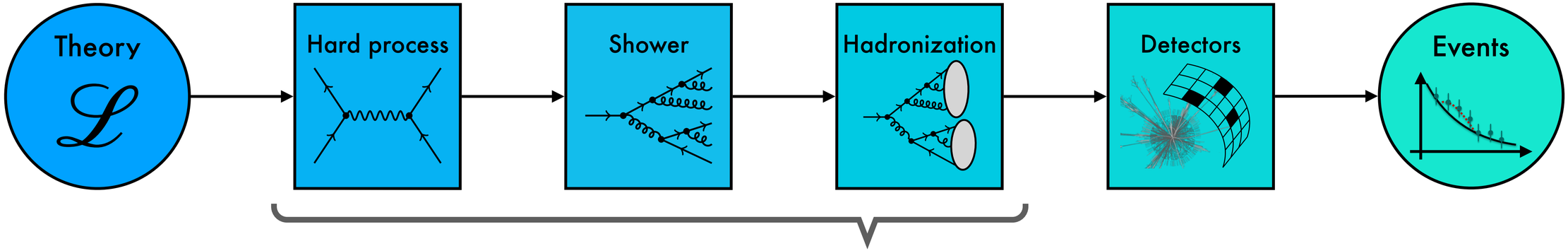
Machine Learning for Event Generation

Making our generators ready for the era of the High-Luminosity LHC

Theo Heimel
CP3, UCLouvain
September 2026

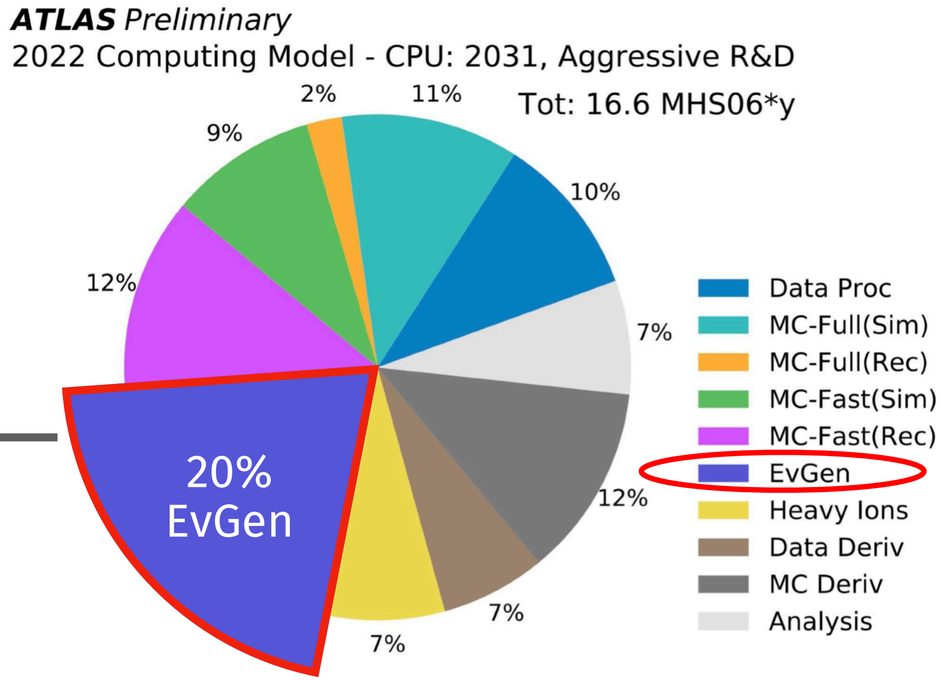
Introduction



Event generation is a major part of the computational cost

↓

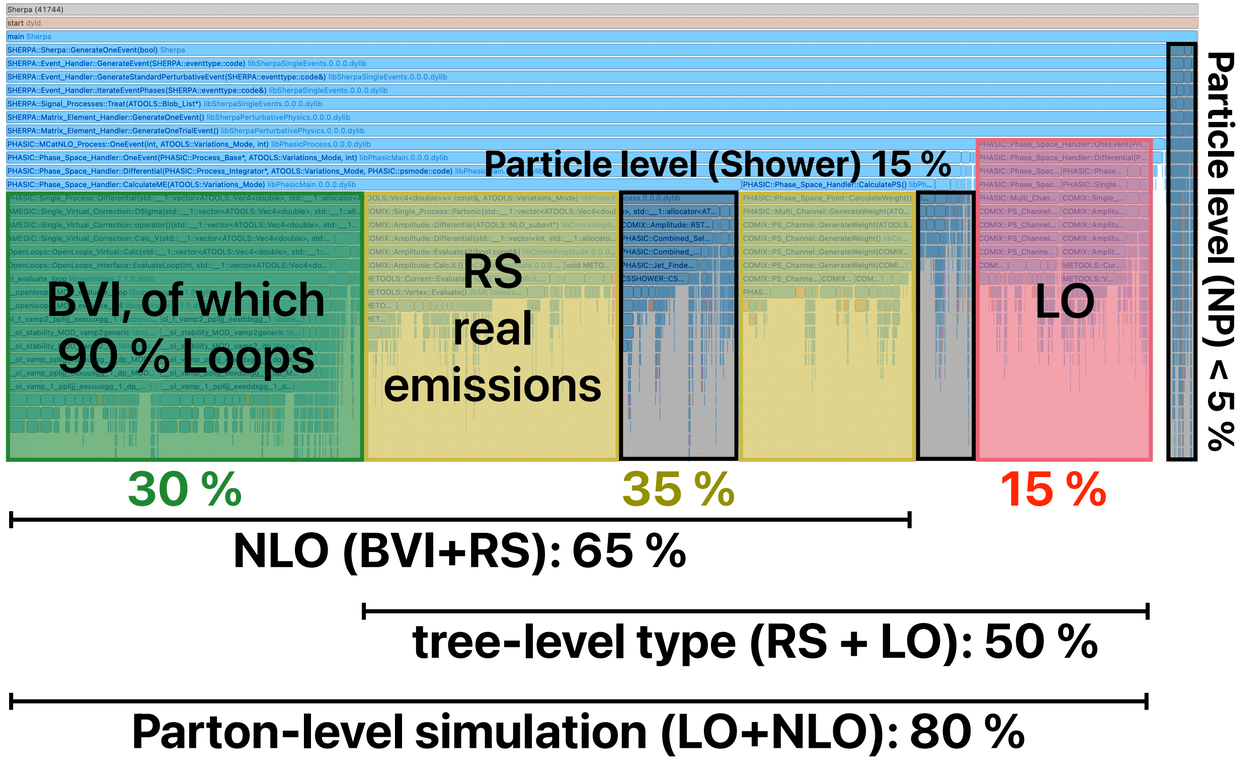
How can we prevent it from becoming a bottleneck at the High-Luminosity LHC?



