)
- Gao et al. [2001.10028]
- Bothmann et al. [2001.05478]

... or training directly on event samples

Event generation

- Generating 4-momenta

→ Phase space sampling, data compression, interpolation, . . .

[1901.00875] Otten et al. **VAE** & **GAN**

[1901.05282] Hashemi et al. **GAN**

[1903.02433] Di Sipio et al. **GAN**

[1903.02556] Lin et al. **GAN**

[1907.03764, 1912.08824] Butter et al. **GAN**

[1912.02748] Martinez et al. **GAN**

[2001.11103] Alanazi et al. **GAN**

[2011.13445] Stienen et al. **NF**

[2012.07873] Backes et al. **GAN**

[2101.08944] Howard et al. **VAE**

Detector simulation

- Jet images
- Fast calorimeter simulation

[1701.05927] de Oliveira et al. **GAN**

[1705.02355, 1712.10321] Paganini et al. **GAN**

[1802.03325, 1807.01954] Erdmann et al. **GAN**

[1805.00850] Musella et al. **GAN**

[ATL-SOFT-PUB-2018-001, ATLAS-SIM-2019-004, ATL-SOFT-PROC-2019-007] ATLAS **VAE** & **GAN**

[1909.01359] Carazza and Dreyer **GAN**

[1912.06794] Belayneh et al. **GAN**

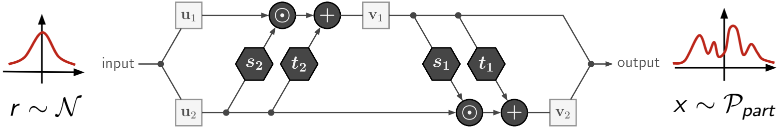
[2005.05334] Buhmann et al. **VAE**

[2009.03796] Diefenbacher et al. **GAN**

[2009.14017] Lu et al.

NO claim to completeness!

Invertible networks



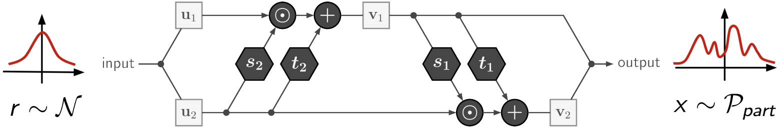
$+$ Tractable Jacobian

$+$ Enable correction for perfect precision

$+$ Fast evaluation in both directions

$$\begin{pmatrix} v_1 \\ v_2 \end{pmatrix} = \begin{pmatrix} u_1 \cdot s_2(u_2) + t_2(u_2) \\ u_2 \end{pmatrix}$$

Invertible networks



$+$ Tractable Jacobian

$+$ Enable correction for perfect precision

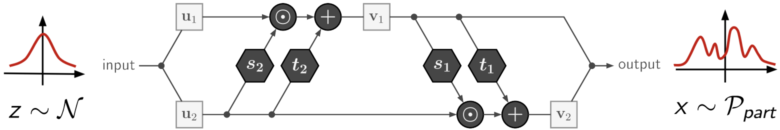
$+$ Fast evaluation in both directions

$$\begin{pmatrix} v_1 \\ v_2 \end{pmatrix} = \begin{pmatrix} u_1 \cdot s_2(u_2) + t_2(u_2) \\ u_2 \end{pmatrix}$$

Many alternative implementations, eg. cubic splines

Training on density

Sherpa [2001.05478, 2001.10028]



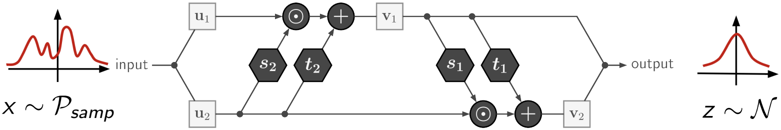
- $z \sim \mathcal{N} \rightarrow$ NN $\rightarrow x \sim p_x$
- $p_x(x) = p_z(z) \cdot J_{\text{NN}}$
- Given target density $t(x)$

$\rightarrow$ Train NN to minimize $\log(p_z(z) \cdot J_{\text{NN}}/t(x))$

- Problem: Calculate $f(x)$ each time

Training on samples

A.B., T. Heimel, S. Hummerich, T. Krebs, T. Plehn, A. Rousselot, S. Vent [arXiv:2110.13632]



