Accurate Surrogate Amplitudes with Calibrated Uncertainties

Henning Bahl

based on
2412.12069 in collaboration with Nina Elmer, Luigi Favaro, Manuel Haußmann, Tilman Plehn, and Ramon Winterhalder

UNIVERSITÄT HEIDELBERG
ZUKUNFT
SEIT 1386

EPS-HEP Marseille, 8.7.2025

ML for particle physics — requirements

Precision

Speed

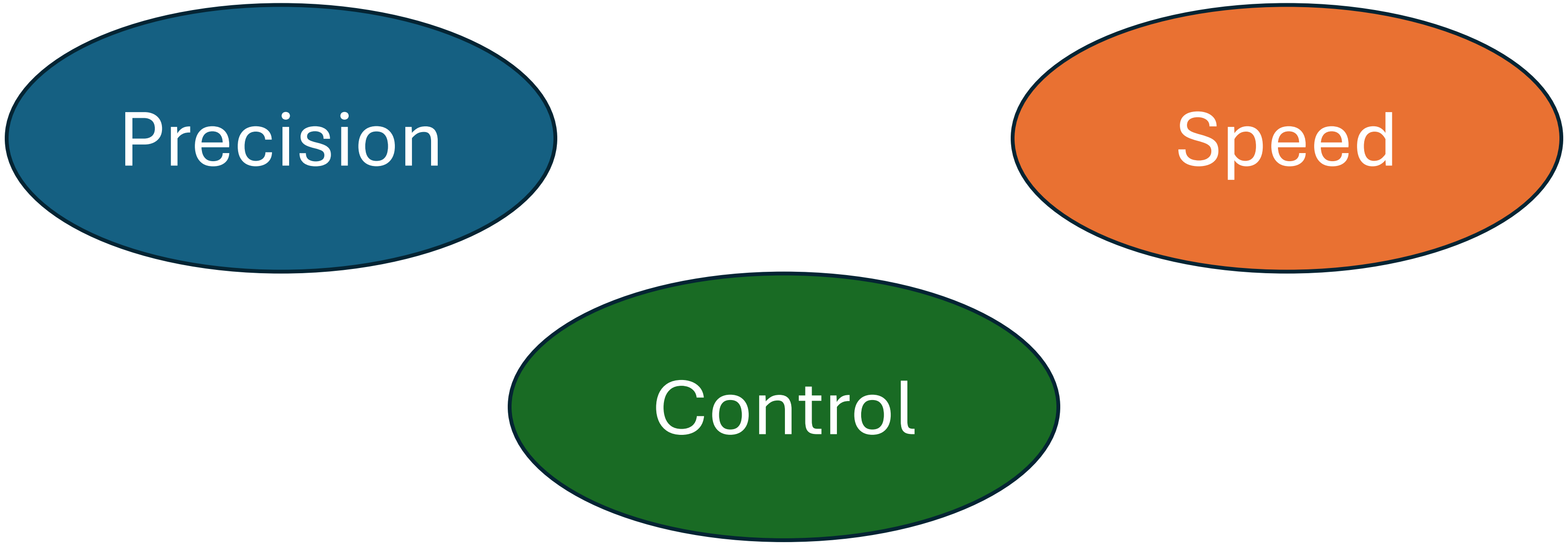
ML for particle physics — requirements

Precision

Speed

Control

ML for particle physics — requirements



"All models are wrong, but some — those that know when they can be trusted — are useful!"

— George Box (adapted)

Case study for amplitude surrogates

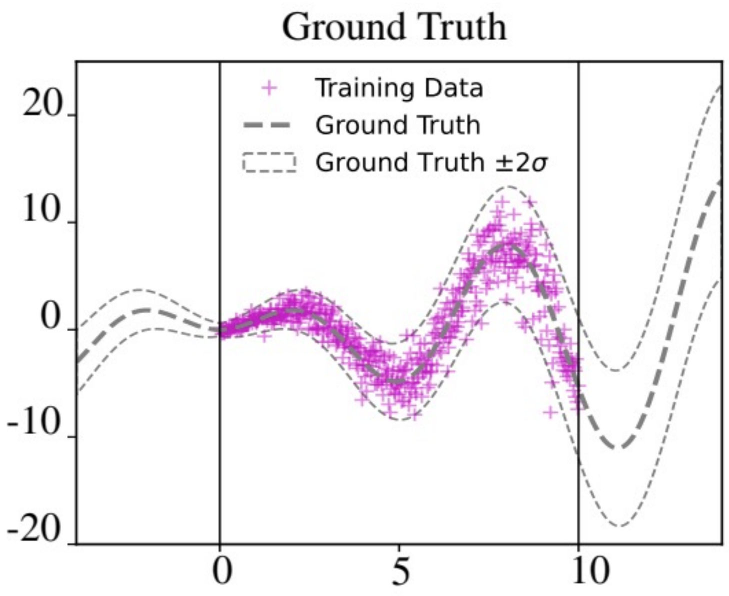
- evaluating analytic expressions for amplitudes $|\mathcal{M}|^2$ can be very expensive due to
  - higher-order corrections
  - large final-state multiplicities
  - studied here: $gg \to \gamma\gamma g$

- solution:
  - generate small training sample using full analytic expression
  - train a NN to approximate $|\mathcal{M}|^2$
  - generate events using NN surrogate, which is much faster to evaluate

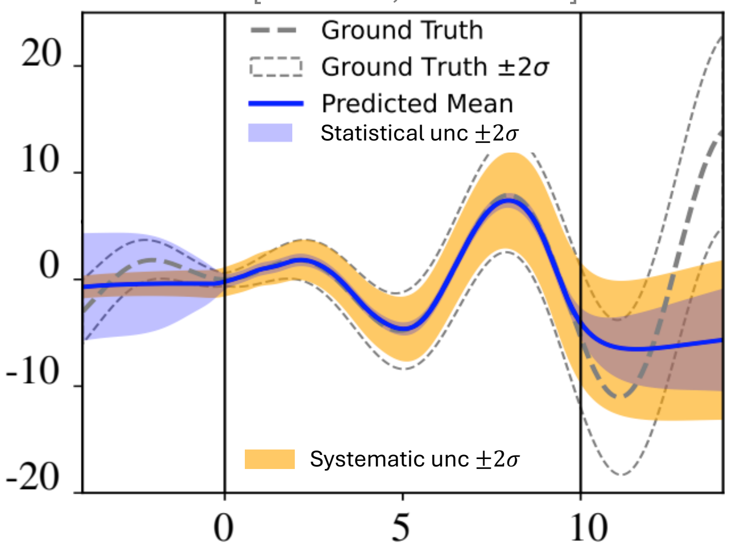
$\rightarrow$ speed-ups of by a factor of $\mathcal{O}(100)$ possible

Can we also control the uncertainties?

