TECHNISCHE UNIVERSITÄT DRESDEN

Efficiency improvements in Monte Carlo algorithms for high-multiplicity processes

Katharina Danziger

Milano, 07.01.2020

DRESDEN concept
Exzellenz aus Wissenschaft und Kultur

Machine Learning based unweighting

Basic Idea

- speed up unweighting by using ML model to predict event weight

  → use “ML based unweighting“

  → save time to explicitly calculate ME and PS

- for 2 → n process

  4-momenta: 4x(2+n) input numbers (features)

Machine Learning Based Unweighting

Classic unweighting

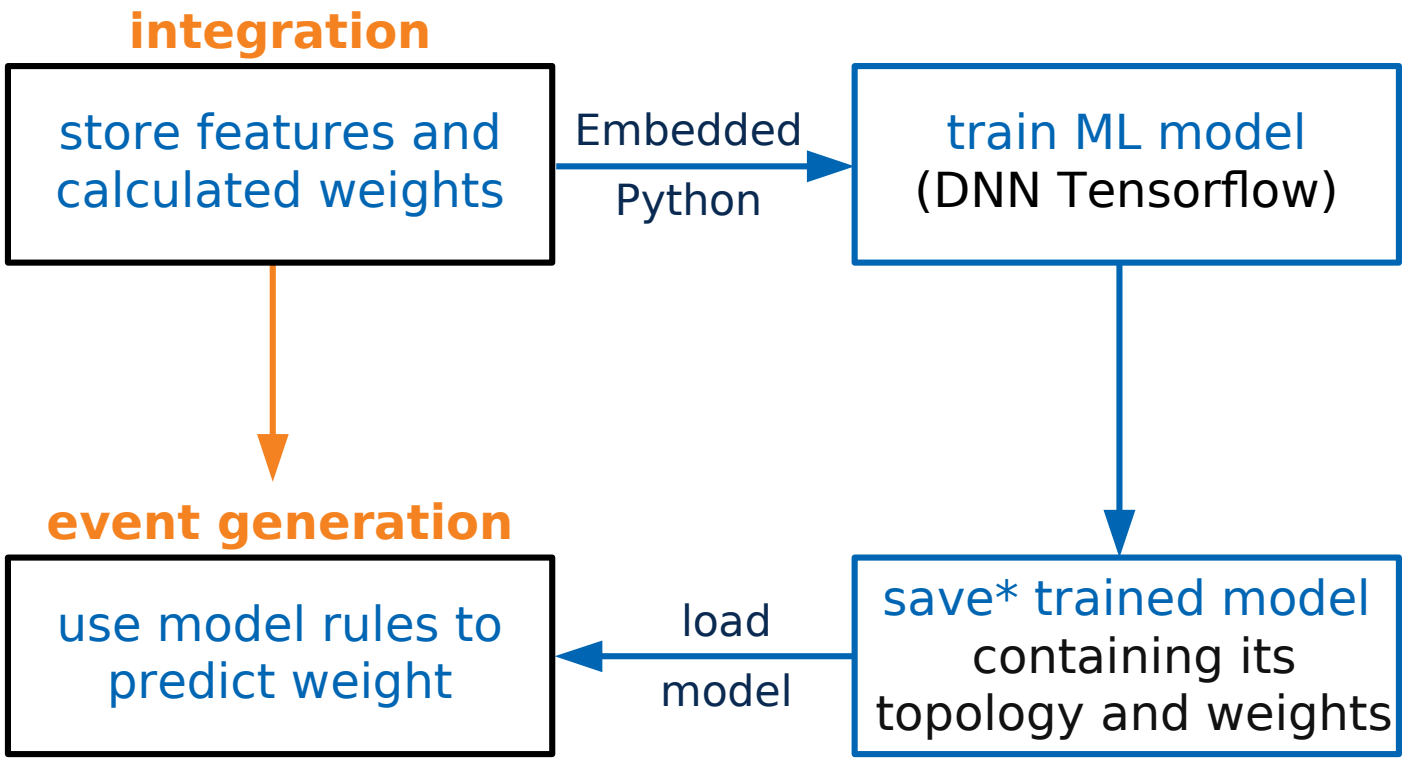
1. generate event
2. calculate corresponding weight w
3. generate uniformly distributed random number R ∈ [0,1]
4. if $w > R \cdot w_{max}$:
   accept event
   else:
   go back to 1.

ML based unweighting

1. generate event
2. calculate approximated weight a
3. generate uniformly distributed random number R ∈ [0,1]
4. if $a > R \cdot w_{max}$:
   go to 5.
   else:
   go back to 1.
5. calculate event weight w and keep the event with weight $\frac{w}{a}$

→ second unweighting needed

Workflow

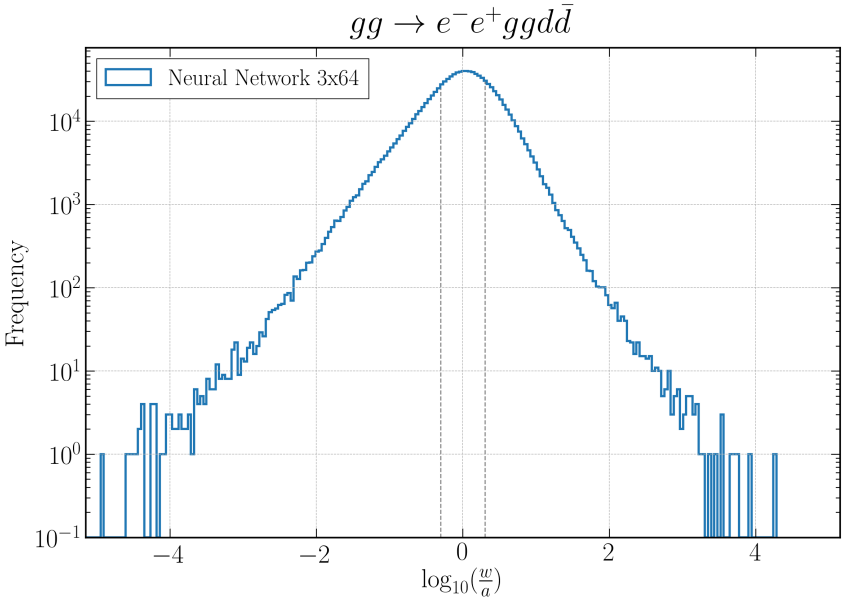


“Classic“

Machine Learning

*as JSON object

Results



$$\frac{\#(\frac{w}{a} > \frac{1}{2} \wedge \frac{w}{a} < 2)}{\#\ total} \approx 51\%$$