- $x \sim p_{\text{samples}} \rightarrow$ NN $\rightarrow z$

$\rightarrow$ Train NN to ensure $z \sim \mathcal{N}$

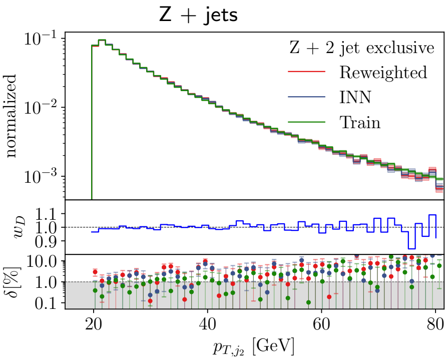
- Loss: Maximize posterior over network weights:

$$\begin{aligned} -\log(p(\theta|x)) &= -\log(p(x|\theta)) - \log(p(\theta)) + \text{const.} \\ &= -\log(p(z|\theta)) - \log(J) - \log(p(\theta)) + \text{const.} \end{aligned}$$

Naive INN results

Inclusive Z+jets production

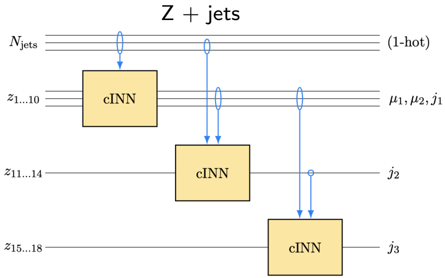
- INN easy trainable, powerful baseline
- Challenges:



Naive INN results

Inclusive Z+jets production

- INN easy trainable, powerful baseline
- Challenges:
  - Variable number of jets

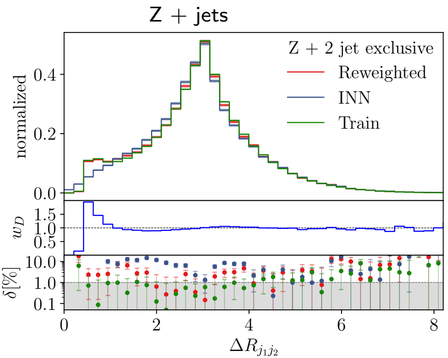


Naive INN results

Inclusive Z+jets production

- INN easy trainable, powerful baseline

- Challenges:
  - Variable number of jets
  - Topological holes



Reweighting

Discriminator

$$\begin{aligned}\mathcal{L} &= -\sum_{x\sim p_{data}} \log(D(x)) - \sum_{x\sim p_{INN}} \log(1-D(x))\\ &= -\int \mathrm{d}x\; p_{data}(x)\log(D(x)) + p_{inn}(x)\log(1-D(x))\end{aligned}$$

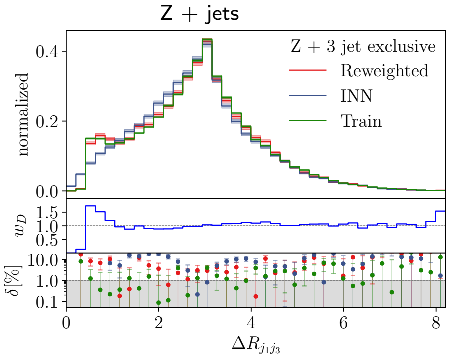
From variation we obtain

$$\begin{aligned}0 &= \frac{p_{data}(x)}{D(x)} - \frac{p_{inn}(x)}{1-D(x)}\\ \Rightarrow \frac{p_{data}(x)}{p_{inn}(x)} &= \frac{D(x)}{1-D(x)}\end{aligned}$$

Reweighting the generated distributions

Inclusive Z+jets production

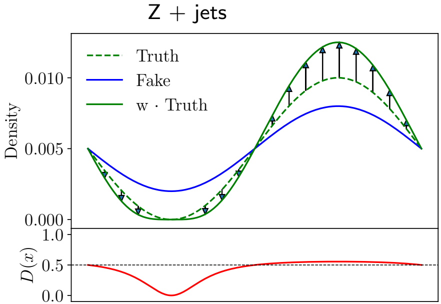
- Reweighted results show significant improvement
- Include discriminator information to improve training



Reweighting the generated distributions

Inclusive Z+jets production

- Reweighted results show significant improvement
- Include discriminator information to improve training
- Discflow