Regression with uncertainties



[Yi&Bessa, 2505.02743]



- statistical or epistemic uncertainty ≙ lack of training data
- systematic or aleatoric uncertainty ≙ noise in the data, lack in model expressivity

Systematic uncertainty: heteroskedastic loss

- log-likelihood loss:

sum over training dataset

true amplitudes

NN parameters

$$\mathcal{L} = -\sum_{x_i, A_i \in D_{\text{train}}} \log p(A_{\text{true}}(x_i) | x_i, \theta)$$

phase-space point

- assume Gaussian likelihood: $p(A|x,\theta) = \mathcal{N}(\bar{A}(x), \sigma_{\text{syst}}^2(x))$
- NN learns both: $\bar{A}(x)$ and $\sigma_{\text{syst}}(x)$

$$\Rightarrow \text{heteroskedastic loss: } \mathcal{L} = \sum_i \left[\frac{\left(\bar{A}(x_i) - A_{\text{true}}(x_i)\right)^2}{2\sigma_{\text{syst}}^2(x_i)} + \log\left(\sigma_{\text{syst}}(x_i)\right) \right]$$

- constant $\sigma_{\text{syst}} \rightarrow$ recovers MSE loss

Statistical uncertainty: repulsive ensemble

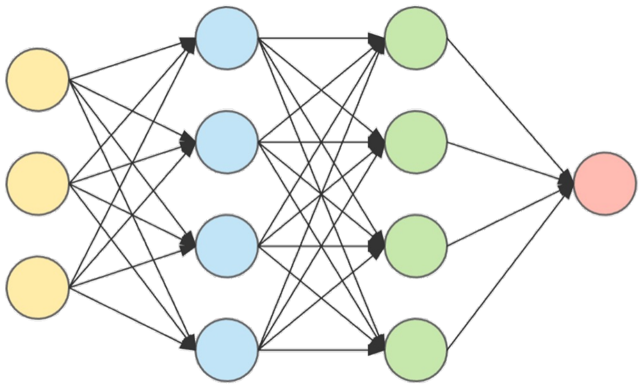
- train ensemble of networks

- ensure convergence to correct posterior via repulsive interaction between ensemble members

- each networks leads to slightly different result

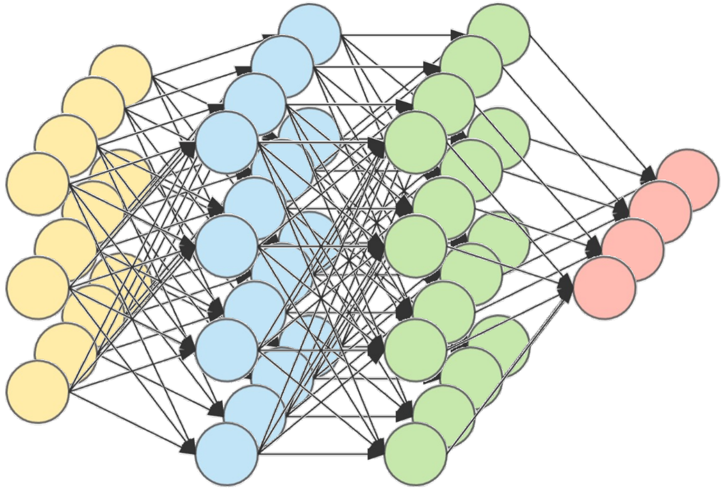
- spread of network predictions $\sim$ statistical uncertainty

- less data $\rightarrow$ higher spread

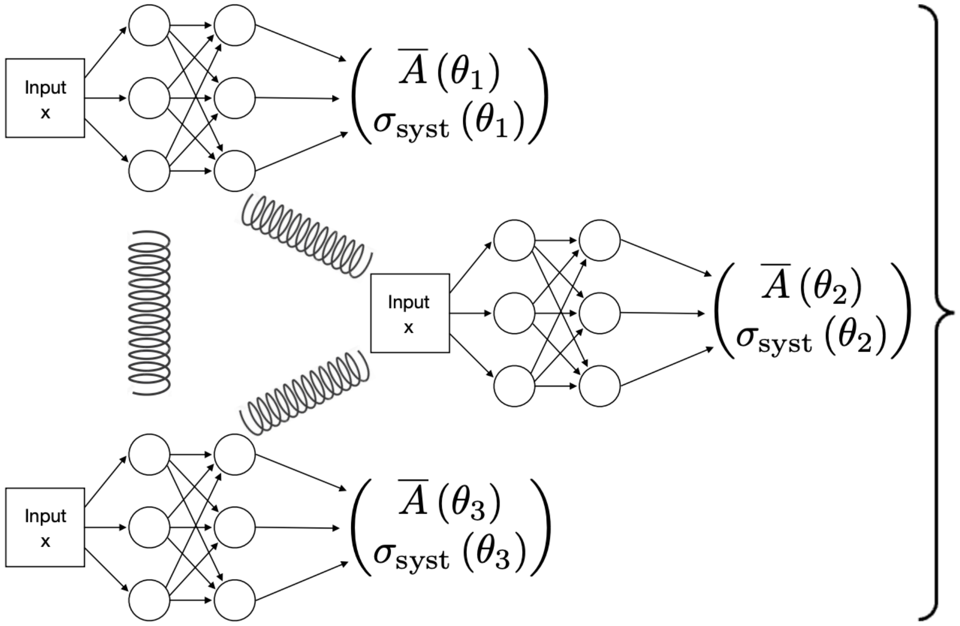


Statistical uncertainty: repulsive ensemble

- train ensemble of networks
- ensure convergence to correct posterior via repulsive interaction between ensemble members
- each networks leads to slightly different result
- spread of network predictions ~ statistical uncertainty
- less data → higher spread



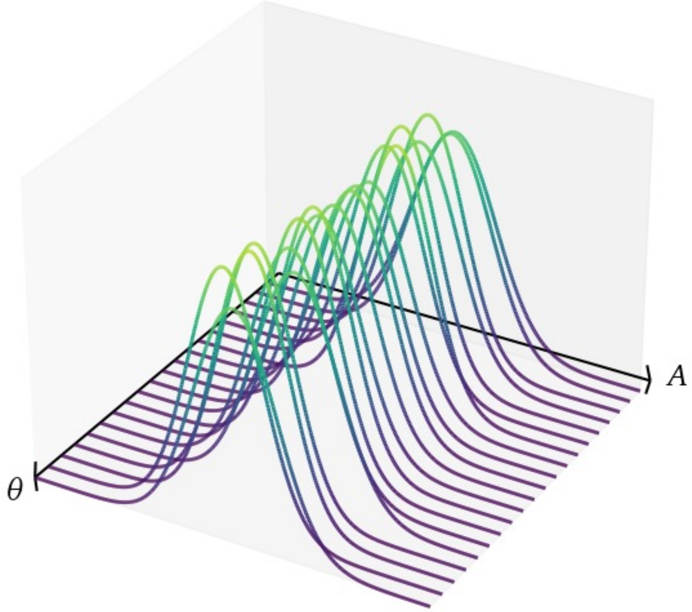
Repulsive ensemble + heteroskedastic loss



$$\langle A \rangle = \frac{1}{N} \sum_i^N \overline{A}(\theta_i)$$

$$\sigma_{\text{syst}}^2 = \frac{1}{N} \sum_i^N \sigma_{\text{syst}}^2(\theta_i)$$

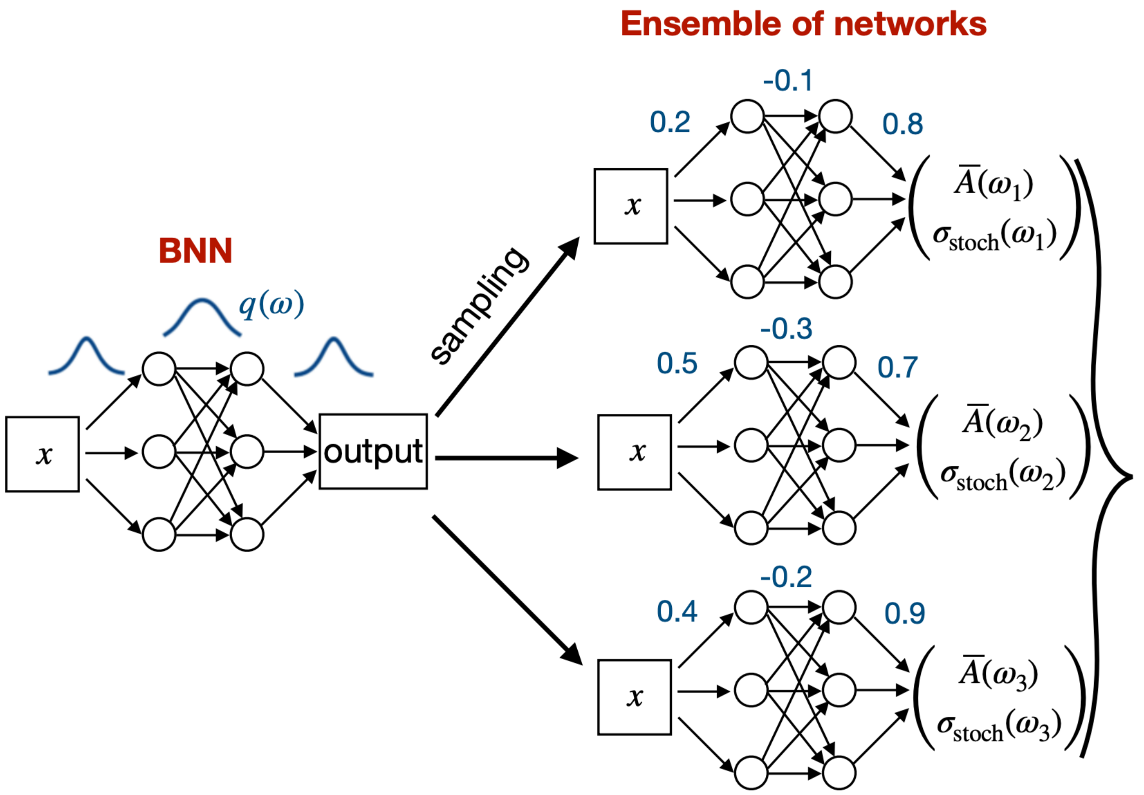
$$\sigma_{\text{stat}}^2 = \frac{1}{N} \sum_i^N \left(\langle A \rangle - \overline{A}(\theta_i) \right)^2$$