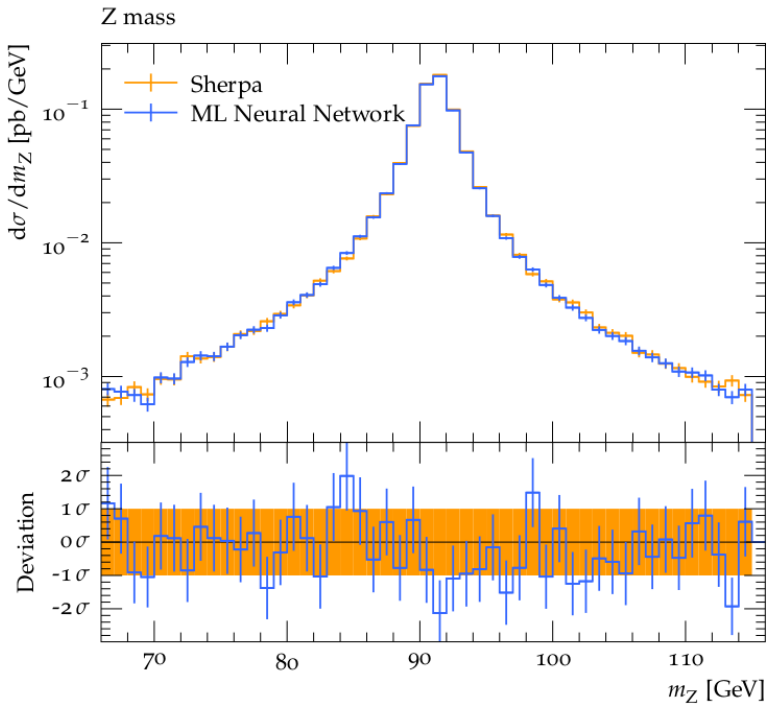
Process	Evt.gen. Time Sherpa / ML
gg→e-e+ggddb	6.1
ug→e-e+gggu	5.5
du→e-e+ggdu	2.1
gd→e-e+gggd	5.6

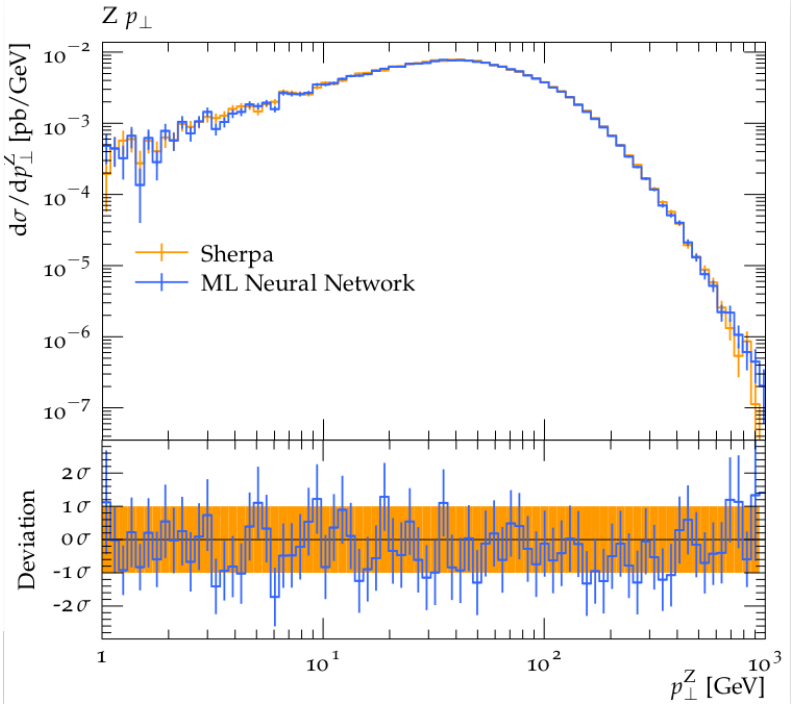
- only Z+jets so far
- further tests for W+jets, ttbar+jets in progress

- event generation time ML up to 6 times faster than Sherpa!
- only Amegic thus far