[CERN-LHCC-2022-005]

Computational cost

Flame graph of Sherpa in ATLAS Drell-Yan setup (0-2j @ NLO, >2j @ LO)



[Figure by Enrico Bothmann]

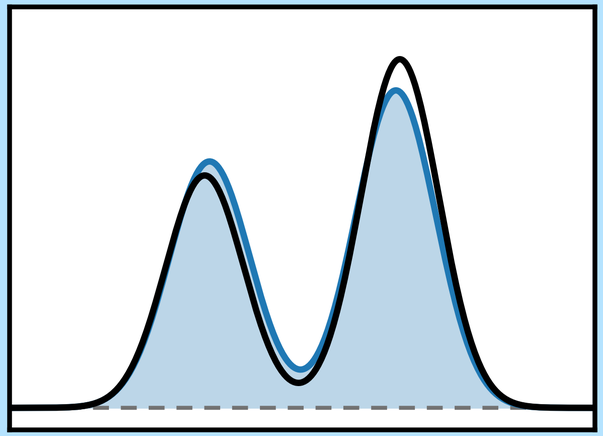
- Most expensive: parton-level
  → focus of this talk
- Standard model backgrounds
  → large rates
  → large multiplicities
  → NLO precision necessary
- Signals
  → lower rates
  → LO often sufficient
  → many different processes

Leading-order event generation with ML

$$\sigma_{\mathrm{LO}} = \int \mathrm{d}\Phi_n \, B$$

Neural importance sampling

- Use **normalizing flows or CFMs** → density available
- Unweighting step → **unbiased**
- Better model = faster sampling



Tree-level amplitudes

- **Hardware acceleration**
  → move to GPU
  → CPU vector instructions
- Optional:
  use **ML surrogates**

[2001.05486, 2001.05478, 2001.10028, 2212.06172, 2311.01548]

Event generators

Two projects for production-ready ML- and hardware-accelerated LHC event generation

MadGraph7
Make end-to-end GPU- and ML-accelerated workflows available for all processes supported by MG5

PEPPER
Complementary to Sherpa with focus on most expensive Standard Model background processes

Both focus on LO for now, NLO will follow

Challenges of ML event generation

Training data

- Don’t want to use legacy generators
- Need to “bootstrap” our own training data

Sampling speed

- Matrix element evaluations have become very fast
- Rejecting more samples with cheaper model might be faster

Variety of processes

- Large number of different processes and settings
  → signals and backgrounds
  → model parameters
  → cuts
  → …

Many trade-offs to consider

fast training vs convergence

expressivity vs throughput

surrogates vs hardware acceleration

pre-training vs training from scratch

Event generation @ ML4Jets 2026

Agentic tools

Neural many-jet generation

Extrapolation & pre-training

Towards deployment

Amplitude surrogates

NLO generation & negative weight reduction

The NNLO frontier

number of jets

order in perturbation theory

Event generation in MadGraph

Calculate (differential) cross sections

$$\mathrm{d}\sigma = \frac{1}{\text{flux}} \mathrm{d}x_a \mathrm{d}x_b \, f(x_a) f(x_b) \, \mathrm{d}\Phi_n \left\langle \, |M_{\lambda,c,\ldots}(p_a, p_b \mid p_1, \ldots, p_n)|^2 \, \right\rangle$$

Sum over channels

MadGraph: build channels from Feynman diagrams

Integrand

MadGraph: $\mathrm{d}\sigma/\mathrm{d}x$

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

Channel weights

MadGraph: $\alpha_i^{\mathrm{MG}}(x) \sim |M_i|^2$

Channel mappings

MadGraph: use amplitude structure, ...

Analytic mappings + refine with **VEGAS** (factorized, histogram based importance sampling)

Event generation in MadNIS

Calculate (differential) cross sections

$$\mathrm{d}\sigma = \frac{1}{\text{flux}} \mathrm{d}x_a \mathrm{d}x_b \, f(x_a) f(x_b) \, \mathrm{d}\Phi_n \left\langle \, |M_{\lambda,c,\ldots}(p_a, p_b \mid p_1, \ldots, p_n)|^2 \, \right\rangle$$

MadNIS

Sum over channels

MadGraph: build channels from Feynman diagrams

Integrand

MadGraph: $\mathrm{d}\sigma/\mathrm{d}x$

$$I = \sum_i \left\langle \alpha_i^\xi(x) \frac{f(x)}{p_i^\omega(x)} \right\rangle_{x \sim p_i^\omega(x)}$$

Learned channel weights

MadGraph: $\alpha_i^{\mathrm{MG}}(x) \sim |M_i|^2$

$$\alpha_i(x) \to \alpha_i^\xi(x) = \alpha_i^{\mathrm{MG}}(x) \cdot K_i^\xi(x)$$

parametrize with NN

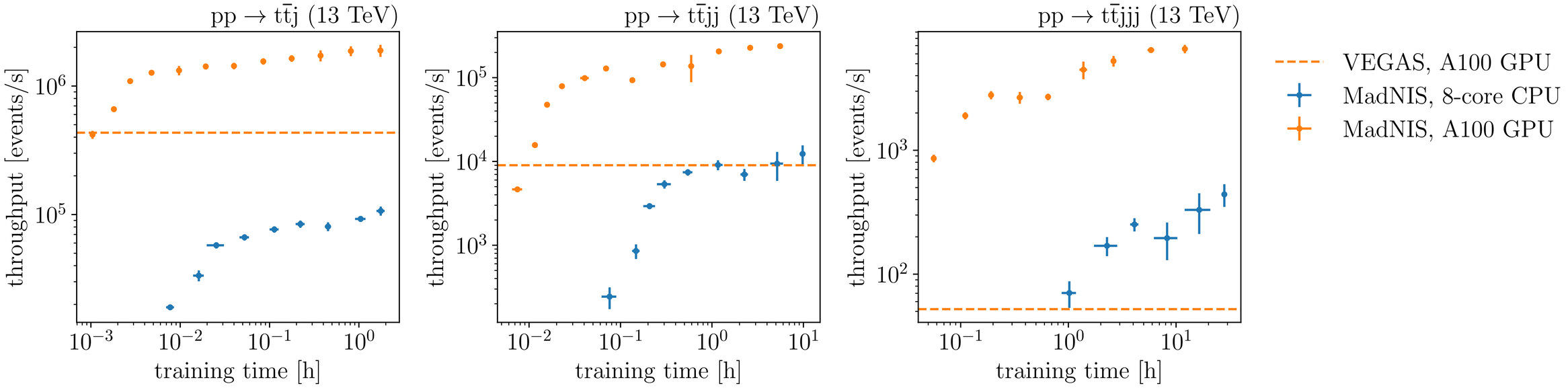
Learned channel mappings

MadGraph: use amplitude structure, ...

Analytic mappings + ~~refine with VEGAS~~

refine with Normalizing Flow

MadNIS performance



(preliminary)

- Large speed-up compared to VEGAS baseline already after a few minutes of training, **10x to 100x speed-up** for long training
- Upcoming publication with much larger range of LHC processes

[2610.xxxxx] TH, Massaro, Mattelaer, Winterhalder

Upcoming MadGraph7 release

MadMatrix

- Matrix element on GPU
- improved CPU performance from SIMD vectorization

[2507.21039]

MadNIS

- neural importance sampling for better unweighting efficiency
- automated defaults: as easy to use as VEGAS
- used automatically for many processes

[2311.01548, 2603.22407]

AmpliCol

- implement ME using recursion relations to enable higher multiplicities
- made efficient using leading- to full-color reweighting

[2409.12128, 2601.19483]