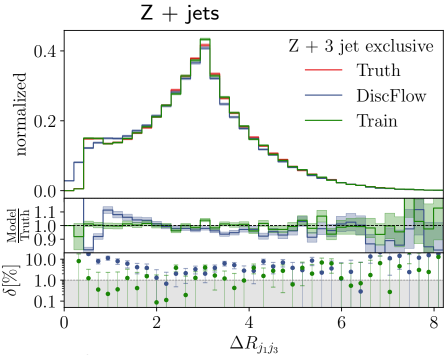


$$\mathcal{L}_{\text{DiscFlow}} = \sum_{i=1}^{B} w_D(x_i)^\alpha \left(\frac{\psi(x_i; c_i)^2}{2} - \log J(x_i) \right)$$
$$\approx \int dx \; \underbrace{w_D(x)^\alpha P(x)}_{\text{reweighted truth}} \left(\frac{\psi(x; c)^2}{2} - \log J(x) \right)$$

Reweighting the generated distributions

Inclusive Z+jets production

- Reweighted results show significant improvement
- Include discriminator information to improve training
- Discflow



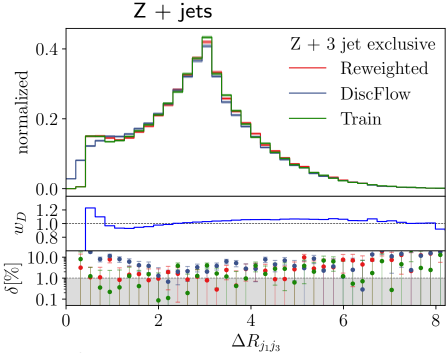
$$\mathcal{L}_{\text{DiscFlow}} = \sum_{i=1}^{B} w_D(x_i)^\alpha \left(\frac{\psi(x_i; c_i)^2}{2} - \log J(x_i) \right)$$

$$\approx \int dx \; \underbrace{w_D(x)^\alpha P(x)}_{\text{reweighted truth}} \left(\frac{\psi(x; c)^2}{2} - \log J(x) \right)$$

Reweighting the generated distributions

Inclusive Z+jets production

- Reweighted results show significant improvement
- Include discriminator information to improve training
- Discflow + Reweighting



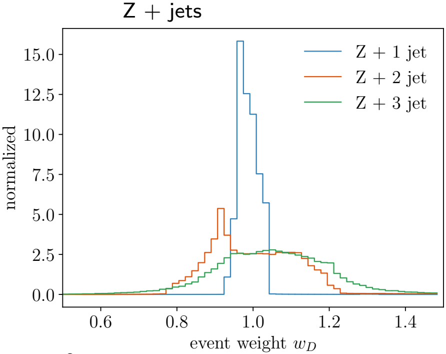
$$\mathcal{L}_{\text{DiscFlow}} = \sum_{i=1}^{B} w_D(x_i)^\alpha \left(\frac{\psi(x_i; c_i)^2}{2} - \log J(x_i) \right)$$

$$\approx \int dx \; \underbrace{w_D(x)^\alpha P(x)}_{\text{reweighted truth}} \left(\frac{\psi(x; c)^2}{2} - \log J(x) \right)$$

Reweighting the generated distributions

Inclusive Z+jets production

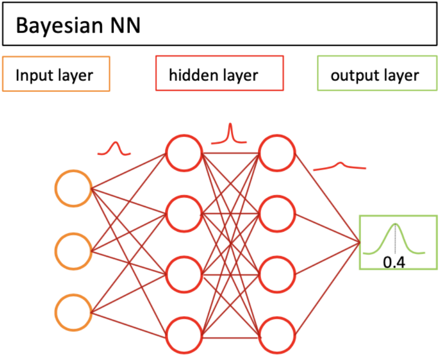
- Reweighted results show significant improvement
- Include discriminator information to improve training
- Discflow



$$\mathcal{L}_{\text{DiscFlow}} = \sum_{i=1}^{B} w_D(x_i)^\alpha \left(\frac{\psi(x_i; c_i)^2}{2} - \log J(x_i) \right)$$

