Accurate and Ultrafast Two-Loop Matrix Elements

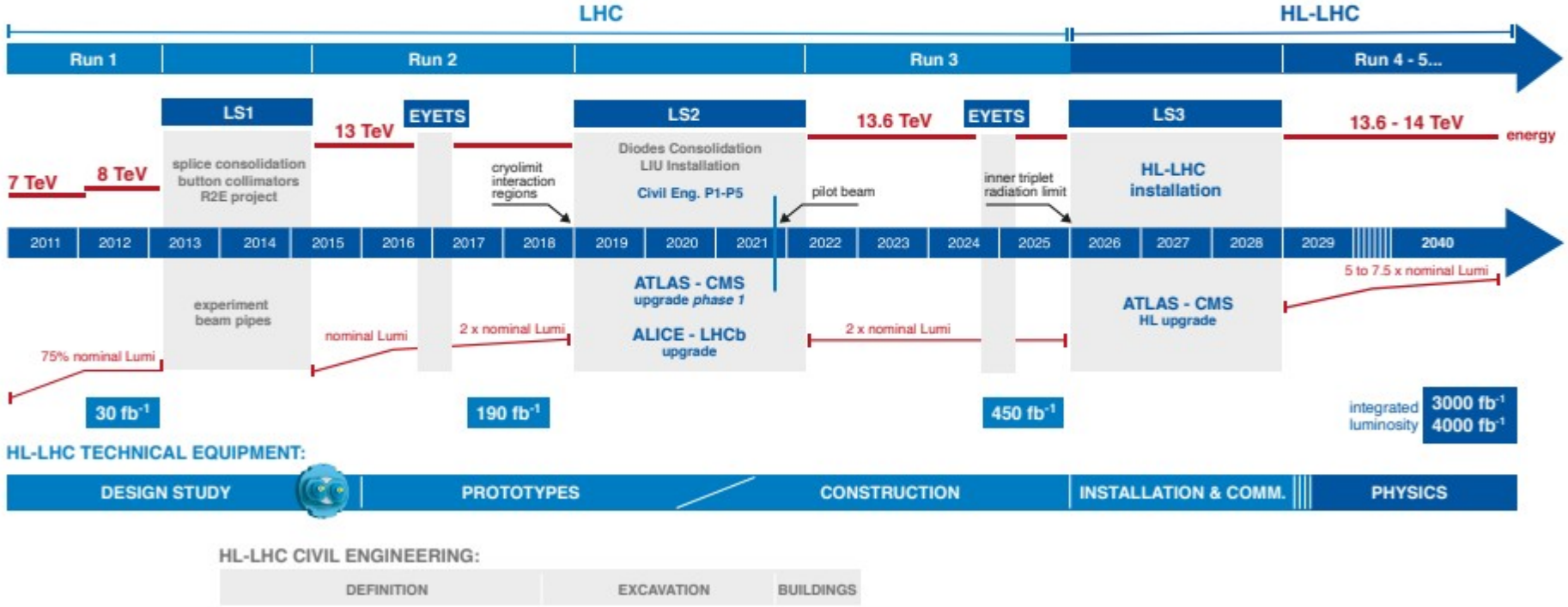
Fady Bishara

IPPP Durham
26 January 2023

HELMHOLTZ RESEARCH FOR GRAND CHALLENGES

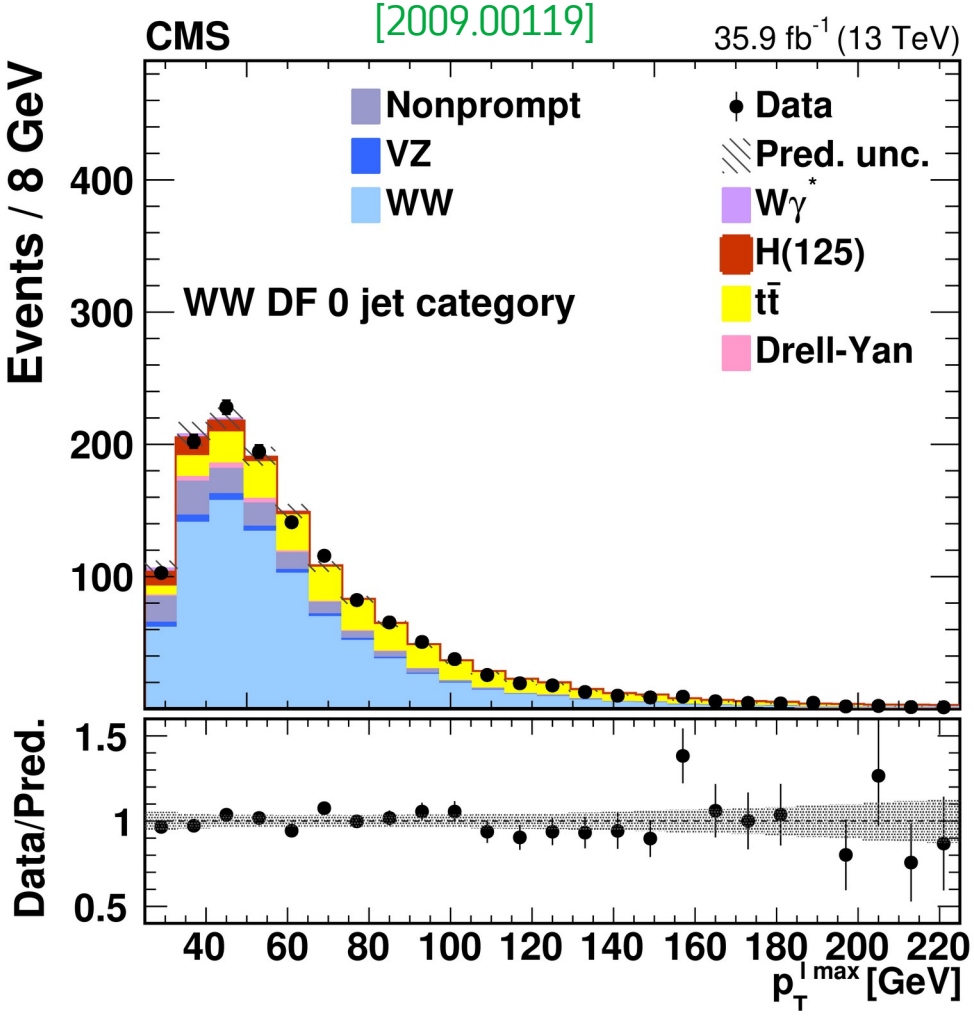
DESY.

LHC / HL-LHC Plan

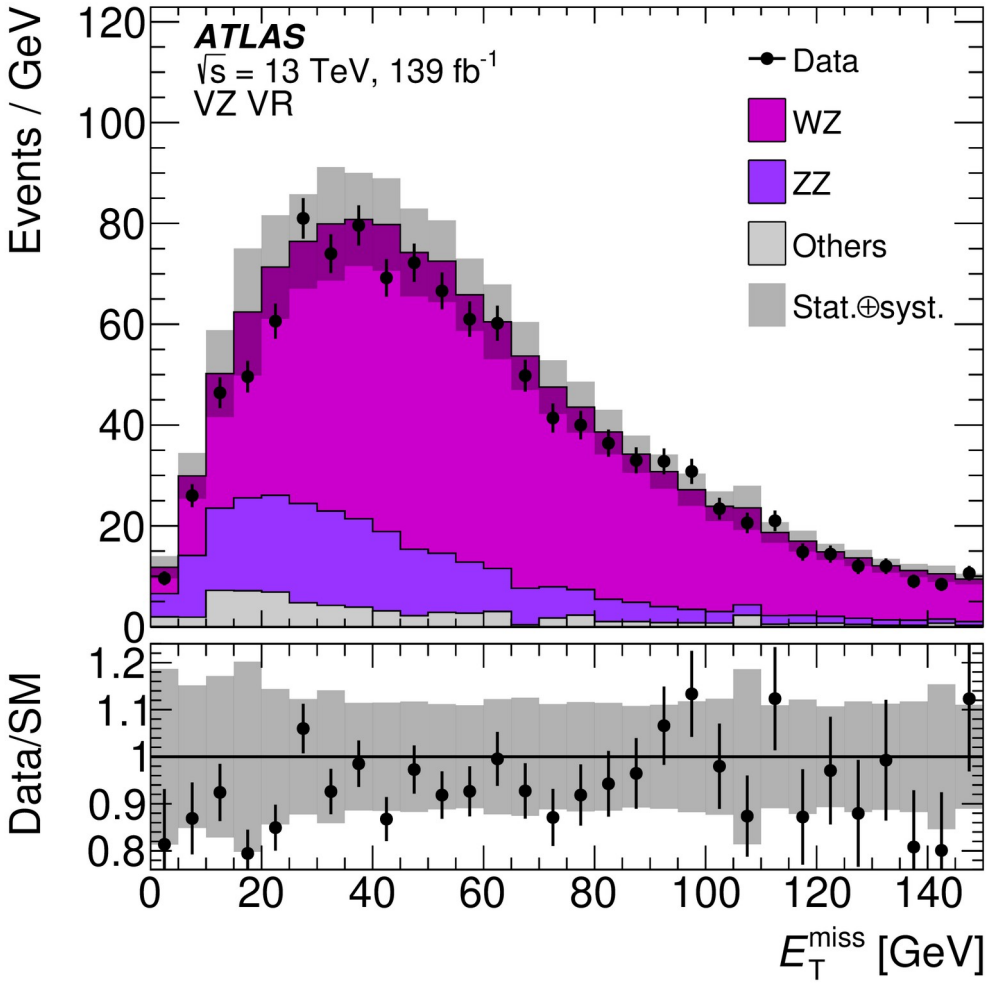


Diboson examples

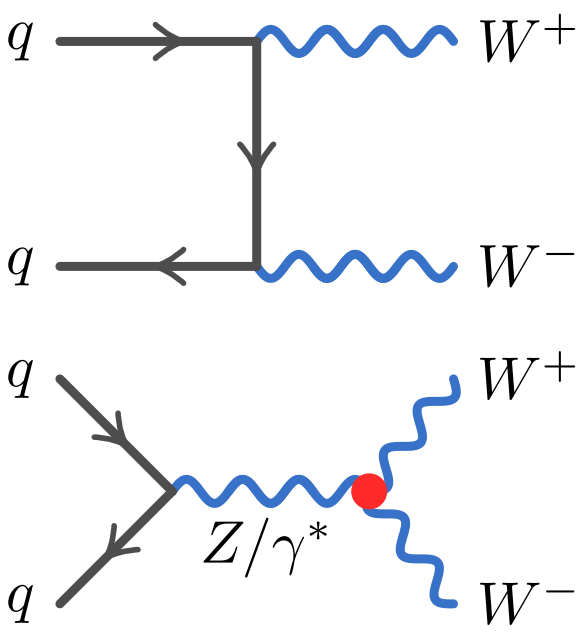
[2009.00119]

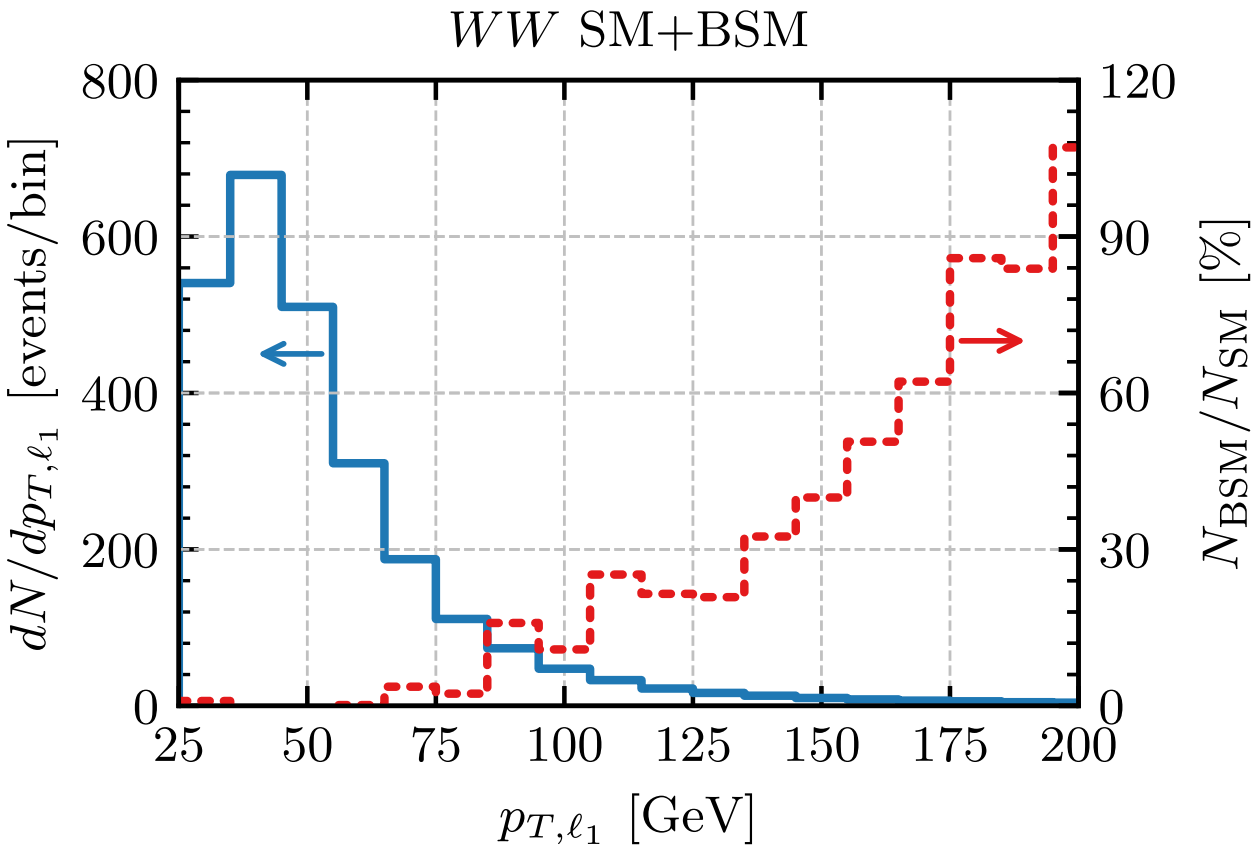


[2103.10319]



WW example in detail





$$N_{\rm SM} = 100,\ \varepsilon_{\rm EXP} = 3\%,\ \varepsilon_{\rm TH} = \{15\%, 3\%\} \Rightarrow \frac{N_{\rm BSM}}{N_{\rm SM}} < \{46\%, 18\%\} @\, 95\%\ \text{C.I.}$$