Results

$$gg \to e^- e^+ ggd\bar{d}$$

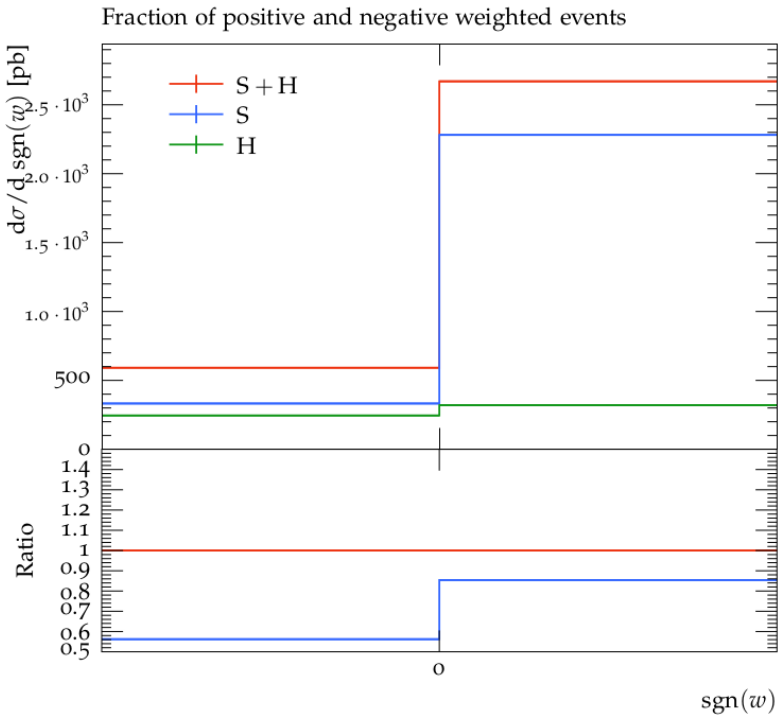




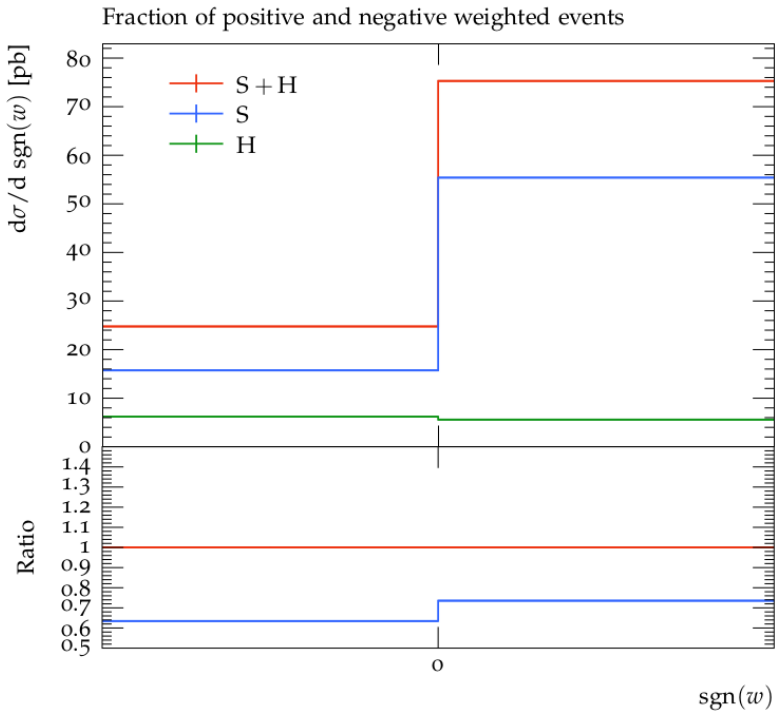
Reduction of negative weight fraction

Negative Weights S vs H Events

$Z + 0, 1, 2j\text{@NLO} + 3\text{@LO}$



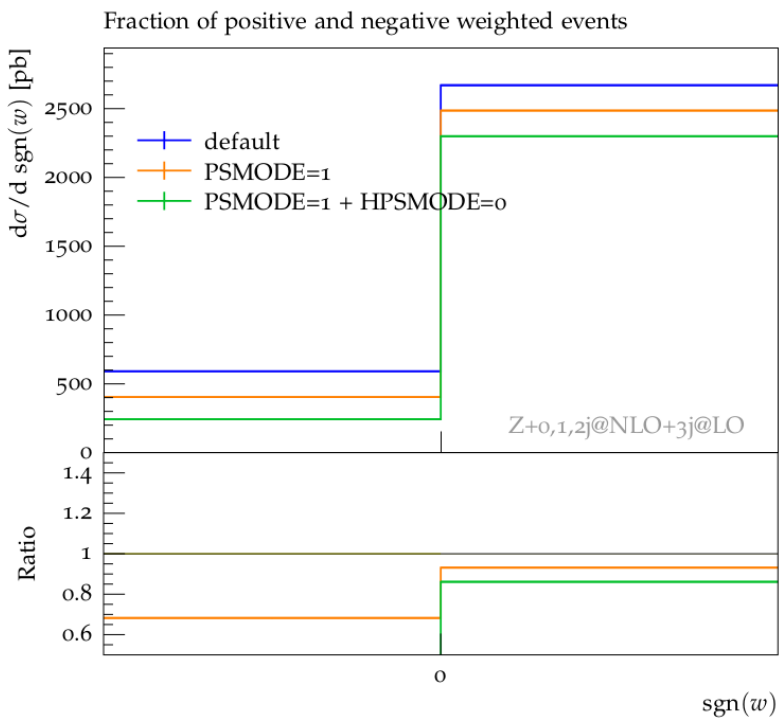
$t\bar{t} + 0, 1j\text{@NLO} + 2, 3\text{@LO}$

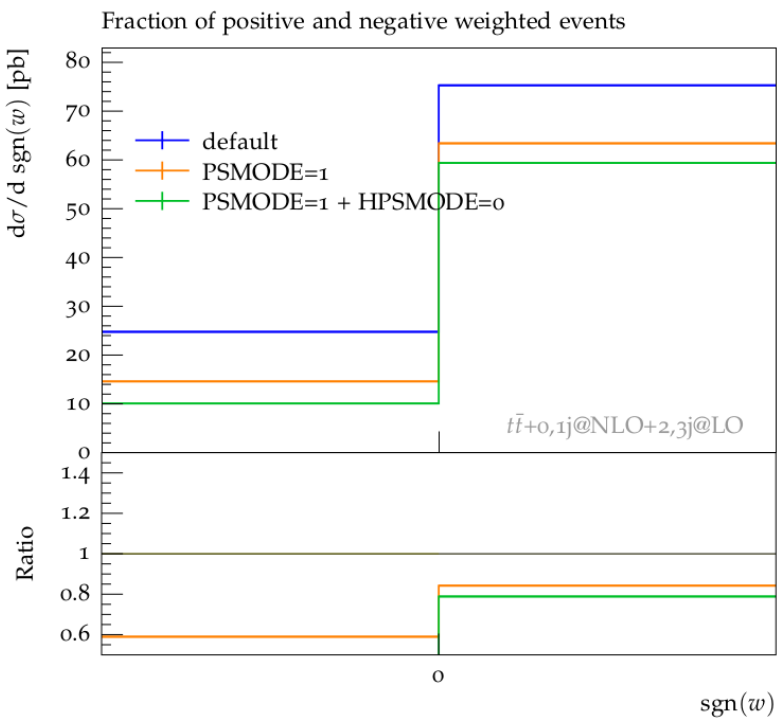


→ negative weights mostly from S events

Possible Improvements

- run MC@NLO in leading color mode → “PSMODE=1“
- Sudakov suppression on H events → “HPSMODE=0“



