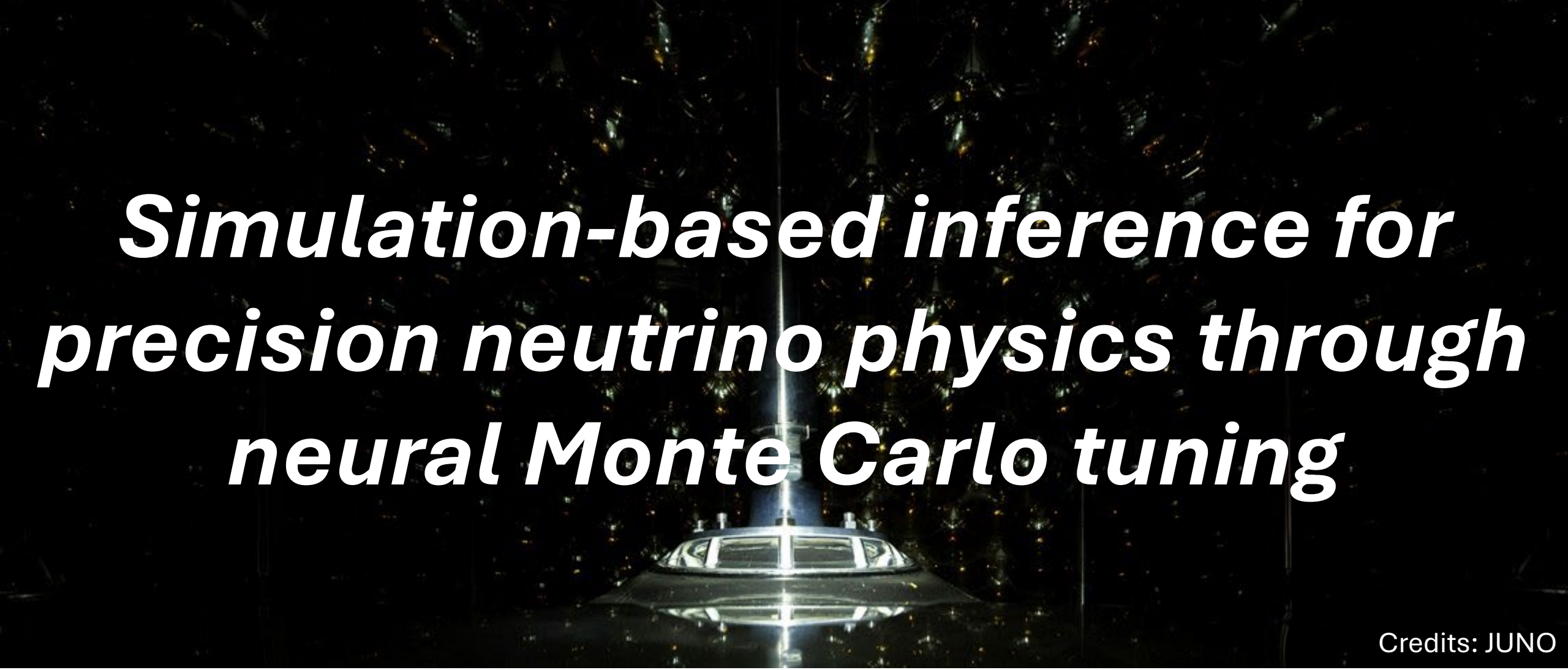
Simulation-based inference for precision neutrino physics through neural Monte Carlo tuning



Credits: JUNO

Arsenii Gavrikov[1,a]

based on Commun Phys **9**, 63 (2026)

[1]Tsung-Dao Lee Institute and School of Physics and Astronomy,
Shanghai Jiao Tong University
[a]*arsenii.gavrikov@sjtu.edu.cn*

QC and ML Workshop 2026. 19 August, 2026, Shenzhen, China

Simulation-based inference: bypassing intractable likelihoods

- In physics (and beyond) we employ mechanistic models (that act as forward models):
  - from fundamental particle interactions to cosmological evolution
  - **known forward process but unknown inverse**

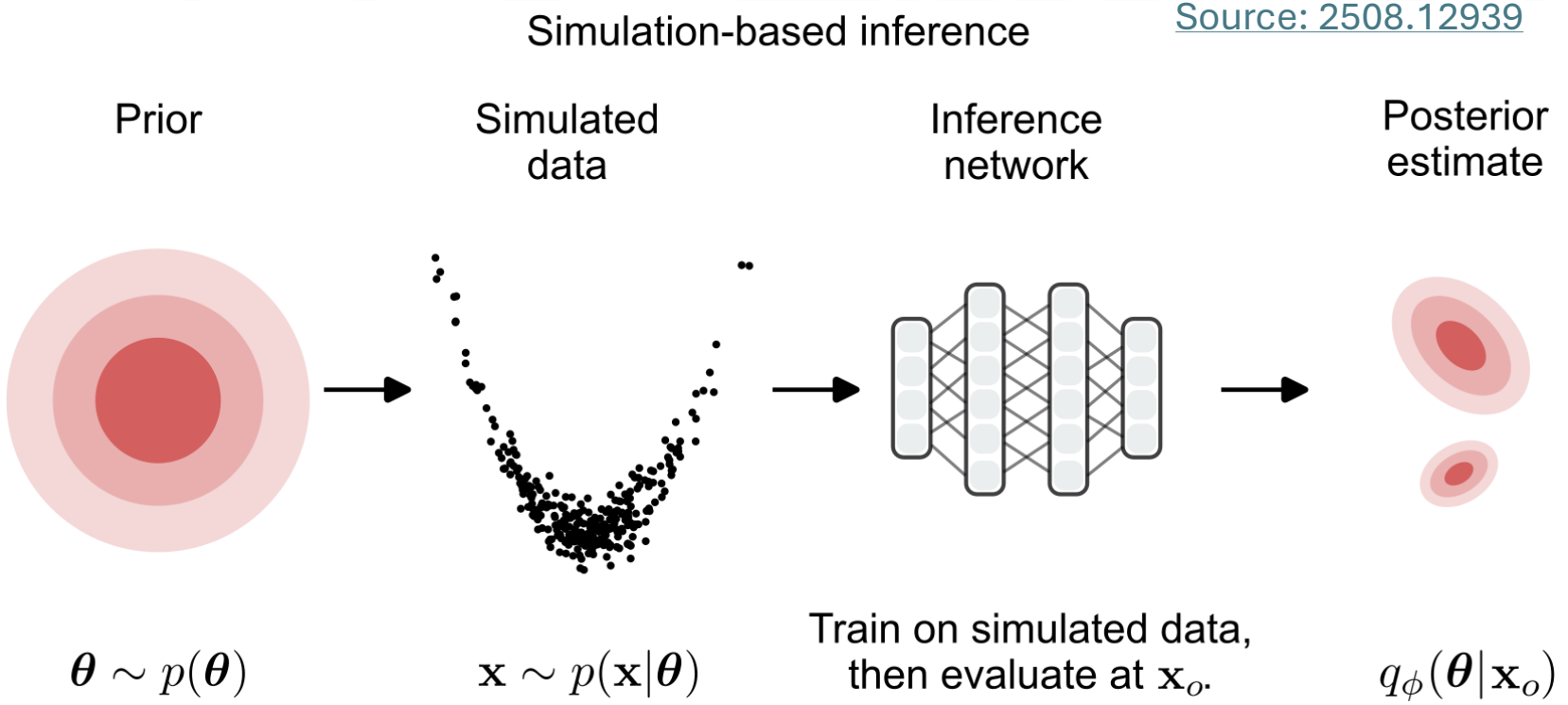
- We **rely on high-fidelity simulators** (like Geant4 or GENIE) that encode fundamental theory to generate synthetic data: ***p(x | θ)***

- ***p(x | θ)* is not explicitly available**: the likelihood intractability problem

- To infer the parameters θ, **we have to estimate *p(θ | x)***

- ML can help here too!

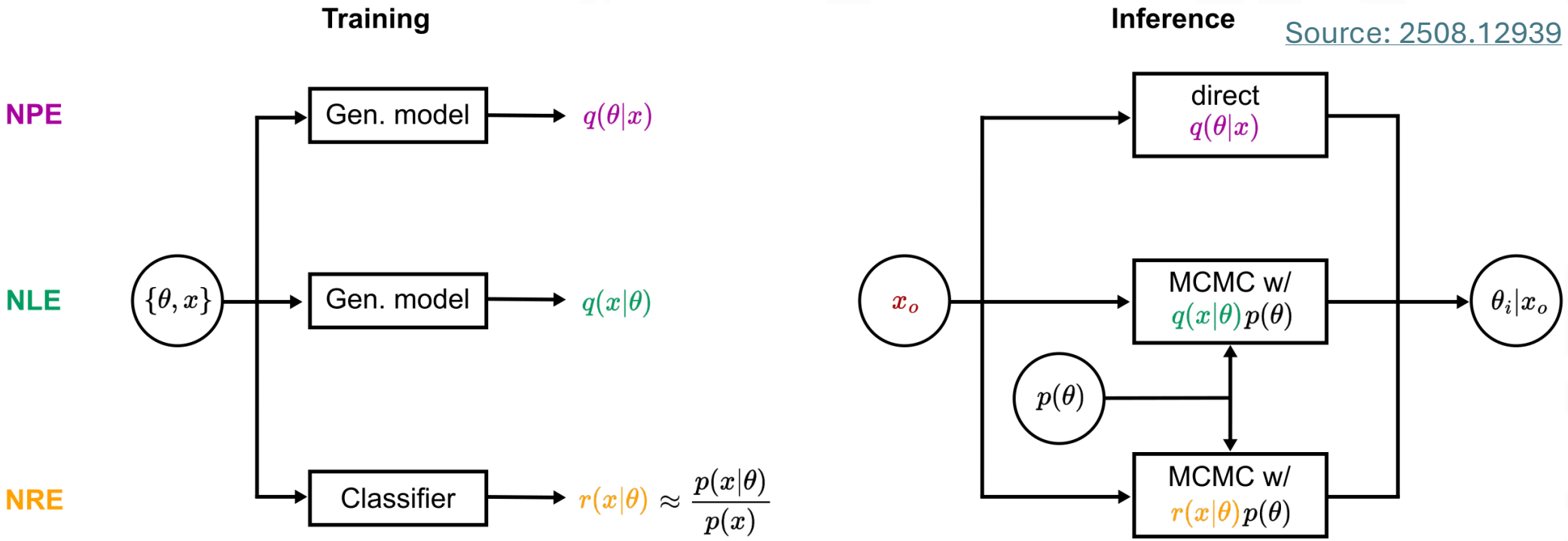
Source: 2508.12939



[1] 2010.06439v2
[2] PNAS 117 (48) 30055-30062
[3] 2508.12939

Simulation-based inference: bypassing intractable likelihoods

- The ML approach: learning the inference quantity from simulator outputs
- Three complementary strategies:
  - Neural Posterior Estimation (**NPE**): directly learns the **posterior** ***p(θ | x)***
  - Neural Likelihood Estimation (**NLE**): learns the **likelihood** ***p(x | θ)***, the posterior is then evaluated with e.g. MCMC
  - Neural Ratio Estimation (**NRE**): trains a binary classifier to approximate the **likelihood ratio** ***r(x | θ) = p(x|θ)/p(x)***