$$\mathcal{L}_{\text{RE}} = \sum_{i=1}^{n} \left[-\frac{1}{B} \sum_{b=1}^{B} \log p(x_b|\theta_i) + \frac{\beta}{N} \frac{\sum_{j=1}^{n} k(A_{\theta_i}(x), \overline{A_{\theta_j}(x)})}{\sum_{j=1}^{n} k(\overline{A_{\theta_i}(x)}, \overline{A_{\theta_j}(x)})} + \frac{\theta_i^2}{2N\sigma^2} \right]$$

Combined learnable modelling of systematic and statistical uncertainties!

Alternative: Bayesian NNs



Output

$$\langle A \rangle = \frac{1}{N}\sum_i^N \bar{A}(\omega_i)$$

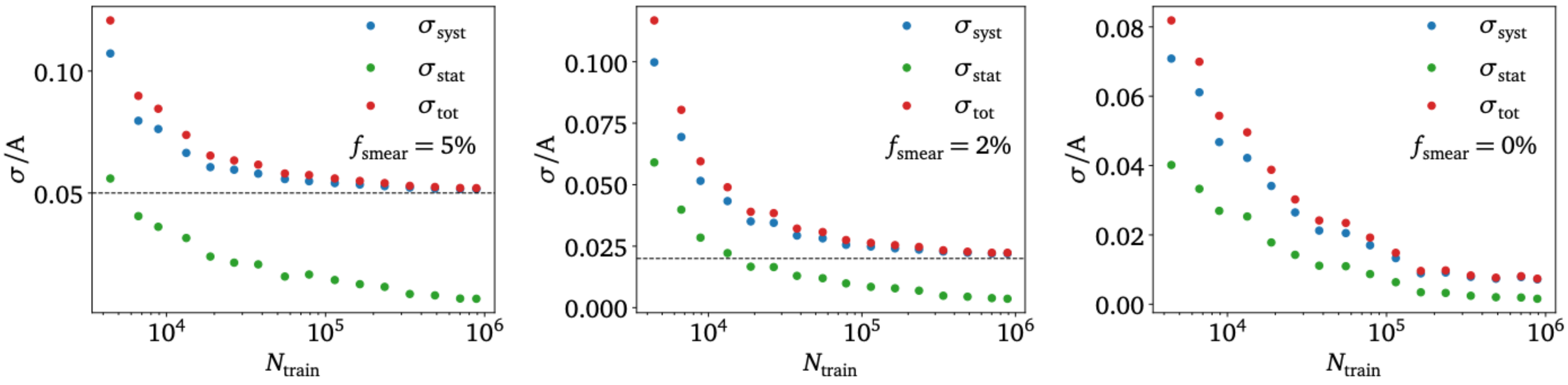
$$\sigma^2_{\text{stoch}} = \frac{1}{N}\sum_i^N \sigma^2_{\text{stoch}}(\omega_i)$$

$$\sigma^2_{\text{pred}} = \frac{1}{N}\sum_i^N \left(\langle A \rangle - \bar{A}(\omega_i)\right)^2$$

- promote NN parameters to Gaussians $q(\theta)$
- for each evaluation, sample from Gaussians
- learn means and widths

$$\mathcal{L}_{\text{BNN}} = \sum_x \left[\text{KL}[q(\theta), p(\theta)] - \left\langle \log p(D_{\text{train}}|\theta) \right\rangle_{\theta \sim q(\theta)} \right]$$

Behavior of learned uncertainties

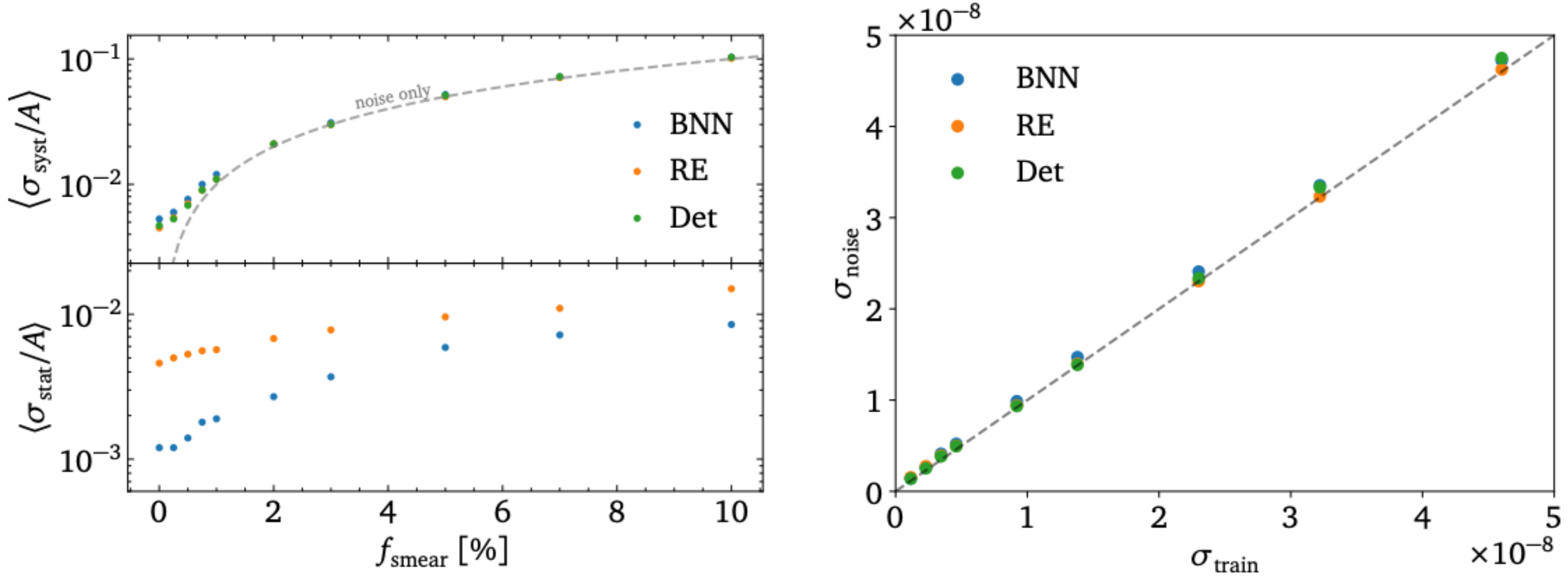


Test: apply different levels of Gaussian noise to amplitudes

$$A_{\text{train}} \sim \mathcal{N}(A_{\text{true}}, \sigma_{\text{train}}^2)$$
$$\sigma_{\text{train}} = f_{\text{smear}} A_{\text{true}}$$

- statistical uncertainty decreases with more training data
- systematic uncertainty converges to level of applied noise

Extracting the noise level



with $\sigma^2_{\text{noise}} = \sigma^2_{\text{syst}} - \sigma^2_{\text{syst},0}$ and $\sigma^2_{\text{syst},0}$ being the systematic unc. due to limited NN expressivity.