Better precision → better ability to discover or exclude BSM

Precision $\rightarrow$ compute higher orders in expansion

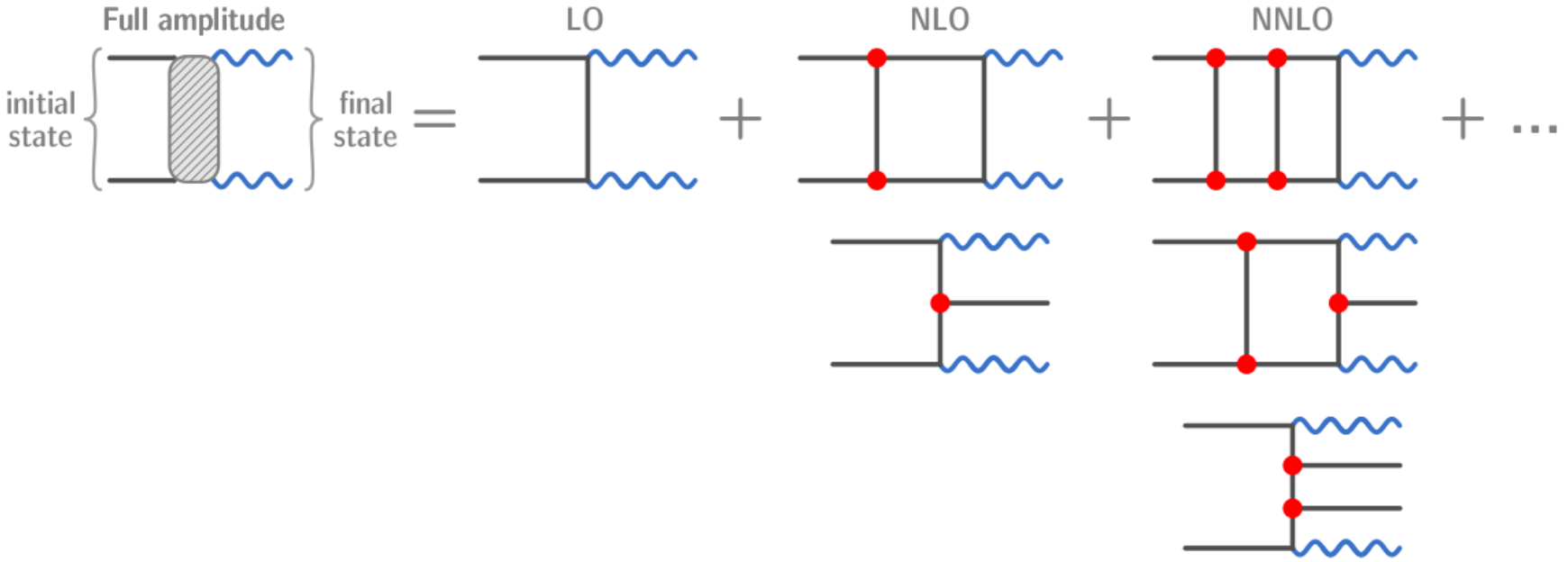
Exact value $= 0.142857143$ LO NLO NNLO N3LO

$$\frac{1}{7} = \frac{1}{10}\left[1 - \frac{3}{10}\right]^{-1} \approx \frac{1}{10}\left[1 + 0.3 + 0.09 + 0.027 + \ldots\right]$$

Cumulative sum: 0.1 0.13 0.139 0.1417

Relative error: 30% 9% 3% 1%

Perturbative expansion in QFT



Slide from Marius Wiesemann

process (${process_id})	LO runtime estimate for 10^{-3} uncertainty	NLO runtime estimate for 10^{-3} uncertainty	NNLO runtime estimate for 10^{-3} uncertainty	
$pp \to H$ (pph21)	2 CPU seconds	1 CPU minute	19 CPU days	Higgs
$pp \to Z$ (ppz01)	4 CPU seconds	1 CPU minute	11 CPU days	DY
$pp \to W^-$ (ppw01)	2 CPU seconds	1 CPU minute	10 CPU days	
$pp \to W^+$ (ppwx01)	5 CPU seconds	2 CPU minutes	11 CPU days	
$pp \to e^- e^+$ (ppeex02)	28 CPU seconds	12 CPU minutes	22 CPU days	
$pp \to \nu_e \bar{\nu}_e$ (ppnenex02)	1 CPU minute	4 CPU minutes	18 CPU days	
$pp \to e^- \bar{\nu}_e$ (ppenex02)	1 CPU minute	16 CPU minutes	21 CPU days	
$pp \to e^+ \nu_e$ (ppexne02)	1 CPU minute	15 CPU minutes	24 CPU days	
$pp \to \gamma\gamma$ (ppaa02)	1 CPU minute	19 CPU minutes	6 CPU days	diphoton
$pp \to e^- e^+ \gamma$ (ppeexa03)	9 CPU minutes	4 CPU hours	167 CPU days	
$pp \to \nu_e \bar{\nu}_e \gamma$ (ppnenexa03)	1 CPU minute	1 CPU hour	17 CPU days	
$pp \to e^- \bar{\nu}_e \gamma$ (ppenexa03)	13 CPU minutes	9 CPU hours	232 CPU days	Wγ
$pp \to e^+ \nu_e \gamma$ (ppexnea03)	17 CPU minutes	1 CPU day	443 CPU days	
$pp \to ZZ$ (ppzz02)	1 CPU minute	4 CPU minutes	25 CPU days	
$pp \to W^+W^-$ (ppwxw02)	1 CPU minute	3 CPU minutes	13 CPU days	
$pp \to e^- \mu^- e^+ \mu^+$ (ppemexmx04)	2 CPU minutes	20 CPU minutes	45 CPU days	
$pp \to e^- e^- e^+ e^+$ (ppeeexex04)	6 CPU minutes	1 CPU hour	193 CPU days	
$pp \to e^- e^+ \nu_\mu \bar{\nu}_\mu$ (ppeexnmnmx04)	3 CPU minutes	29 CPU minutes	31 CPU days	
$pp \to e^- \mu^+ \nu_\mu \bar{\nu}_e$ (ppemxnmnex04)	7 CPU minutes	3 CPU hours	119 CPU days	
$pp \to e^- e^+ \nu_e \bar{\nu}_e$ (ppeexnenex04)	10 CPU minutes	4 CPU hours	52 CPU days	
$pp \to e^- \mu^- e^+ \bar{\nu}_\mu$ (ppemexnmx04)	3 CPU minutes	26 CPU minutes	19 CPU days	
$pp \to e^- e^- e^+ \bar{\nu}_e$ (ppeeexnex04)	6 CPU minutes	1 CPU hour	39 CPU days	
$pp \to e^- e^+ \mu^+ \nu_\mu$ (ppeexmxnm04)	4 CPU minutes	1 CPU hour	21 CPU days	
$pp \to e^- e^+ e^+ \nu_e$ (ppeexexne04)	6 CPU minutes	3 CPU hours	44 CPU days	

MATRIX CPU budget (total runtime)

[Grazzini, Kallweit, MW '17]

from seconds at LO
to minutes at NLO
to days at NNLO

(MATRIX not optimized for simple processes)

diphoton fastest NNLO process

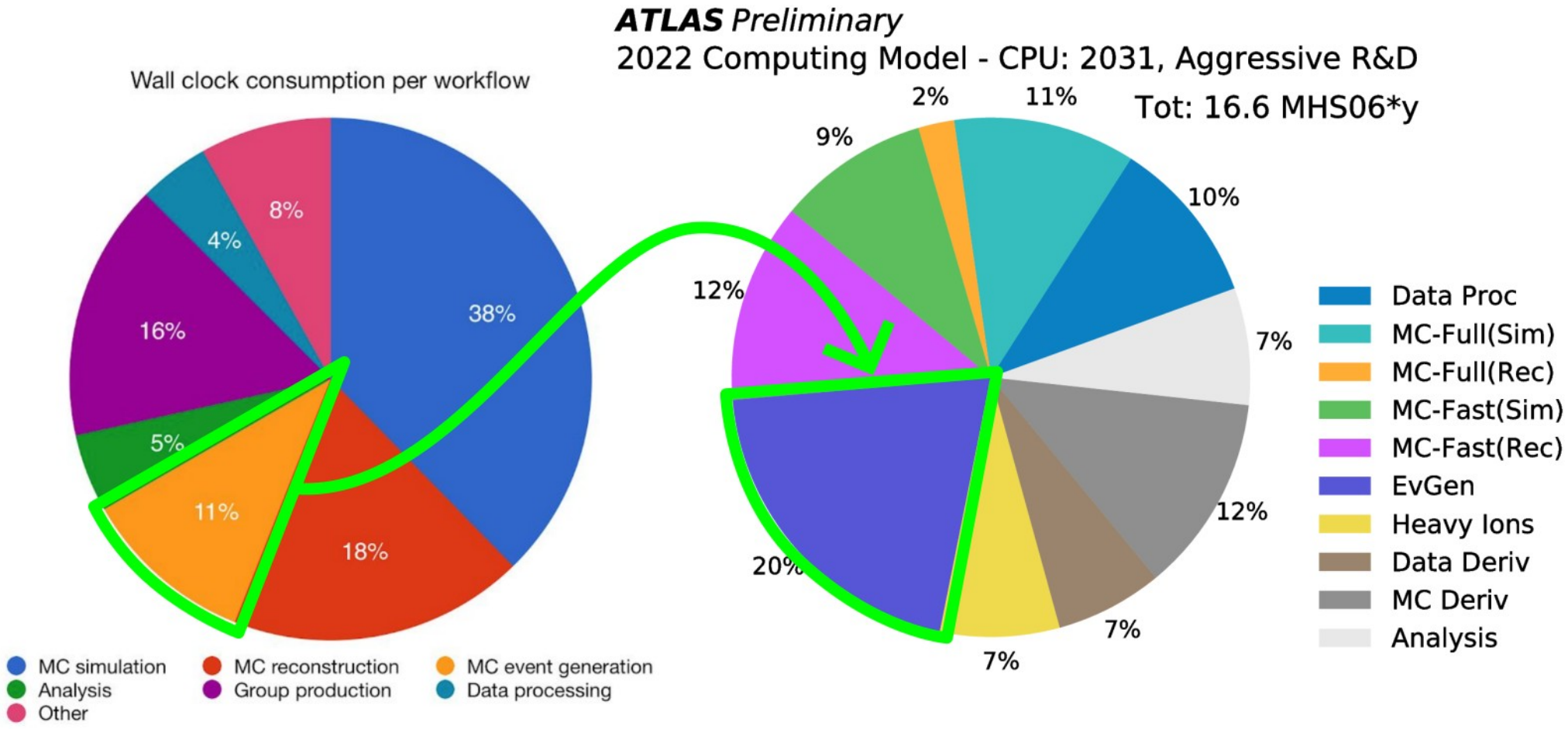
Wγ slowest NNLO process

(dependents on fiducial cuts!)

off-shell diboson processes

from minutes at LO
to hours at NLO
to days at NNLO

Computing budget, e.g.



- Some NNLO computations are extremely slow to evaluate numerically (soln's: interpolate or reweight)

- Bottleneck is evaluation of scalar loop functions (form factors)

- UNIVERSAL APPROXIMATION THEOREM: "…any multivariate continuous function can be represented as a superposition of one–dimensional functions" (Neural Networks/sigmoid)
  From Braun, J. & Griebel, M. Constr Approx (2009)

- In practice, convergence is non-trivial (and not guaranteed)

- ✔ Gradient boosting machines are ideal for this unique application and perform extremely well

And... the time is right!