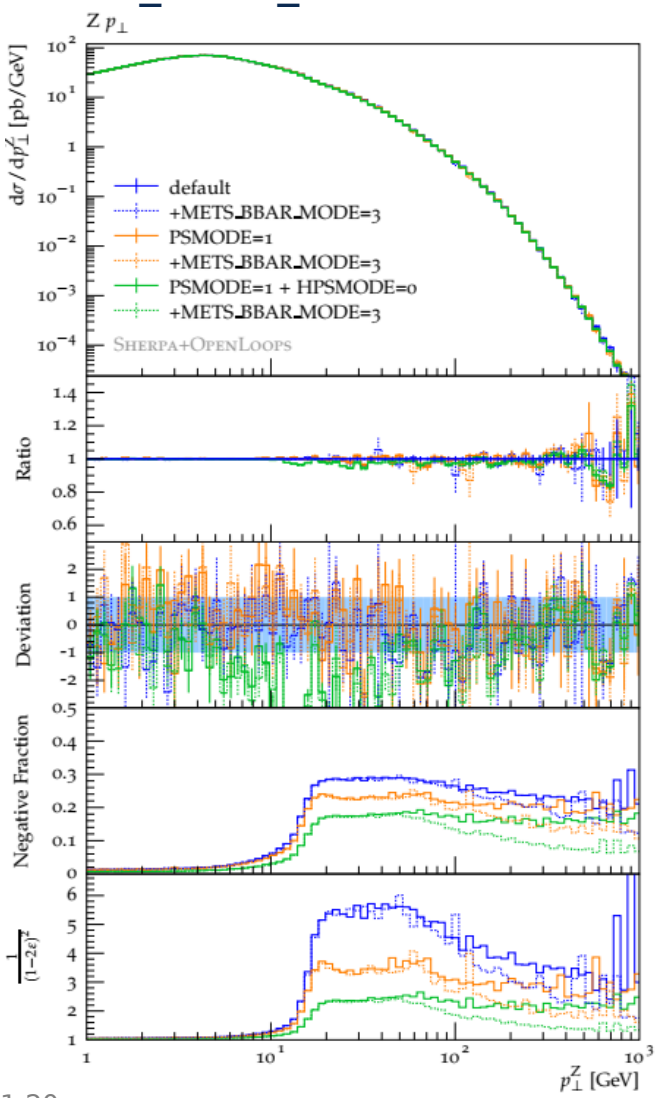


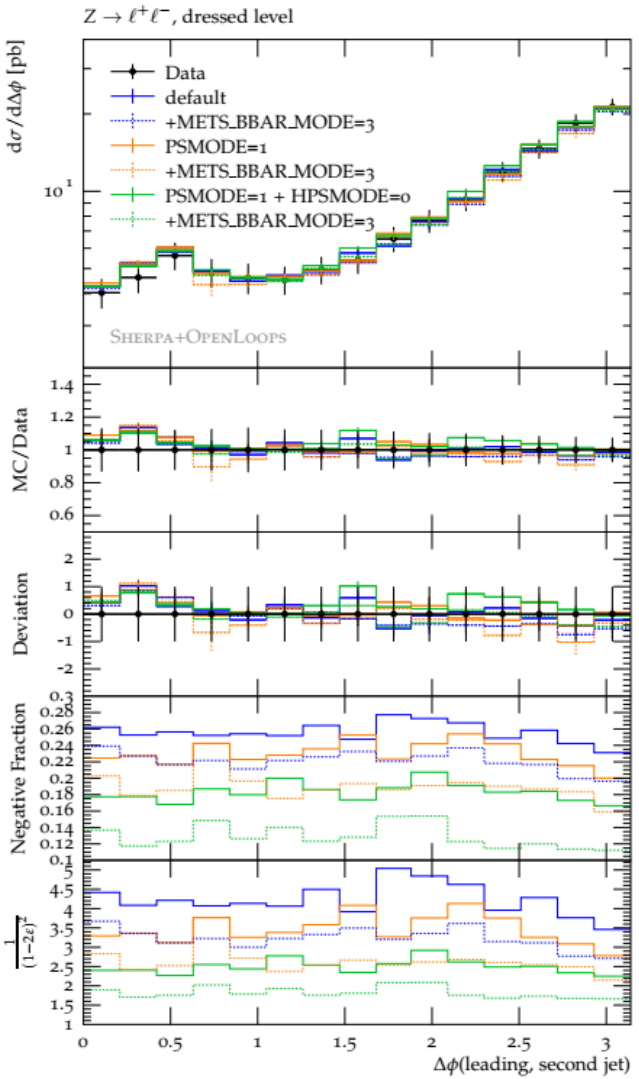
Negative weight fractions

	default	PSMODE=1	HPSMODE=0
Z+jets	18.14%	13.97%	9.53%
ttbar+jets	24.75%	18.72%	14.50%