→ Able to reliably extract noise level!

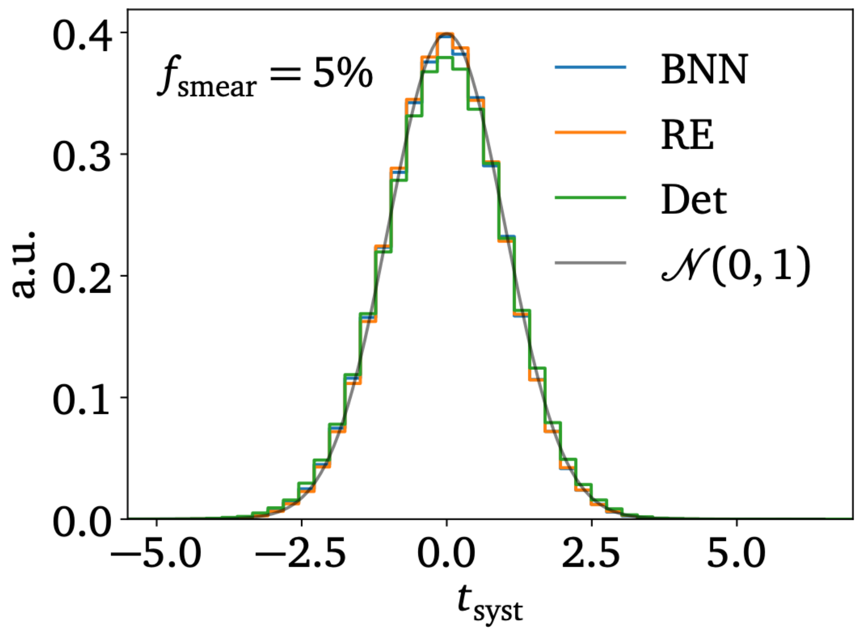
Are these uncertainties calibrated?

- statistical uncertainties play minor role for amplitude regression

- define systematic pull:

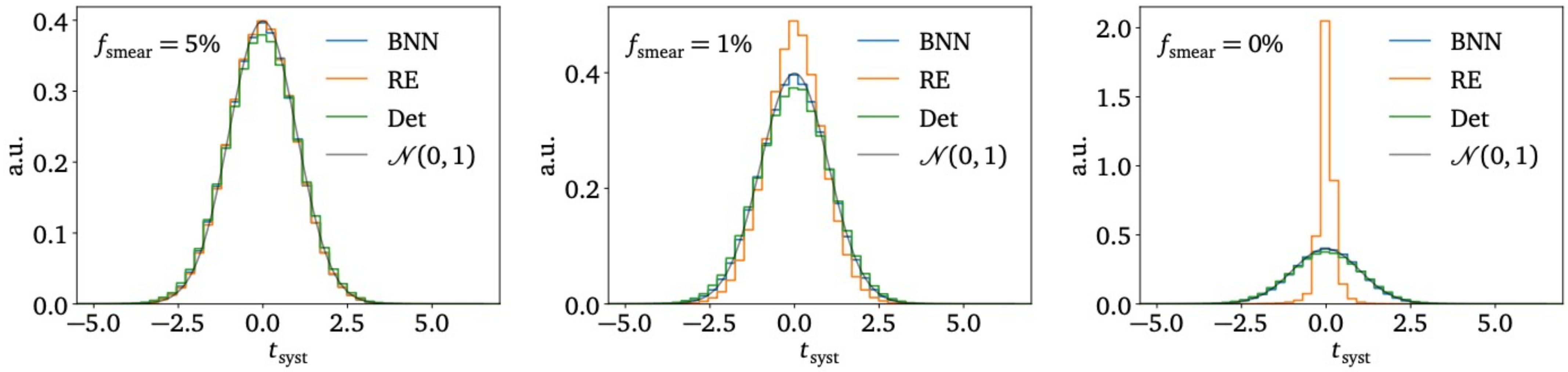
$$t_{\text{syst}} = \frac{\langle A \rangle(x) - A_{\text{train}}(x)}{\sigma_{\text{syst}}(x)}$$

- if calibrated, t_{syst} distribution should follow $\mathcal{N}(0,1)$



Almost perfectly calibration → reliable uncertainty estimate

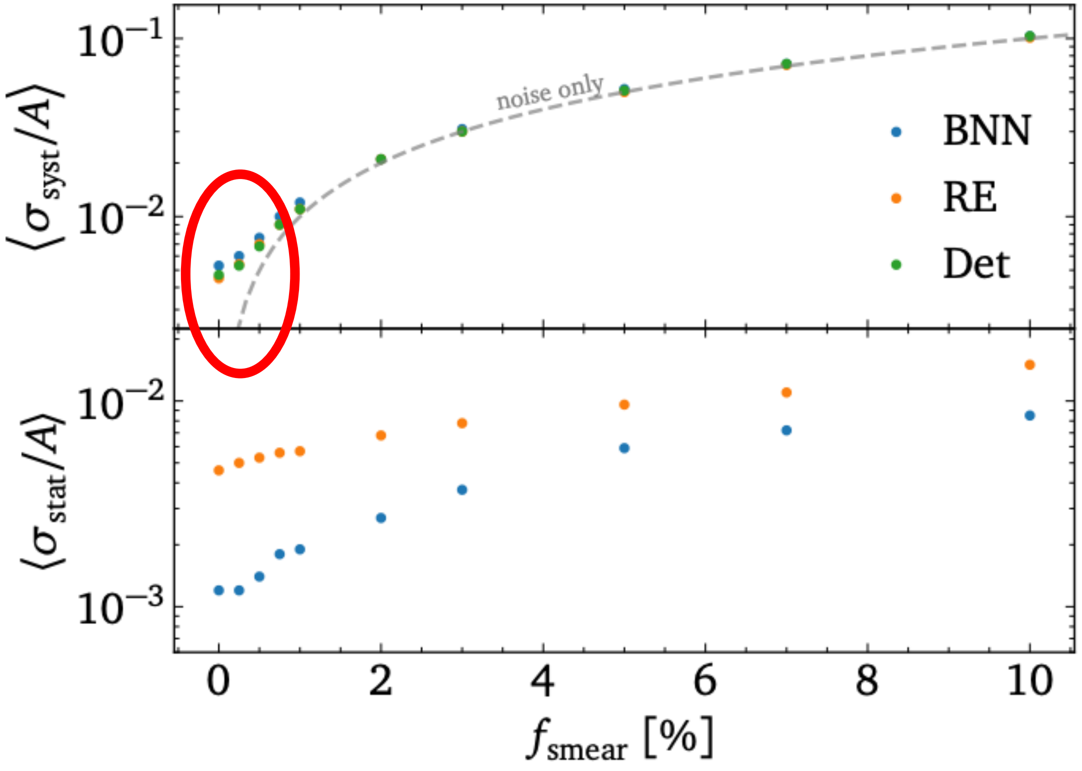
Dependence on smearing



- BNN and deterministic models
  → well calibrated for different smearing levels

- Repulsive ensemble overestimates uncertainty for low smearing
  → consequence of ensemble being more precise than single member

