How to GAN Event Unweighting

Mathias Backes

ITP Heidelberg

Based on arXiv:2012.07873 [hep-ph]

with Anja Butter, Tilman Plehn and Ramon Winterhalder.

Monte Carlo Event Simulation

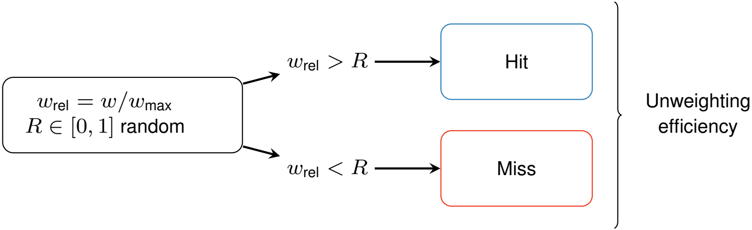
Necessity of Unweighting Procedures

- Consider integrated cross section

$$\sigma = \int \mathrm{d}x \, \frac{\mathrm{d}\sigma}{\mathrm{d}x} := \int \mathrm{d}x \, w(x)$$

- Sampling of N phase space points: weighted events with weight $w(x)$

  $\Rightarrow$ Unweighting procedures are necessary



Monte Carlo Event Simulation

Bottlenecks

- Differential cross section might vary strongly: $\langle w \rangle \ll w_{\text{max}}$

  $\Rightarrow$ Small unweighting efficiency:

$$\epsilon_{\text{uw}} = \frac{\langle w \rangle}{w_{\text{max}}}$$

$\swarrow$ $\searrow$

Monte Carlo Integration

- Phase space remapping (e.g. VEGAS)
- Neural importance sampling

  [1810.11509] Klimek and Perelstein

  [2001.05478] Bothmann et al. **NF**

  [2001.05486, 201.10028] Gao et al. **NF**

Unweighting GAN approach

- Central idea: Implement weights in training procedure
- Not a classical unweighting approach, instead "generalised GAN"

  [2011.13445] Stienen and Verheyen **AF**

Generative Adversarial Networks (GANs)

Training procedure

- Discriminator distinguishes $\{\mathbf{x}_T\}$ and $\{\mathbf{x}_G\}$ $[D(x_T) \to 1,\ D(x_G) \to 0]$

$$L_D = \langle -\log D(\mathbf{x}) \rangle_{\mathbf{x}\sim P_T} + \langle -\log(1 - D(\mathbf{x})) \rangle_{\mathbf{x}\sim P_G}$$

- Generator mimics true data $[D(x_G) \to 1]$

$$L_G = \langle -\log D(\mathbf{x}) \rangle_{\mathbf{x}\sim P_G}$$

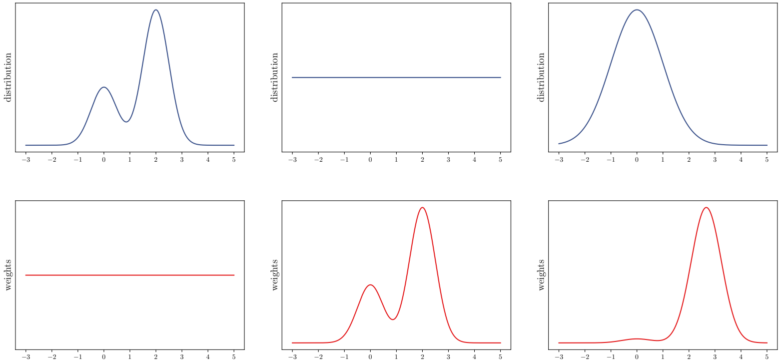
Implementing Weights

- True distribution factorizes into $P_T = Q_T \cdot w(x)$
- Redefine loss function L_D of the discriminator:

$$L_D^{(\text{uw})} = \frac{\langle -w(x)\ \log D(x) \rangle_{x\sim Q_T}}{\langle w(x) \rangle_{x\sim Q_T}} + \langle -\ \log(1 - D(x)) \rangle_{x\sim P_G}$$