MadSpace

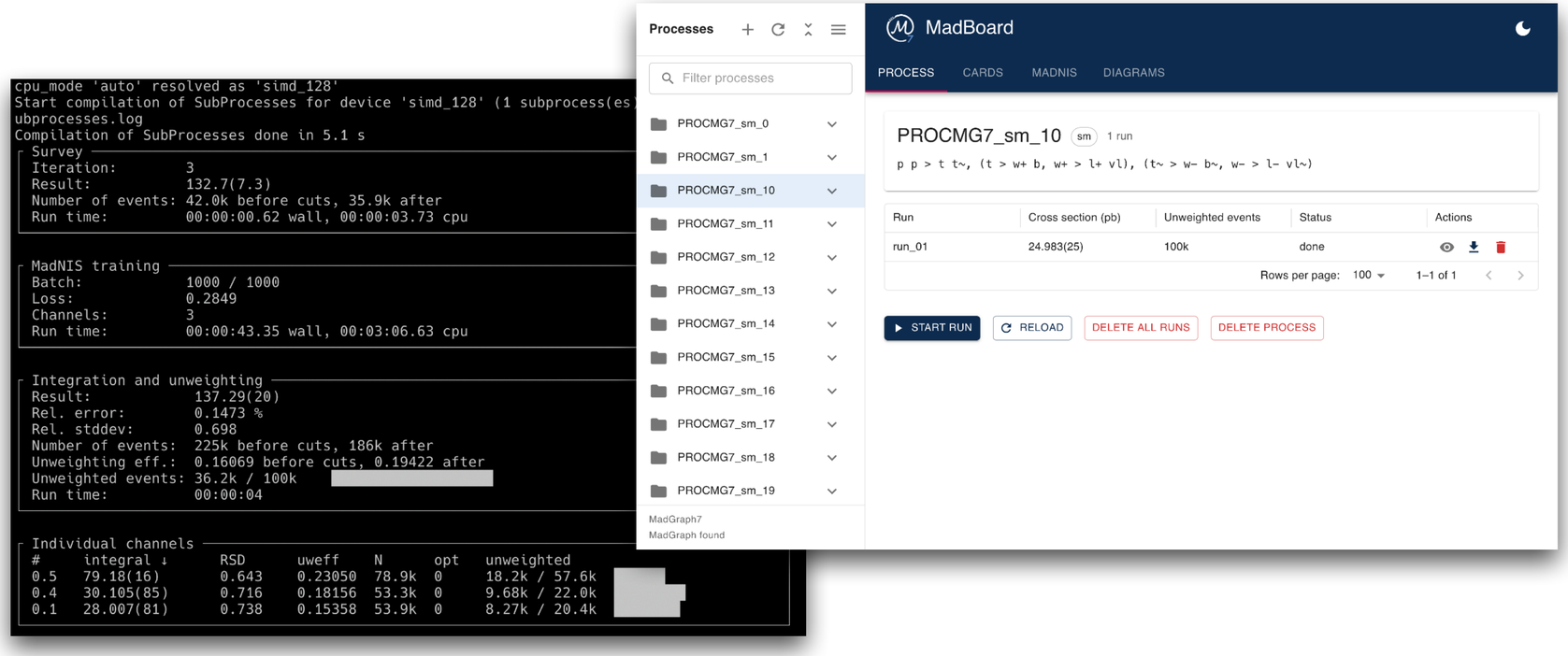
- new flexible and modular phase space generator
- improved throughput
- GPU- and ML-enabled
- usable beyond MadGraph

[2602.06895]

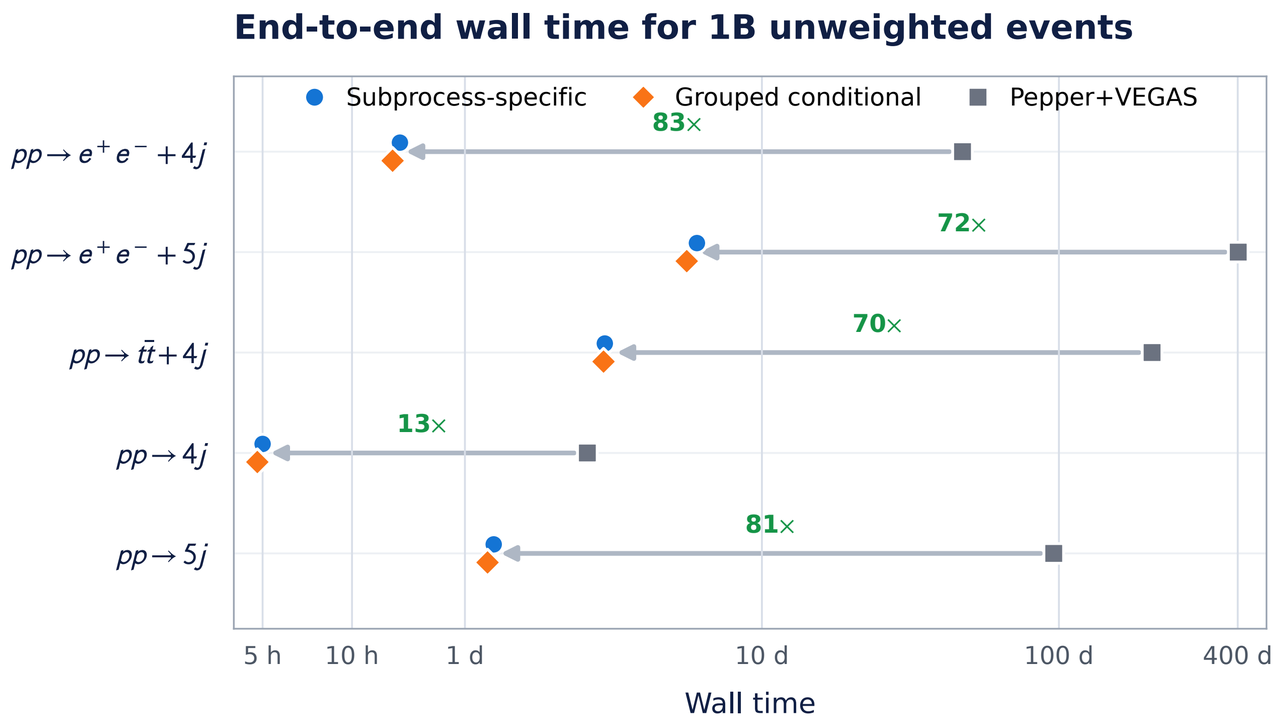
MadGraph7 alpha release

Alpha release of MadGraph7 out now!

https://github.com/MadGraphTeam/MadGraph7



NIS @ high multiplicity



- Neural importance sampling with normalizing flows in PEPPER
- **speed-ups of 10x to 100x**
- End-to-end GPU setup

- Earlier work also with flow matching: high expressivity, but expensive sampling [2506.06203]
  → for expensive integrands

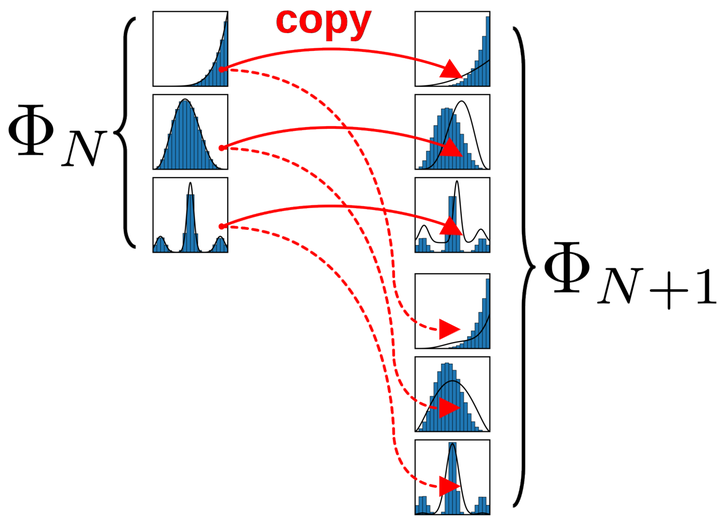
[2608.21338] Bothmann, Isaacson, Krause, López-Zurita, Spannring, Wang