Enhancing NN expressivity



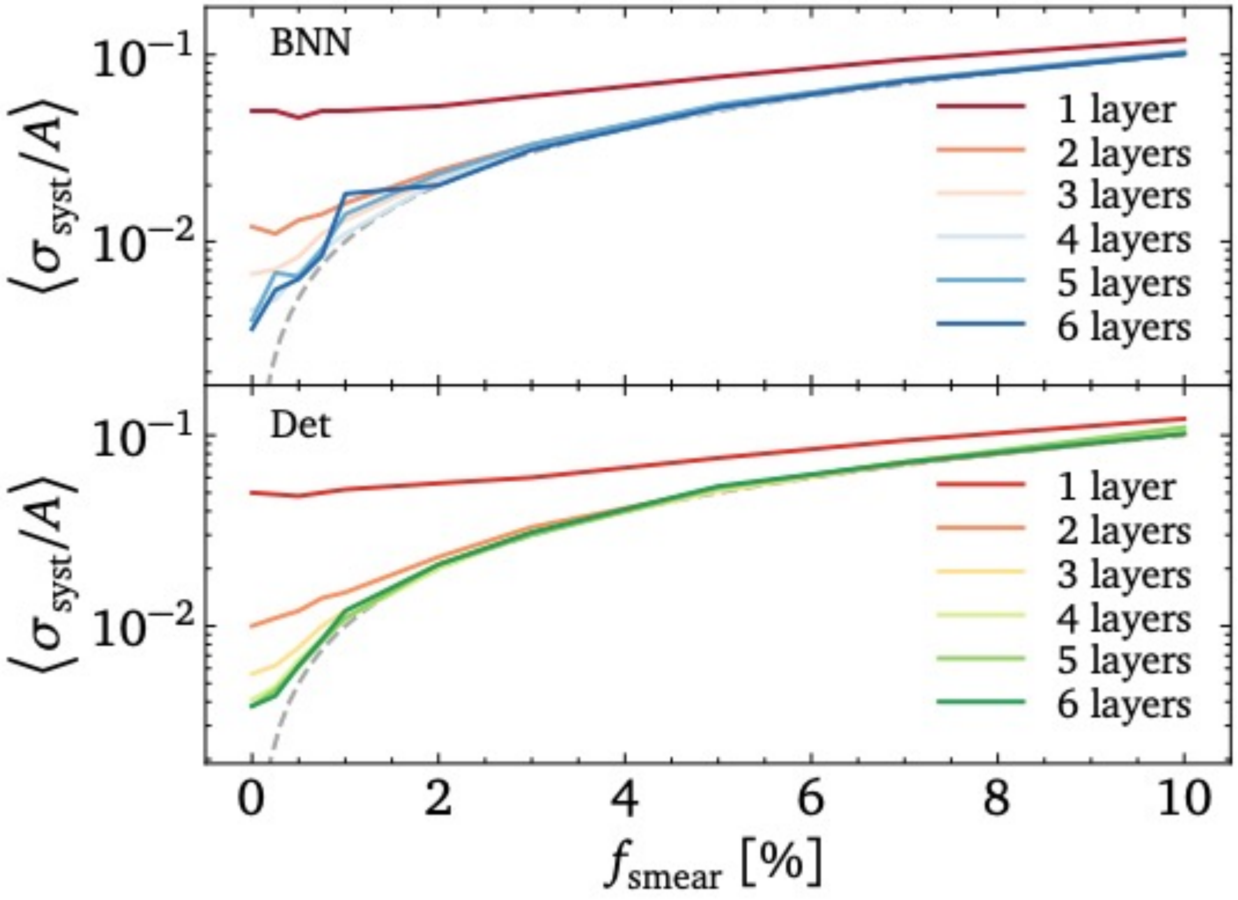
Can we improve accuracy for low smearing?

Improve NN expressivity:

- more layers,
- Lorentz invariance,
- permutation invariance between identical particles

Also tested: KANs, activation functions, ...

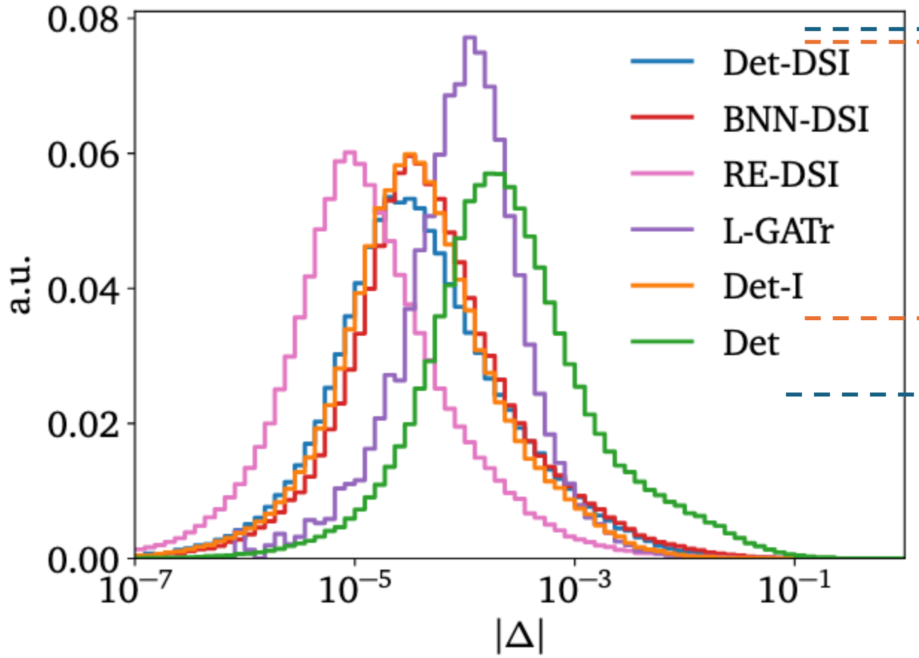
Adding more layers



adding more layers

convergence towards noise level

Encoding our physics knowledge



Permutation invariant

Lorentz invariant

relative deviation: $\Delta = \frac{A_{\mathrm{NN}} - A_{\mathrm{true}}}{A_{\mathrm{true}}}$

Large gain in NN accuracy! Also found uncertainties still to be well calibrated.

Conclusions

- amplitude surrogates → speed up MC generation
- uncertainty-aware NNs allow for controlled modelling
- systematic uncertainties: heteroskedastic loss
- statistical uncertainties: Bayesian neural networks + repulsive ensembles
- found well-calibrated uncertainties
- encoding physics knowledge increase accuracy with still well-calibrated unc.

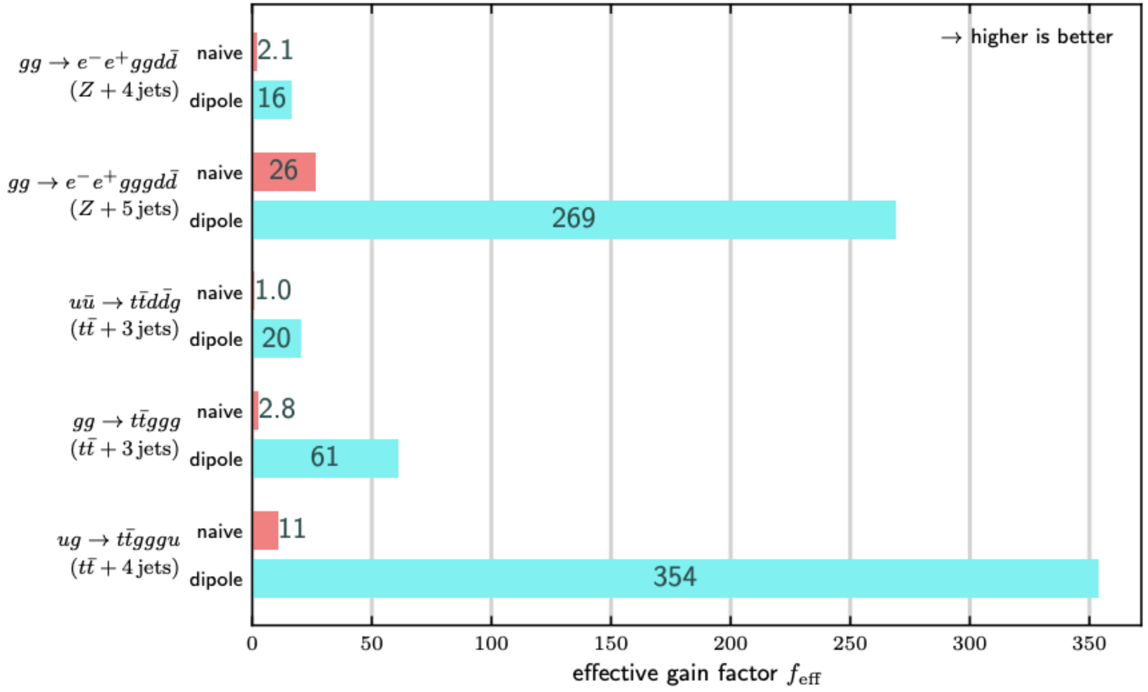
➡ Same techniques also applicable to all kind of other problems

Appendix

Does it work?