Results Z+jets

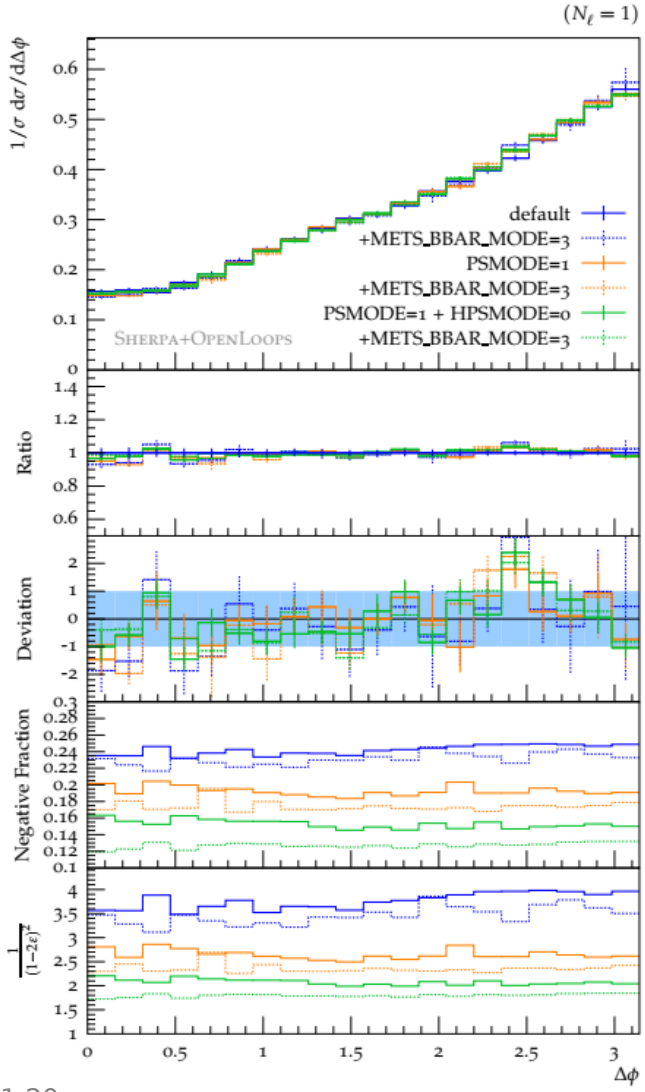
"METS_BBAR_MODE=3" → using local K-factor from core process

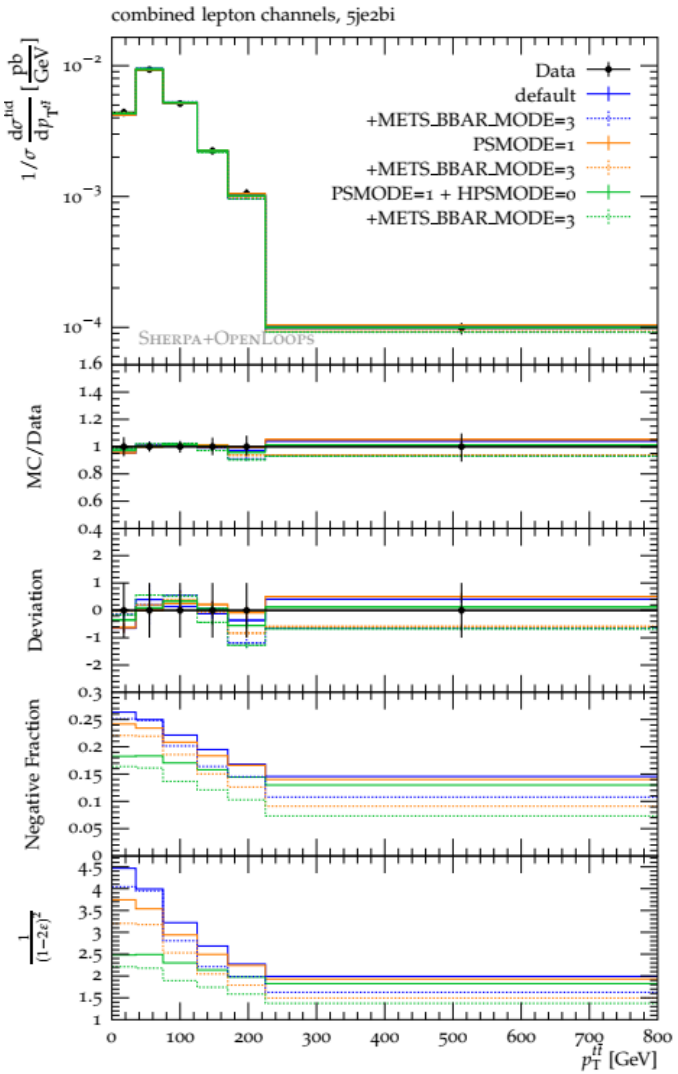




Results ttbar+jets

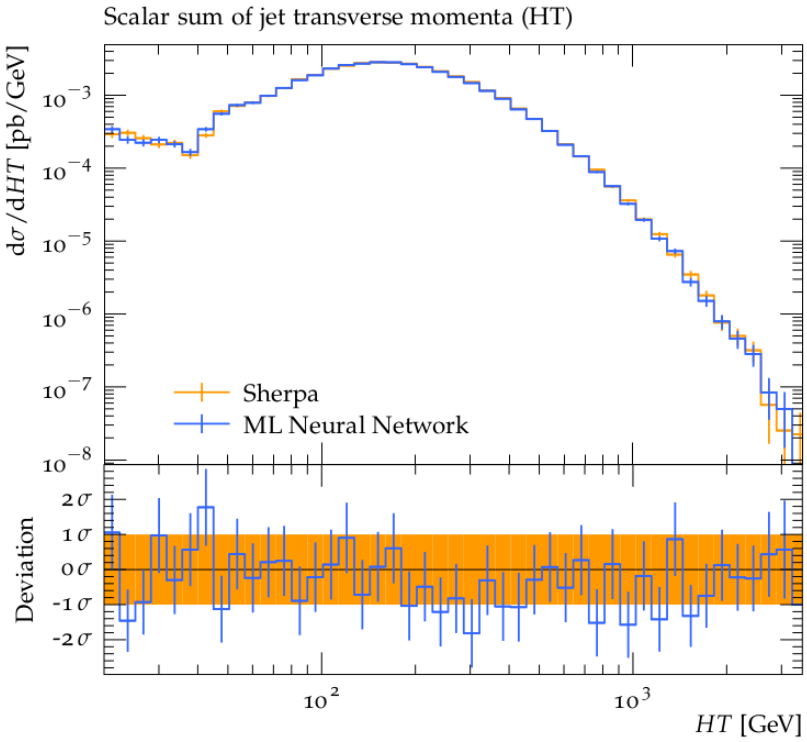
"METS_BBAR_MODE=3" → using local K-factor from core process