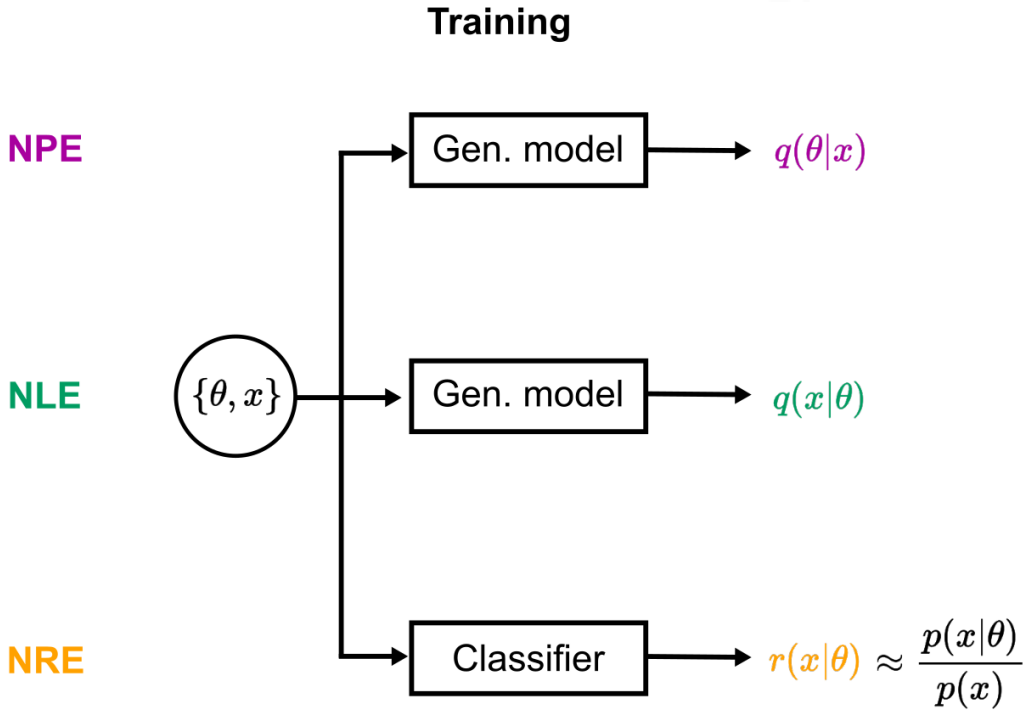


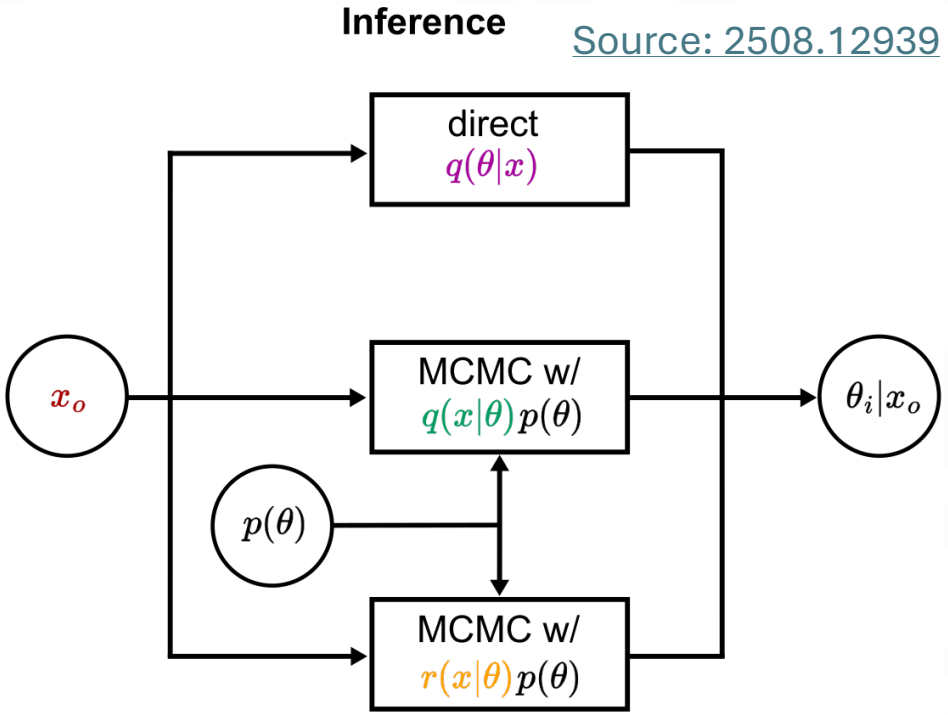
[1] 2010.06439v2
[2] PNAS 117 (48) 30055-30062
[3] 2508.12939

Simulation-based inference: bypassing intractable likelihoods

- The ML approach: learning the inference quantity from simulator outputs
- Three complementary strategies:
  - Neural Posterior Estimation (**NPE**): directly learns the **posterior *p(θ | x)***
  - Neural Likelihood Estimation (**NLE**): learns the **likelihood *p(x | θ)***, the poste
  - Neural Ratio Estimation (**NRE**): trains a binary classifier to approximate the **likelihood ratio *r(x | θ) = p(x|θ)/p(x)***

I) Fast inference
II) Probabilistic reconstruction





Source: 2508.12939

[1] 2010.06439v2
[2] PNAS 117 (48) 30055-30062
[3] 2508.12939

Simulation-based inference: bypassing intractable likelihoods

- The ML approach: learning the inference quantity from simulator outputs
- Three complementary strategies:
  - Neural Posterior Estimation (**NPE**): directly learns the **posterior** $p(\theta \mid x)$
  - Neural Likelihood Estimation (**NLE**): learns the **likelihood** $p(x \mid \theta)$, the posterior is then evaluated with e.g. MCMC
  - Neural Ratio Estimation (**NRE**): trains a binary classifier to approximate the **likelihood ratio** $r(x \mid \theta) = p(x|\theta)/p(x)$

Source: 2508.12939

Criterion	NPE	NLE	NRE
Inference speed	++ (ms)	– (sec-min)	– (sec-min)
Simulation efficiency for i.i.d. data	––	++	++
Embedding network compatibility	++	––	++
Training computational cost	Medium	Medium	Low
Invalid simulation handling	++	–	–
Hyperparameter tuning complexity	Low	High	High

[1] 2010.06439v2
[2] PNAS 117 (48) 30055-30062
[3] 2508.12939

Simulation-based inference: bypassing intractable likelihoods

- The ML approach: learning the inference quantity from simulator outputs
- Three complementary strategies:
  - Neural Posterior Estimation (**NPE**): directly learns the **posterior** $p(\theta \mid x)$
  - Neural Likelihood Estimation (**NLE**): learns the **likelihood** $p(x \mid \theta)$, the posterior is then evaluated with e.g. MCMC
  - Neural Ratio Estimation (**NRE**): trains a binary classifier to approximate the **likelihood ratio** $r(x \mid \theta) = p(x|\theta)/p(x)$

Source: 2508.12939

Criterion	NPE	NLE	NRE
Inference speed	++ (ms)	– (sec-min)	– (sec-min)
Simulation efficiency for i.i.d. data	––	++	++
Embedding network compatibility	++	––	++
Training computational cost	Medium	Medium	Low
Invalid simulation handling	++	–	–
Hyperparameter tuning complexity	Low	High	High

[1] 2010.06439v2
[2] PNAS 117 (48) 30055-30062
[3] 2508.12939

The JUNO experiment

Credits: JUNO

The JUNO experiment

Jiangmen Underground Neutrino Observatory (**JUNO**):

- **multi-purpose** neutrino experiment
- located in **China**
- largest **liquid scintillator (LS) detector** ever built

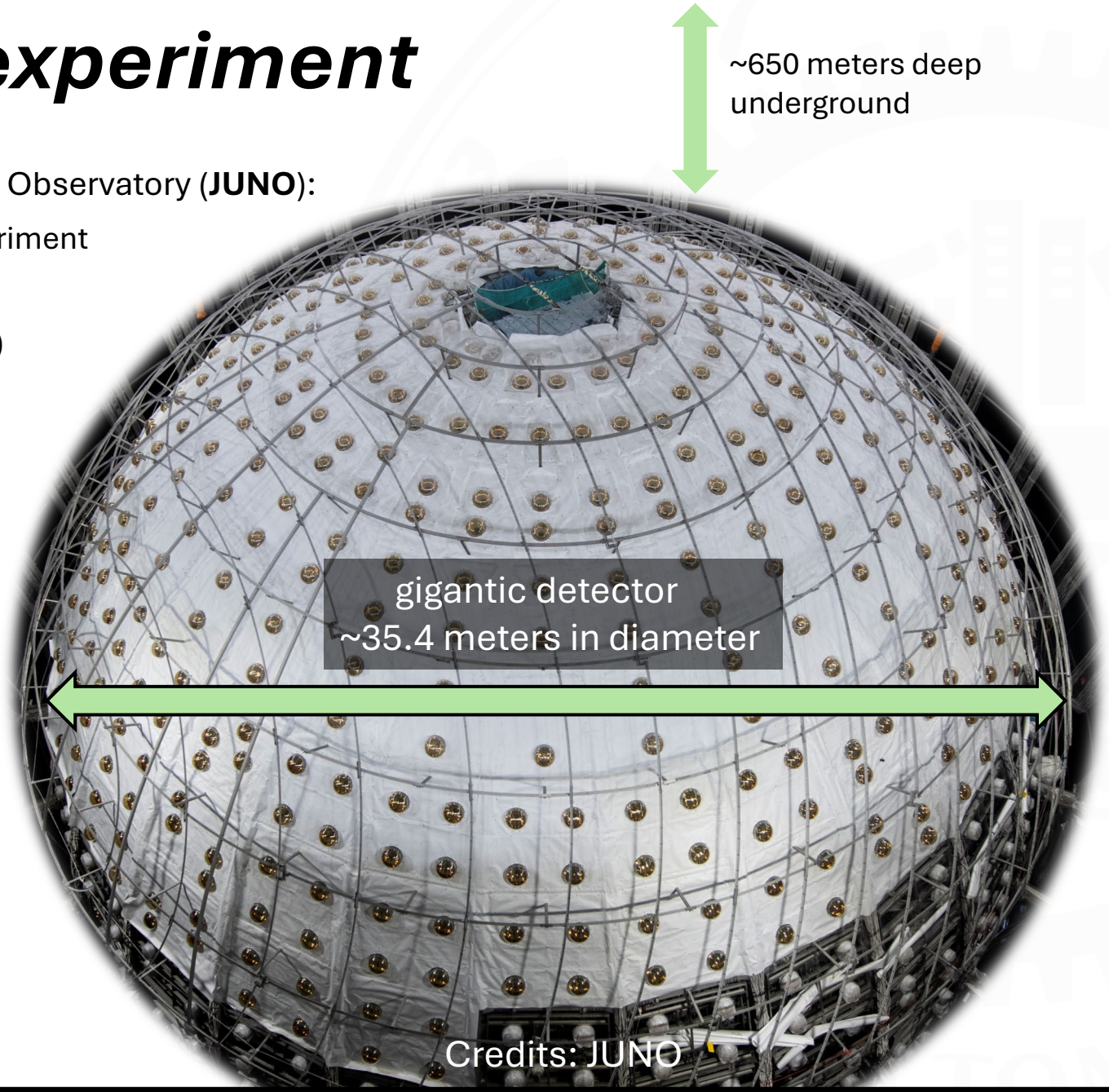
Credits: JUNO

Data-taking is ongoing since August 2025 and first physics results are already out!

The JUNO experiment

Jiangmen Underground Neutrino Observatory (**JUNO**):

- **multi-purpose** neutrino experiment
- located in **China**
- largest **liquid scintillator (LS) detector** ever built

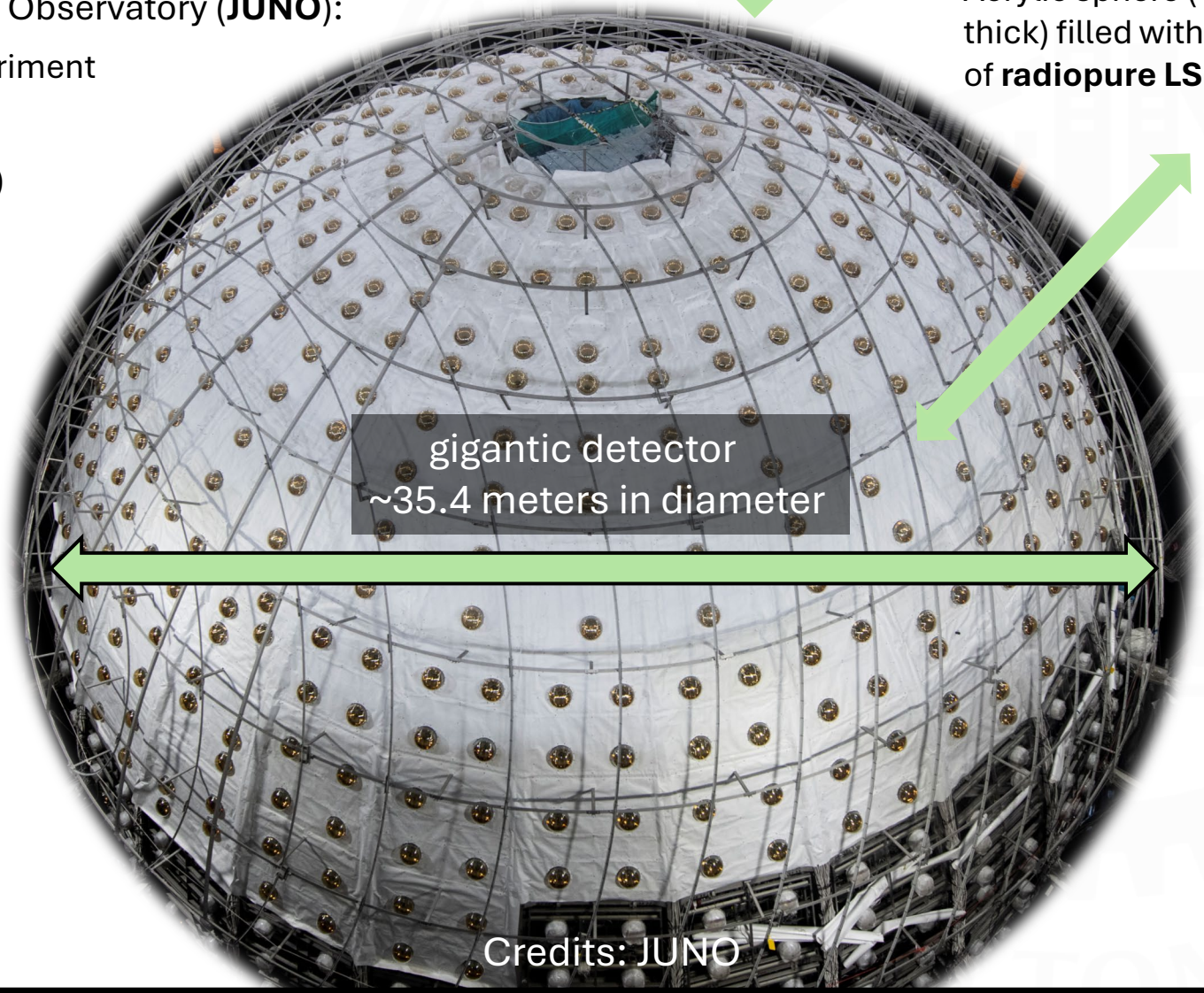


Data-taking is ongoing since August 2025 and first physics results are already out!