[Janßen et al.,2301.13562]

$$f_{\text{eff}} = \frac{T_{\text{standard}}}{T_{\text{surrogate}}}$$



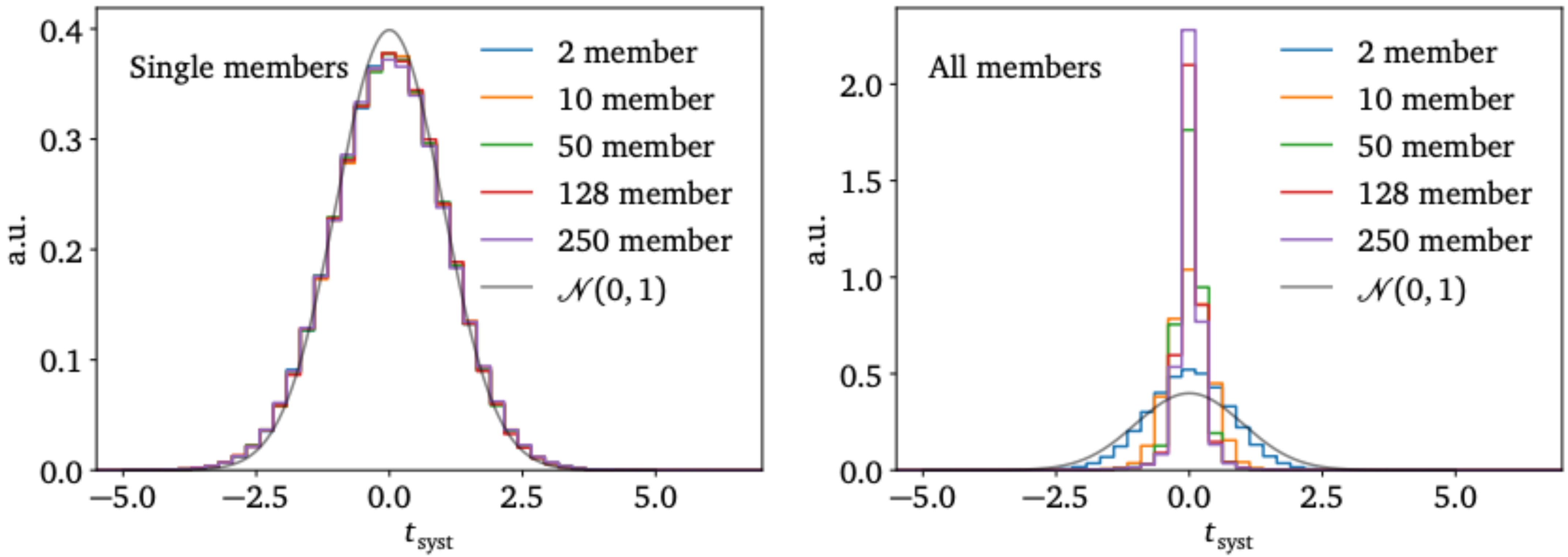
	SHERPA default			with dipole-model surrogate				
Process	t_{ME}[ms]	t_{PS}[ms]	ϵ_{full}	t_{surr}[ms]	x_{max}	$\epsilon_{\text{1st,surr}}$	$\epsilon_{\text{2nd,surr}}$	f_{eff}
$gg \to e^- e^+ gg d\bar{d}$	54	0.40	1.411 %	0.14	2.6	1.418 %	39 %	16
$gg \to e^- e^+ ggg d\bar{d}$	16 216	5.70	0.076 %	0.20	3.6	0.085 %	29 %	269

→ Large speed-ups possible!