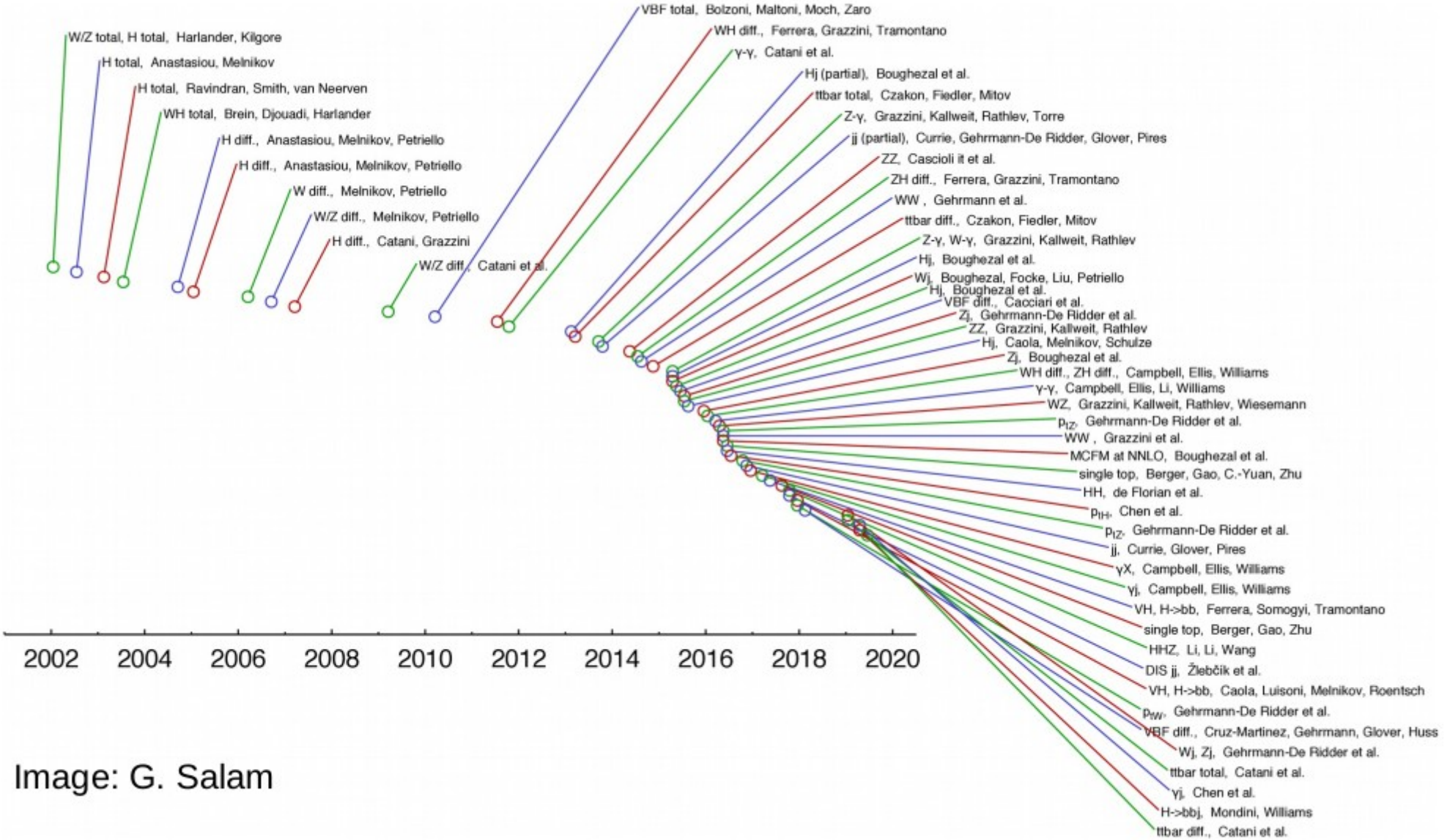


Image: G. Salam

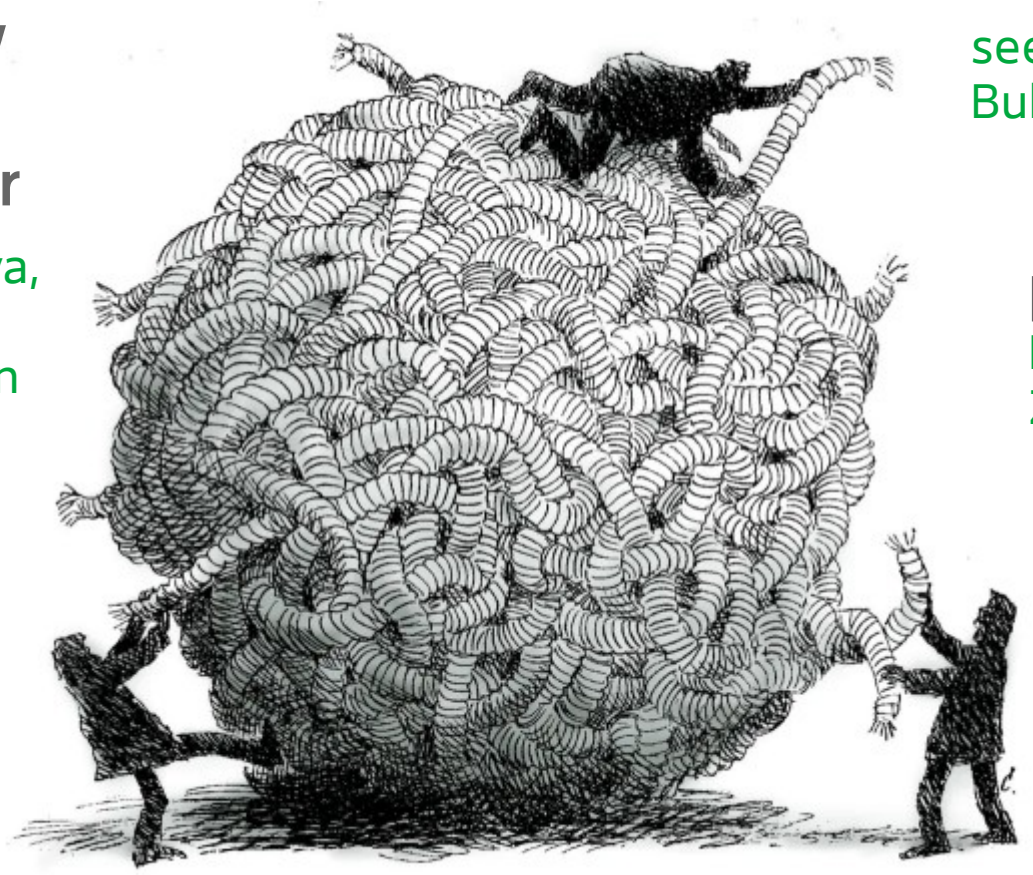
Lots of (ML) activity and approches

Approximating amplitudes

see also Badger & Bullock [2002.07516]

Numerical stability by well-chosen integration contour

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



Hadronization

Ilten, Menzo, Youssef, Zupan [2203.04983]

Symbolic simplification of polylogs using language models

Dersy, Schwartz, Zhang [2206.04115]

And much more...

Integration and sampling efficiency

see e.g., Bendavid [1707.00028]; Klimek, Perelstein [1810.11509]; Gao, Isaacson, Krause [arXiv:2001.05486]; Gao, Höche, Isaacson, Krause, Schulz [2001.10028], Maitre, Santos-Mateos [2211.02834]

https://ischoolonline.berkeley.edu/blog/what-is-machine-learning/

Definition of Machine Learning

The basic concept of machine learning in data science involves using statistical learning and optimization methods that let computers analyze datasets and identify patterns (view a visual of machine learning via R2D3). Machine learning techniques leverage data mining to identify historic trends and inform future models.

The typical supervised machine learning algorithm consists of roughly three components:

1. **A decision process:** A recipe of calculations or other steps that takes in the data and "guesses" what kind of pattern your algorithm is looking to find.
2. **An error function:** A method of measuring how good the guess was by comparing it to known examples (when they are available). Did the decision process get it right? If not, how do you quantify "how bad" the miss was?
3. **An updating or optimization process:** A method in which the algorithm looks at the miss and then updates how the decision process comes to the final decision, so next time the miss won't be as great.

$$L(\theta) = \sum_i (y_i - \hat{y}_i)^2$$

Boosted decision trees (BDTs)

Sequential, additive corrections to previous result

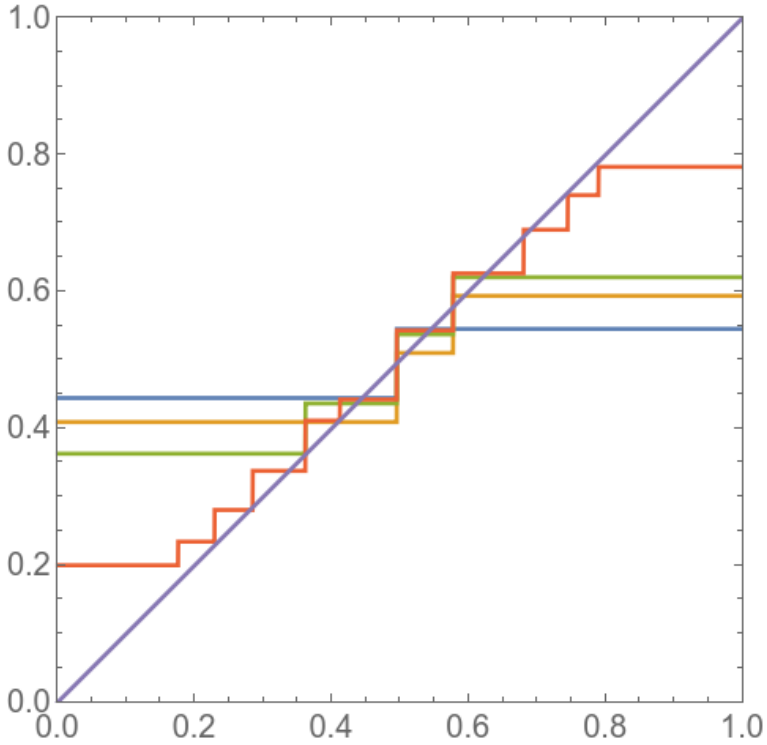
$$\hat{y}_i^{(0)} = 0$$

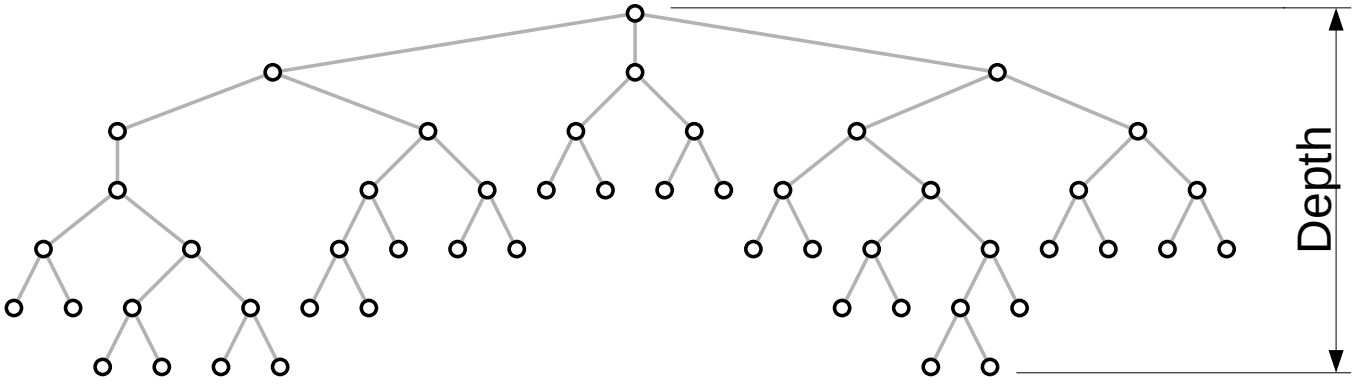
$$\hat{y}_i^{(1)} = f_1(x_i) = \hat{y}_i^{(0)} + f_1(x_i)$$

$$\hat{y}_i^{(2)} = f_1(x_i) + f_2(x_i) = \hat{y}_i^{(1)} + f_2(x_i)$$

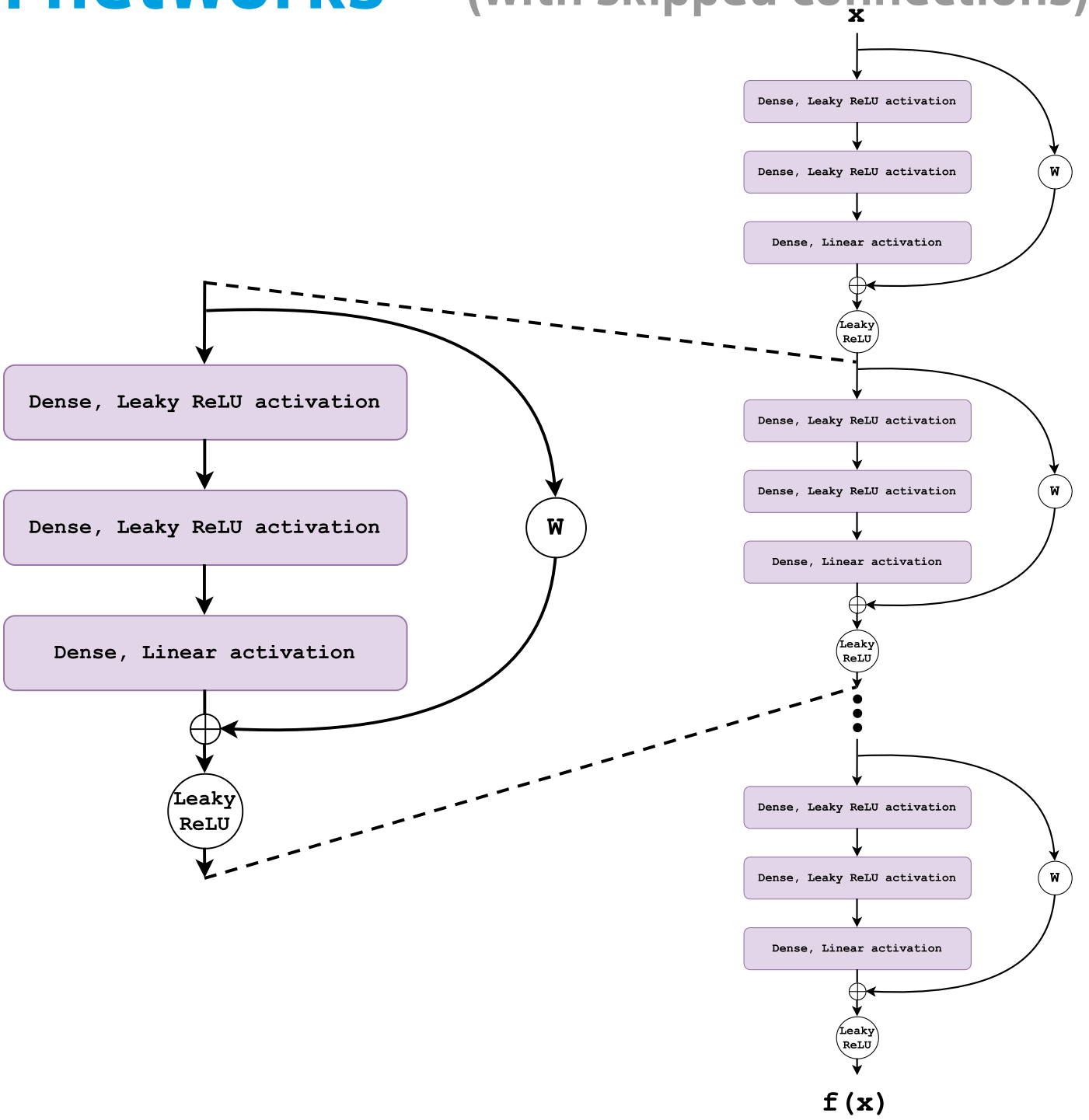
$$\dots$$

$$\hat{y}_i^{(t)} = \sum_{k=1}^{t} f_k(x_i) = \hat{y}_i^{(t-1)} + f_t(x_i)$$



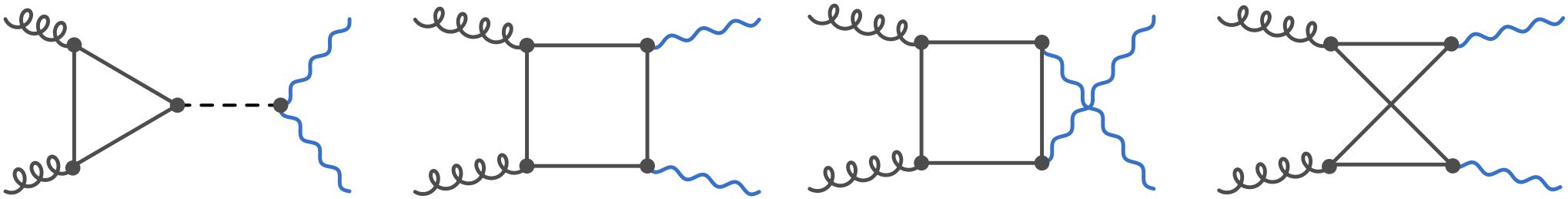


Neural networks (with skipped connections)



A Proof of Principle

Loop-induced matrix element @ L.O.