The JUNO experiment

Jiangmen Underground Neutrino Observatory (**JUNO**):

- **multi-purpose** neutrino experiment
- located in **China**
- largest **liquid scintillator (LS) detector** ever built

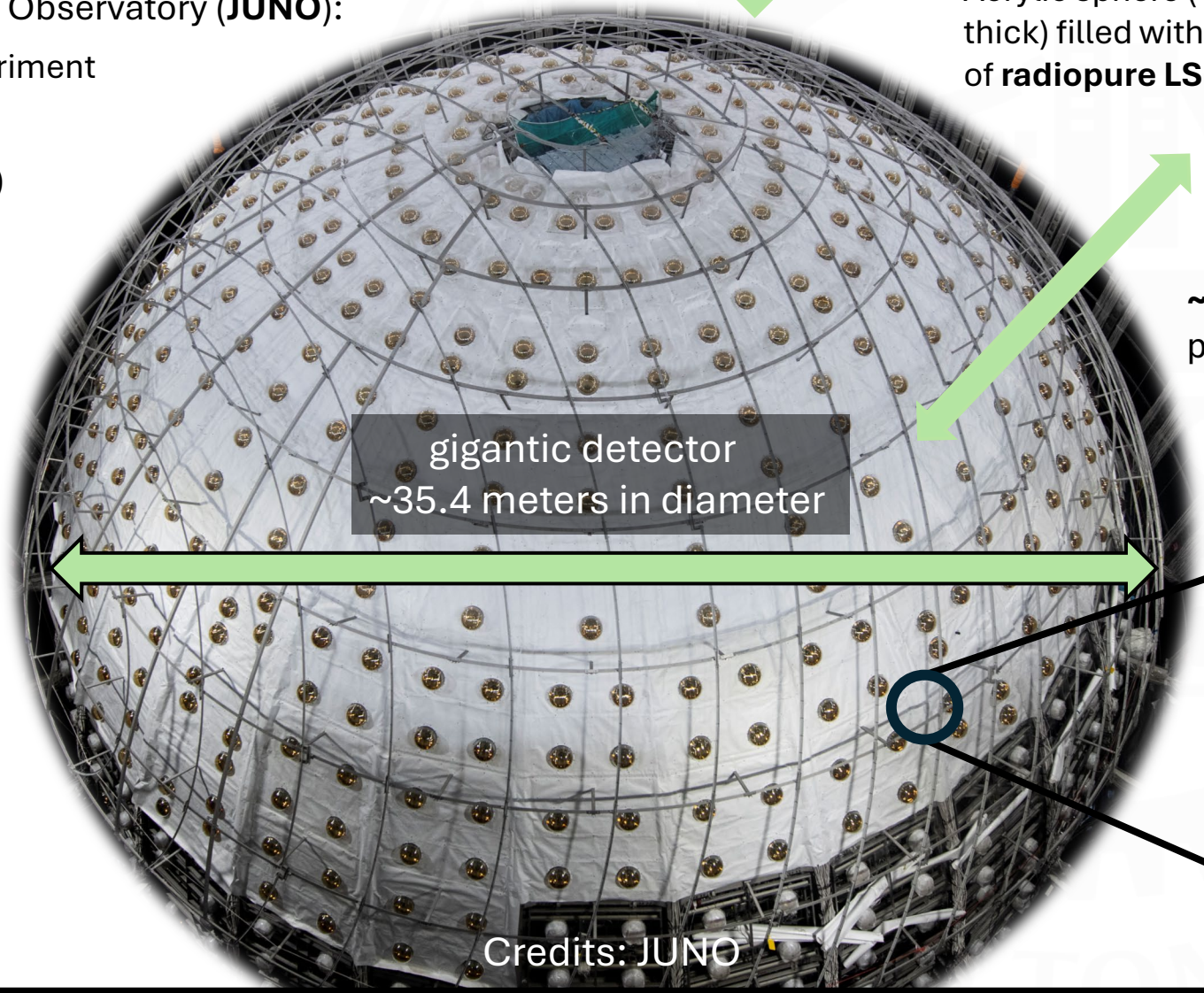


Data-taking is ongoing since August 2025 and first physics results are already out!

The JUNO experiment

Jiangmen Underground Neutrino Observatory (**JUNO**):

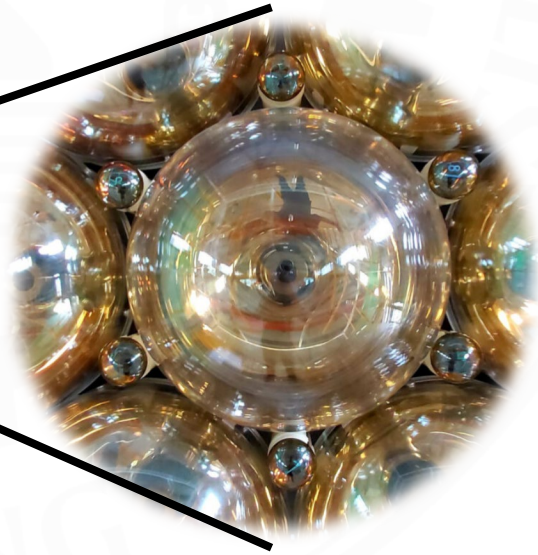
- **multi-purpose** neutrino experiment
- located in **China**
- largest **liquid scintillator (LS) detector** ever built



LS of JUNO is organic: **LAB +** 2.5 g/L **PPO + 3** mg/L **bis-MSB**

~78% photo-coverage by photo-multiplier tubes (PMTs):

- 17596 Large 20'' PMTs
- 25587 small 3'' PMTs



Data-taking is ongoing since August 2025 and first physics results are already out!

The JUNO experiment

Jiangmen Underground Neutrino Observatory (**JUNO**):

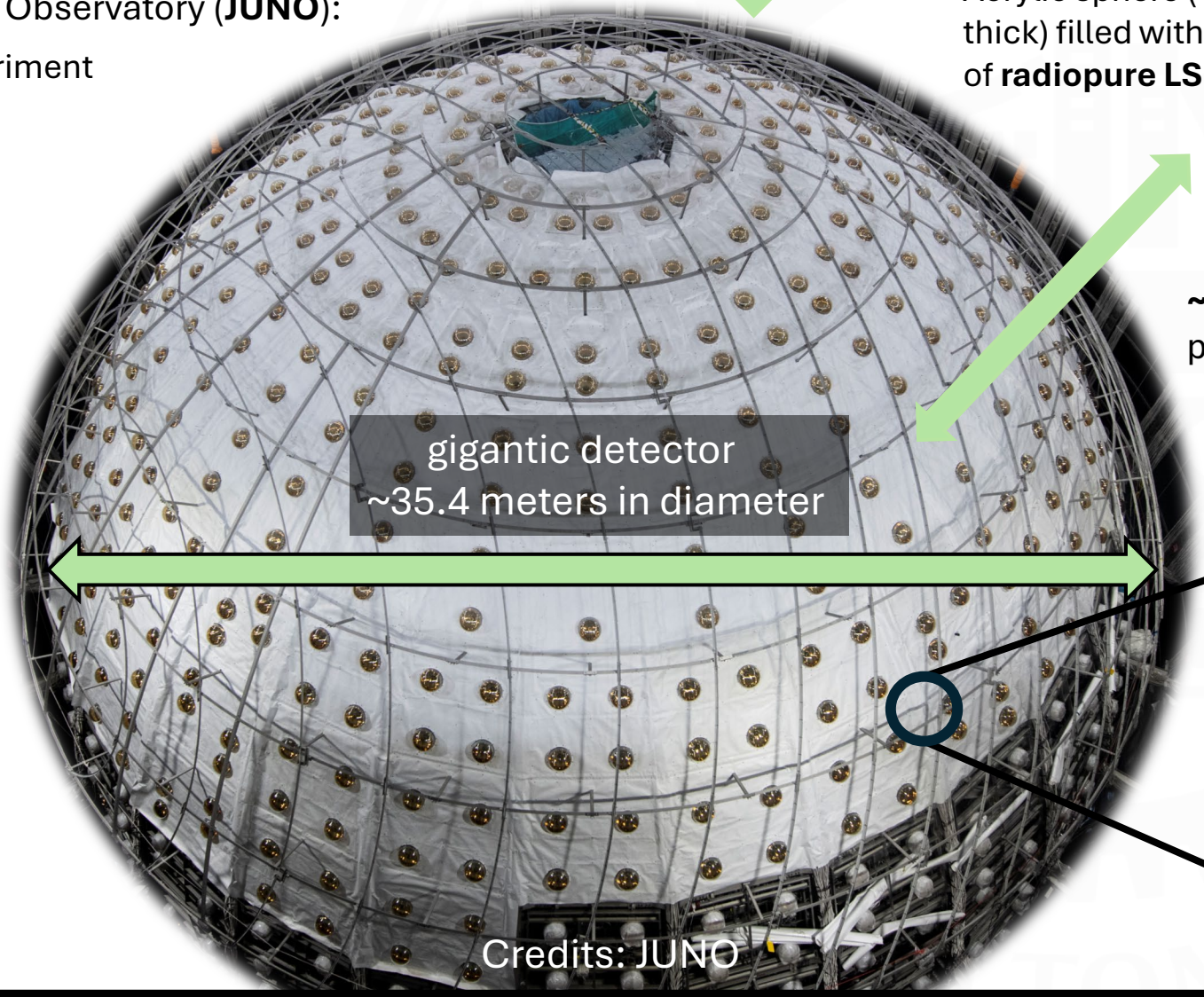
- **multi-purpose** neutrino experiment
- located in **China**
- largest **liquid scintillator (LS) detector** ever built

The main goals of JUNO:

- **neutrino mass ordering** at 3σ after ~6-7 years
- sub-percent measurements **of the oscillation parameters**: $\Delta m^2_{21}, \Delta m^2_{31}, \sin^2\theta_{12}$
- using **reactor** $\overline{\nu}_e$

Overall physics potential and program is broader

Data-taking is ongoing since August 2025 and first physics results are already out!



The JUNO experiment

Jiangmen Underground Neutrino Observatory (**JUNO**):

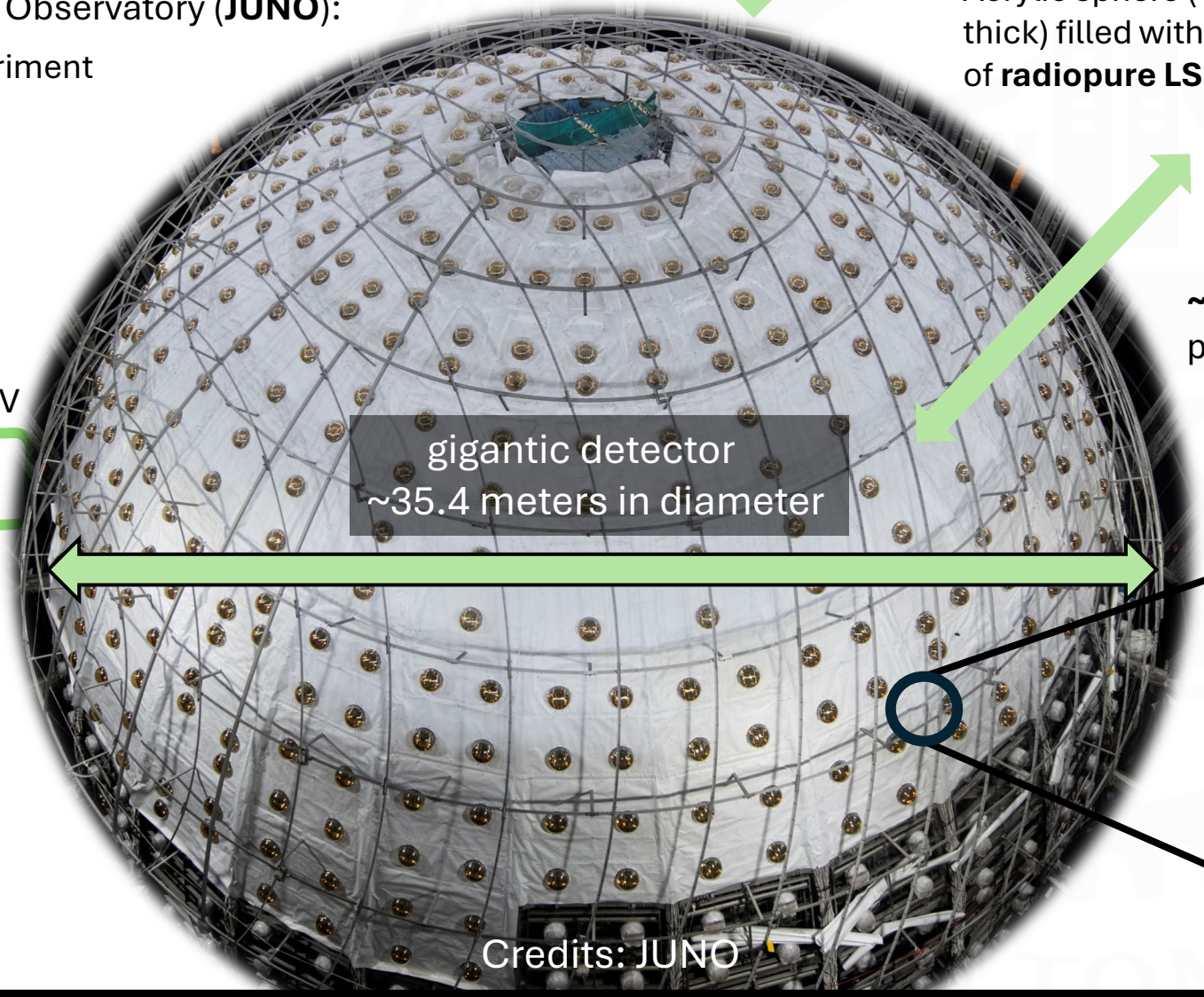
- **multi-purpose** neutrino experiment
- located in **China**
- largest **liquid scintillator (LS) detector** ever built

The core challenge:

- **Energy resolution**: 3% at 1 MeV
- <1% **control over energy scale** systematic uncertainty

need for **highly accurate and well-tuned Monte Carlo** (MC) simulations

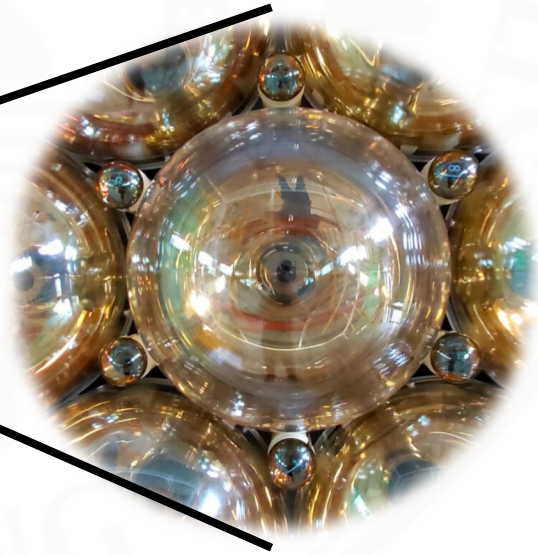
Data-taking is ongoing since August 2025 and first physics results are already out!