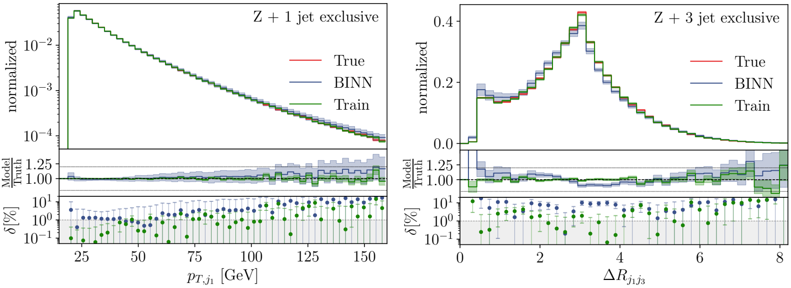
$$\approx \int dx \; \underbrace{w_D(x)^\alpha P(x)}_{\text{reweighted truth}} \left(\frac{\psi(x; c)^2}{2} - \log J(x) \right)$$

Addressing uncertainties



$$\mathcal{L} = \mathcal{L}_{INN} + KL_{prior}$$

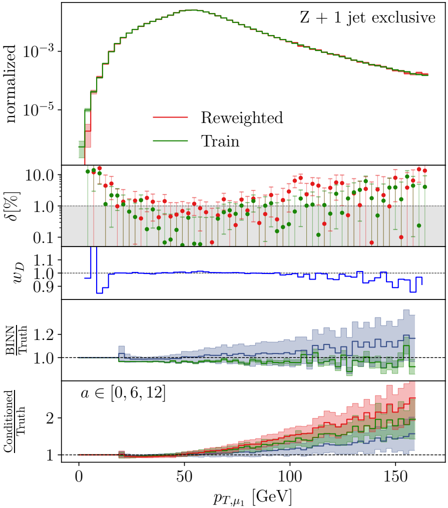
BINN results



⇒ BINN uncertainty captures convergence of the network ✓

⇒ BINN uncertainty does NOT capture where network fails

Overview on uncertainties



Short summary

INNs can ..

$\rightarrow$ learn event distributions and correlations

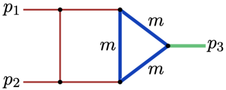
$\rightarrow$ achieve higher precision through reweighting and Discflow

$\rightarrow$ be extended to BINN to assign uncertainties

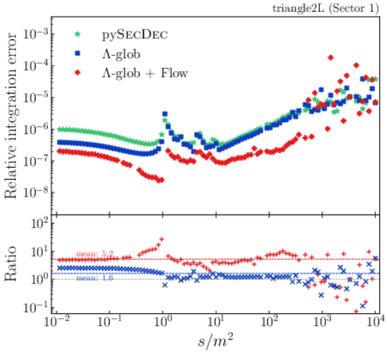
Targeting loop amplitudes

- Neural networks can learn amplitudes

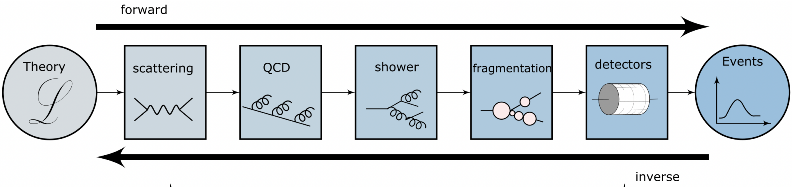
→ Precision?



- Feynman integrals often contain singularities
- Solved by contour deformation due to Cauchy's theorem
- Parametrize with NN
- Minimize variance of the integral



Can we invert the simulation chain?



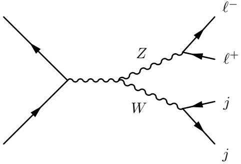
What we
want to know

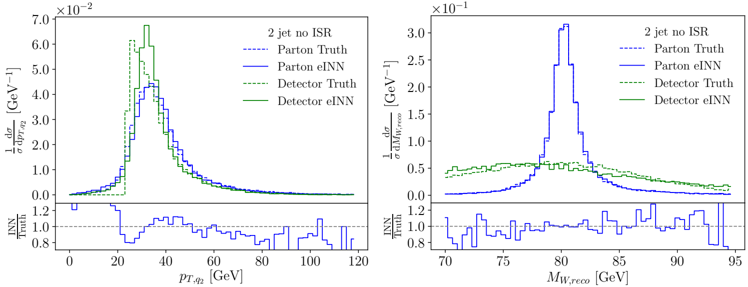
What we
measure or simulate

wish list:
- ☐ multi-dimensional
- ☐ bin independent
- ☐ statistically well defined

Inverting detector effects

$$\begin{pmatrix} x_{part} \\ r_{part} \end{pmatrix} \xleftrightarrow[\leftarrow\ \text{unfolding}:\bar{g}]{\textsc{Pythia},\textsc{Delphes}:g\rightarrow} \begin{pmatrix} x_{det} \\ r_{det} \end{pmatrix}$$





multi-dimensional ✓ bin independent ✓ statistically well defined ?