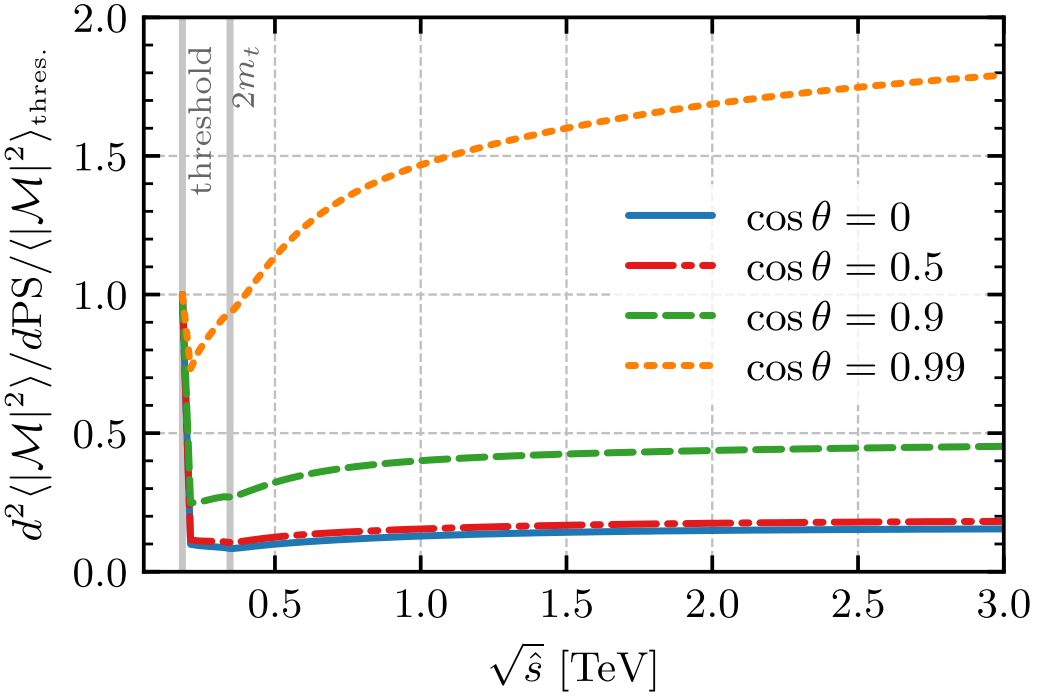
$$gg \to ZZ$$

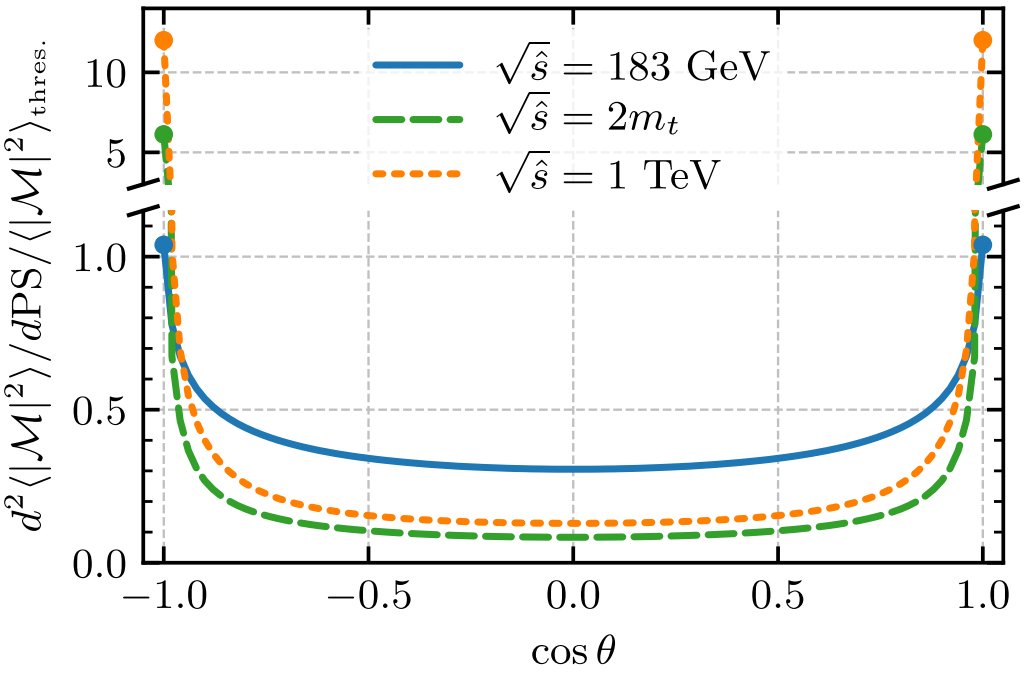
- Phase-space is 2-dimensional: $\{\sqrt{\hat{s}}, \cos\theta\}$
  - squared/averaged matrix element $\langle|\mathcal{M}|^2\rangle : \mathbb{R}^2 \mapsto \mathbb{R}_{>0}$
  - for on-shell Z's, invariant under $\cos\theta \to -\cos\theta$

- It is important to normalize $\langle|\mathcal{M}|^2\rangle$ because of loss function
  - simple sol'n: divide by max. value in large sample
  - better: divide by std. deviation of large sample
  - even better: take the $\log$

[FB & Marc Montull [1912.11055]]

- $[2m_Z \oplus p_{T,Z} > 1\,\text{GeV}, 3\,\text{TeV}] \mapsto [0,1]^2$
  - the p_T cut regulates an integrable singularity @ $\cos\theta = \pm 1$ as in `MCFM` and `MG5_aMC@NLO` otherwise **no cuts** on P.S.!
  - extending $\sqrt{\hat{s}}$ to the full 14 TeV is trivial





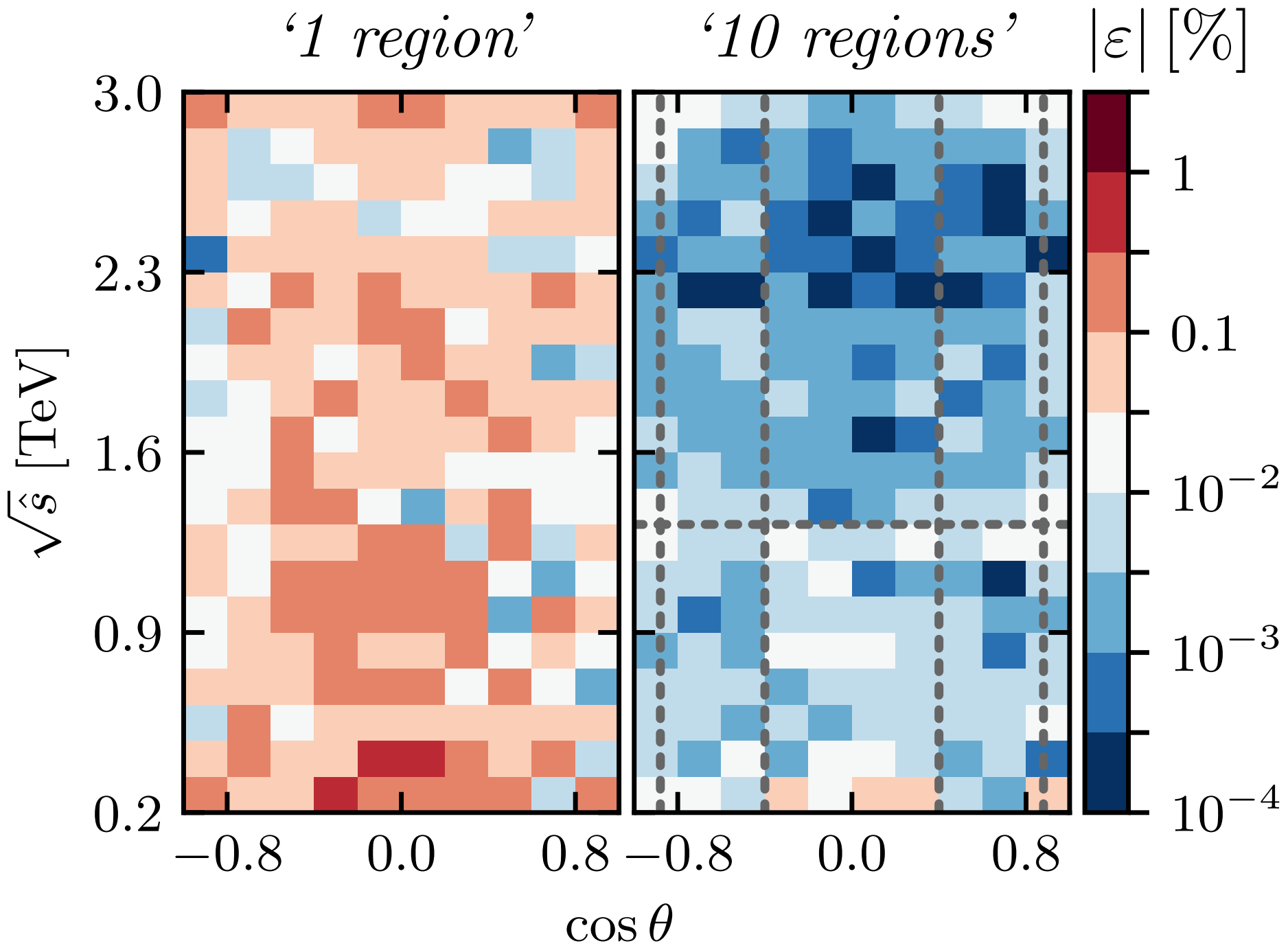
The training and validation sets

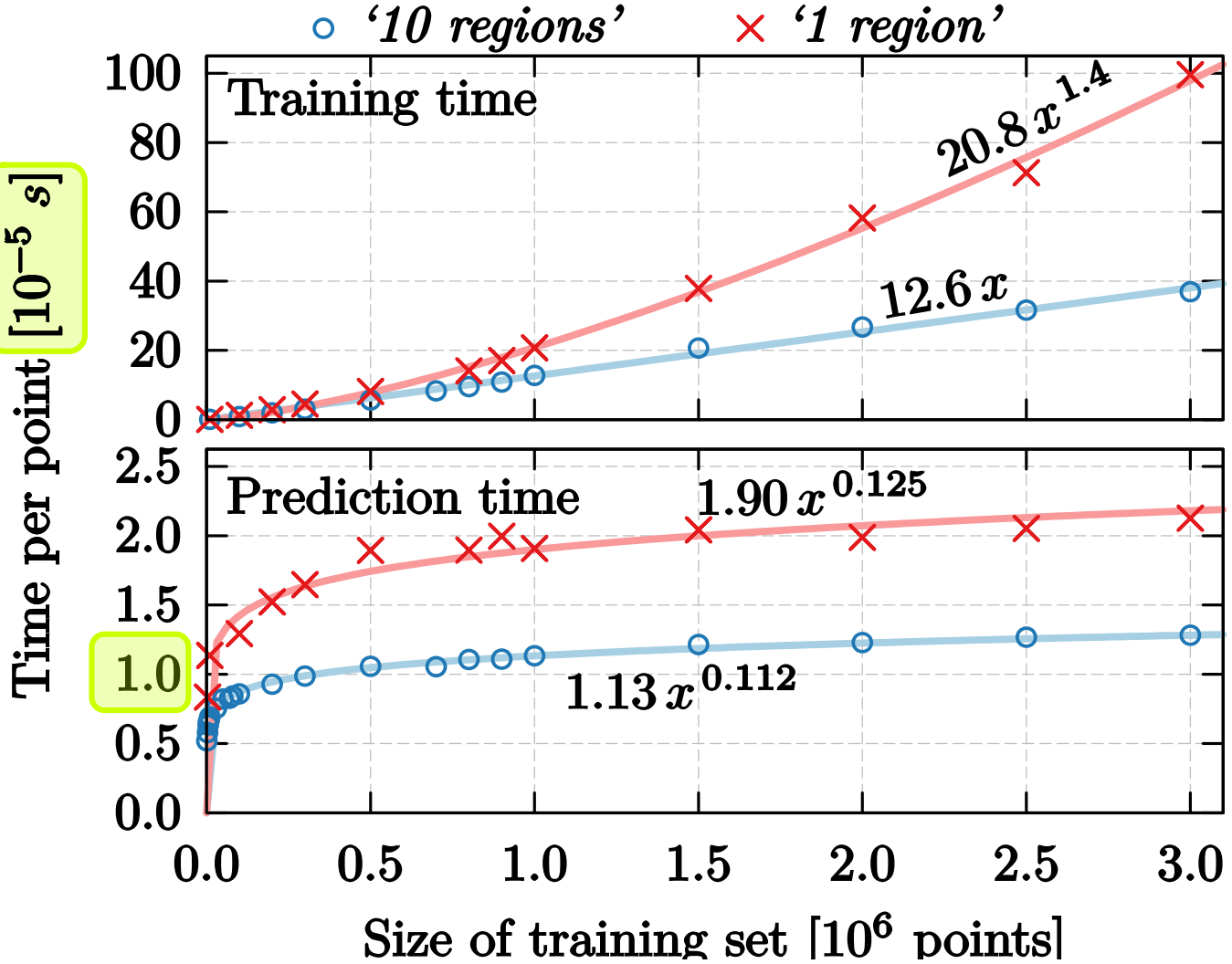
- Populate full phase-space uniformly
  - Training: 1.5M points
  - Validation: 15M points (10× to catch rare events)

- Compute $\langle|\mathcal{M}|^2\rangle$ using `OpenLoops2`
  [Buccioni, Lang, Lindert, Maierhöfer, Pozzorini, Zhang, Zoller [1907.13071]]

- Approximation error defined as

$$\varepsilon = 1 - \frac{\texttt{approx.}}{\texttt{exact}}$$


‘1 region’
‘10 regions’
$|\varepsilon|$ [%]
3.0
2.3
1.6
0.9
0.2
$\sqrt{\hat{s}}$ [TeV]
−0.8
0.0
0.8
−0.8
0.0
0.8
$\cos\theta$
1
0.1
10^{-2}
10^{-3}
10^{-4}



★ Compare with `OpenLoops` 8.7×10^{-3} [s/point] $\Rightarrow$ 1000-fold speedup!

Two-loop matrix element

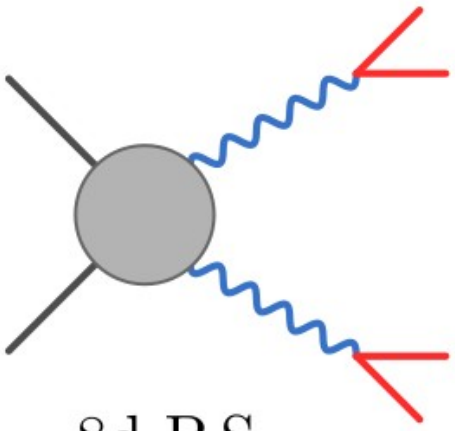
10 tensor structures
@LO only 4: $T_7, \ldots, T_{10}$

form factors = scalar
functions of kinematic vars.