LS of JUNO is organic: **LAB +** 2.5 g/L **PPO + 3** mg/L **bis-MSB**

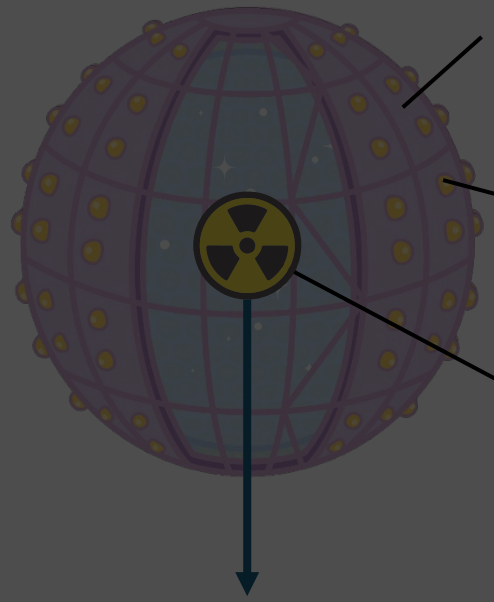
~78% photo-coverage by photo-multiplier tubes (PMTs):

- 17596 Large 20'' PMTs
- 25587 small 3'' PMTs

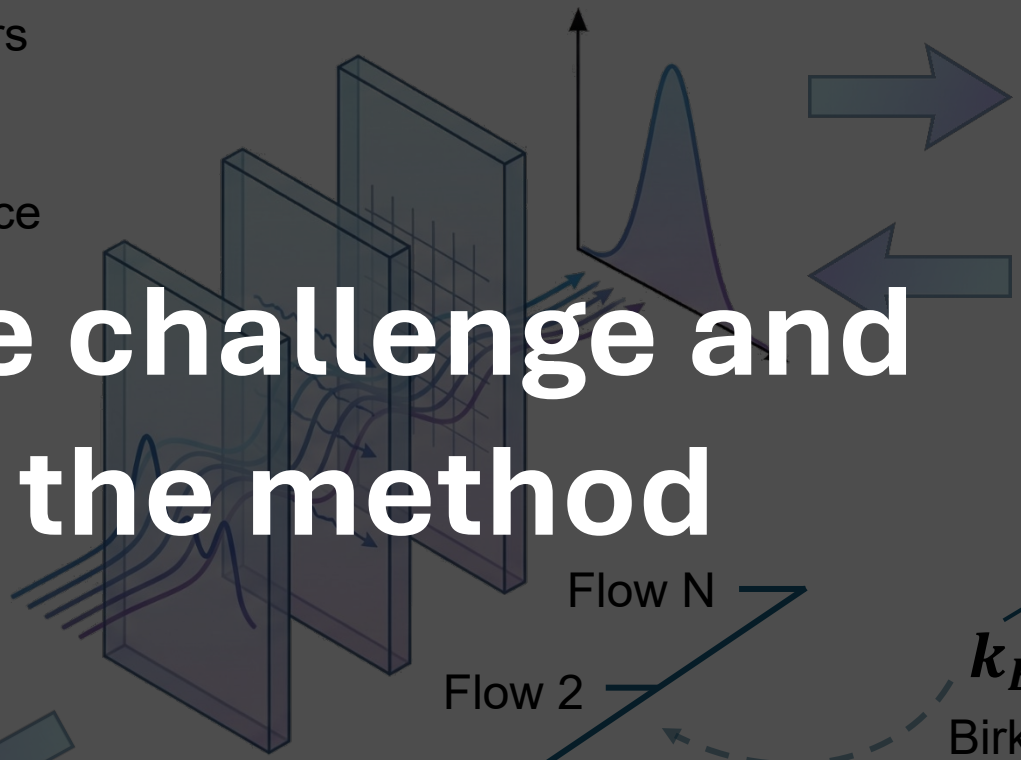


The challenge and the method

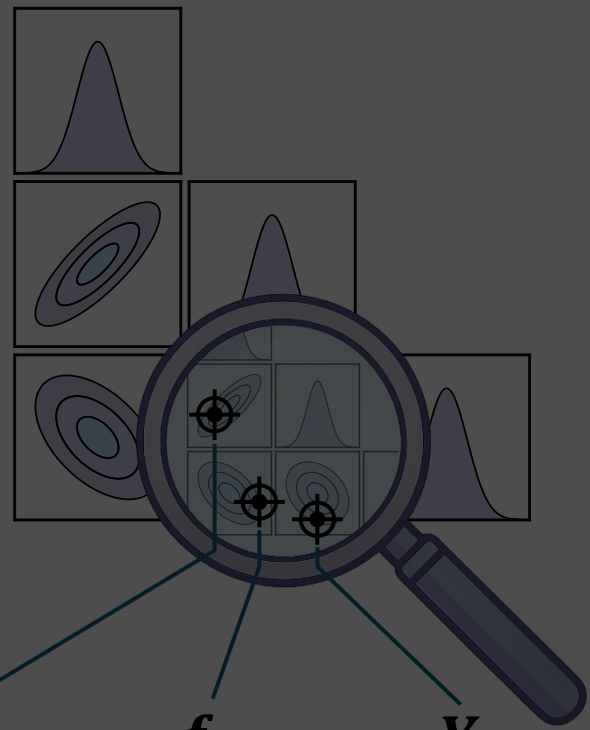
JUNO calibration data



Normalizing flows
Turning the spectrum to a normal distribution



Parameter estimation
Bayesian inference with Nested Sampling



k_B Birks' constant

f_C Cherenkov factor

Y Light yield

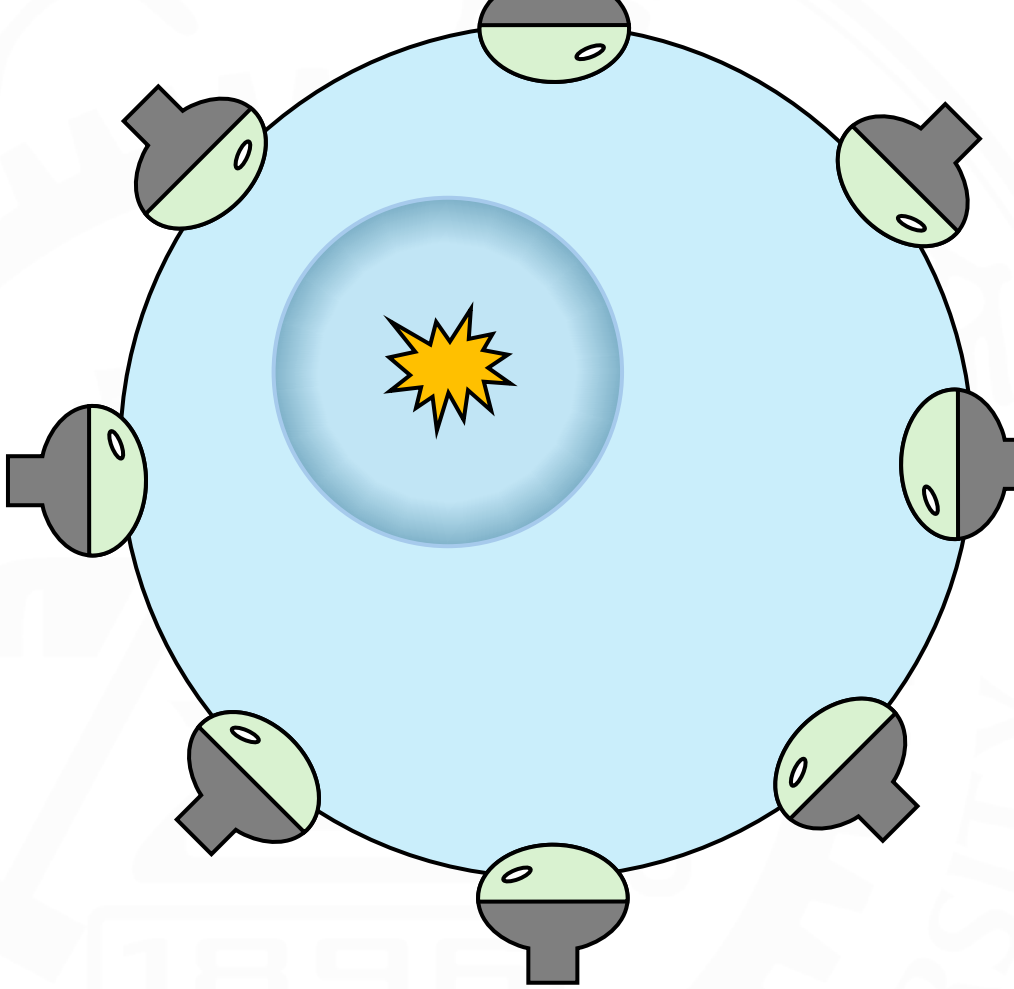
Best fit parameters
Tuned configuration better matching calibration data

Complex energy spectrum

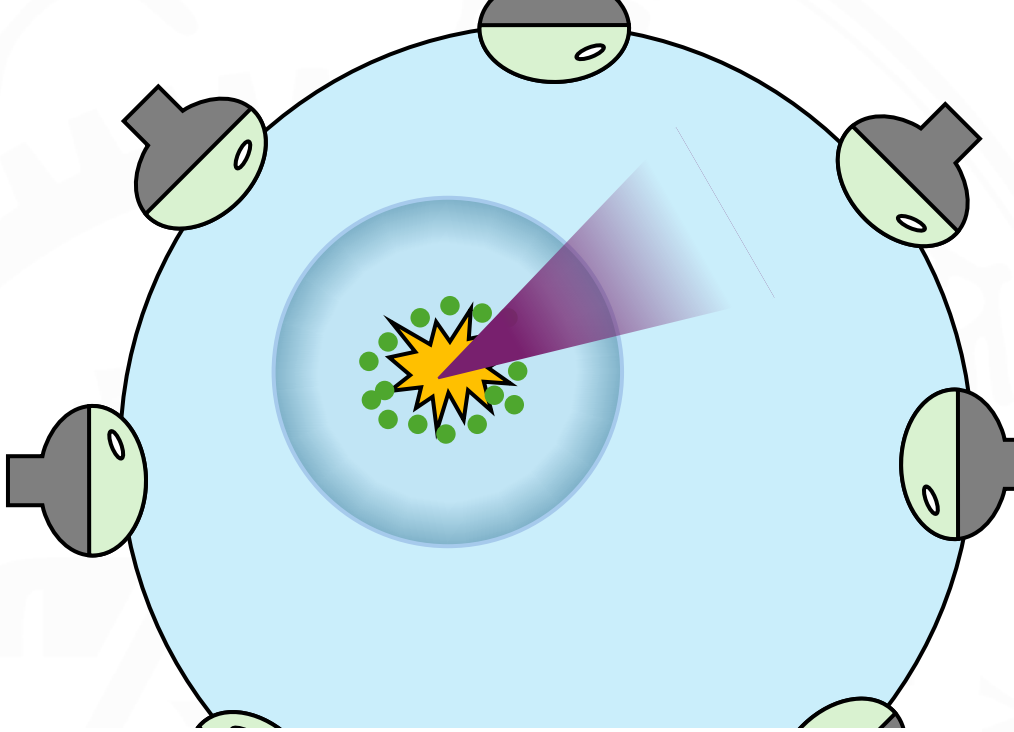
Data simulated by tuned Monte Carlo describing calibration data

Physics challenge

- **Detection principle of JUNO:**
  - JUNO is a **calorimeter**: a particle deposits energy in the scintillator.
  - The scintillator emits **light collected by PMTs** and measured as a total **number of photo-electrons (NPE)** → our proxy for the visible energy

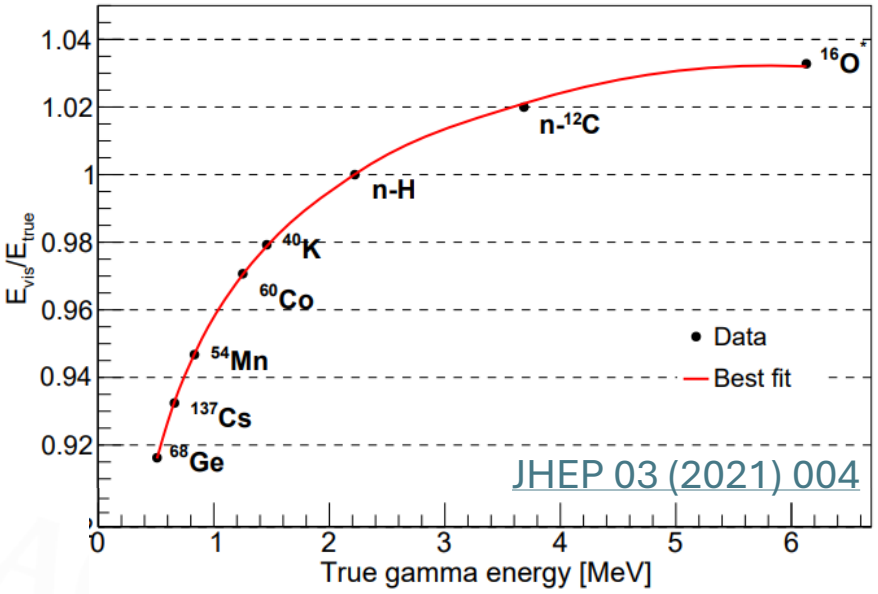


Physics challenge



- **Detection principle of JUNO:**
  - JUNO is a **calorimeter**: a particle deposits energy in the scintillator.
  - The scintillator emits **light collected by PMTs** and measured as a total **number of photo-electrons (NPE)** → our proxy for the visible energy