→ Daohan Wang's talk

Avoid training from scratch

Too good to go

[2608.27104] Helms, Janßen, Schumann

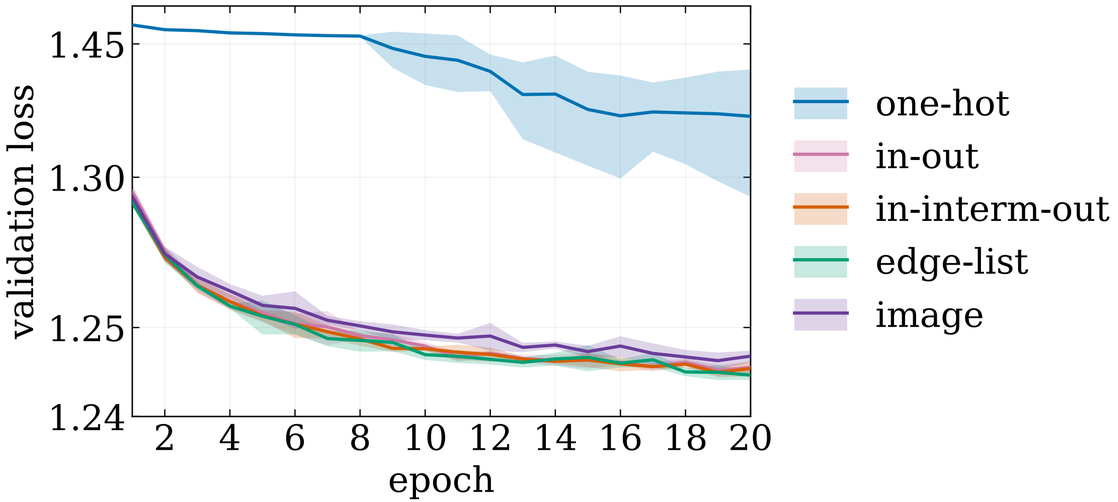


- Cost increases towards more jets
  → reuse results from cheaper trainings
- Initialize (N+1)-particle network based on N-particle network

→ Konrad Helms' talk

LLM conditioning

[2606.23791] Bahl, Plehn, Schiller, Sivagnanalingam



- LLM pre-trained on various SM processes
- Feynman diagrams as input
  → learn physics, not just distribution

→ Thanush Sivagnanalingam' talk

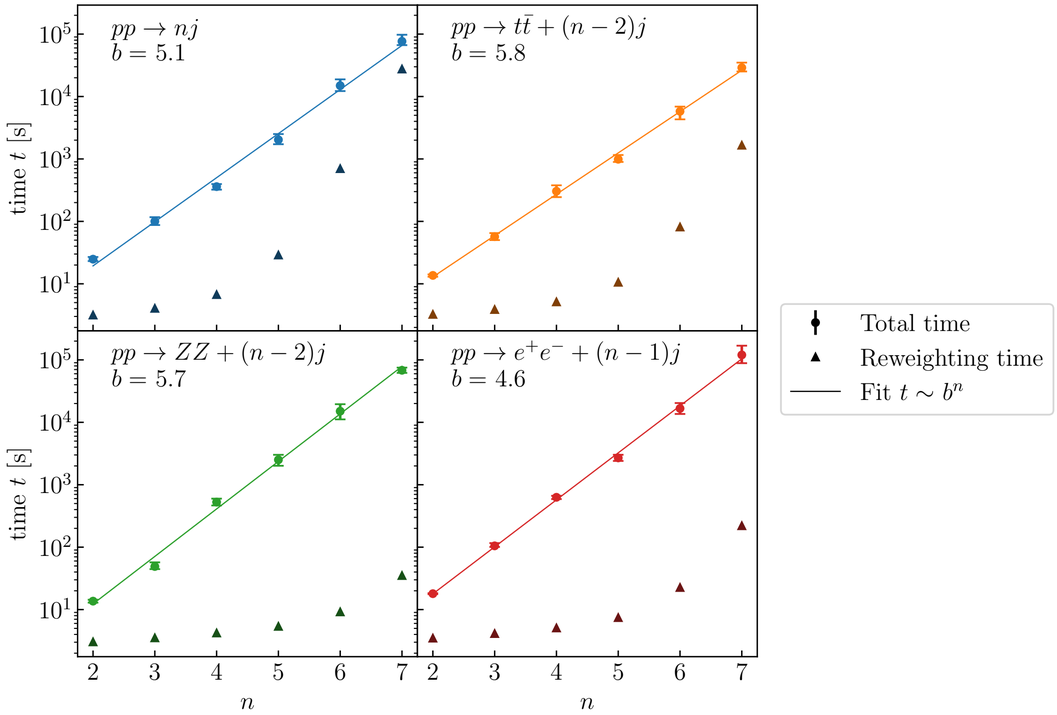
Surrogates: double-stage unweighting

- Generate unweighted events using surrogate
  → many rejected samples, but fast
- Evaluate truth and unweight again
  → less rejected samples
  → **unbiased result**
- Largest gains for high multiplicities

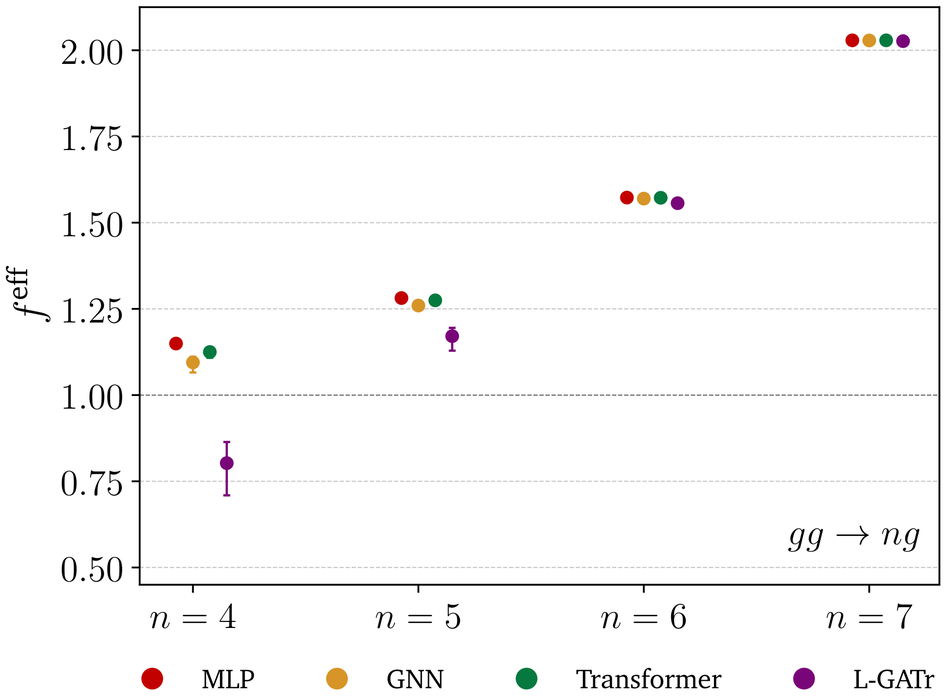
Process	$T^{[\text{approach}]}_{\text{merged process}}$ [MHS23y]			Ratio
	Baseline_Σ	Baseline	Surrogate	
$Z + \leq 0$ jets	0.04	0.04	-	
$Z + \leq 1$ jets	0.11	0.11	-	
$Z + \leq 2$ jets	0.32	0.34	-	
$Z + \leq 3$ jets	0.67	1.23	-	
$Z + \leq 4$ jets	1.60	5.44	-	
$Z + \leq 5$ jets	16.09	47.83	4.91	3.3
$Z + \leq 6$ jets	297.73	130.51	11.78	11.1