Squared amplitude:

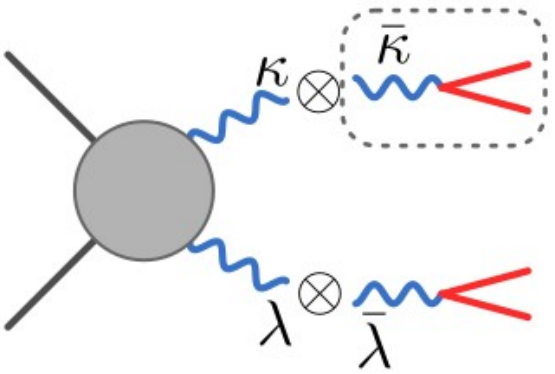
$$\mathcal{T}\left(s,t,p_3^2,p_4^2\right) = \sum_{\text{pols,cols}} \left| \sum_{j=1}^{10} A_j\left(s,t\right) T_j^{\mu\nu}\, \varepsilon_\lambda^\mu\left(p_3\right) \varepsilon_{\lambda'}^\nu\left(p_4\right) \right|^2$$

$$\mathcal{T}^{(2)}(s,t,p_3^2,p_4^2) = 2\Re\left(\langle \mathcal{M}^{(0)} | \mathcal{M}^{(2)} \rangle\right) + \langle \mathcal{M}^{(1)} | \mathcal{M}^{(1)} \rangle$$



8d P.S.

$$= \sum_{\kappa,\lambda}$$

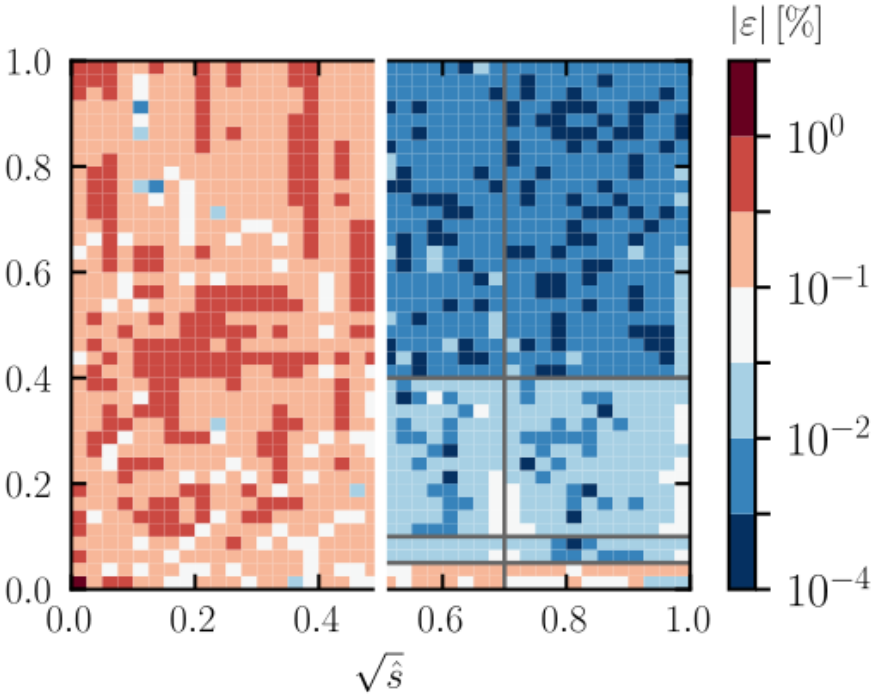


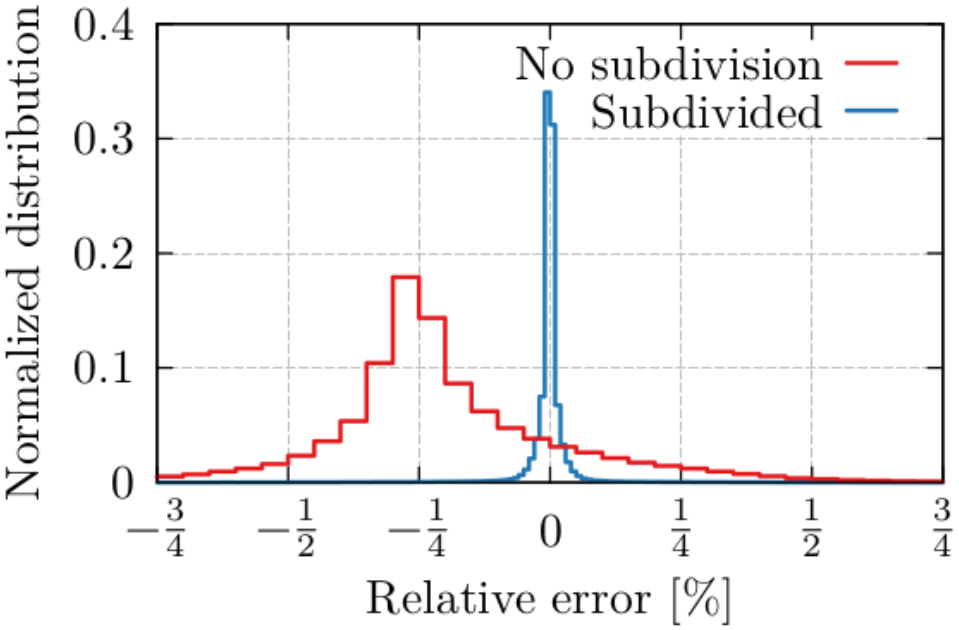
4d P.S. + 2 × 2d P.S.

lepton amps. are simple:
$\propto \langle 3\gamma_\mu 4]$ and $[3\gamma_\mu 4\rangle$

- Extract A_i from `VVAMP` (computed to $\mathcal{O}(\alpha_s^2)$)
  [Gehrmann, von Manteuffel, Tancredi [1503.04812]]

- Compute and approximate $\mathcal{T}^{(2)}$ (full or 2L$\times$ Born)





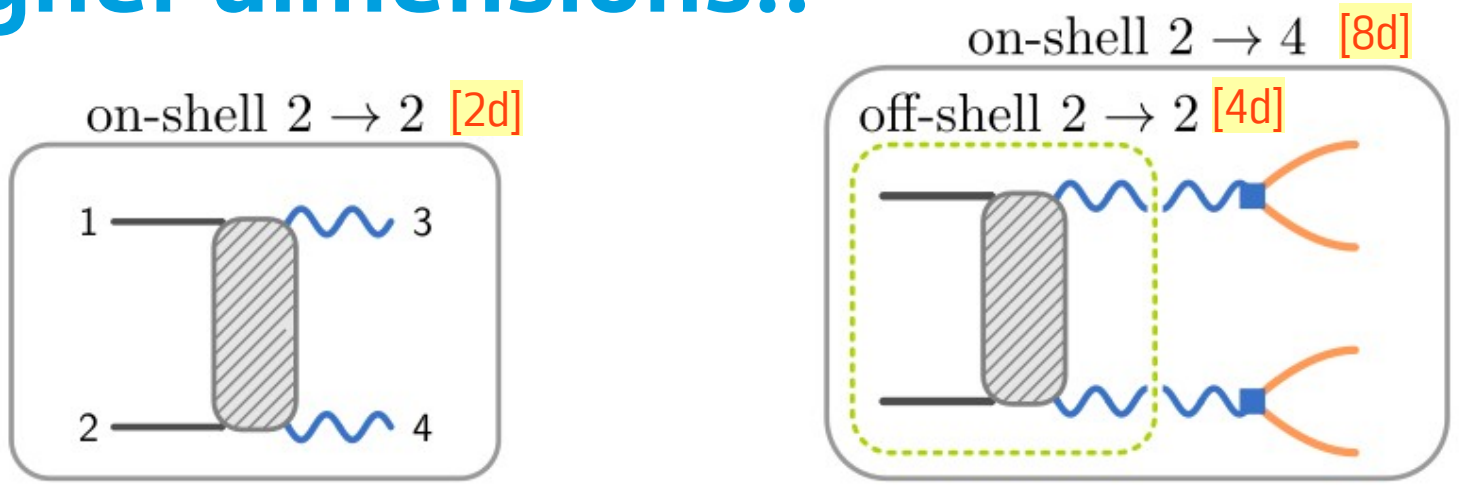
Exact: $\sim$ 16s / point

Approximate: $\sim$ 16s / **1M** points

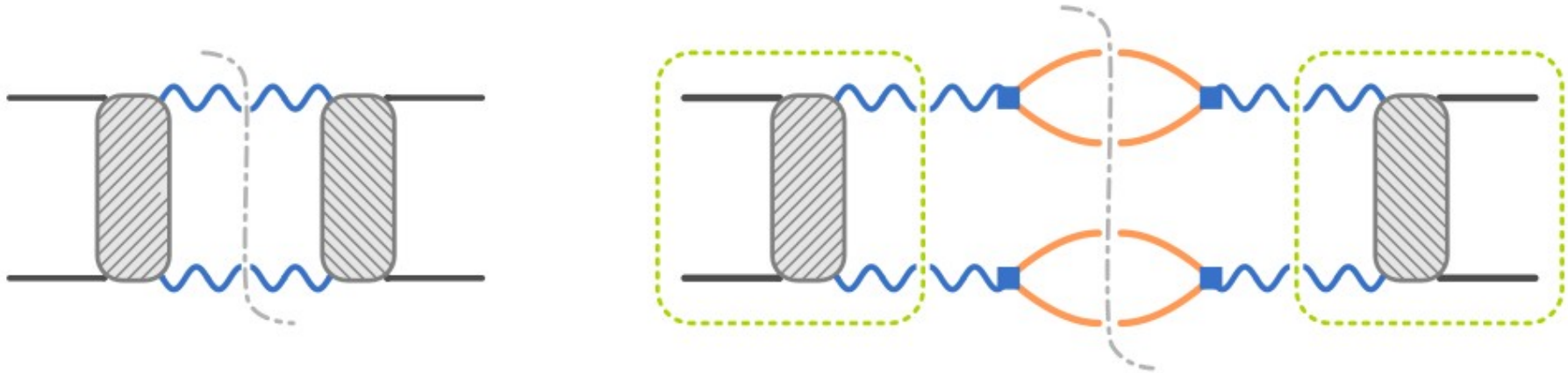
Awesome! But... does it generalize to higher dimensions!?

Higher dimensions!?

AMPLITUDE

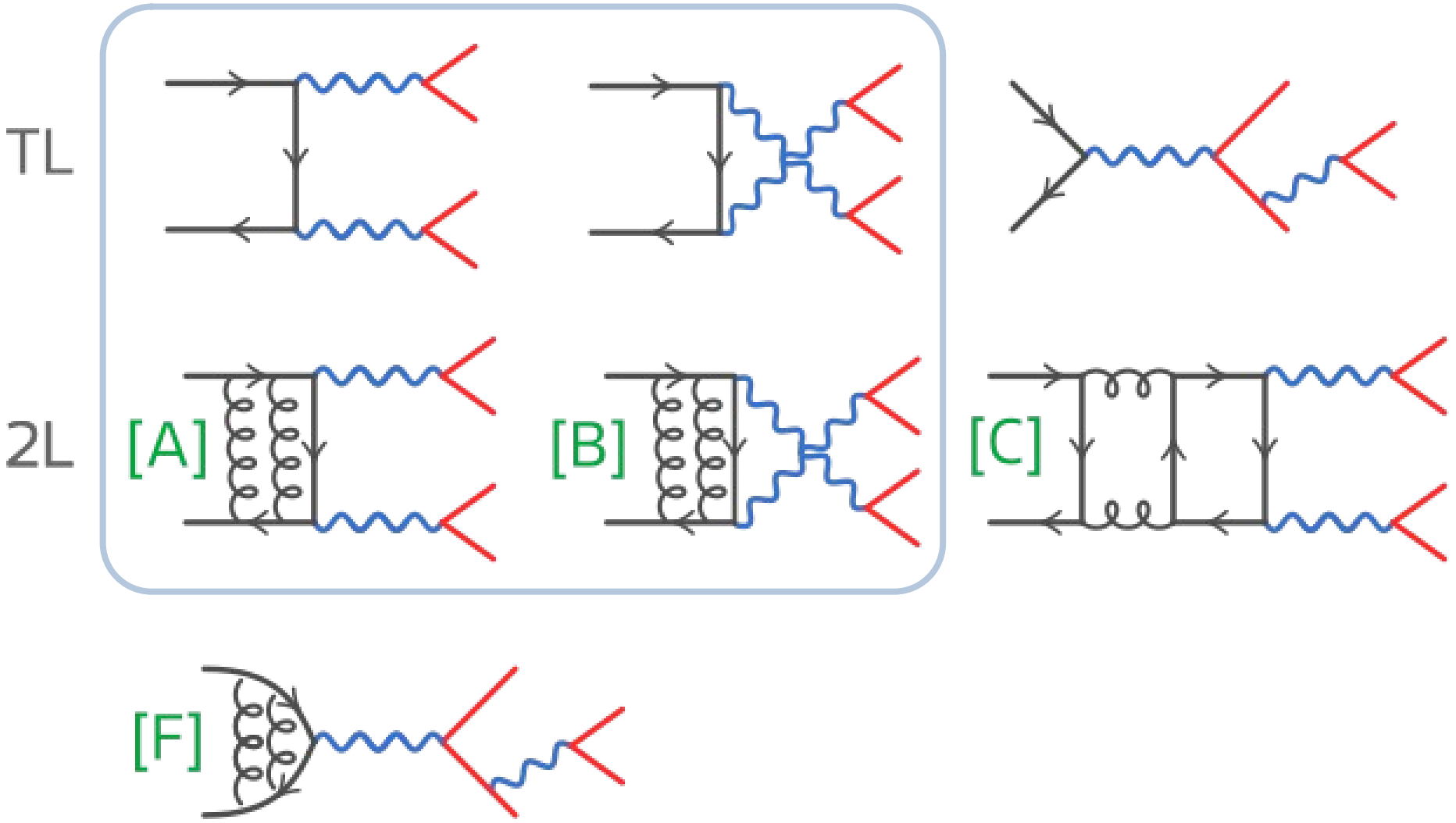


SQUARED AMPLITUDE

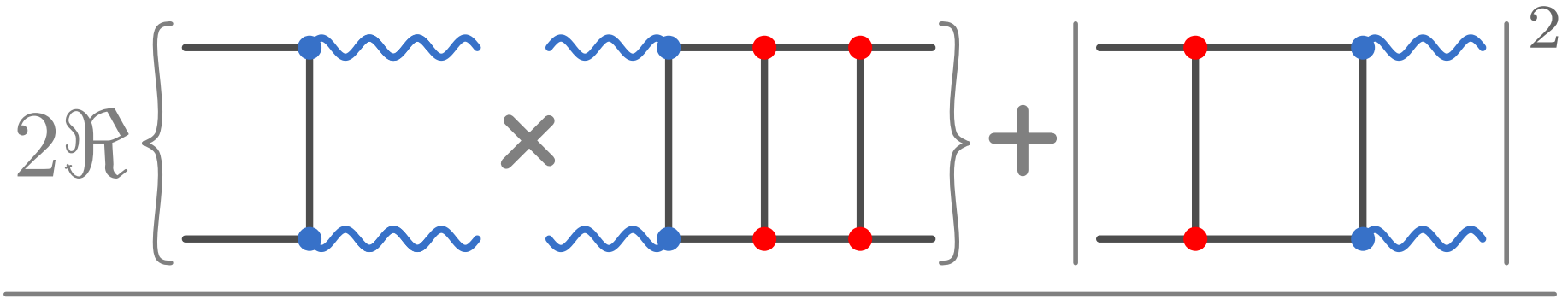


- Always average over initial and sum over final polarizations
- Off-shell case: "pretend" $Z(p_3)$ and $Z(p4)$ are on-shell but with masses $m_3 \neq m_4 \neq m_Z$