- **Challenge:**
  - The relationship between true deposited energy and NPE **is not linear**
  - In simulation, we model this non-linear relationship with **3 parameters:**
    - **Birks' constant** (k_B) ↓: models quenching, which reduces visible NPE
    - **Cherenkov factor** (f_C) ↑: model the contribution of Cherenkov light
    - **Light yield** (Y) ↑: models the overall scaling factor (after quenching)



JHEP 03 (2021) 004

calibration sources

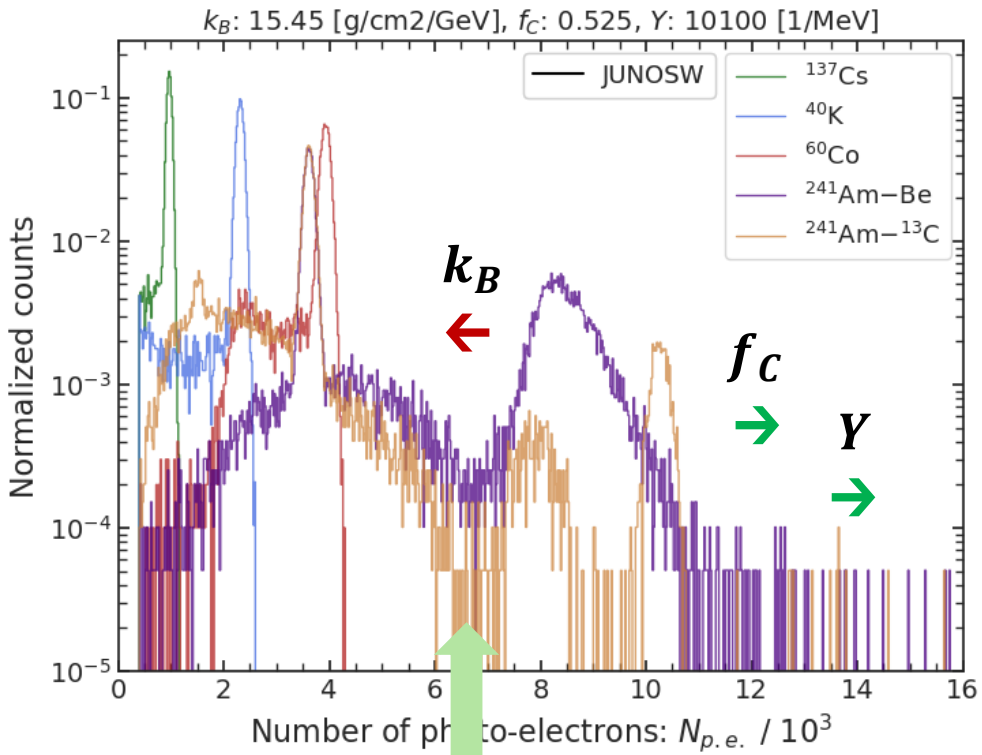
Physics challenge

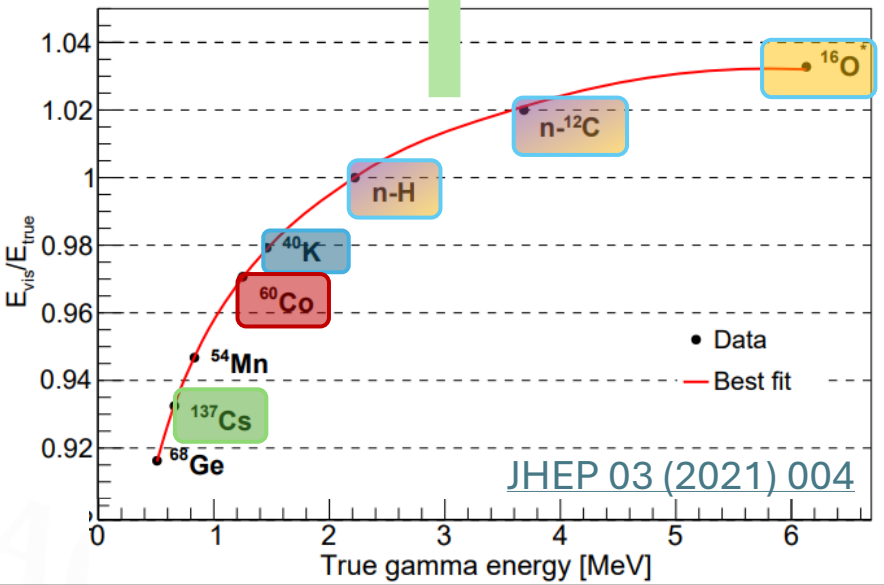
- **Detection principle of JUNO:**
  - JUNO is a **calorimeter**: a particle deposits energy in the scintillator.
  - The scintillator emits **light collected by PMTs** and measured as a total **number of photo-electrons (NPE)** → our proxy for the visible energy

- **Challenge:**
  - The relationship between true deposited energy and NPE **is not linear**
  - In simulation, we model this non-linear relationship with **3 parameters:**
    - **Birks' constant** (k_B) ↓: models quenching, which reduces visible NPE
    - **Cherenkov factor** (f_C) ↑: model the contribution of Cherenkov light
    - **Light yield** (Y) ↑: models the overall scaling factor (after quenching)

Task: tune (**k_B, f_C, Y**) so that our MC matches real calibration data

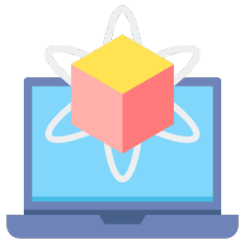
pivotal to control systematic uncertainties





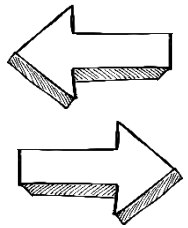
Monte Carlo tuning strategy

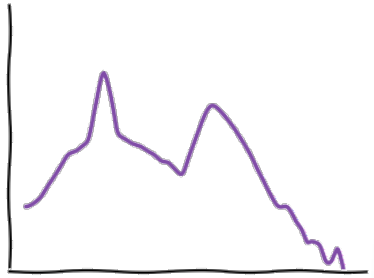
The **most straightforward approach** would be…



MC

$\phi = \{k_B, f_C, Y\}_{\mathrm{g}}$

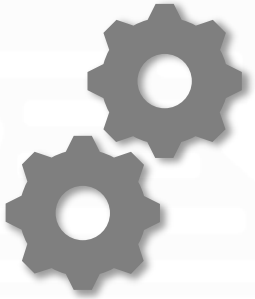




Simulated data M

$\mathcal{L}(M(\phi)|D)$

Likelihood



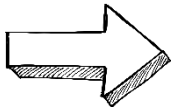
Fitter

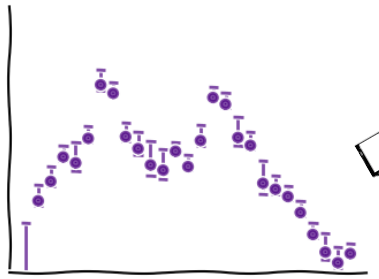
- Minuit
- MCMC
- Nested sampling

Results

best fit $\{k_B, f_C, Y\}$

JUNO

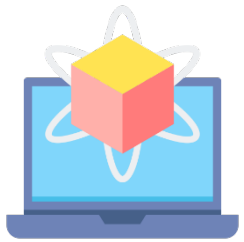




Calibration data D

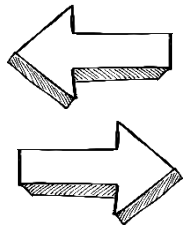
Monte Carlo tuning strategy

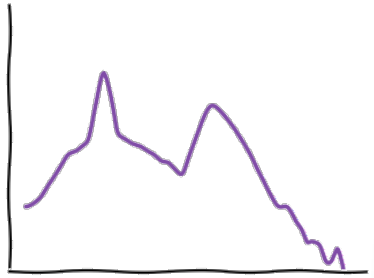
The **most straightforward approach** would be…



MC

$\phi = \{k_B, f_C, Y\}_g$



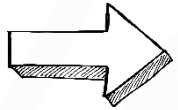


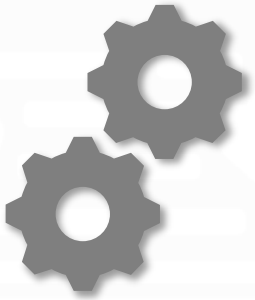
Simulated data M

At **each step** of the fit, **we generate a lot of events with full MC** to build spectra of the sources with the corresponding kB_g fC_g LY_g

$\mathcal{L}(M(\phi)|D)$

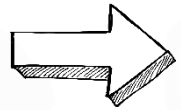
Likelihood





Fitter

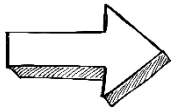
- Minuit
- MCMC
- Nested sampling

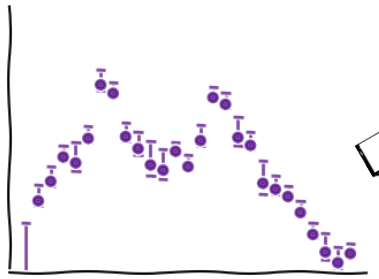


Results

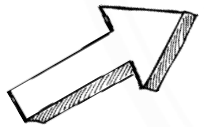
best fit $\{k_B, f_C, Y\}$

JUNO





Calibration data D

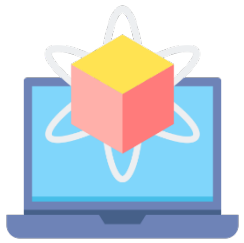


Monte Carlo tuning strategy

The **most straightforward approach** would be…

too impractical and slow! Can we replace it with a surrogate model?

Neural Likelihood Estimation (NLE)



MC

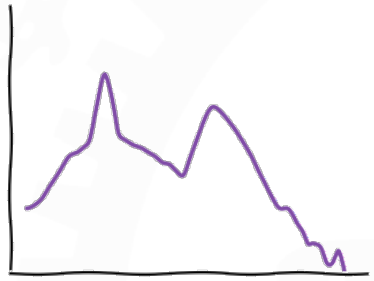
$\phi = \{k_B, f_C, Y\}_{\mathrm{g}}$

Training

Fast ML model to get spectra for ϕ

$p(x|\phi)$

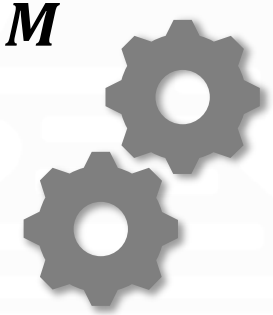
(able to interpolate in param space)



ML generated data M

$\mathcal{L}(M(\phi)|D)$

Likelihood



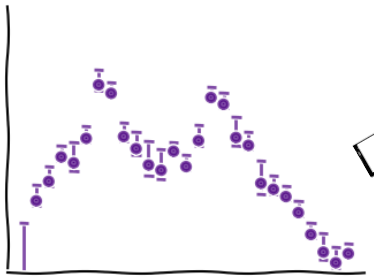
Fitter

- Minuit
- MCMC
- Nested sampling

Results

best fit $\{k_B, f_C, Y\}$

JUNO



Calibration data D

A. Gavrikov, A. Serafini, D. Dolzhikov et al., *Commun Phys* **9**, 63 (2026)

Two Neural Likelihood Estimators

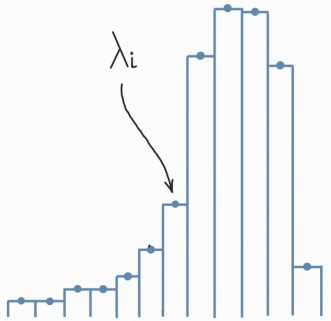
+ straight and reliable model
- requires pre-defined binning

Multi-output regressor

Aims to directly learn **a mapping** from the parameters and a source type to the histogram representing the **spectra PDF**

*Provides **PDF estimation** by outputting a histogram*

TEDE



We use a **Transformer encoder-based** model as the density estimator (**TEDE**)

Conditions:
kB, fC, Y + source type **S**

Normalizing Flow-based model

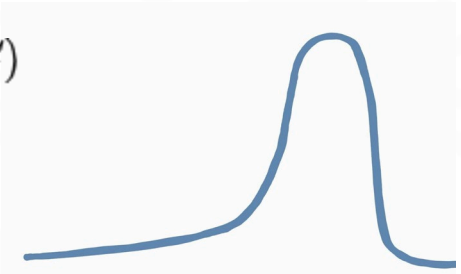
Aims to learn the exact **conditional probability** of the energies* for a given set of parameters and a source type:

$P(E|k_B, f_C, L_Y, S)$

*Provides the **exact conditional PDF** for any energy value:*

$P(E \mid k_B, f_C, L_y, S)$

NFDE



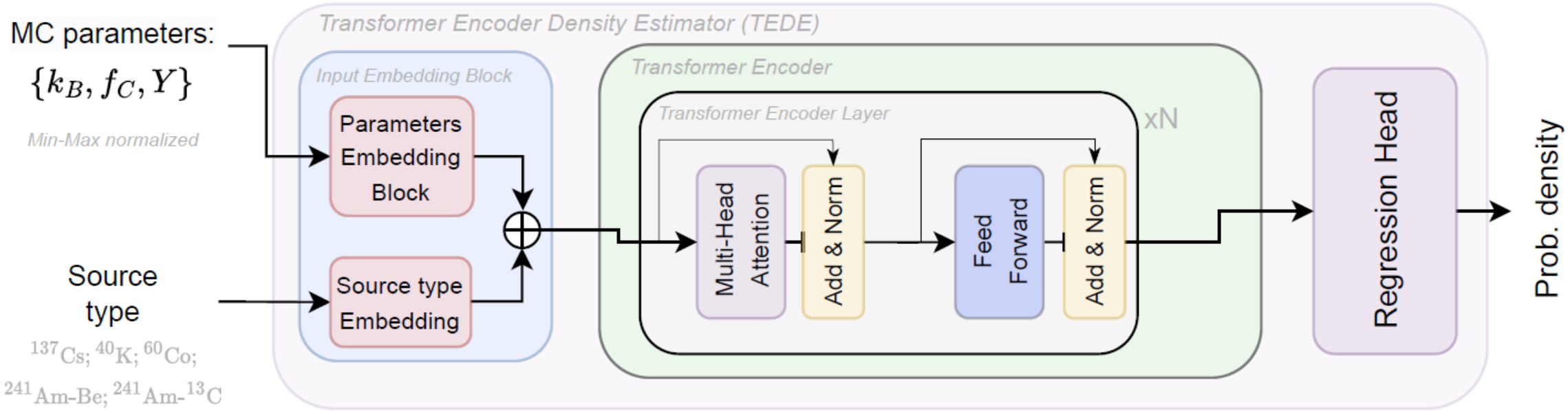
+ exact probability density function
+ no pre-defined binning
+ opens a possibility of an unbinned fit

We use **Normalizing Flows**-based model (NFDE) to evaluate the exact conditional probability for the events and build the PDF

**represented by amount of light collected*

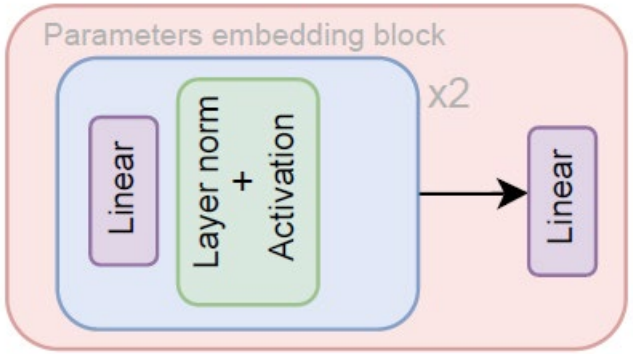
A. Gavrikov, A. Serafini, D. Dolzhikov et al., *Commun Phys* **9**, 63 (2026)

Transformer Encoder Density Estimator (TEDE)



Multi-output regressor is based on the **Transformer's encoder** :

- A powerful architecture, well known for NLP and CV applications
- Each condition (MC parameter or source type) is treated as a token
- Regressor Head block to convert outputs to the spectra PDF with pre-defined binning (**bin size of 20 PEs**)

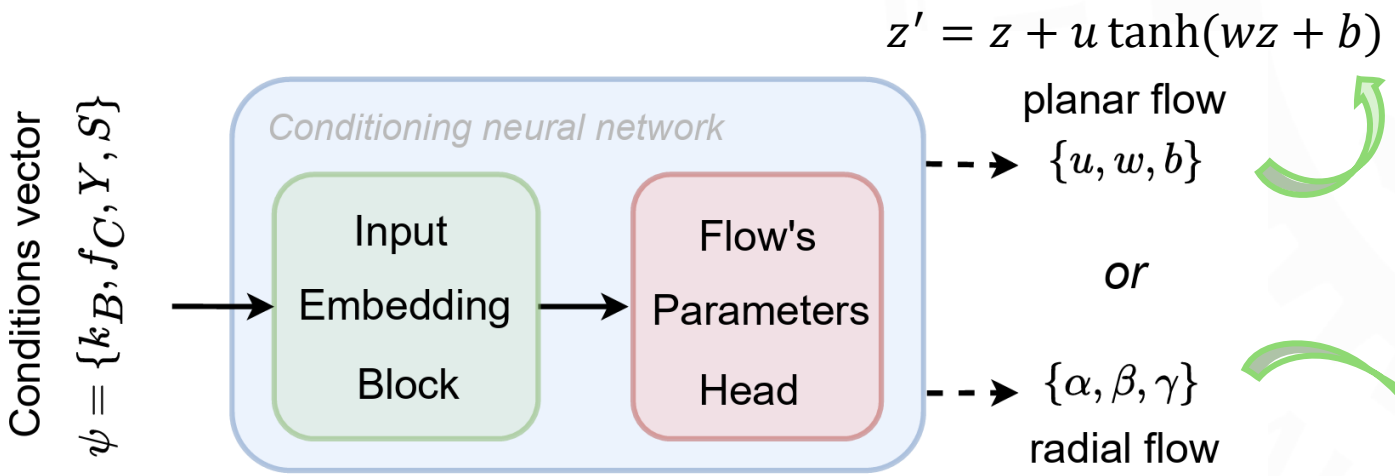


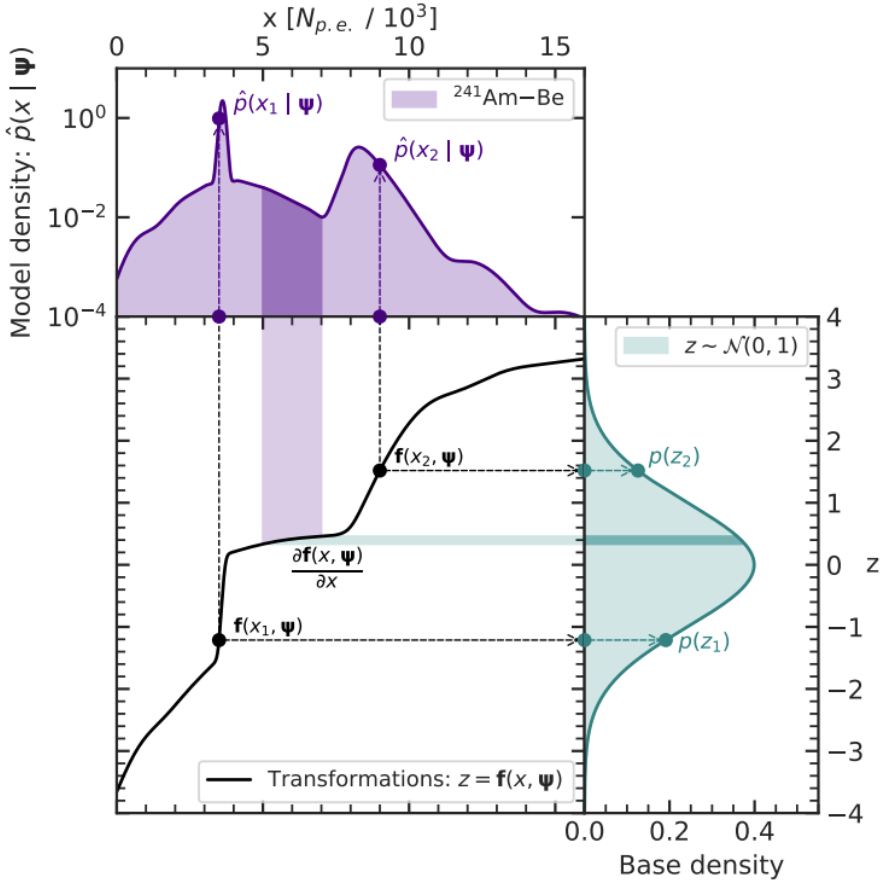
A. Gavrikov, A. Serafini, D. Dolzhikov et al., *Commun Phys* **9**, 63 (2026)

Normalizing Flows Density Estimator (NFDE)

Normalizing flows:

- **generative** ML model aimed to **explicitly model a density estimation**
- generalizable for modeling **a conditional density estimation** as well (via a dedicated **conditioning neural network**)
- based on an **idea of transforming a simple known distribution** (*e.g. standard Gaussian*) **to the complex, desired distribution** by applying multiple learnable (*using data*) and invertible transformations called **flows.**





A. Gavrikov, A. Serafini, D. Dolzhikov et al., *Commun Phys* **9**, 63 (2026)

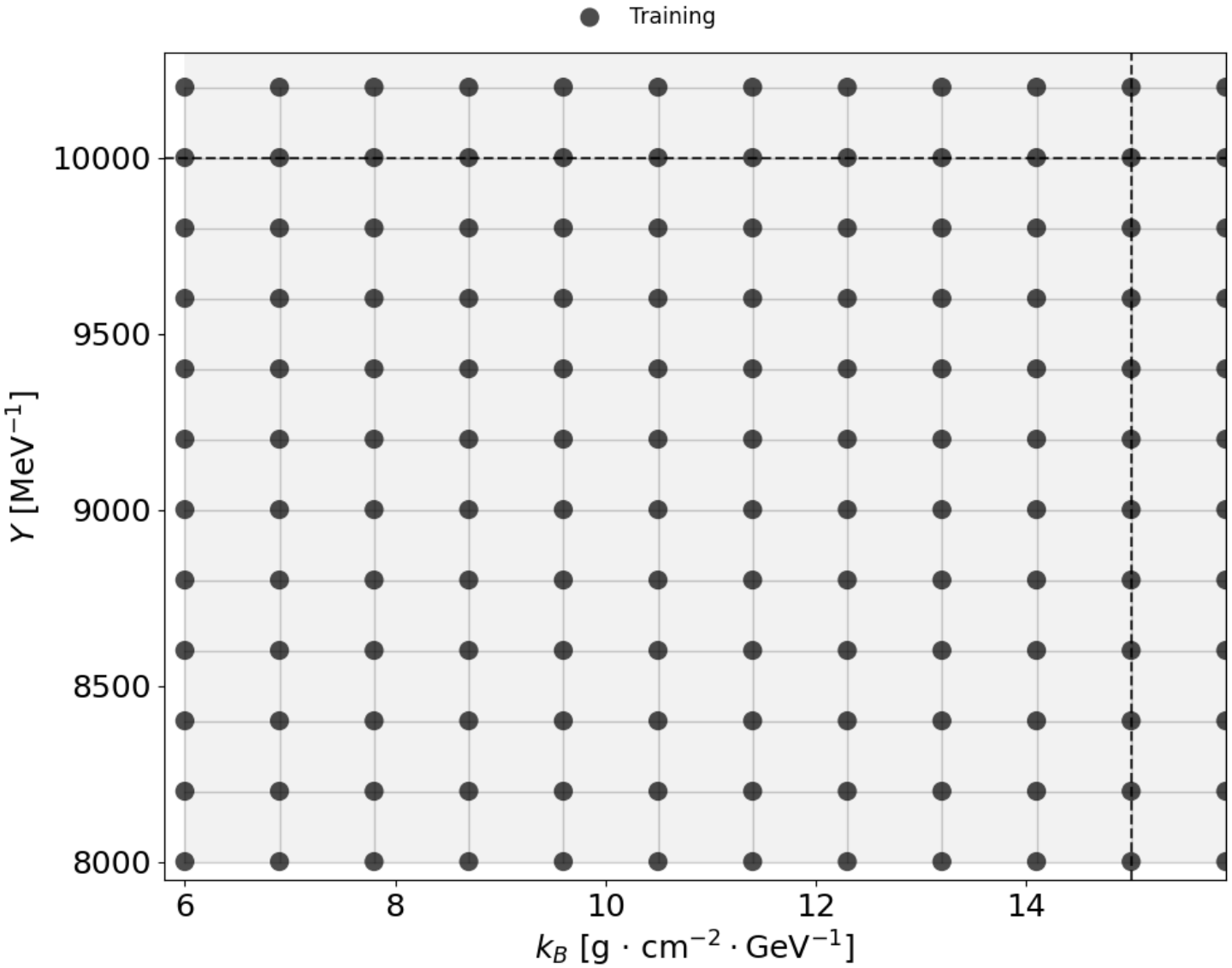
Datasets: training/validation strategy

Always in the center of the detector

Training data

- 21 points per param (21^3 combs)
- 10k events x 5 sources per point (~500M events)

Y and k_B example



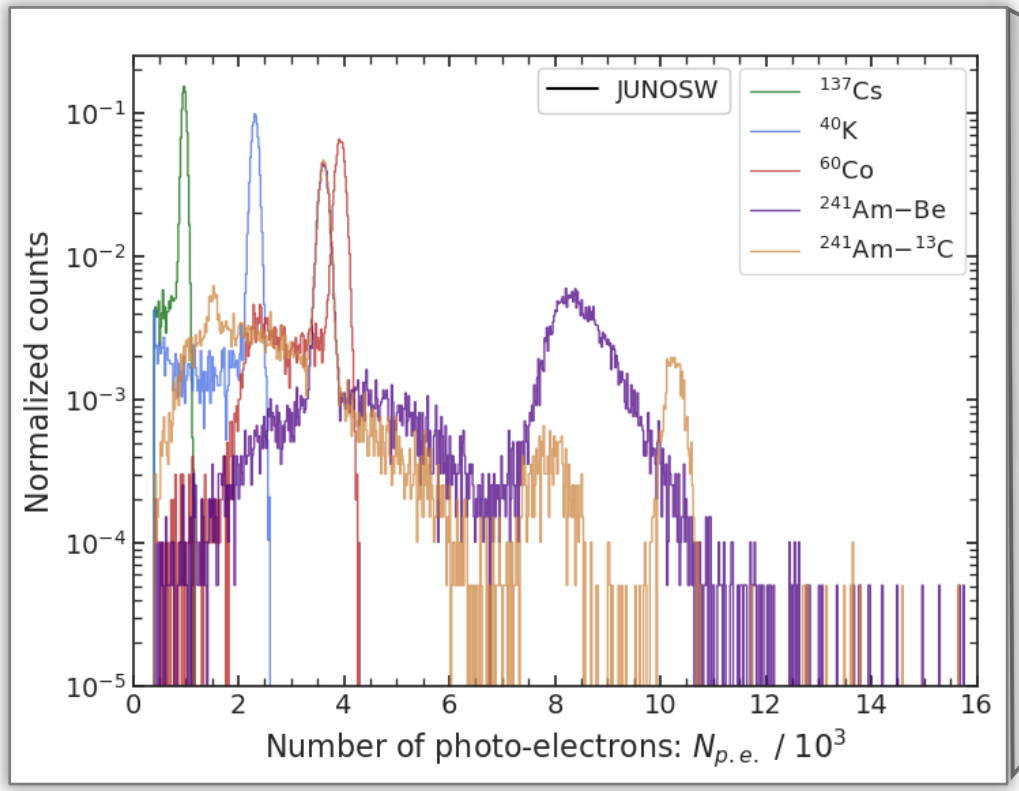
A. Gavrikov, A. Serafini, D. Dolzhikov et al., *Commun Phys* **9**, 63 (2026)