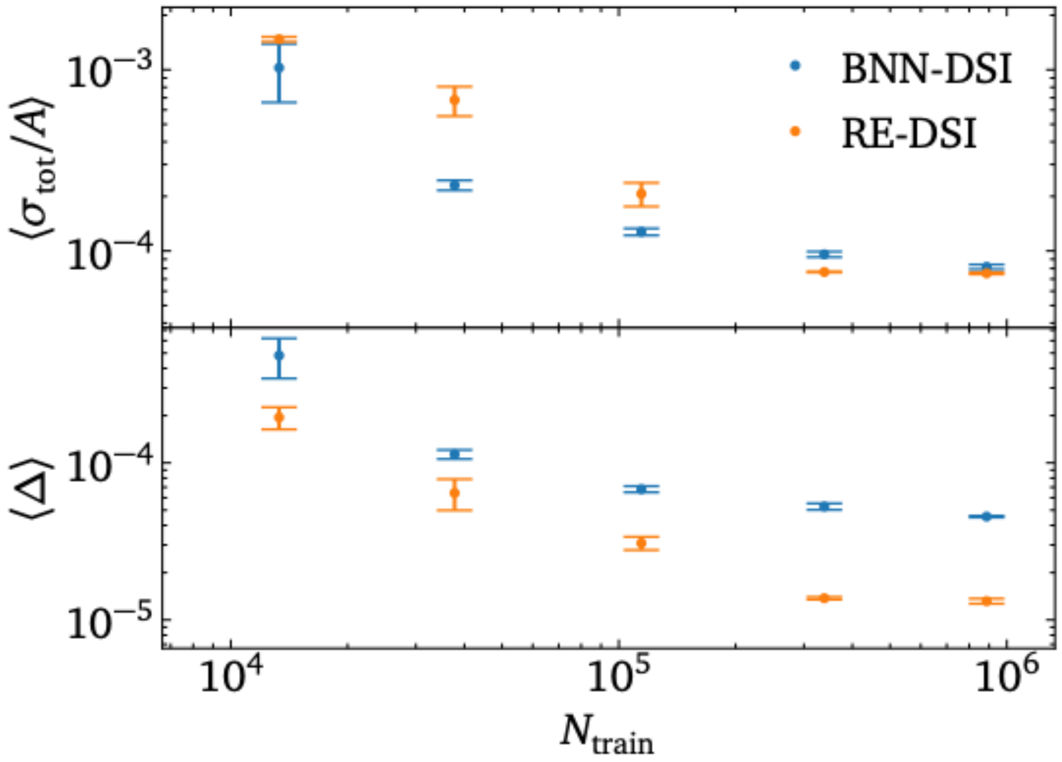
↓

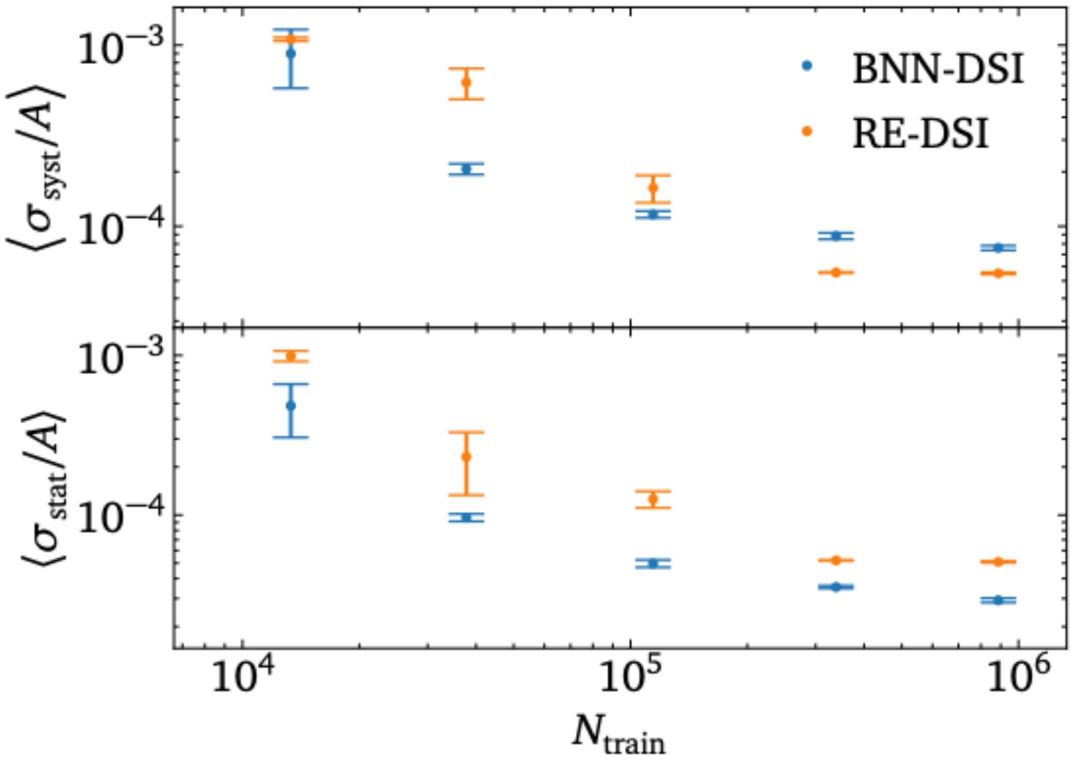
Can we also control the uncertainties?

Repulsive ensemble calibration

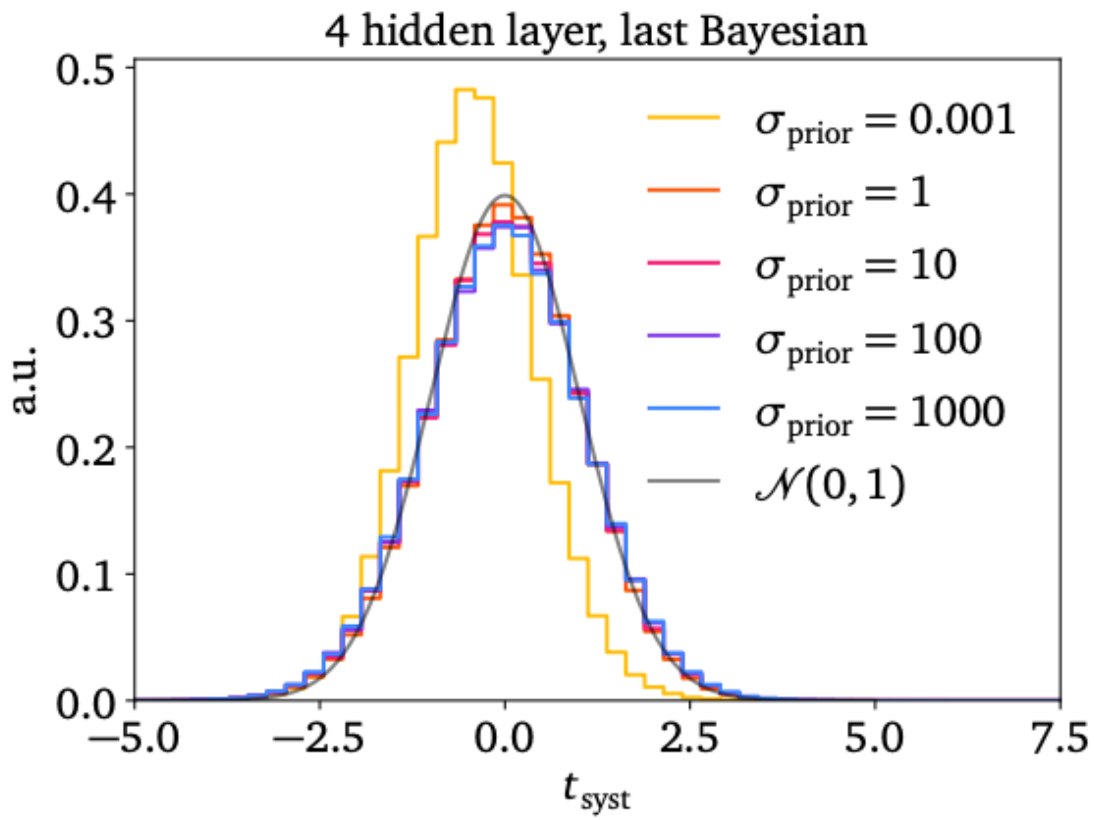


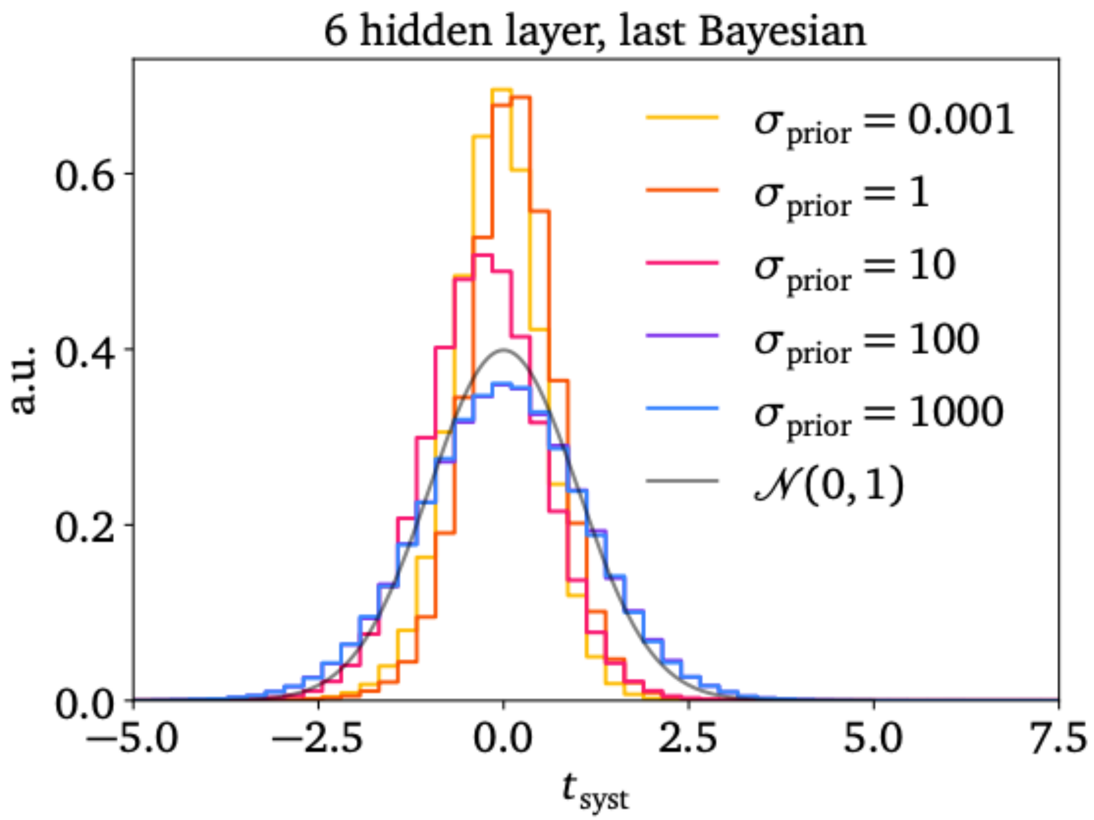
Uncertainty overview





BNN prior dependence





Kolmogorov-Arnold Networks

- KAN layer:

$$x_{l+1} = \underbrace{\begin{pmatrix} \phi_{l,1,1}(\cdot) & \phi_{l,1,2}(\cdot) & \cdots & \phi_{l,1,n_l}(\cdot) \\ \phi_{l,2,1}(\cdot) & \phi_{l,2,2}(\cdot) & \cdots & \phi_{l,2,n_l}(\cdot) \\ \vdots & \vdots & \ddots & \vdots \\ \phi_{l,n_{l+1},1}(\cdot) & \phi_{l,n_{l+1},2}(\cdot) & \cdots & \phi_{l,n_{l+1},n_l}(\cdot) \end{pmatrix}}_{\equiv \Phi_l} x_l \, ,$$