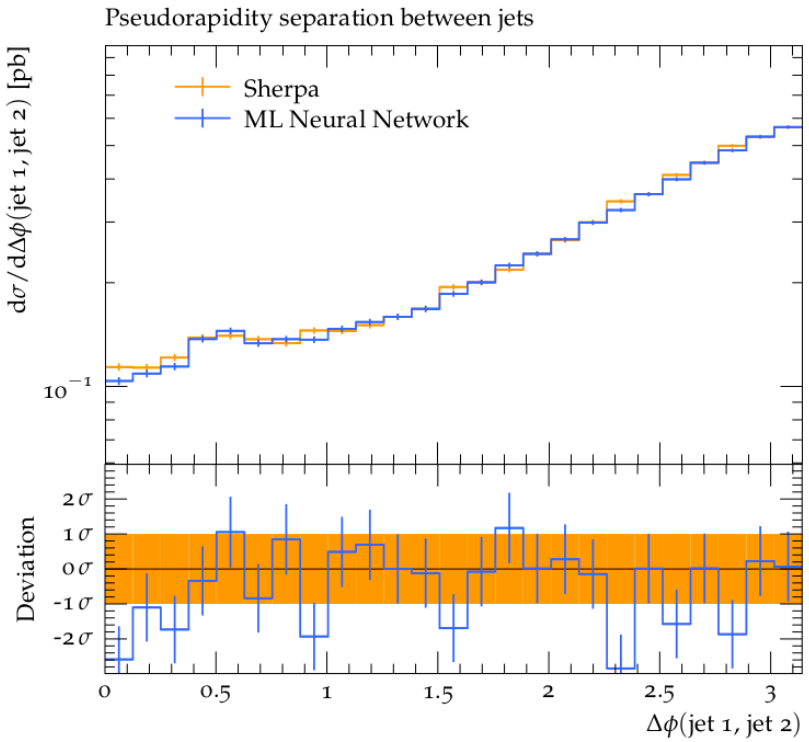




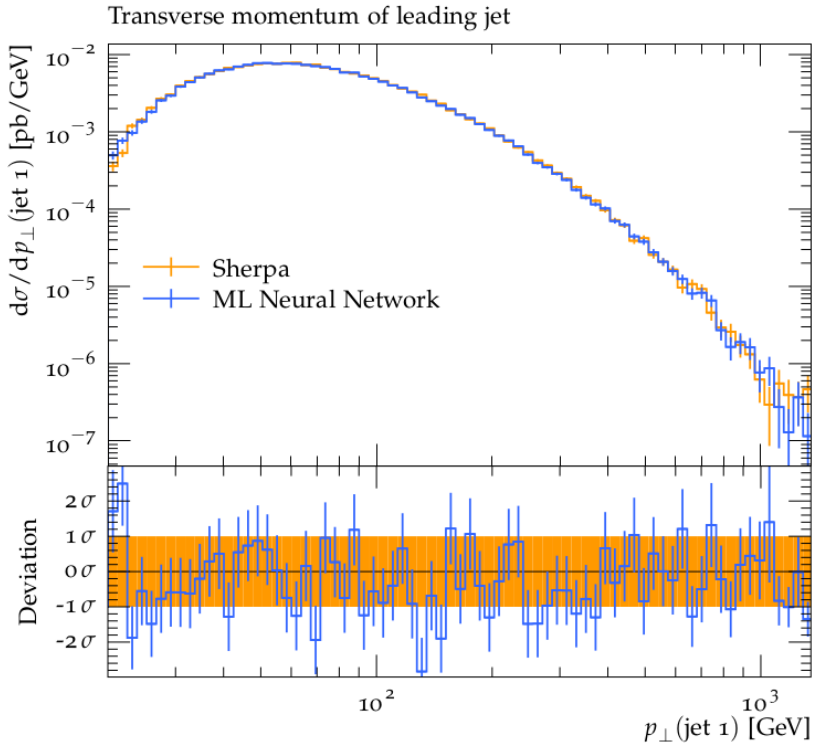
Backup

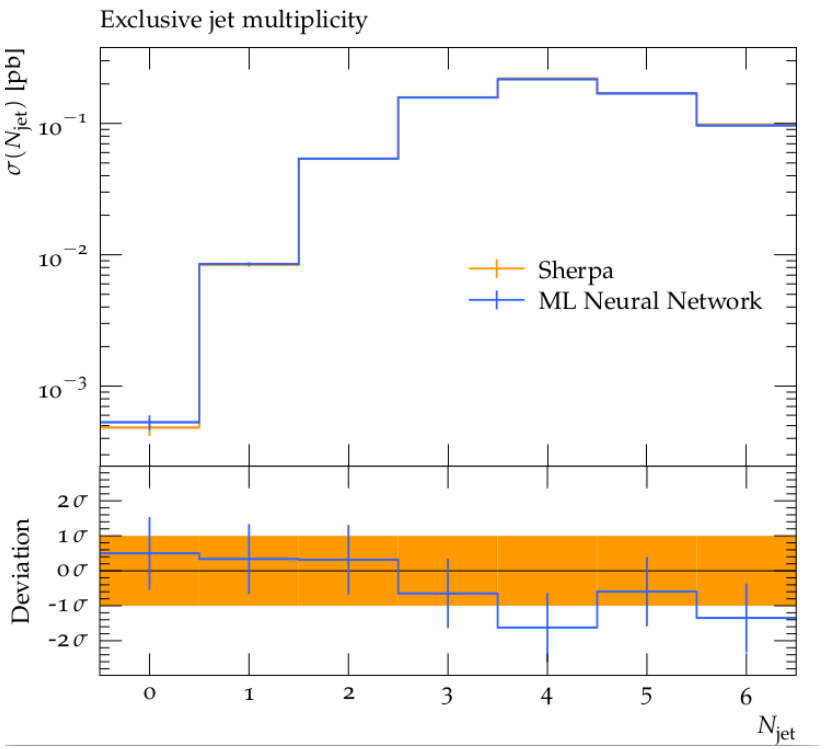
Backup: Results ML Sherpa



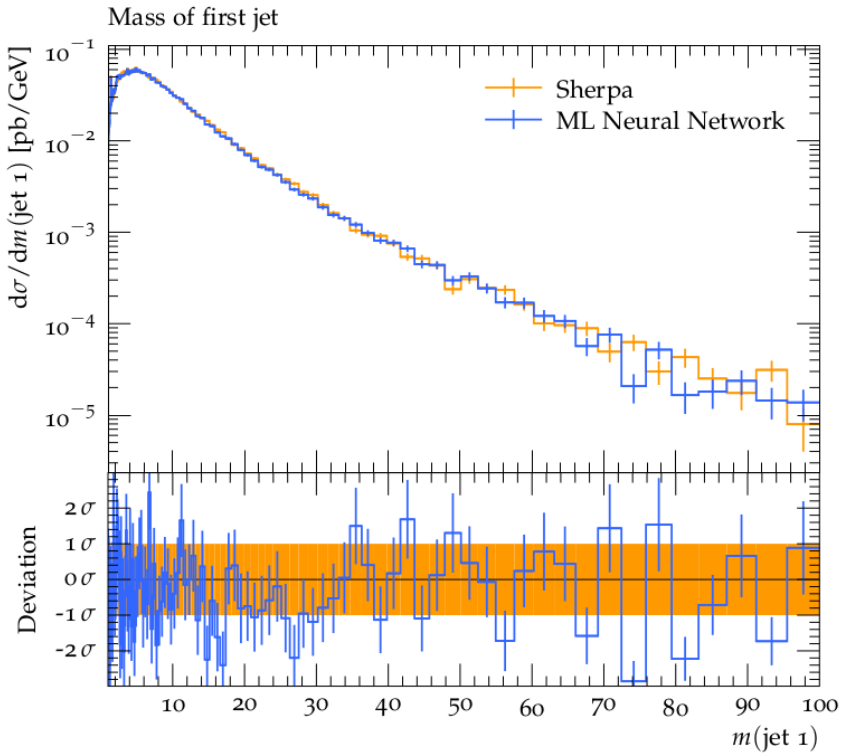


Backup: Results ML Sherpa