Datasets: training/validation strategy

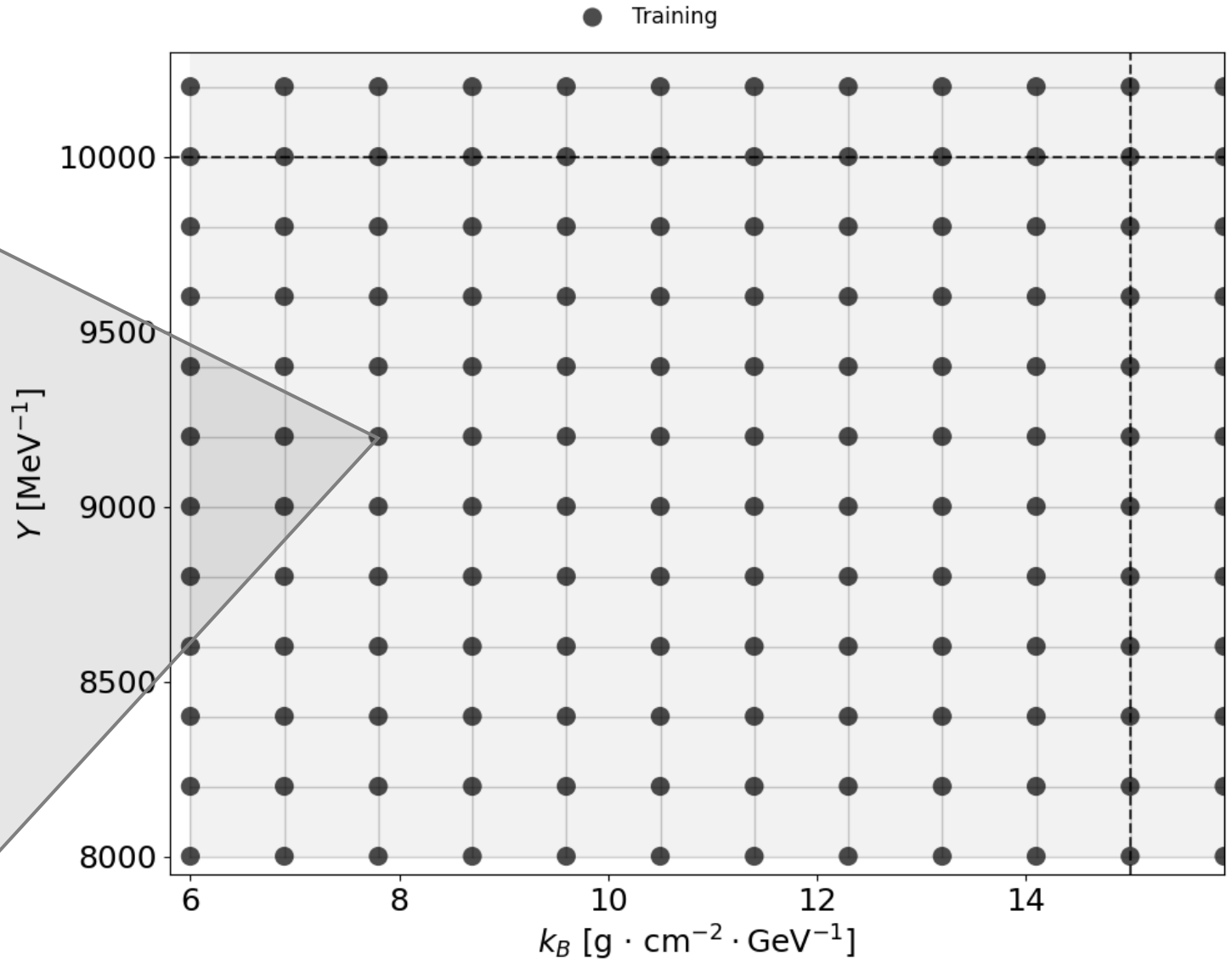
Always in the center of the detector

Training data

- 21 points per param (21^3 combs)
- 10k events x 5 sources per point (~500M events)



Y and k_B example



A. Gavrikov, A. Serafini, D. Dolzhikov et al., *Commun Phys* **9**, 63 (2026)

Datasets: training/validation strategy

Always in the center of the detector

Training data

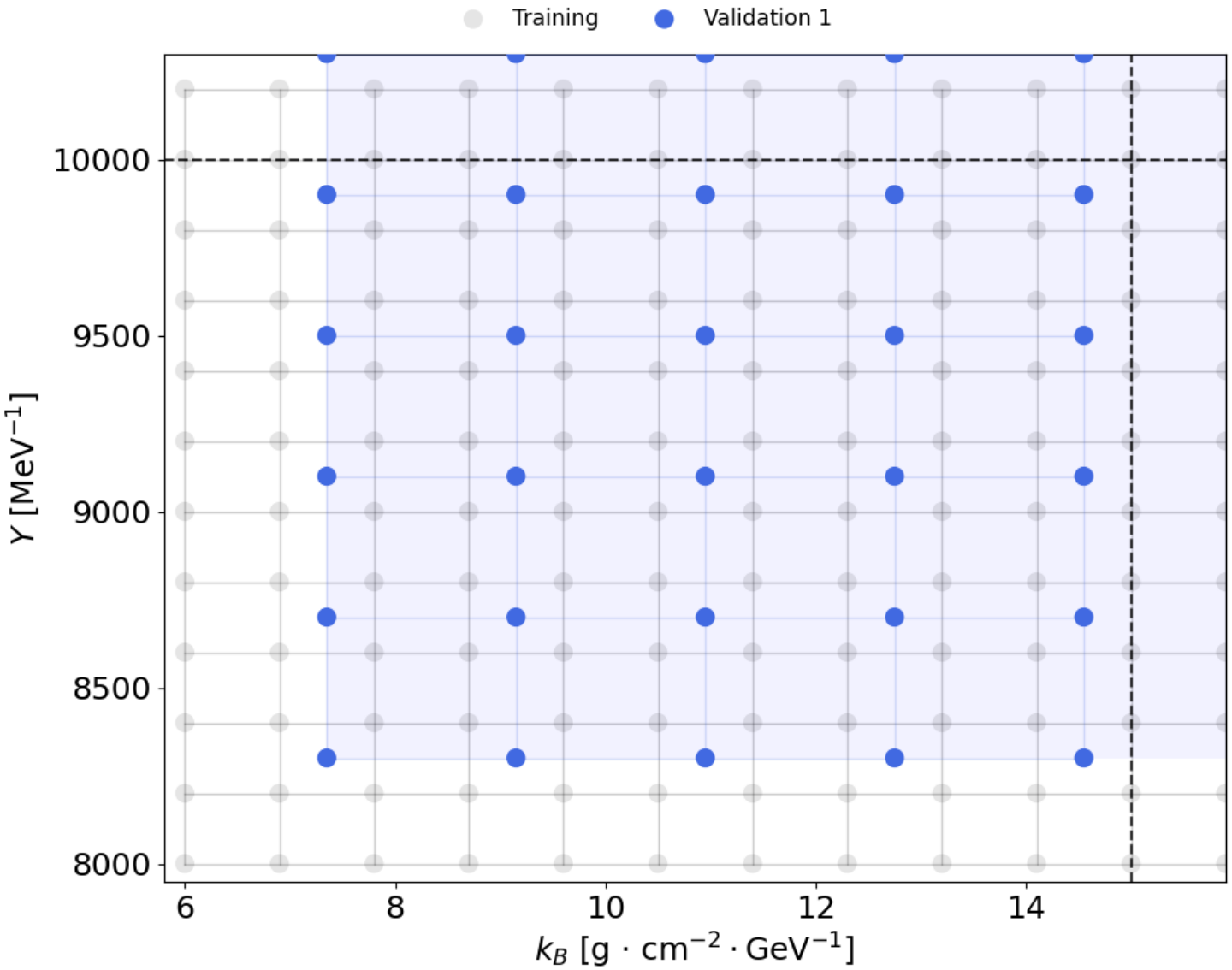
- 21 points per param (21^3 combs)
- 10k events x 5 sources per point (~500M events)

Validation data #1

- 10 points per param (10^3 combs)
- 10k x 5 events per point (50M events)

→ cover most of parameter space

Y and k_B example



A. Gavrikov, A. Serafini, D. Dolzhikov et al., *Commun Phys* **9**, 63 (2026)

Datasets: training/validation strategy

Always in the center of the detector

Training data
- 21 points per param (21^3 combs)
- 10k events x 5 sources per point (~500M events)

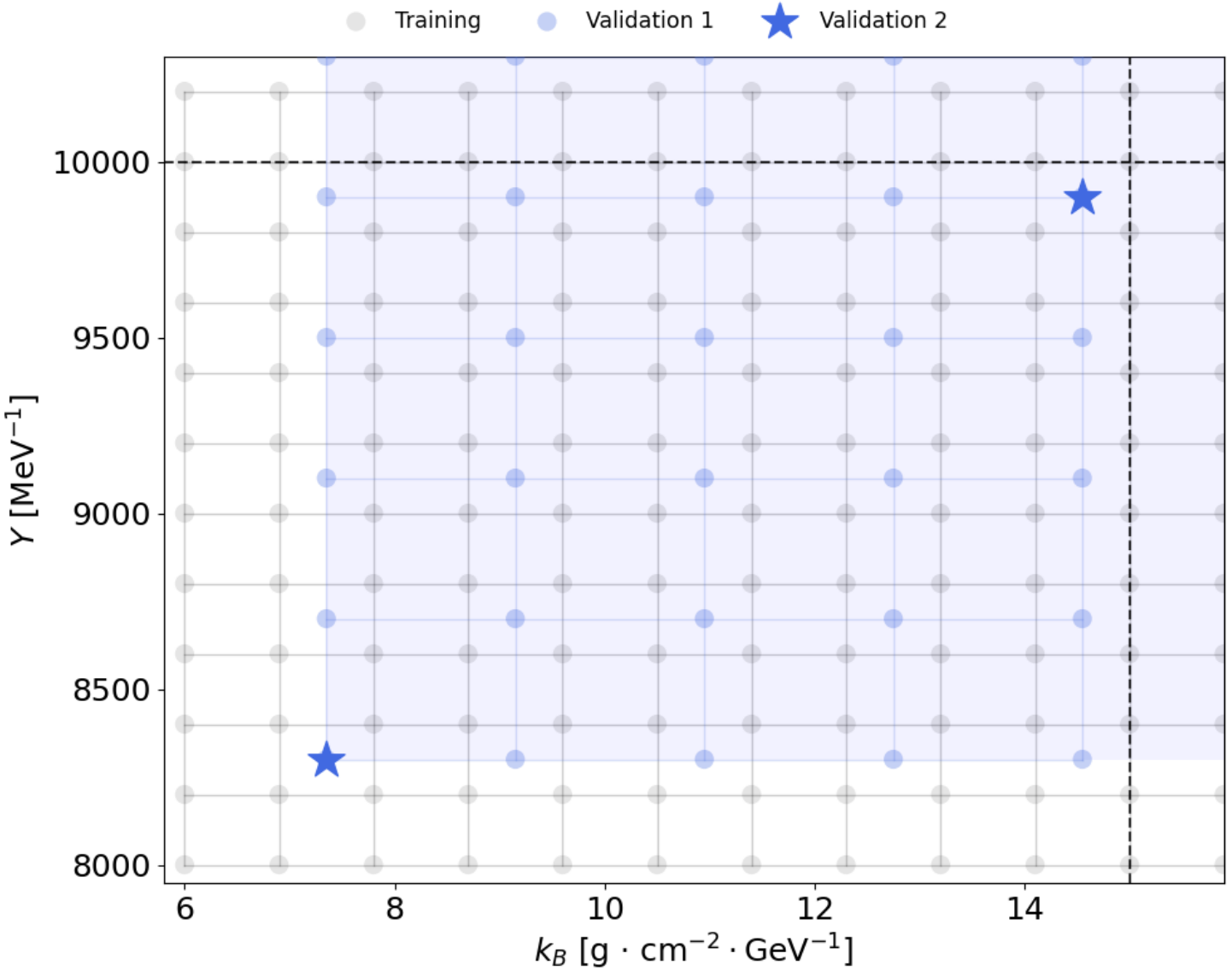
Validation data #1
- 10 points per param (10^3 combs)
- 10k x 5 events per point (50M events)

→ cover most of parameter space

Validation data #2
- 3 points in total
- 10M x 5 events per point (150M events)