[2603.22407] Herrmann, Janßen, Schenker, Schumann, Siegert
[2109.11964] Danziger, Janßen, Schumann, Siegert

Surrogates: leading-color reweighting



- AmpliCol: leading-color approximation as fast physics-based surrogate [2601.19483]



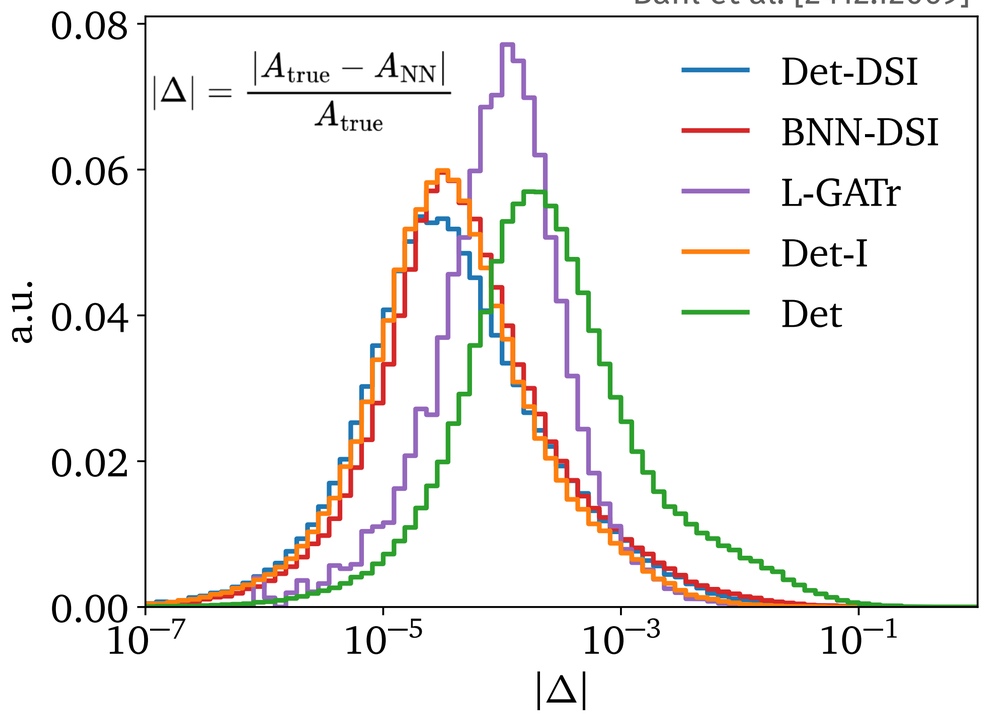
- ML to speed up reweighting to full color
- Gains of up to **factor 2** for large n

[2509.07068] Villadamigo, Frederix, Plehn, Vitos, Winterhalder

→ Javier Villadamigo's talk

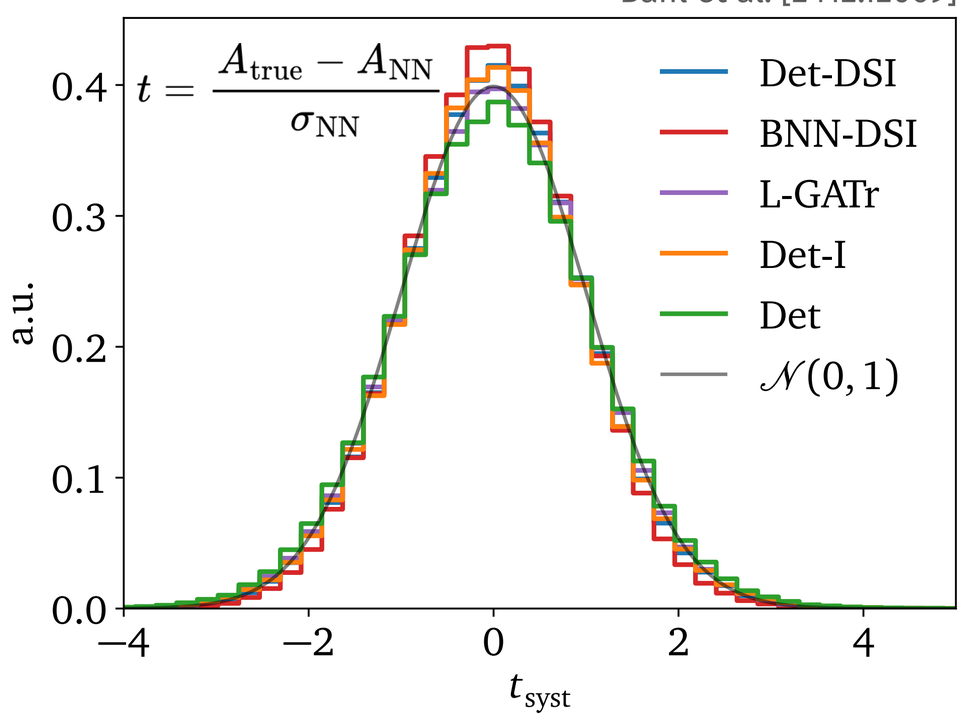
Surrogates: learned uncertainties

Bahl et al. [2412.12069]



- Symmetry- and physics-aware networks perform better
- Uncertainties small compared to theory uncertainties like scale, PDF, …

Bahl et al. [2412.12069]



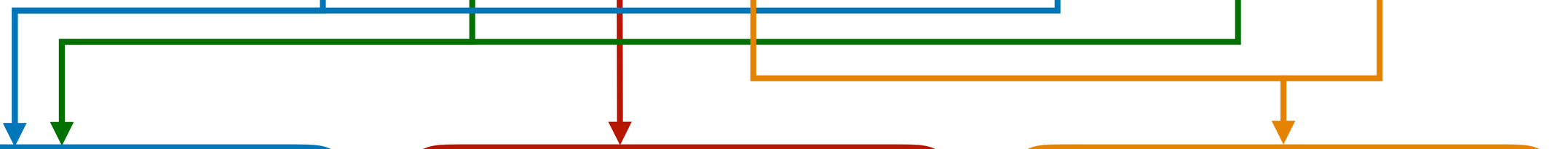
- Well-calibrated uncertainties
- Learned uncertainty also useful to catch low-accuracy events

[2002.07516, 2106.09474, 2206.14831, 2509.00155, 2512.11036, 2601.00950, 2601.13308, 2607.01354, 2608.06450]

→ Talks by Suprio Dubey, Rebecca Revelli, Joaquin Ramirez

Next-to-leading order integration

$$\sigma_{\mathrm{NLO}} = \int \mathrm{d}\Phi_n \, (B + V + I) + \int \mathrm{d}\Phi_{n+1} \, (R - S)$$



Importance sampling & tree-level amplitudes

Similar to leading order:

- Neural importance sampling
- Hardware acceleration

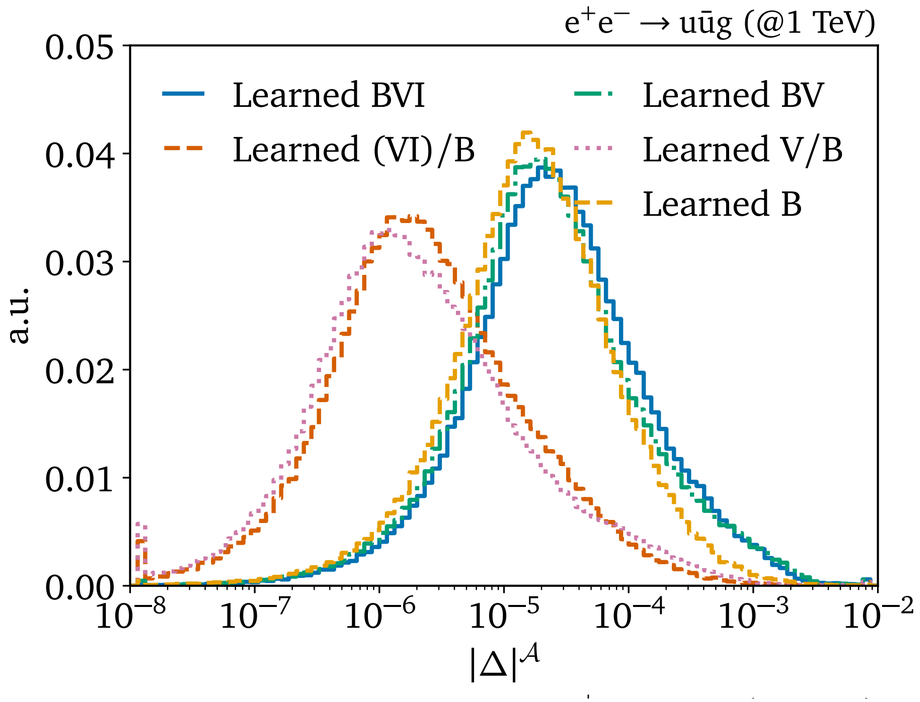
Loop amplitudes

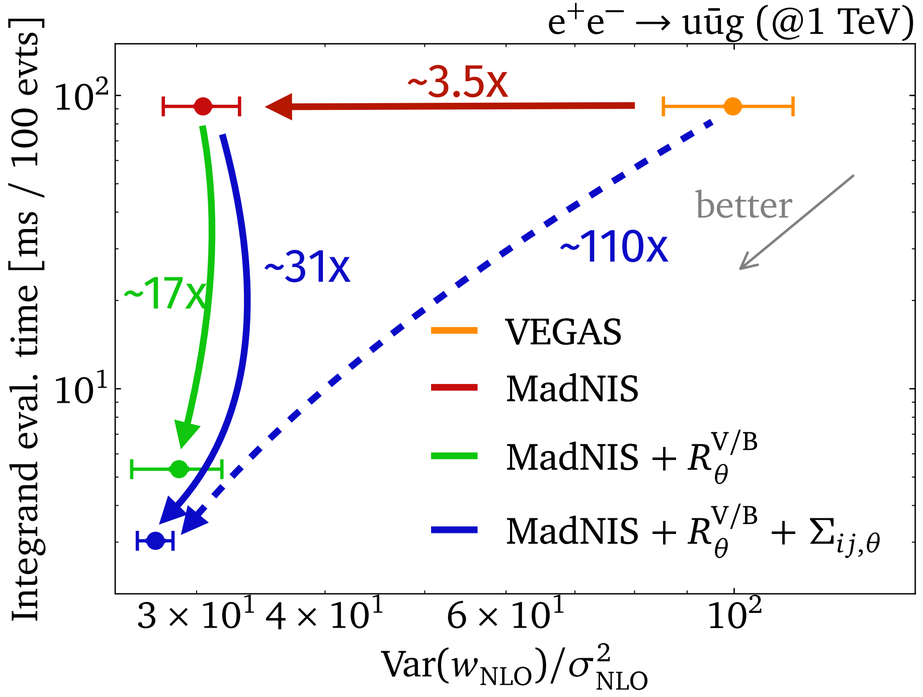
- Most expensive part
- Often need quadruple precision → bad for GPUs
- Speed up with **surrogates**

Subtraction scheme

- Causes **negative weights** → large cost downstream
- Some freedom in how to define it → improve with ML

MadNIS @ NLO



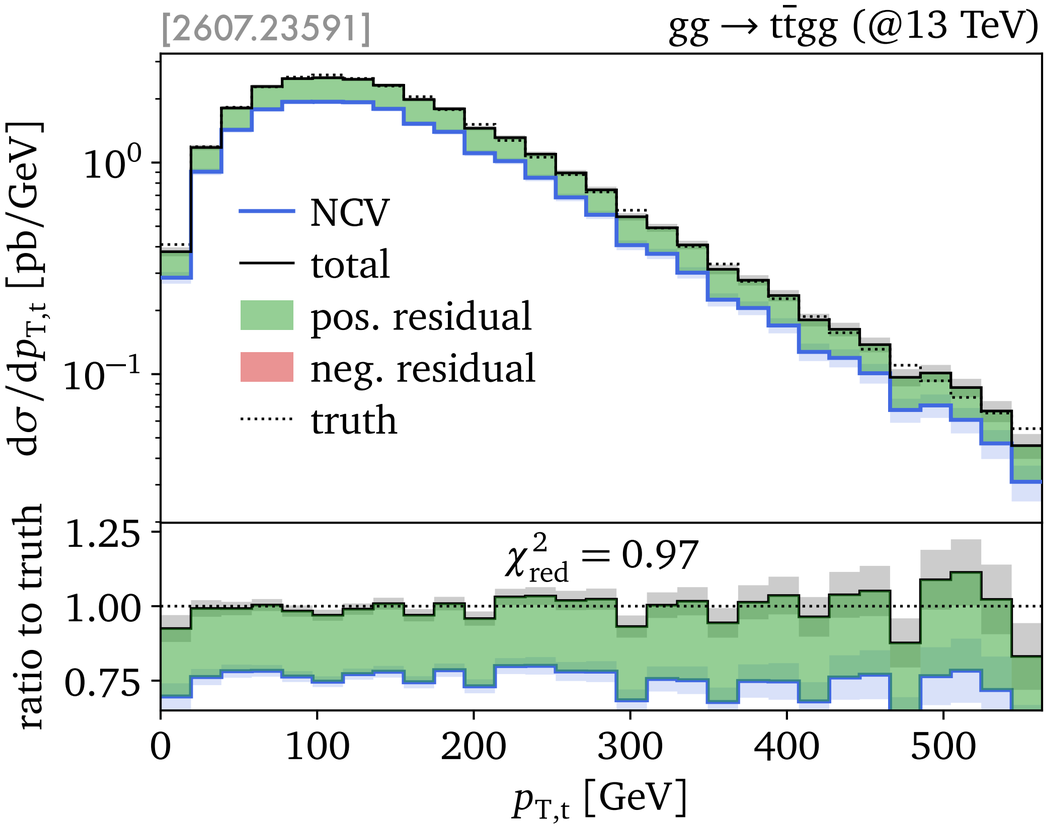


- Variance reduction from MadNIS → less samples for given precision
- Replace ME evaluation with surrogate → less time per sample
- Virtual contribution learned accurately, best for ratio virtual / Born

[2603.22407] De Crescenzo, Mariño Villadamigo, Elmer, TH, Plehn, Winterhalder, Zaro