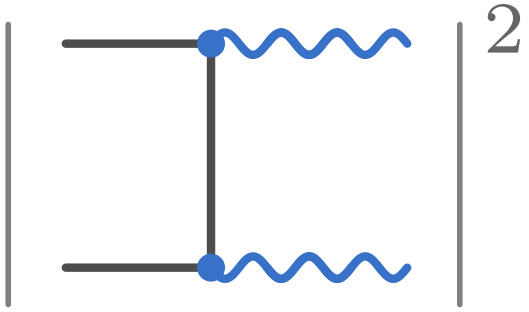
A few more details before we go for it…



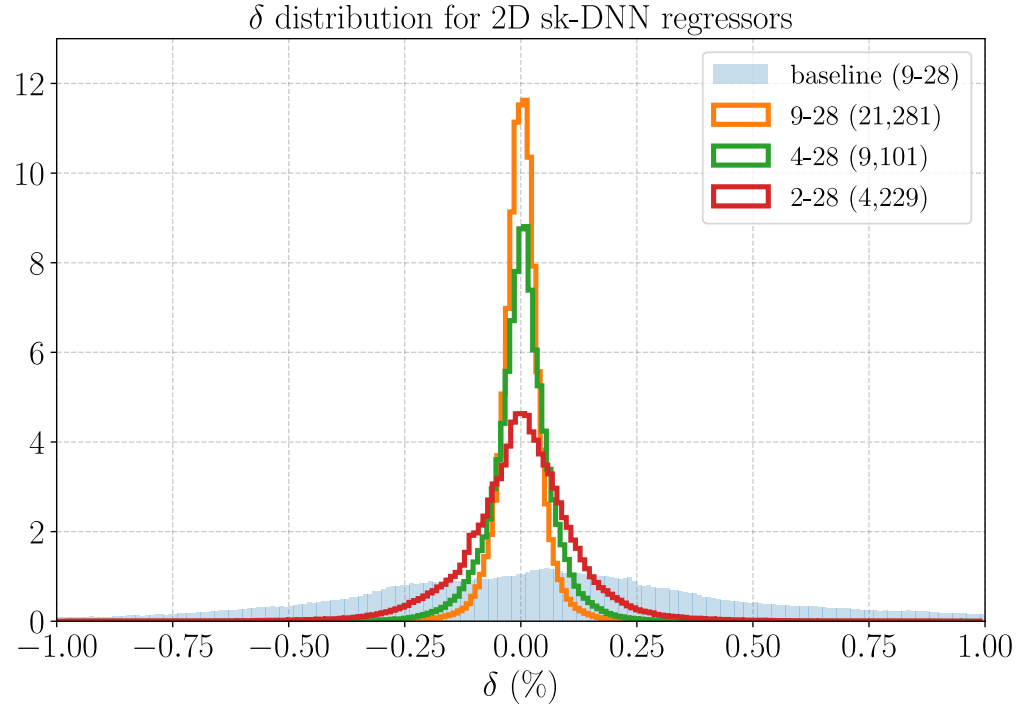
- "Toy" process just to establish generalization to higher dims.
  [FB, Ayan Paul, Jennifer Dy; https://ml4physicalsciences.github.io/2022/files/NeurIPS_ML4PS_2022_164.pdf]
  [FB, Ayan Paul, Jennifer Dy; [2301.XXXXX]]

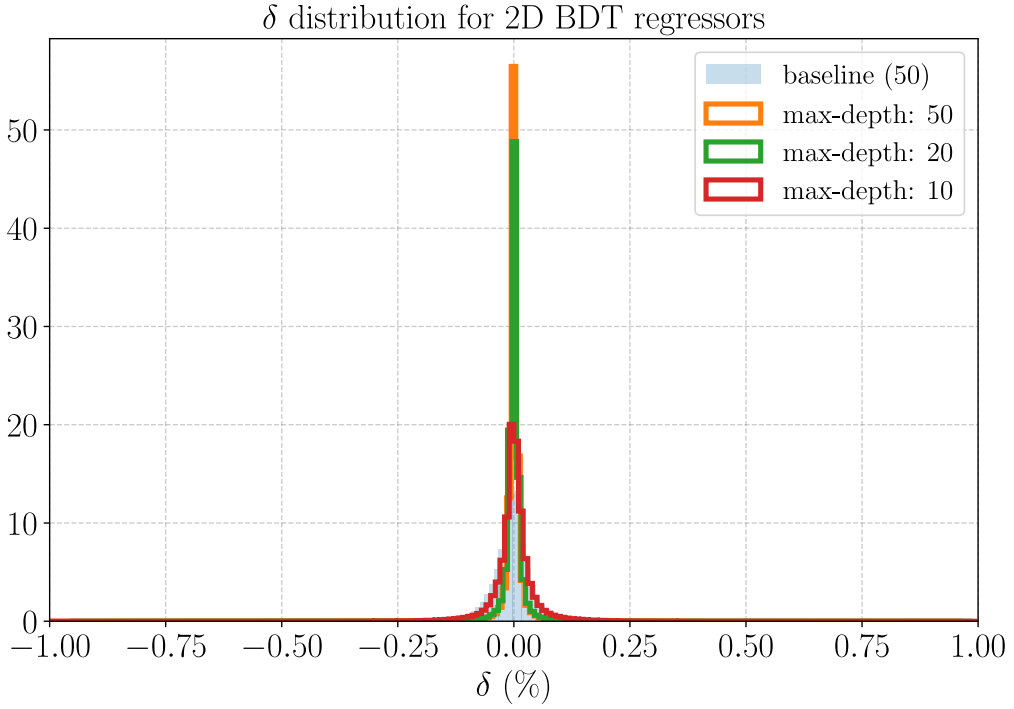


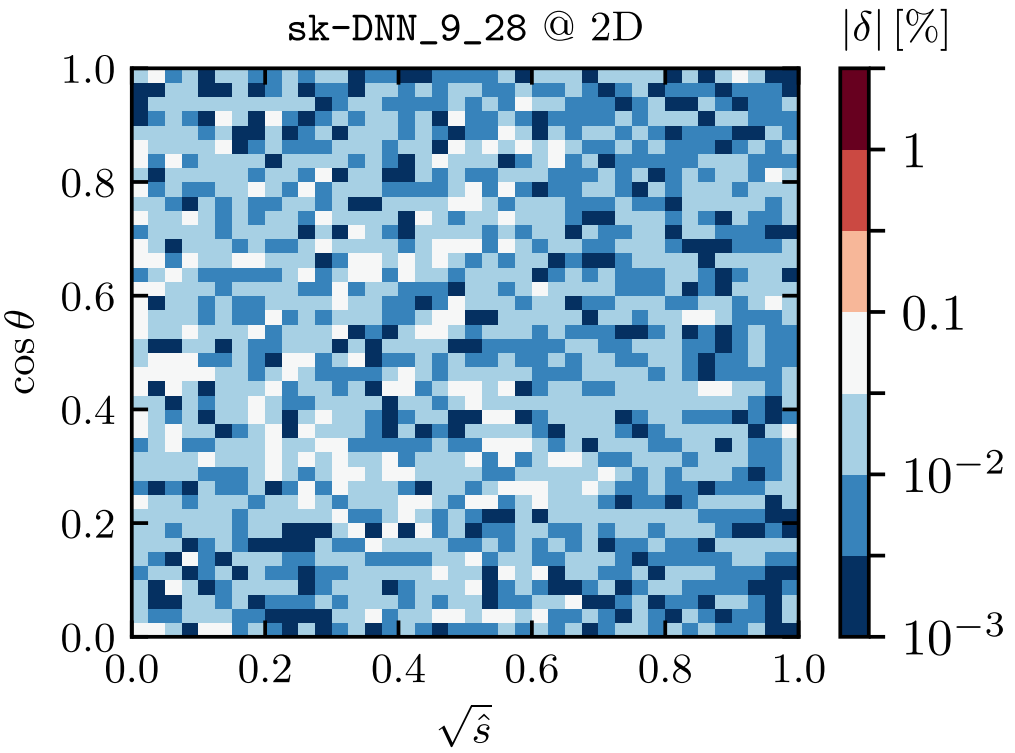
This study only includes classes [A] + [B]

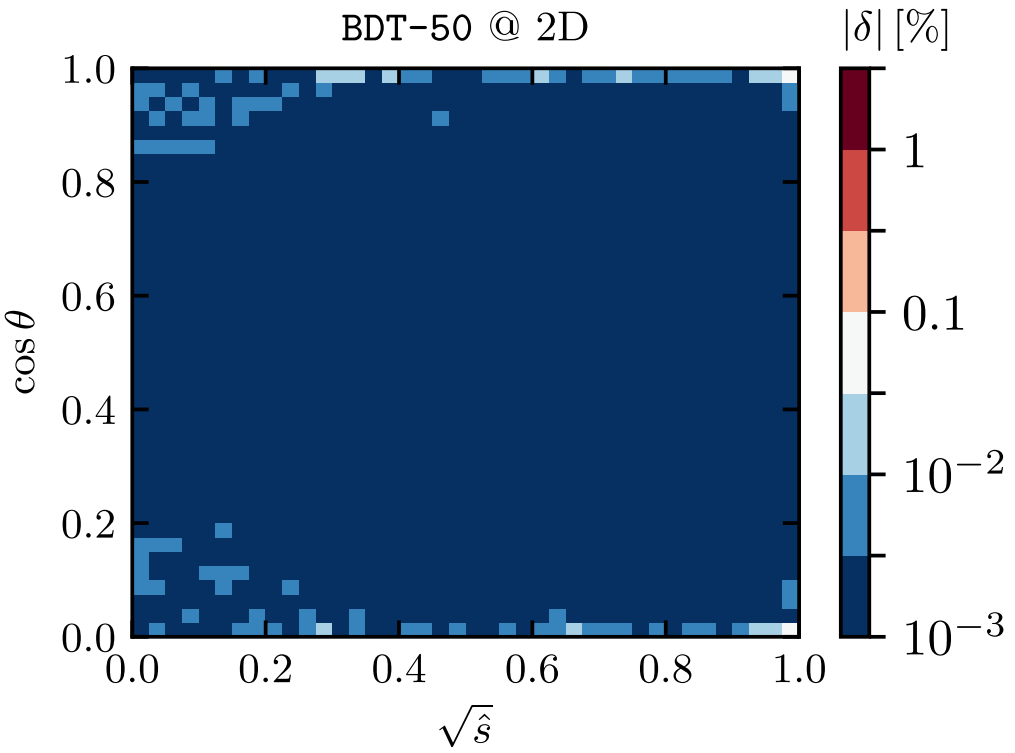


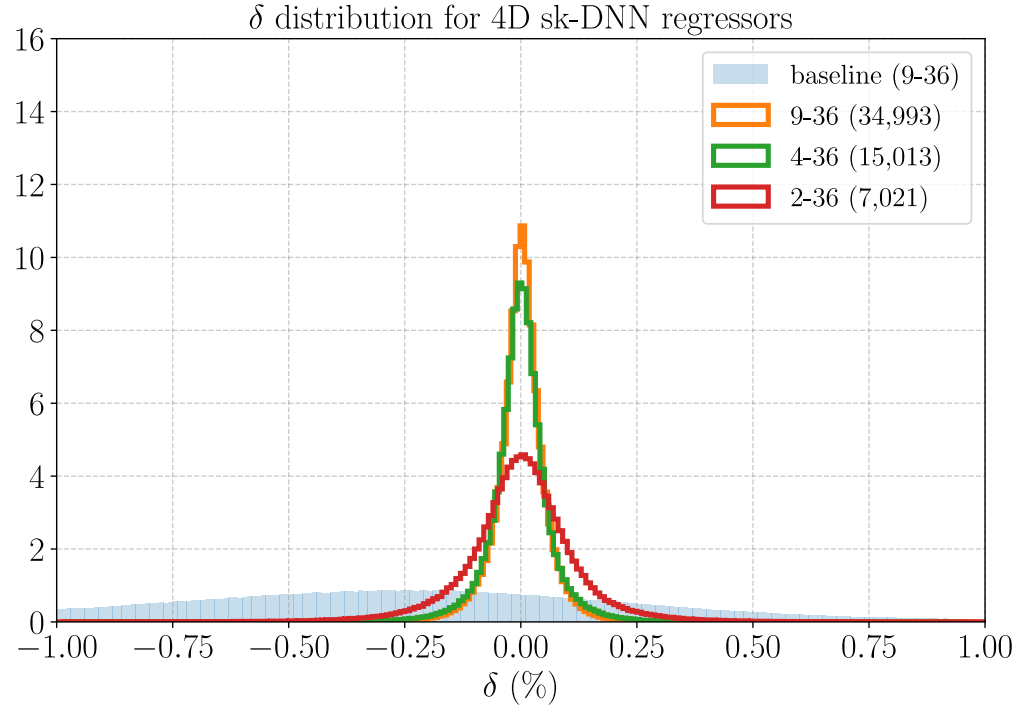
- Compute $\langle |\mathcal{M}|^2 \rangle$ for $qq \to ZZ(\to 4\ell)$ using `VVAMP`
  - Training: 4.8M points
  - Validation: 3.2M
  - Testing: 2M