Testing the Unweighting Procedure

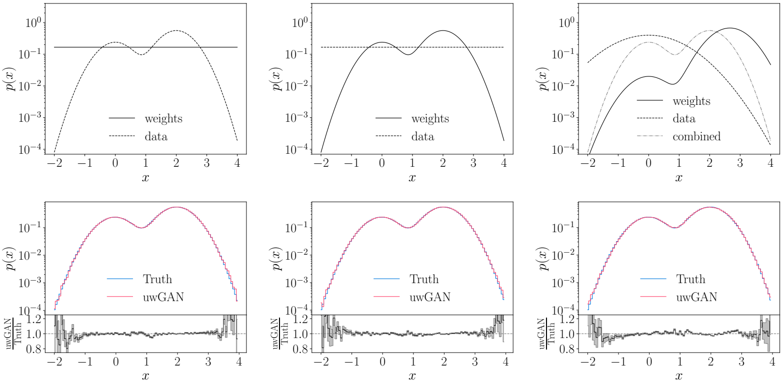
Distribution Q_T



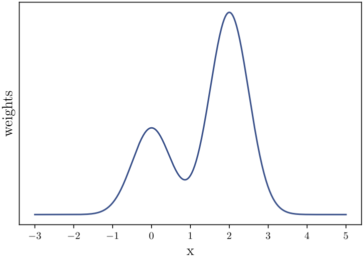
Weights $w(x)$

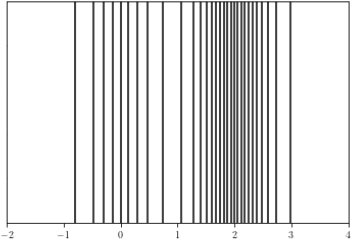
$$P_T = Q_T \cdot w(x)$$

Testing the Unweighting Procedure



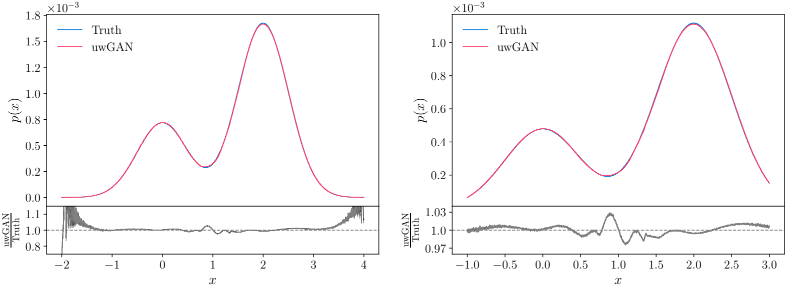
Unweighting with VEGAS





$\Rightarrow$ Grid factorizes in higher dimensions

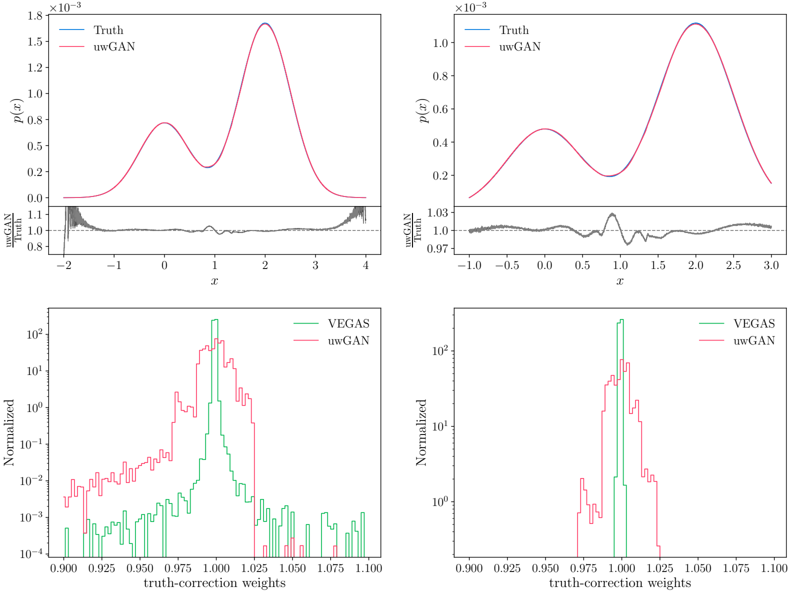
Unweighting Quality



Calculate truth-correction weights:

$$w_G(x) = \frac{P_T(x)}{P_G(x)}$$

Unweighting Quality

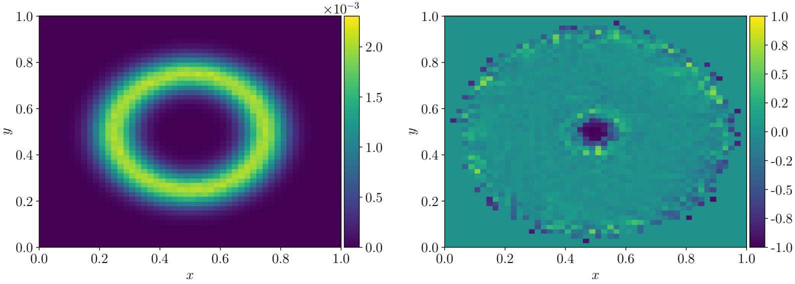


Outperforming VEGAS in Higher Dimensions

Two Dimensional Toy Model

- Circle-shaped weighted input data:

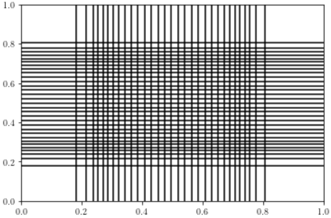
$$P_{\text{circle}}(x,y) = N \exp\left[-\frac{1}{2\sigma^2}\left(\sqrt{(x-x_0)^2+(y-y_0)^2}-r_0\right)^2\right]$$

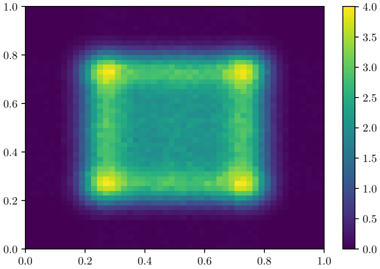


Outperforming VEGAS in Higher Dimensions

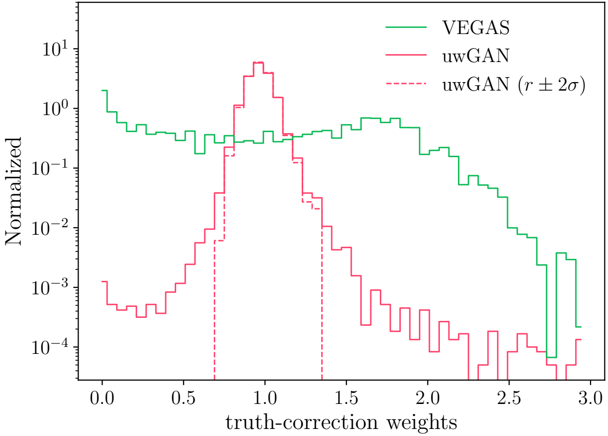
VEGAS Unweighting

- Algorithm restricted to cartesian coordinates
- Impact of the applied grid is visible





Outperforming VEGAS in Higher Dimensions

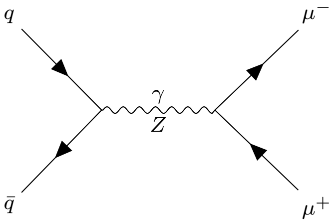


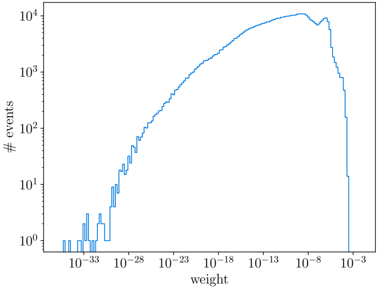
Unweighting Drell-Yan

Drell-Yan Process

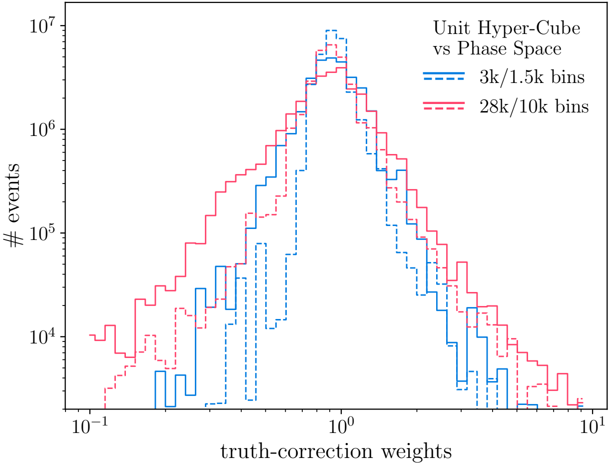
$$p\,p \to \mu^+\mu^-$$

- Four degrees of freedom for outgoing muons: $p_T, p_{z_1}, p_{z_2}, \phi$

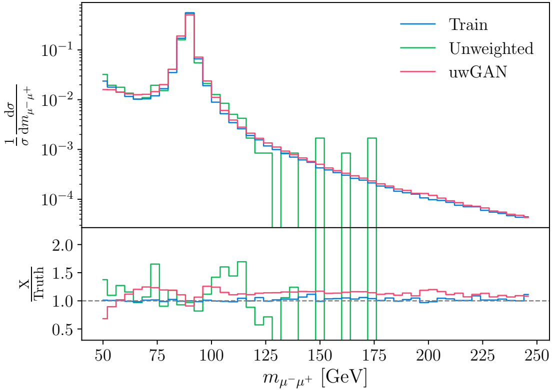




Quality of Drell-Yan Unweighting



Unweighting Drell-Yan



- Higher statistics for uwGAN than for classical Hit-or-Miss (especially for high $m_{\mu^-\mu^+}$)

  $\Rightarrow$ Expandable to more complex physical applications

Conclusion

- More efficient way to perform unweighting procedures
- Quality can be estimated by the truth-correction weights

$\Rightarrow$ Improved GAN-framework to consider weights in training processes

Thank you for your attention!