→ Giovanni de Creszenzo's talk

Neural Control Variates @ LO



Network with known integral (NF)
× learned constant

$$I = \left\langle f(x) - C_\theta\, p_\theta(x) \right\rangle_{x\sim\text{unif}} + \left\langle C_\theta p_\theta(x) \right\rangle_{x\sim\text{unif}}$$

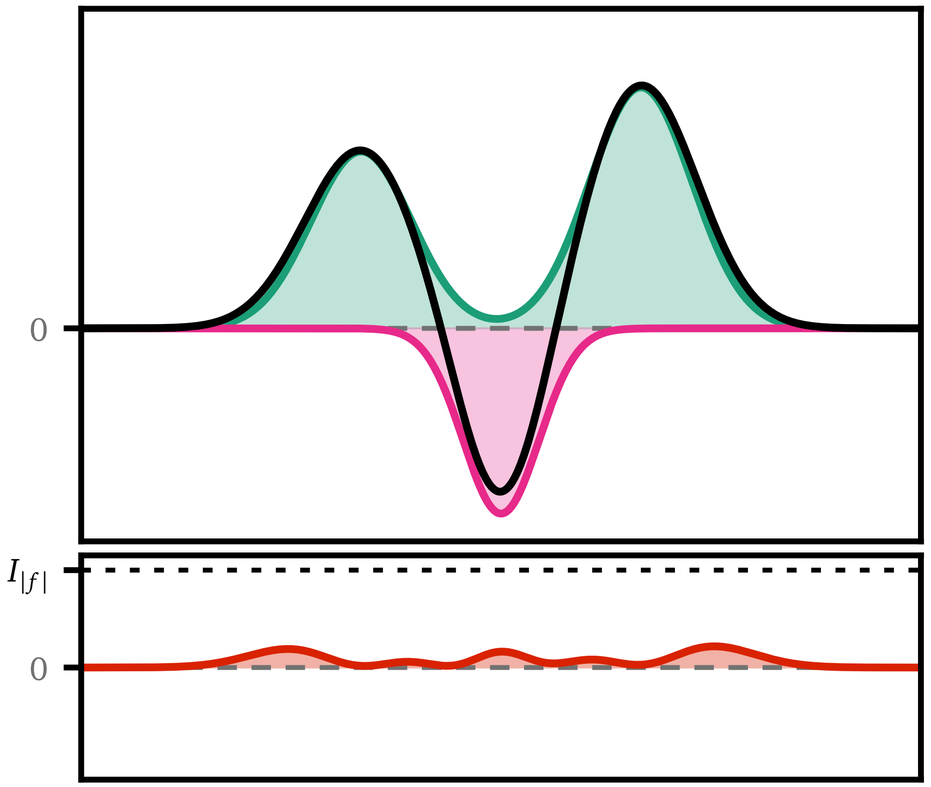
add MadNIS
for res. integral

$$I = \left\langle \frac{f(x) - C_\theta\, p_\theta(x)}{g_\phi(x)} \right\rangle_{x\sim g_\phi} + \left\langle C_\theta \right\rangle_{x\sim p_\theta}$$

100% unweighting eff. by construction!

- Unweighting efficiency: 29% (NIS) to 78% (NIS+NCV)
  → on top of 20× speedup compared to MG5 baseline
- Need penalty loss to avoid negative weights

Neural Control Variates @ NLO



$$I = \left\langle f(x) - c_\theta(x) \right\rangle_{x\sim\text{unif}} + \int \mathrm{d}x\, c_\theta(x)$$

Parametrize *signed-NCV* with
1 positive NCV + 1 negative NCV

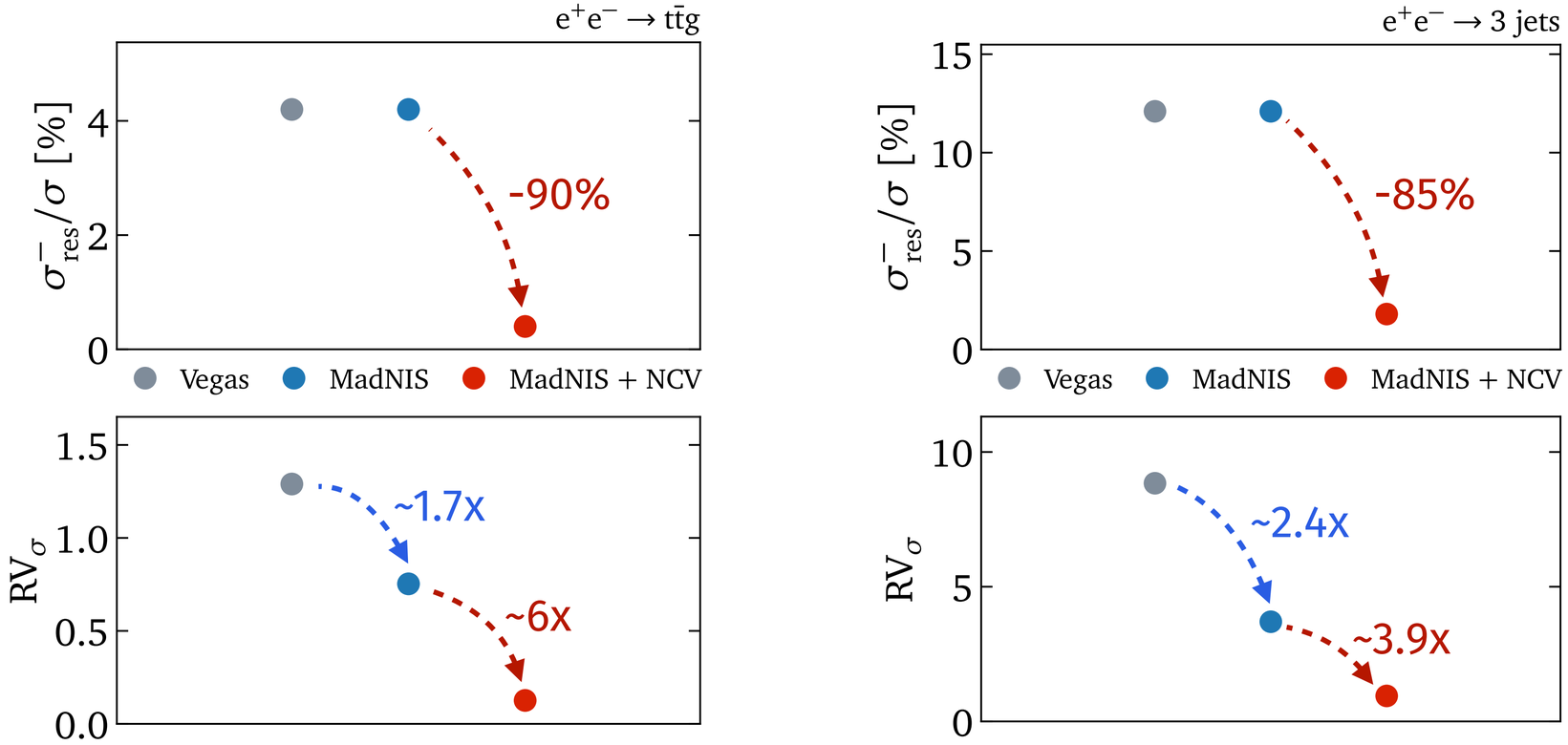
$$I = \left\langle f(x) - C_\theta^+\, p_\theta^+(x) + C_\theta^-\, p_\theta^-(x) \right\rangle_{x\sim\text{unif}} + \left\langle C_\theta^+ \right\rangle_{x\sim p_\theta^+} - \left\langle C_\theta^- \right\rangle_{x\sim p_\theta^-}$$

add MadNIS
for res. integral

$$I = \left\langle \frac{f(x) - C_\theta^+\, p_\theta^+(x) + C_\theta^-\, p_\theta^-(x)}{g_\phi(x)} \right\rangle_{x\sim g_\phi} + \left\langle C_\theta^+ \right\rangle_{x\sim p_\theta^+} - \left\langle C_\theta^- \right\rangle_{x\sim p_\theta^-}$$