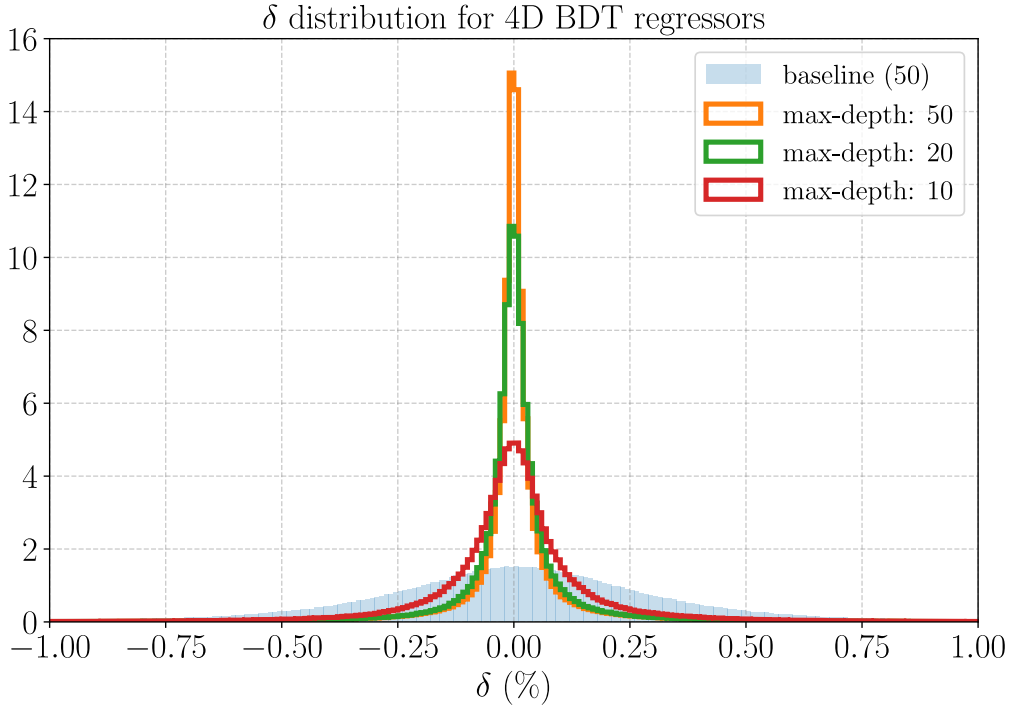

δ distribution for 2D sk-DNN regressors
baseline (9-28)
9-28 (21,281)
4-28 (9,101)
2-28 (4,229)
δ (%)

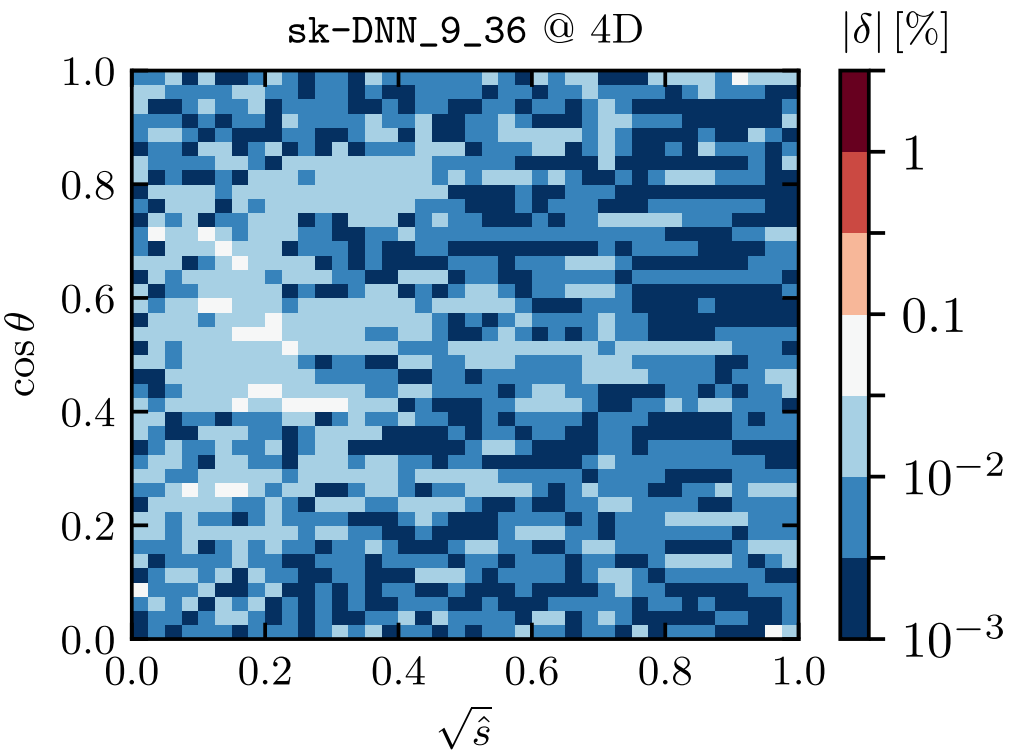

δ distribution for 2D BDT regressors
baseline (50)
max-depth: 50
max-depth: 20
max-depth: 10
δ (%)

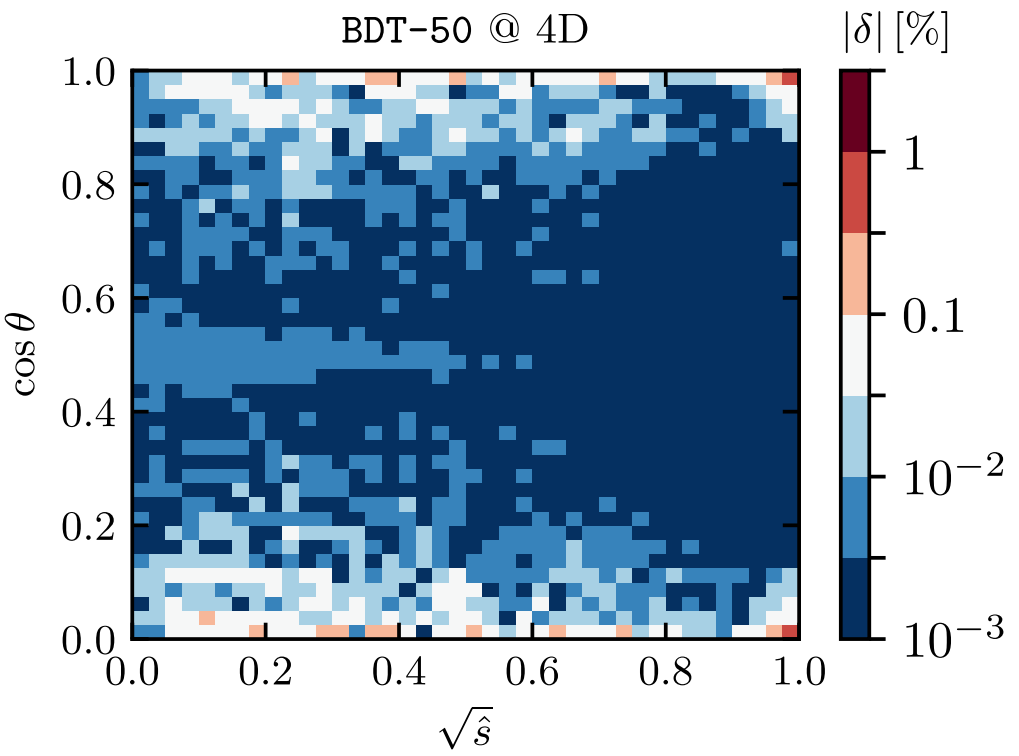

sk-DNN_9_28 @ 2D
|δ| [%]
cos θ
√ŝ

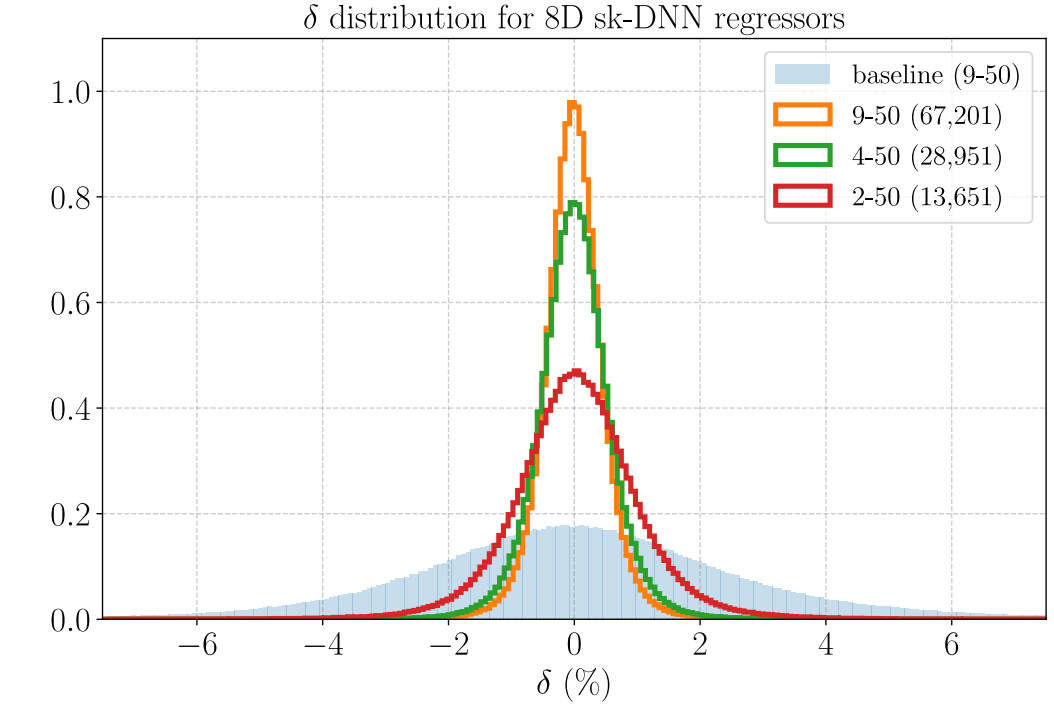

BDT-50 @ 2D
|δ| [%]
cos θ
√ŝ

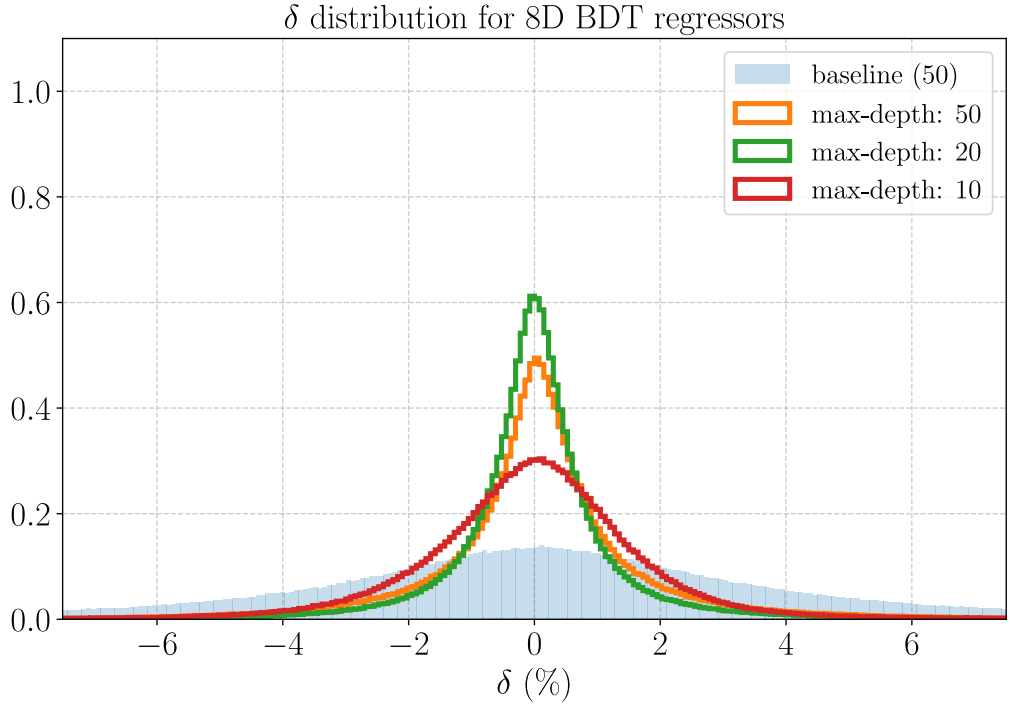

δ distribution for 4D sk-DNN regressors
baseline (9-36)
9-36 (34,993)
4-36 (15,013)
2-36 (7,021)
δ (%)

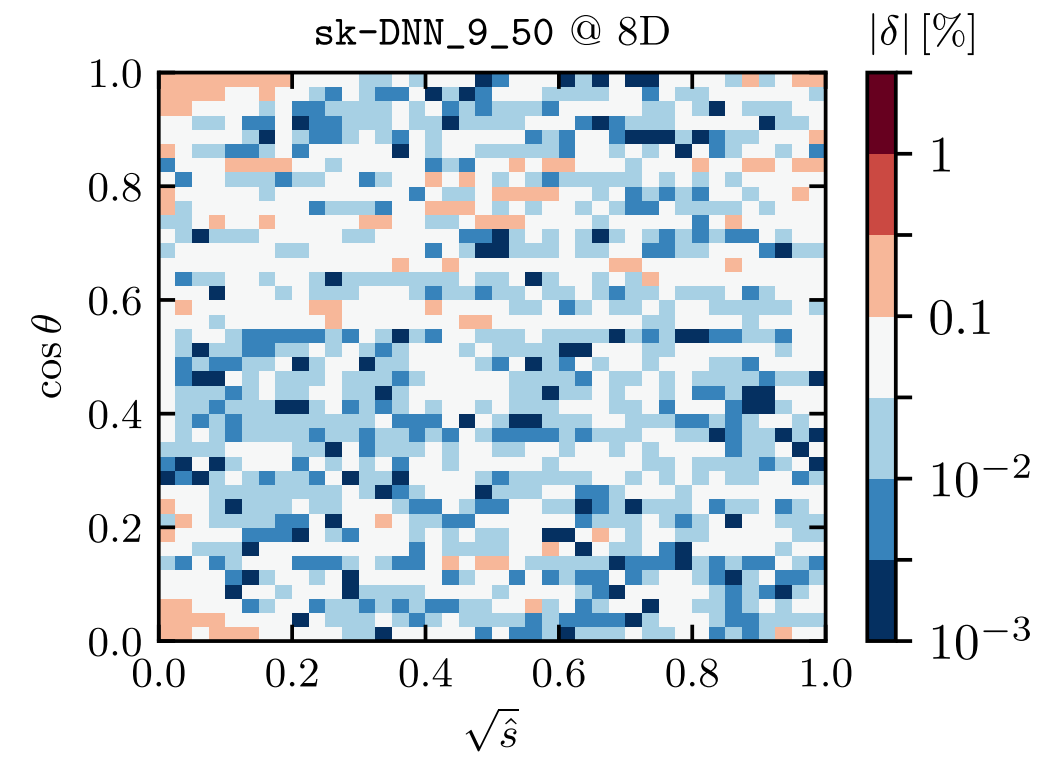

δ distribution for 4D BDT regressors
baseline (50)
max-depth: 50
max-depth: 20
max-depth: 10
δ (%)