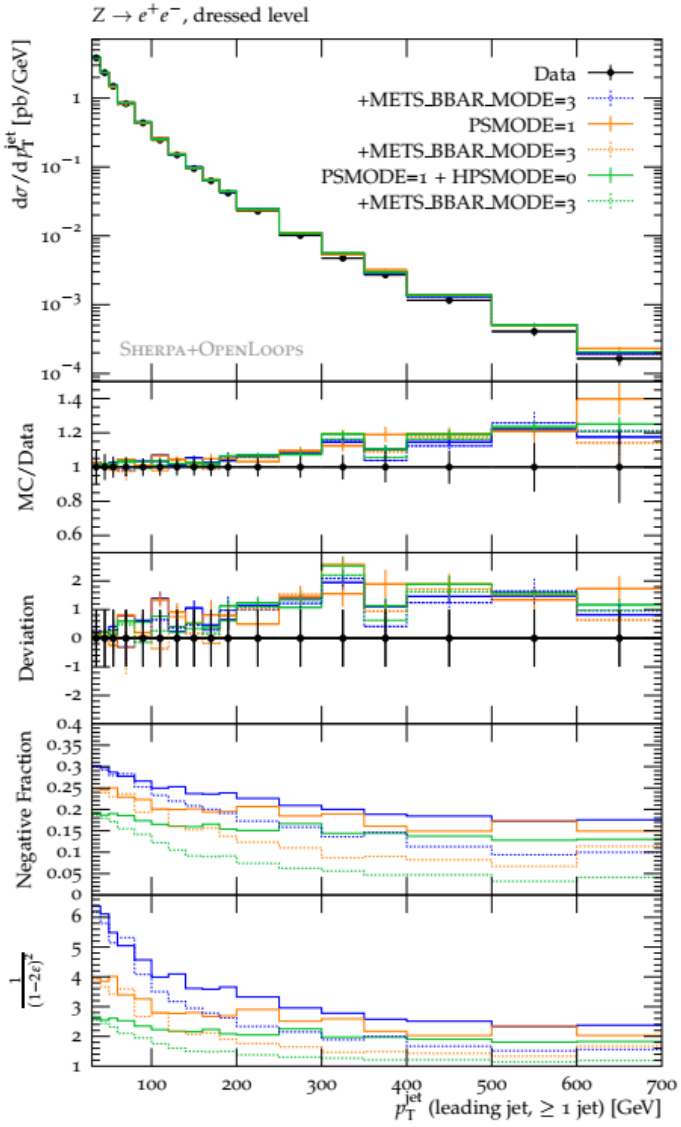


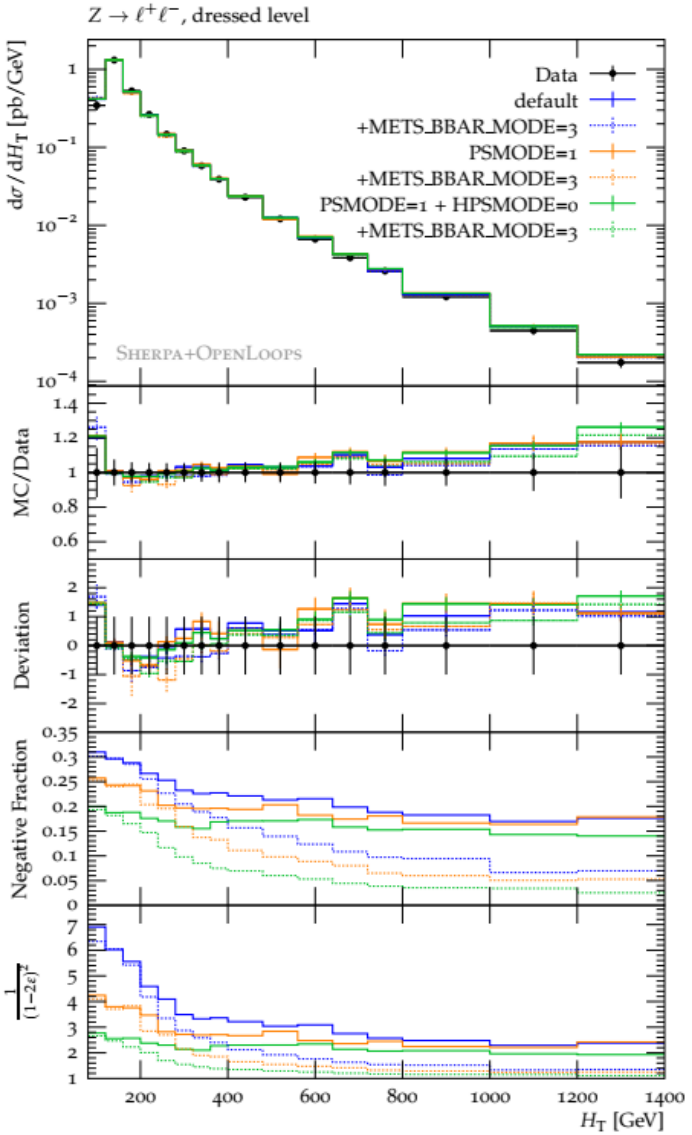


Backup: Results ML Sherpa

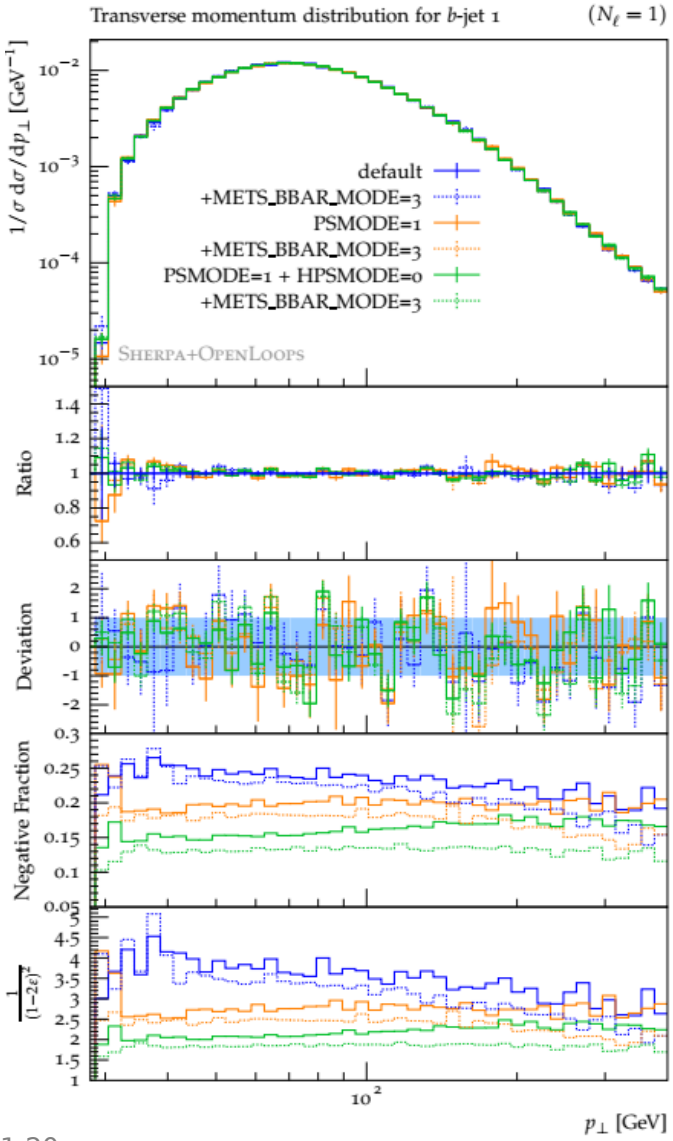


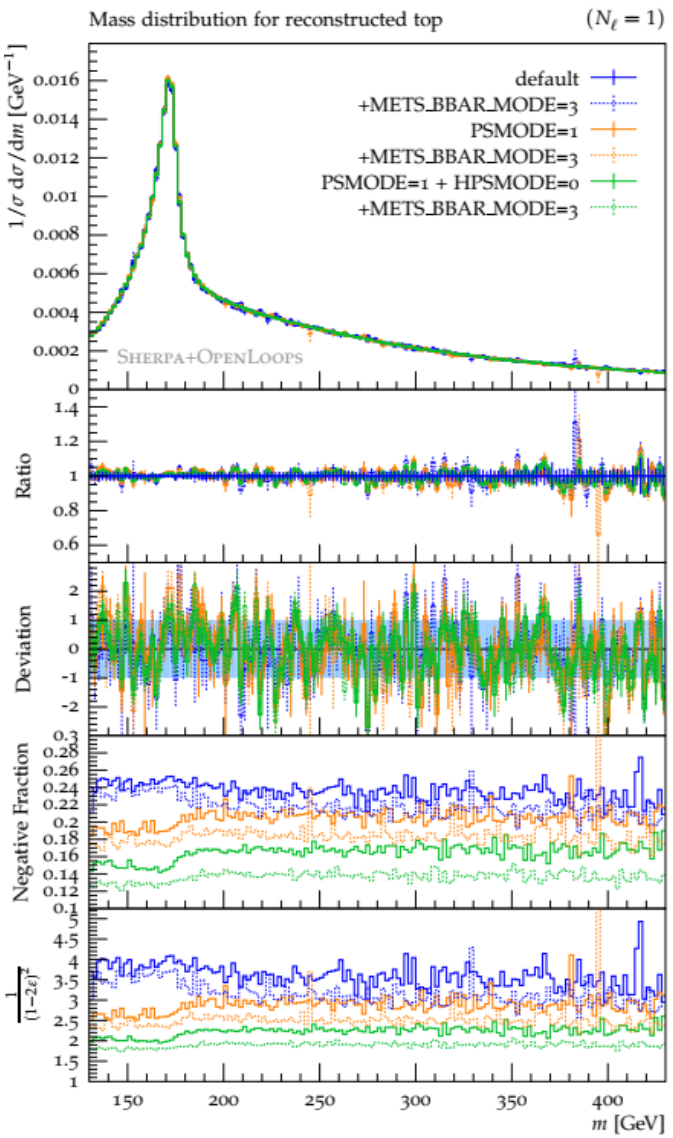
Backup: Results Z+jets





Backup: Results ttbar+jets





Determination of Second Maximum

- employ “second integration“ phase:

  → generate as many points as used for training

- iteratively, determine median of maxima throughout events passing first unweighting