Additional Material

Generative Adversarial Networks (GANs)

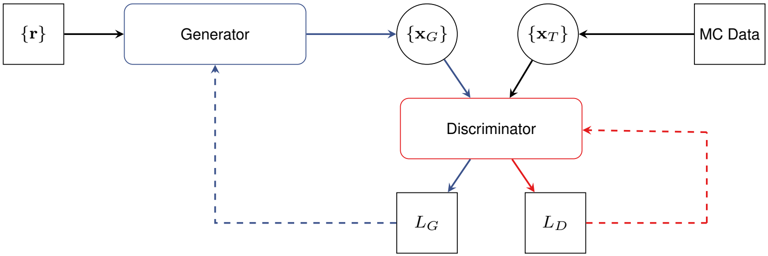
Training procedure

- Discriminator distinguishes $\{\mathbf{x}_T\}$ and $\{\mathbf{x}_G\}$ $[D(x_T) \to 1,\, D(x_G) \to 0]$

$$L_D = \langle -\log D(\mathbf{x}) \rangle_{\mathbf{x}\sim P_T} + \langle -\log(1 - D(\mathbf{x})) \rangle_{\mathbf{x}\sim P_G}$$

- Generator mimics true data $[D(x_G) \to 1]$

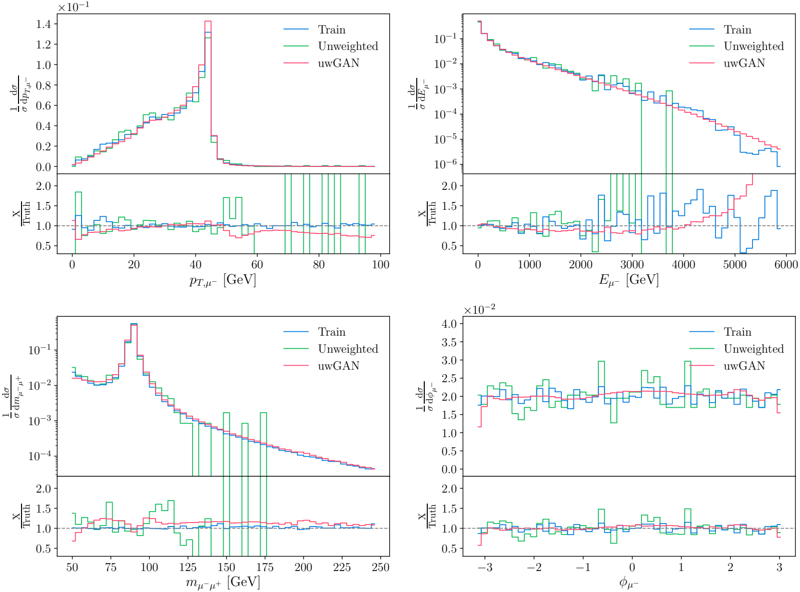
$$L_G = \langle -\log D(\mathbf{x}) \rangle_{\mathbf{x}\sim P_G}$$



Network Architecture for Drell-Yan

Parameter	Value
Layers	6
Kernel initializer	He uniform
G units per layer	414
D units per layer	187
G activation function	ReLU
D activation function	leaky ReLU
D updates per G	2
λ_{wMMD}	2.37
Learning rate	0.0074
Decay	0.42
Batch size	1265
Epochs	500
Iterations per epoch	200

Unweighting Drell-Yan



Drell-Yan Sampling

Sampling weighted events according to

$$\sigma = \int \frac{\mathrm{d}x_1}{x_1} \int \frac{\mathrm{d}x_2}{x_2} \sum_{a,b} x_1 f_a(x_1) x_2 f_b(x_2)\, \hat{\sigma}_{ab}(x_1, x_2, s)$$

Using $\tau = x_1 x_2$ we get:

$$\sigma = 2 \log \tau_{\text{min}} \int_0^1 dr_1\, r_1 \int_0^1 dr_2\, \sum_{a,b} x_1 f_a(x_1)\, x_2 f_b(x_2)\, \hat{\sigma}_{ab}(x_1 x_2 s)$$

With an additional random number $r_3 = (\cos\theta + 1)/2$ we can parametrize the 4-dimensional phase space as

$$\begin{aligned} p_T &= 2E_{\text{beam}}\, \tau_{\text{min}}^{r_1/2} \sqrt{r_3(1-r_3)} \\ p_{z_1} &= E_{\text{beam}}\, \left(\tau_{\text{min}}^{r_1 r_2} r_3 + \tau_{\text{min}}^{r_1(1-r_2)}(r_3 - 1)\right) \\ p_{z_2} &= E_{\text{beam}}\, \left(\tau_{\text{min}}^{r_1 r_2}(1-r_3) - \tau_{\text{min}}^{r_1(1-r_2)} r_3\right) \\ \phi &= 2\pi r_4 \end{aligned}$$