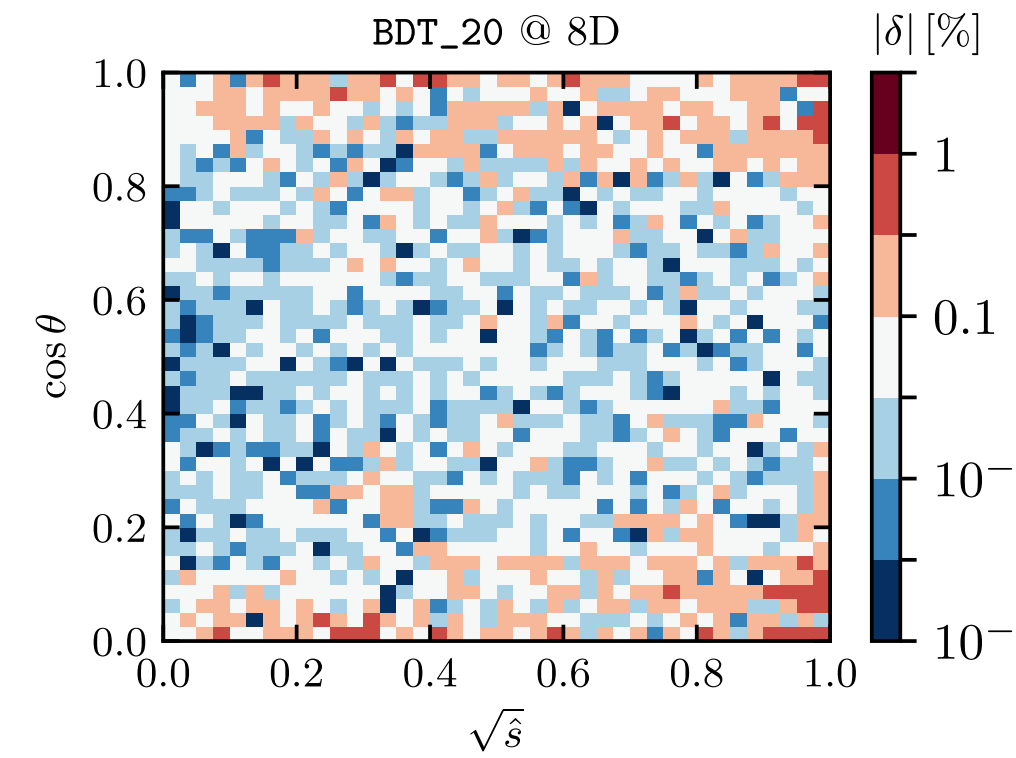

sk-DNN_9_36 @ 4D
|δ| [%]
cos θ
√ŝ


BDT-50 @ 4D
|δ| [%]
cos θ
√ŝ


δ distribution for 8D sk-DNN regressors
baseline (9-50)
9-50 (67,201)
4-50 (28,951)
2-50 (13,651)
1.0
0.8
0.6
0.4
0.2
0.0
−6
−4
−2
0
2
4
6
δ (%)


δ distribution for 8D BDT regressors
baseline (50)
max-depth: 50
max-depth: 20
max-depth: 10
1.0
0.8
0.6
0.4
0.2
0.0
−6
−4
−2
0
2
4
6
δ (%)


sk-DNN_9_50 @ 8D
|δ| [%]
1
0.1
10^{-2}
10^{-3}
1.0
0.8
0.6
0.4
0.2
0.0
cos θ
0.0
0.2
0.4
0.6
0.8
1.0
$\sqrt{\hat{s}}$


BDT_20 @ 8D
|δ| [%]
1
0.1
10^{-2}
10^{-3}
1.0
0.8
0.6
0.4
0.2
0.0
cos θ
0.0
0.2
0.4
0.6
0.8
1.0
$\sqrt{\hat{s}}$

So, it can be done…
but what about physics!?

Even more details before we go on...

[Work in progress with Ayan Paul]

- Goal: **implement into MC generators**, many details to consider
  - want functions that can be recycled $\rightarrow$ couplings factored out
  - must approximate **amplitudes** (i.e. not squared)
  - amplitudes are complex objects $f : \mathbb{R}^d \mapsto \mathbb{C}$
  - want V_1 and V_2 **off-shell** but don't want leptons so **4d**

- Therefore, have $2 \times 3 \times 3 = 18$ amplitudes in principle
  - ✔ Amplitudes have symmetries $\rightarrow$ reduced set

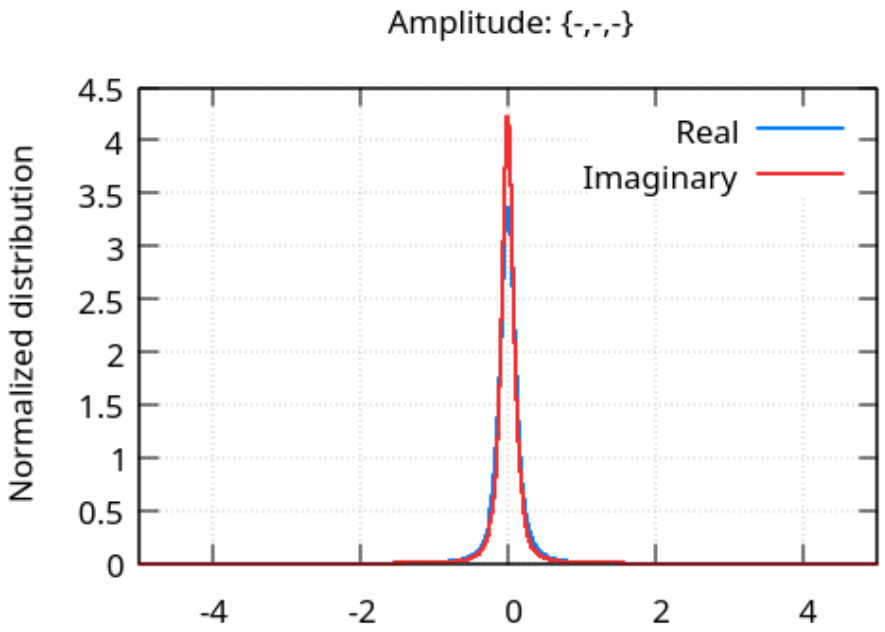
- Nice choice of reference momenta $\rightarrow$ more symmetry
  - simultaneous light-cone decomposition of p_3 and p_4 leads to

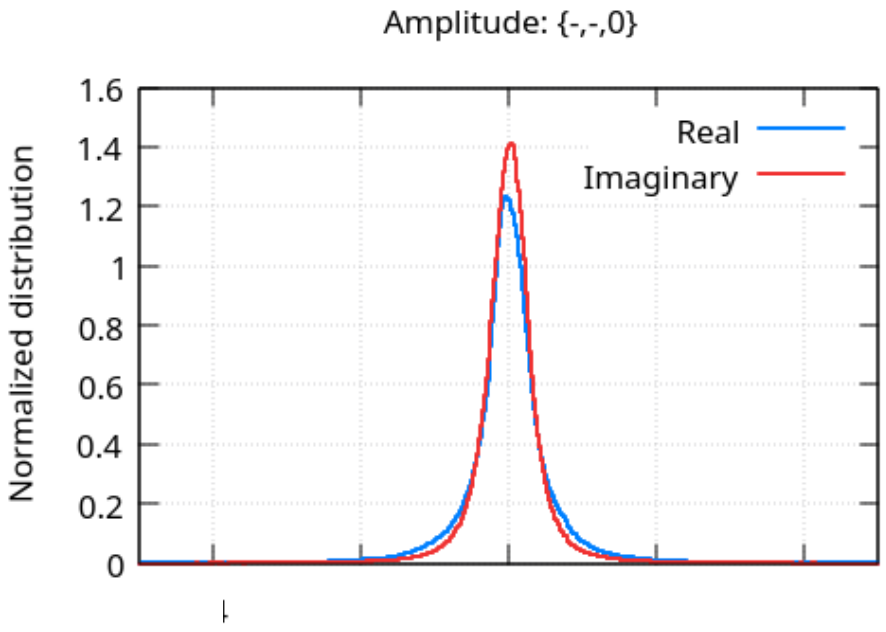
$$\epsilon^-_{3,\mu} = \frac{\langle 4\gamma_\mu 3]}{\sqrt{2}\langle 43\rangle}, \quad \epsilon^+_{3,\mu} = \frac{\langle 3\gamma_\mu 4]}{\sqrt{2}[34]}, \quad \epsilon^-_{4,\mu} = \frac{\langle 3\gamma_\mu 4]}{\sqrt{2}\langle 34\rangle}, \quad \epsilon^+_{4,\mu} = \frac{\langle 4\gamma_\mu 3]}{\sqrt{2}[43]}$$

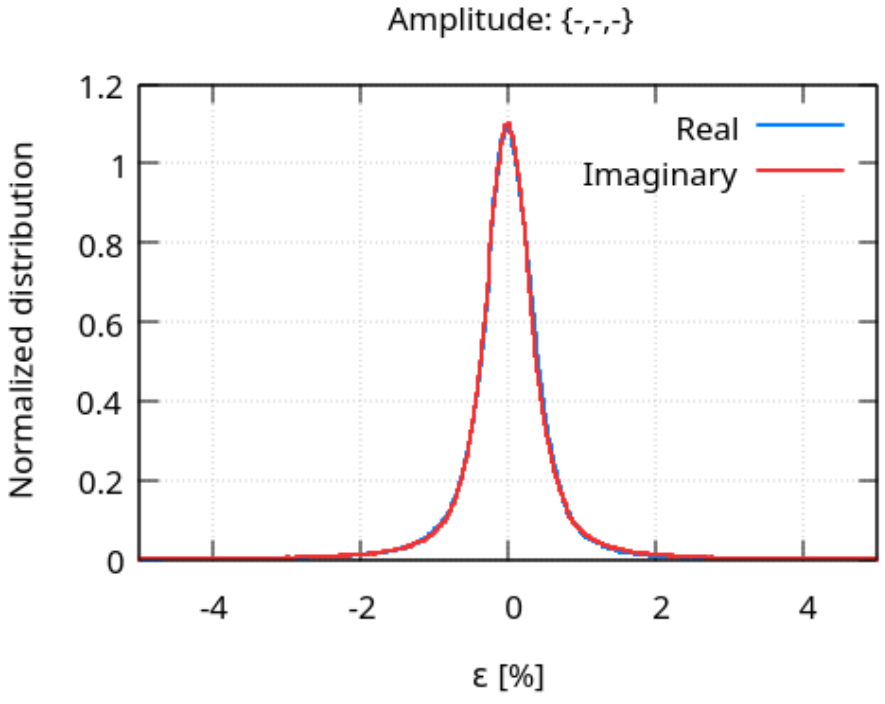
  - in C.M. frame with p_3 and p_4 pointed along $\pm\hat{z}$ direction and with appropriate choice of spinor phases, $\langle 34\rangle = [43]$

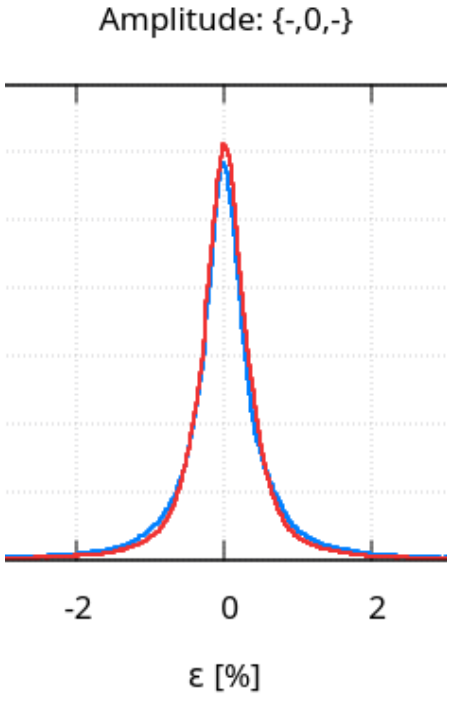
- In the end, only need **5** / 18 amplitudes
- $\Re$ and $\Im$ parts of the amplitudes are correlated $\rightarrow$ natural to output them together (trivial for NNs)
- In the future, could be a good application for complex activation functions
- For now, ignore complications that arise if the two pairs of leptons have the same flavor

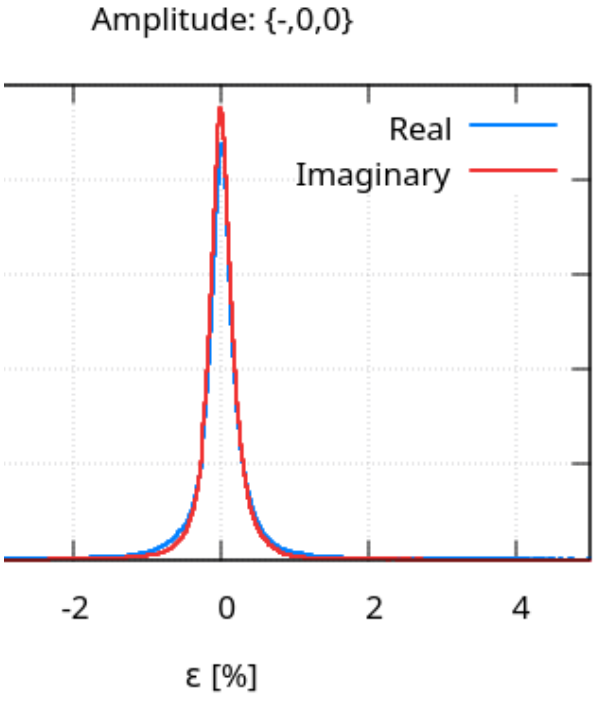
[PRELIMINARY] Fresh off the press!











Summary and outlook

- Approximate virtual amplitudes can leapfrog MC generation times
- Implementation, e.g., in GENEVA very soon
- Many many future directions and application to other amplitudes, e.g., including gluon-induced di-bosons @NLO, top mass in the loop, 5-point 3-photon two loop amplitude, etc.