[2607.23591] TH, Plehn, Revelli, Vent, Winterhalder

Neural Control Variates @ NLO

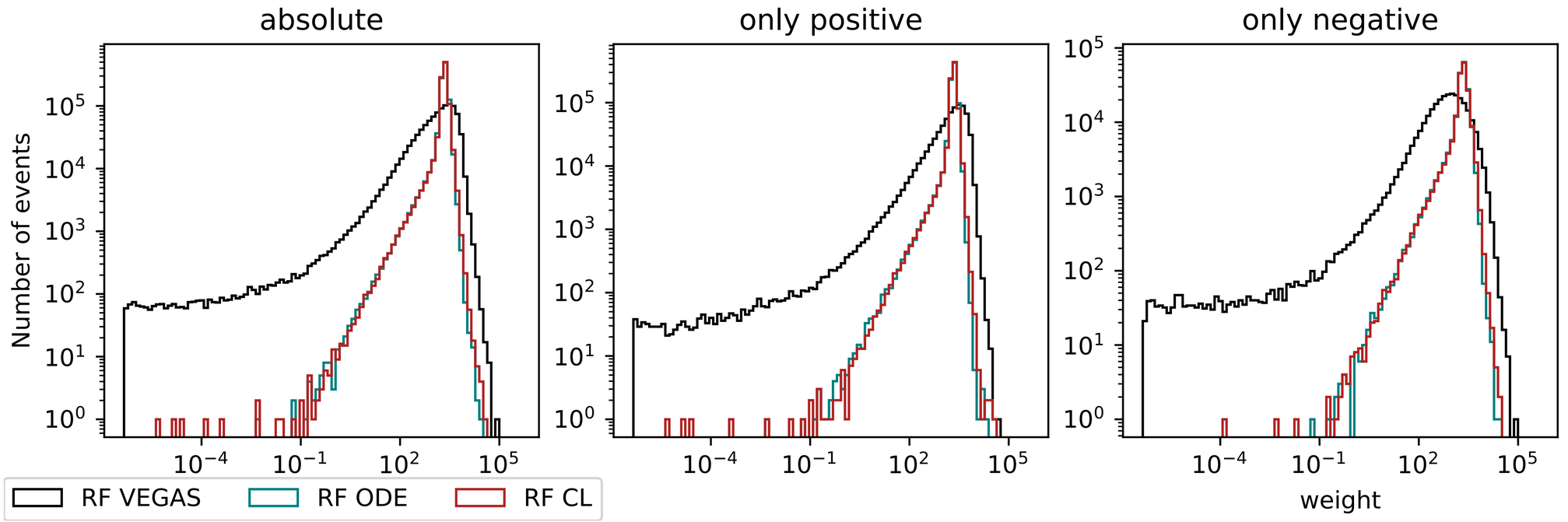


- Better variance reduction than NIS alone
- Loss in effective statistics from sign change: $N_{\text{eff}} = N(1 - 2N_{\text{neg}}/N)^2$
  → save compute in downstream tasks

[2607.23591] TH, Plehn, Revelli, Vent, Winterhalder

→ Sophia Vent's talk

Neural Importance Sampling @ NNLO

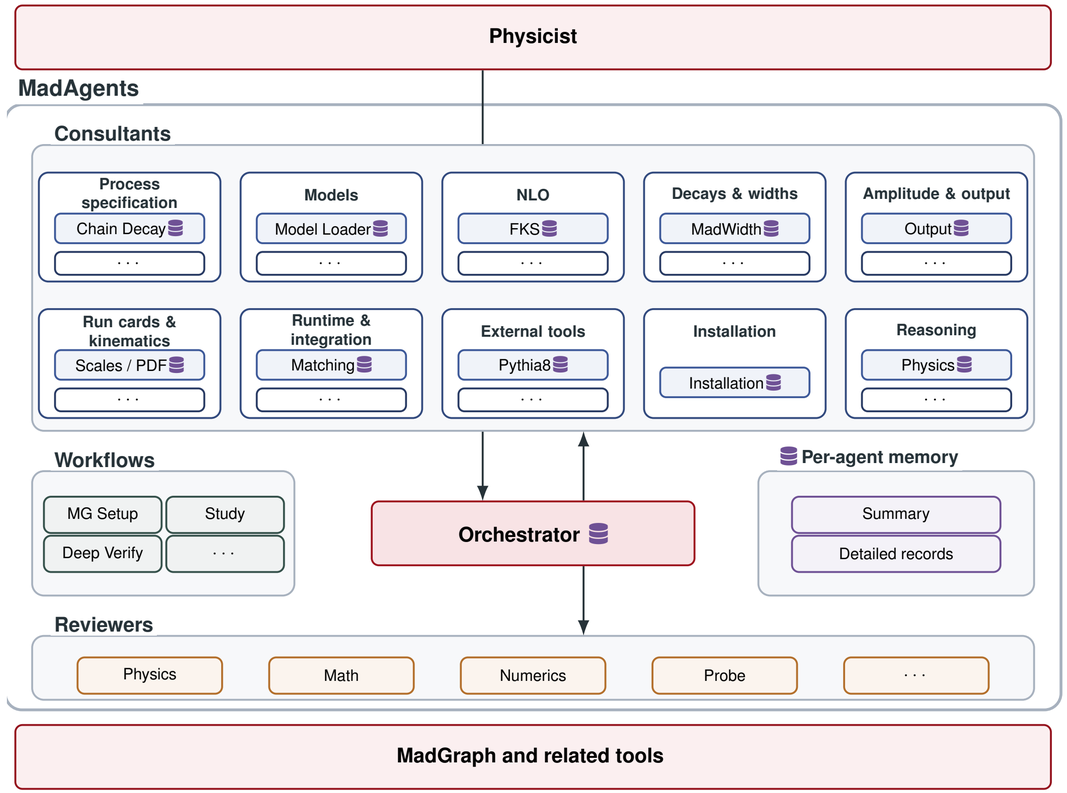


- Neural importance sampling with flow matching
- Tackles negative weights by splitting in positive and negative integrands
  → more efficient than common sampler
- Speed-up by factor 8 compared to STRIPPER baseline

[2505.13608] Janßen, Poncelet, Schumann

→ Timo Janßen's talk

MadAgents: agentic interfaces



- Event generators are complex tools
  → many options unknown even to experts
  → ML+GPU workflows add even more
- Build agentic interfaces specialized for event generation
  → teach beginners
  → help experts
  → reproduce results
  → automate global fits

[2601.21015] Plehn, Schiller, Schmal

→ Daniel Schiller's talk

Outlook

- Neural importance sampling for leading order event-generation is (close to) **production-ready**
  → **speed-ups up to 100x**, on top of hardware acceleration
- Methods developed for LO also **succeed at NLO and NNLO**
- ML solves problems exclusive to higher orders
  → negative weights
  → expensive loop amplitudes
- Agentic interfaces make event generators easier to use

ML will be an integral part of event generation at the High-Lumi LHC and beyond!