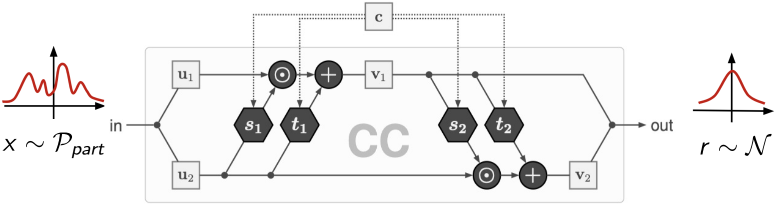
Asking the right question

Given an event x_d, what is the probability distribution at parton level?
$\rightarrow$ event generation conditioned on x_d

$$x_p \overset{g(x_p, f(x_d)) \rightarrow}{\underset{\leftarrow \text{ unfolding: } \bar{g}(r, f(x_d))}{\longleftrightarrow}} r$$

Minimizing the posterior

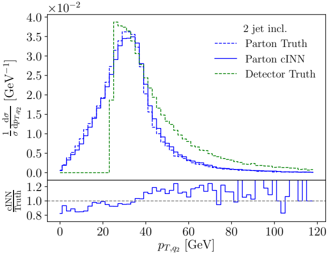
$$L = \left\langle 0.5 ||\bar{g}(x_p, f(x_d))||_2^2 - \log |J| \right\rangle_{x_p \sim P_p, x_d \sim P_d} - \log p(\theta)$$



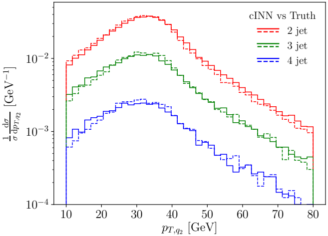
Inverting the full event

$pp > WZ > q\bar{q}l^+l^- + \text{ISR}$
$\to$ 2/3/4 jet events

Train on inclusive dataset



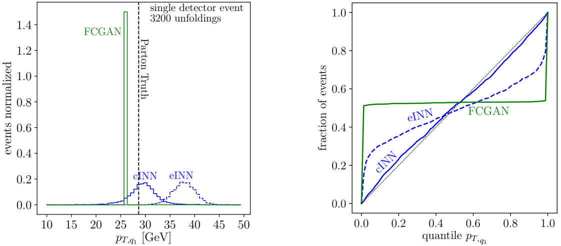
Evaluate
exclusive 2/3/4 jet channels



multi-dimensional ✓ bin independent ✓ statistically well defined ✓

Condition INN on detector data [2006.06685]

$$x_p \xleftrightarrow[\leftarrow \text{ unfolding: } \bar{g}(r, f(x_d))]{g(x_p, f(x_d)) \rightarrow} r$$

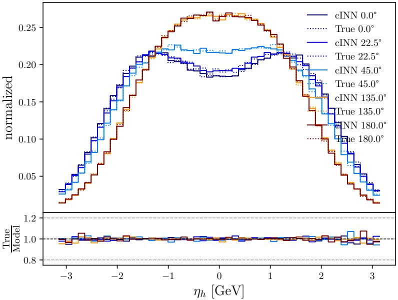


Application to MEM

current work in progress with T. Martini, T. Heimel, S. Peitzsch, T. Plehn

- Single top production in association with Higgs
- Measure CP-phase in the top Yukawa coupling

$$\mathcal{L}(\alpha) = \prod_{i=1}^{N} \frac{1}{\sigma_{\text{fid}}(\alpha)} \int d^m z \; \frac{d^m\sigma(\alpha)}{dz_1 ... dz_m} \; T(\vec{y}^{(i)}, \vec{z}) \; .$$



We can use neural networks ...