- KAN network: $\mathrm{KAN}(x) = (\Phi_{L-1} \circ \Phi_{L-2} \circ \ldots \circ \Phi_1 \circ \Phi_0)x$

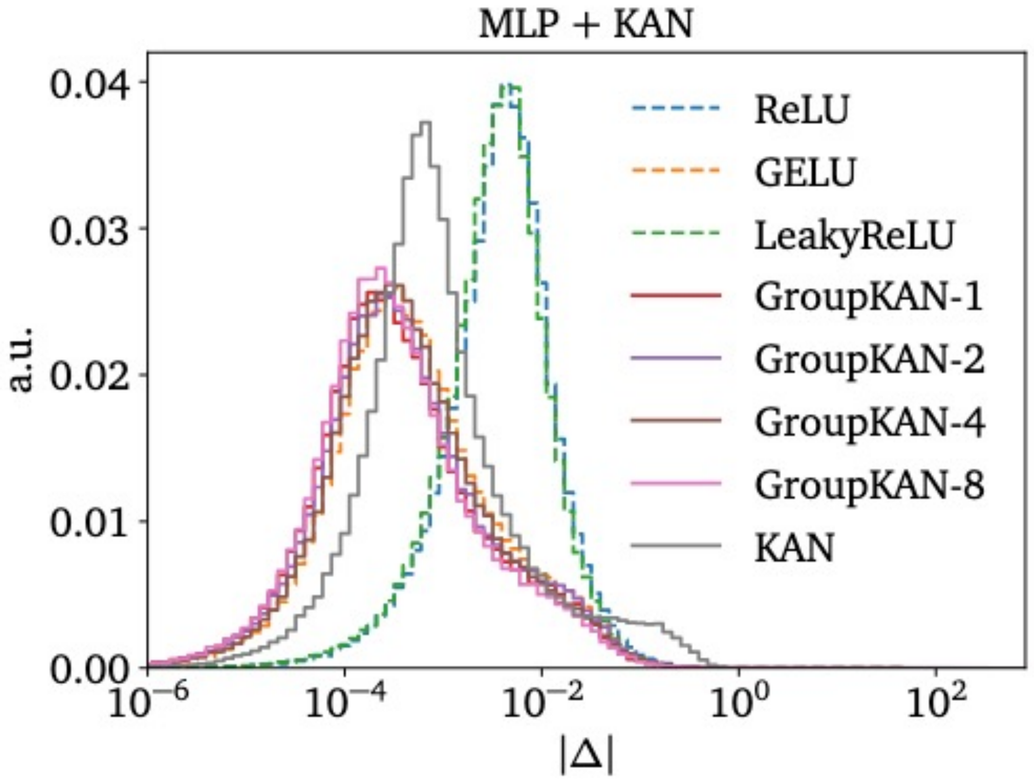
- GroupKAN layer:

$$\mathrm{activation}(x) \to \mathrm{GroupKANlayer}_l(x) = \begin{pmatrix} \phi_{l,g_l(1)}(x_1) \\ \phi_{l,g_l(2)}(x_2) \\ \vdots \\ \phi_{l,g_l(n_l)}(x_{n_l}) \end{pmatrix}$$

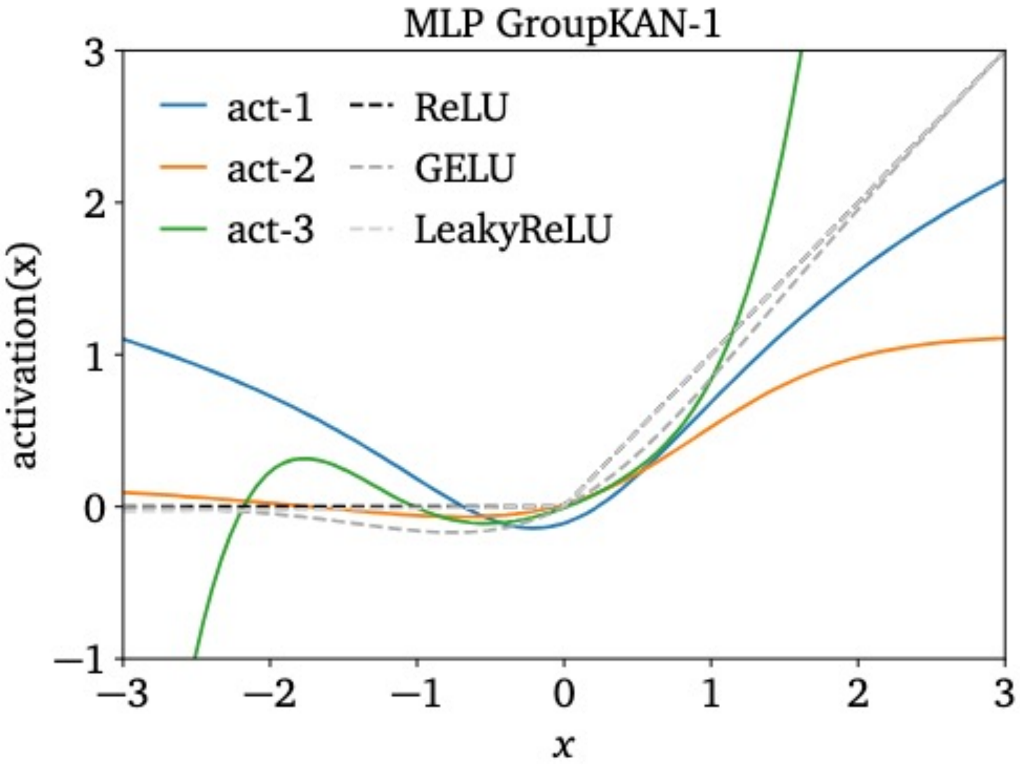
- GroupKAN network: $\mathrm{GroupKAN}(x) = (W_{L-1} \circ \mathrm{GroupKANlayer}_{L-2} \circ W_{L-2} \circ \ldots \circ \mathrm{GroupKANlayer}_0 \circ W_0)x$

- Normal MLP network $\mathrm{MLP}(x) = (W_{L-1} \circ \mathrm{activation} \circ W_{L-2} \circ \ldots \circ \mathrm{activation} \circ W_0)x$

KAN results



MLP + KAN
0.04
0.03
0.02
0.01
0.00
a.u.
10^{-6}
10^{-4}
10^{-2}
10^{0}
10^{2}
$|\Delta|$
ReLU
GELU
LeakyReLU
GroupKAN-1
GroupKAN-2
GroupKAN-4
GroupKAN-8
KAN



MLP GroupKAN-1
3
2
1
0
−1
activation(x)
−3
−2
−1
0
1
2
3
x
act-1
act-2
act-3
ReLU
GELU
LeakyReLU