... to improve precision simulations in forward direction

... to achieve **precision** with discriminators

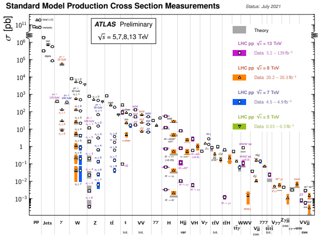
... to estimate the corresponding **uncertainties**

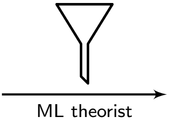
... to learn and calculate **loop amplitudes**

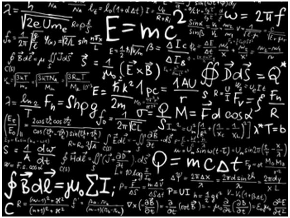
... to **invert** the simulation chain statistically

It doesn't always have to be a neural network

Can we learn theory from data?







Let's try...

arXiv:2109.10414

Johann Brehmer, A.B., Tilman Plehn, Nathalie Soybelman

Two concepts

1. Problem
$\rightarrow$ Optimal observable to measure parameter θ

2. Mechanism
$\rightarrow$ Learn analytic OO with symbolic regression

What is the optimal observable to measure θ?

- Experiments measure high-dimensional data x_{reco}
- 1D representation (p_T, m_{jj}) looses information

What is the optimal observable to measure θ?

- Experiments measure high-dimensional data x_{reco}
- 1D representation (p_T, m_{jj}) looses information
- Starting point: likelihood

$$p(x_{reco}|\theta) = \frac{1}{\sigma(\theta)} \frac{\mathrm{d}\sigma(x_{reco}|\theta)}{\mathrm{d}x_{reco}}$$

- Optimal observable

$$\mathcal{O}_i^{\text{opt}}(x) = \left.\frac{\partial \log p(x|\theta)}{\partial \theta_i}\right|_{\theta_0} \equiv t(x|\theta_0) \quad (\rightarrow \textit{score})$$

$\rightarrow$ contains all information on θ $\rightarrow$ *sufficient statistics*

What is the optimal observable to measure θ?

- Experiments measure high-dimensional data x_{reco}
- 1D representation (p_T, m_{jj}) looses information
- Starting point: likelihood

$$p(x_{reco}|\theta) = \frac{1}{\sigma(\theta)} \frac{\mathrm{d}\sigma(x_{reco}|\theta)}{\mathrm{d}x_{reco}}$$

- Optimal observable

$$\mathcal{O}_i^{\text{opt}}(x) = \left. \frac{\partial \log p(x|\theta)}{\partial \theta_i} \right|_{\theta_0} \equiv t(x|\theta_0) \quad (\rightarrow \textit{score})$$

$\rightarrow$ contains all information on θ $\rightarrow$ *sufficient statistics*

- Problem: $p(x_{reco}|\theta)$ is untractable

$$p(x_{reco}|\theta) = \int \mathrm{d}z \; p(x_{reco}|z_{det}) p(z_{det}|z_{shower}) p(z_{shower}|z_{parton}) p(z_{parton}|\theta)$$

How to compute the optimal observable

- Solution:
  - Consider *joint score* $\rightarrow$ NO integral

$$t(x,z|\theta) = \nabla_\theta \log p(x,z|\theta)$$

$$= \frac{\nabla_\theta |\mathcal{M}(z|\theta)|^2}{|\mathcal{M}(z|\theta)|^2} - \frac{\nabla_\theta \sigma_{\text{tot}}(\theta))}{\sigma_{\text{tot}}(\theta)} \qquad \text{using } p(z|\theta) = \frac{1}{\sigma(\theta)} \frac{\mathrm{d}\sigma(z|\theta)}{\mathrm{d}z}$$

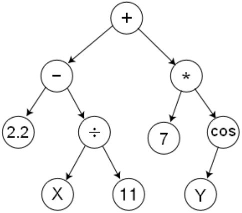
  - Optimal observable is given by

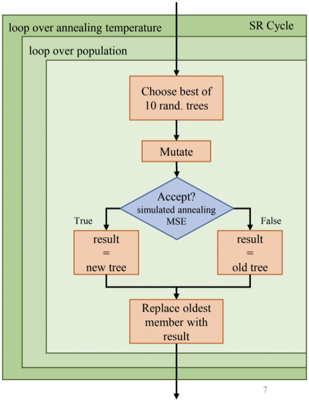
$$t(x|\theta) = \arg\min_{g(x)} \mathcal{E}_{x,z\sim p(x,z|\theta)} \left|g(x) - t(x,z|\theta)\right|^2$$

- Option 1: Minimization with NN $\rightarrow$ SALLY J. Brehmer, et al. [1805.12244]
- *new* Option 2: Learn *analytic* formula to minimize $g(x)$

Symbolic regression with PySR Miles Cranmer, et al.

Tree representation of $g(x)$





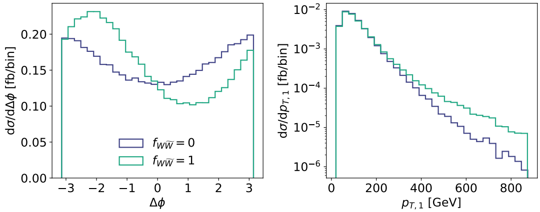
$$\texttt{pysr score} = \frac{\sum_{data}(g(x) - t(x,z|\theta))^2}{\texttt{baseline}} + \texttt{parsimony} \cdot \texttt{complexity}$$

$$p_{\text{accept}} = \exp\left(-\frac{\texttt{score}_{\text{new}} - \texttt{score}_{\text{old}}}{\texttt{alpha} \cdot T \cdot \texttt{score}_{\text{old}}}\right)$$ ← modified wrt. original PySR

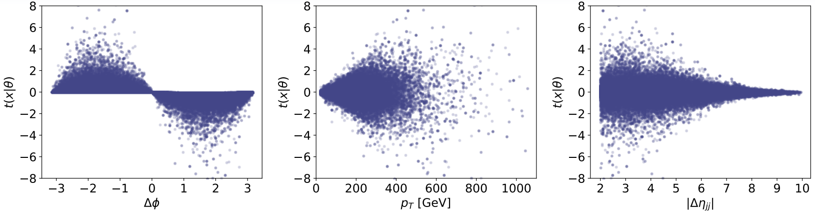
WBF Higgs production with CP violation

Test VVH vertex in WBF Higgs production

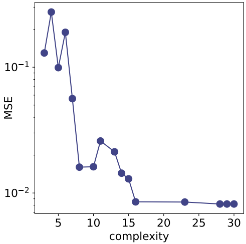
$$\mathcal{L} = \mathcal{L}_{\text{SM}} + \frac{f_{W\widetilde{W}}}{\Lambda^2}\mathcal{O}_{W\widetilde{W}} \qquad \text{with} \qquad \mathcal{O}_{W\widetilde{W}} = -(\phi^\dagger\phi)\,\widetilde{W}^k_{\mu\nu}W^{\mu\nu k}$$



Result $f_{W\widetilde{W}} = 0$



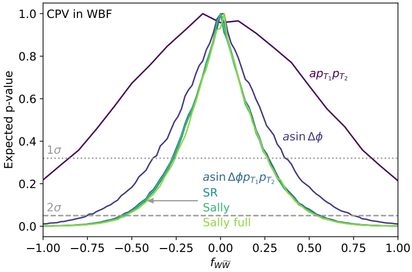
compl	function	MSE
3	$a\,\Delta\phi$	$1.30 \cdot 10^{-1}$
4	$\sin(a\Delta\phi)$	$2.75 \cdot 10^{-1}$
5	$a\Delta\phi x_{p,1}$	$9.93 \cdot 10^{-2}$
6	$-x_{p,1}\sin(\Delta\phi + a)$	$1.90 \cdot 10^{-1}$
8	$(a - x_{p,1})x_{p,2}\sin(\Delta\phi)$	$1.61 \cdot 10^{-2}$
16	$-x_{p,1}(a - b\Delta\eta)(x_{p,2} + c)\sin(\Delta\phi + d)$	$8.50 \cdot 10^{-3}$
28	$(x_{p,2} + a)(bx_{p,1}(c - \Delta\phi)$ $-x_{p,1}(d\Delta\eta + ex_{p,2} + f)\sin(\Delta\phi + g))$	$8.18 \cdot 10^{-3}$



$$t(p_{T,j_1}, p_{T,j_2}, \Delta\phi, \Delta\eta | f_{W\widetilde{W}} = 0) = -p_{T,j_1}\left(p_{T,j_2} + c\right)(a - b\Delta\eta)\sin(\Delta\phi + d)$$

with $a = 1.086(11)$ $b = 0.10241(19)$ $c = 0.24165(20)$ $d = 0.00662(32)$

Including detector effects

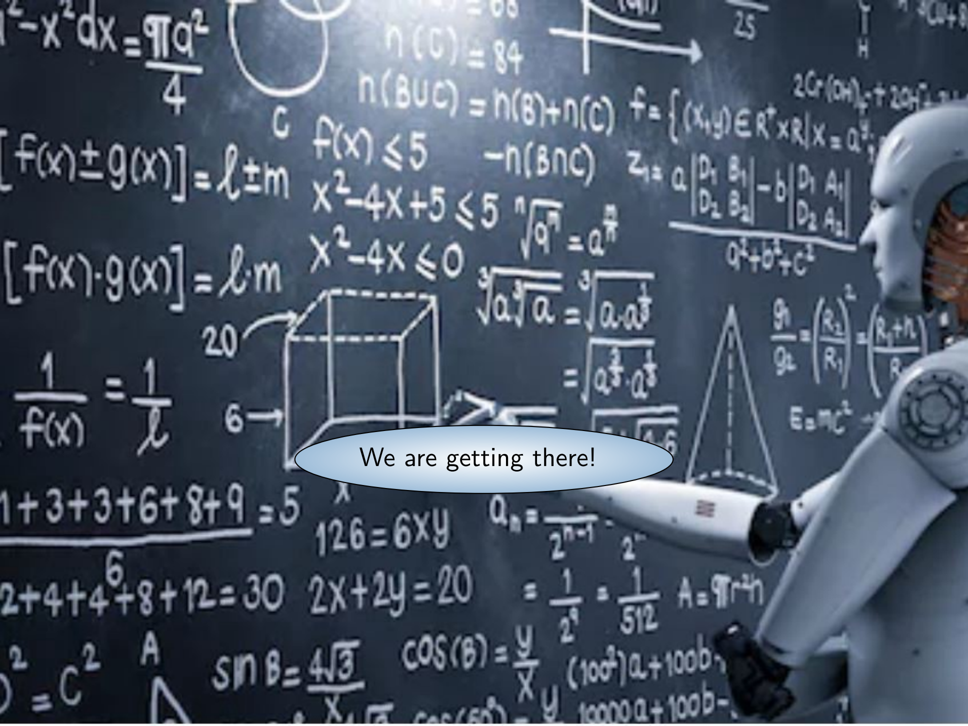


(optimal) observable	MSE all	reach $1\ \sigma$	reach $2\ \sigma$
$ap_{T1}p_{T2}$	0.1576	[-0.86,0.86]	—
$a\sin\phi$	0.0885	[-0.38,0.36]	[-0.76,0.74]
$a\sin\phi p_{T1}p_{T2}$	0.0217	[-0.28,0.28]	[-0.56,0.56]
SR complexity 16	0.0145	[-0.26,0.26]	[-0.54,0.54]
SALLY	0.0129	[-0.26,0.26]	[-0.56,0.54]
SALLY full	0.0048	[-0.26,0.26]	[-0.54,0.54]

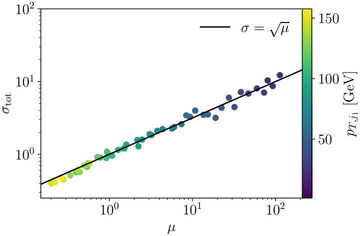
What about ML4Theory?

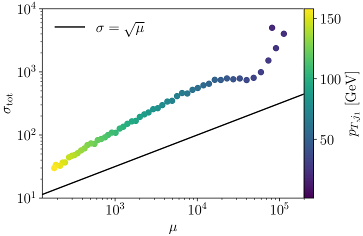
Better predictions?
→ ML for precision simulations
✓

Better understanding?
→ Turn data into theory
(✓)


We are getting there!

A closer look at the uncertainties





$$\sigma_{\text{tot}}^2 == \langle (n - \langle n \rangle)^2 \rangle = \sigma_{\text{stoch}}^2 + \sigma_{\text{pred}}^2$$

$$\sigma_{\text{stoch}}^2 = \int d\theta \; q(\theta) \left[\langle n^2 \rangle_\theta - \langle n \rangle_\theta^2 \right] = \langle n \rangle$$

$$\sigma_{\text{pred}}^2 = \int d\theta \; q(\theta) \left[\langle n \rangle_\theta - \langle n \rangle \right]^2 \; ,$$