→ anchor PDF to high-stat spectra

Y and k_B example



A. Gavrikov, A. Serafini, D. Dolzhikov et al., *Commun Phys* **9**, 63 (2026)

Datasets: training/validation strategy

Always in the center of the detector

Y and k_B example

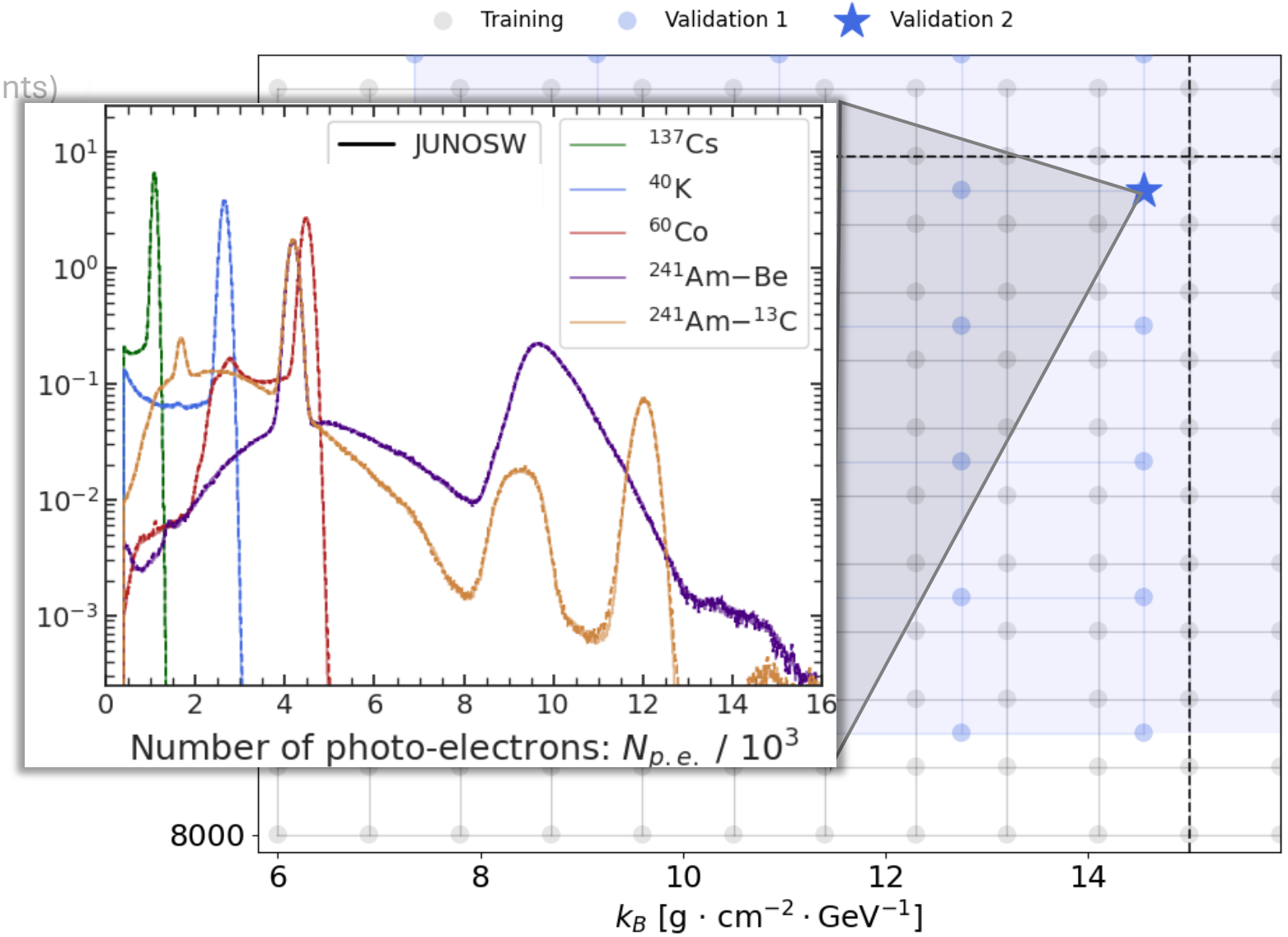
Training data
- 21 points per param (21^3 combs)
- 10k events x 5 sources per point (~500M events)

Validation data #1
- 10 points per param (10^3 combs)
- 10k x 5 events per point (50M events)

→ cover most of parameter space

Validation data #2
- 3 points in total
- 10M x 5 events per point (150M events)

→ anchor PDF to high-stat spectra



A. Gavrikov, A. Serafini, D. Dolzhikov et al., *Commun Phys* **9**, 63 (2026)

Datasets: training/validation strategy

Always in the center of the detector

Training data
- 21 points per param (21^3 combs)
- 10k events x 5 sources per point (~500M events)

Validation data #1
- 10 points per param (10^3 combs)
- 10k x 5 events per point (50M events)

→ cover most of parameter space

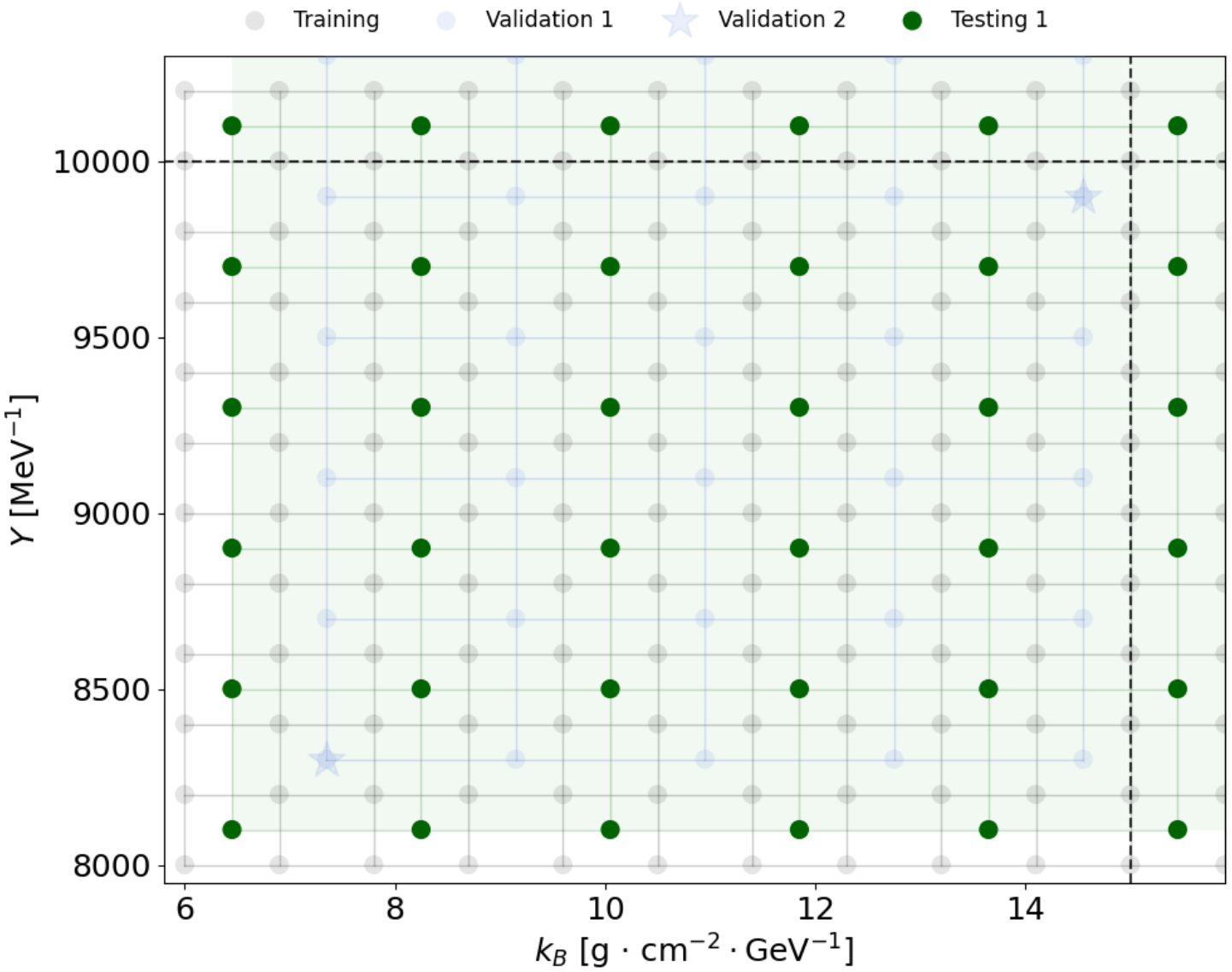
Validation data #2
- 3 points in total
- 10M x 5 events per point (150M events)

→ anchor PDF to high-stat spectra

Testing data #1
- 10 points per param (10^3 combs)
- 10k x 5 events per point (50M events)

→ check possible **biases over parameter space**

Y and k_B example



A. Gavrikov, A. Serafini, D. Dolzhikov et al., *Commun Phys* **9**, 63 (2026)

Datasets: training/validation strategy

Always in the center of the detector

Training data
- 21 points per param (21^3 combs)
- 10k events x 5 sources per point (~500M events)

Validation data #1
- 10 points per param (10^3 combs)
- 10k x 5 events per point (50M events)

→ cover most of parameter space

Validation data #2
- 3 points in total
- 10M x 5 events per point (150M events)

→ anchor PDF to high-stat spectra

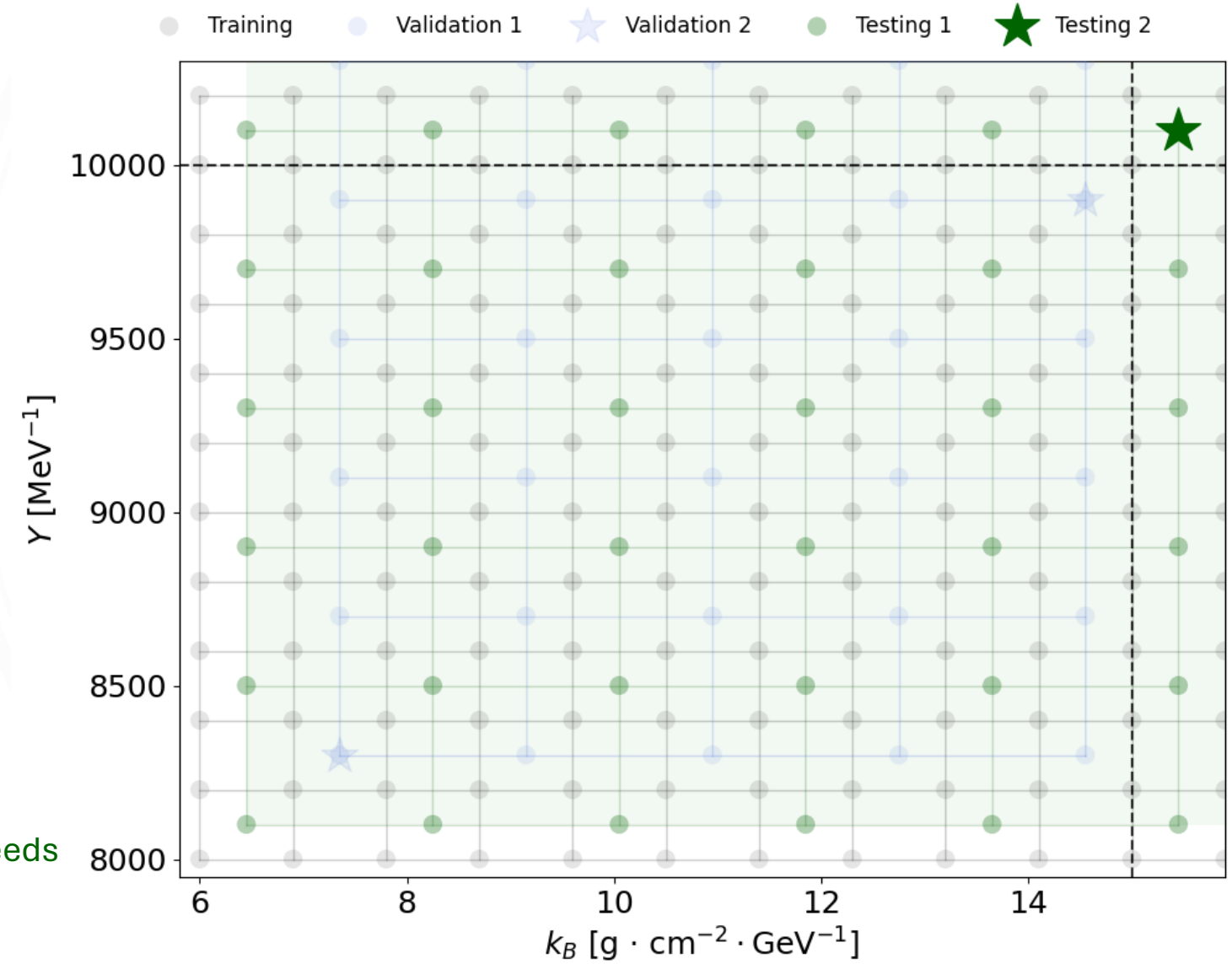
Testing data #1
- 10 points per param (10^3 combs)
- 10k x 5 events per point (50M events)

→ check possible **biases over parameter space**

Testing data #2
- 1 point
- 1000 x [1k; 2k; 5k; 10k; 25k] events x 5 with diff. seeds

→ Analysis of **systematic uncertainty** of the model

Y and k_B example



A. Gavrikov, A. Serafini, D. Dolzhikov et al., *Commun Phys* **9**, 63 (2026)

Datasets: training/validation strategy

Always in the center of the det

Training data
- 21 points per param (21^3 combs)
- 10k events x 5 sources per point (~500M events)

Validation data #1
- 10 points per param (10^3 combs)
- 10k x 5 events per point (50M events)

→ cover most of parameter space

Validation data #2
- 3 points in total
- 10M x 5 events per point (150M events)

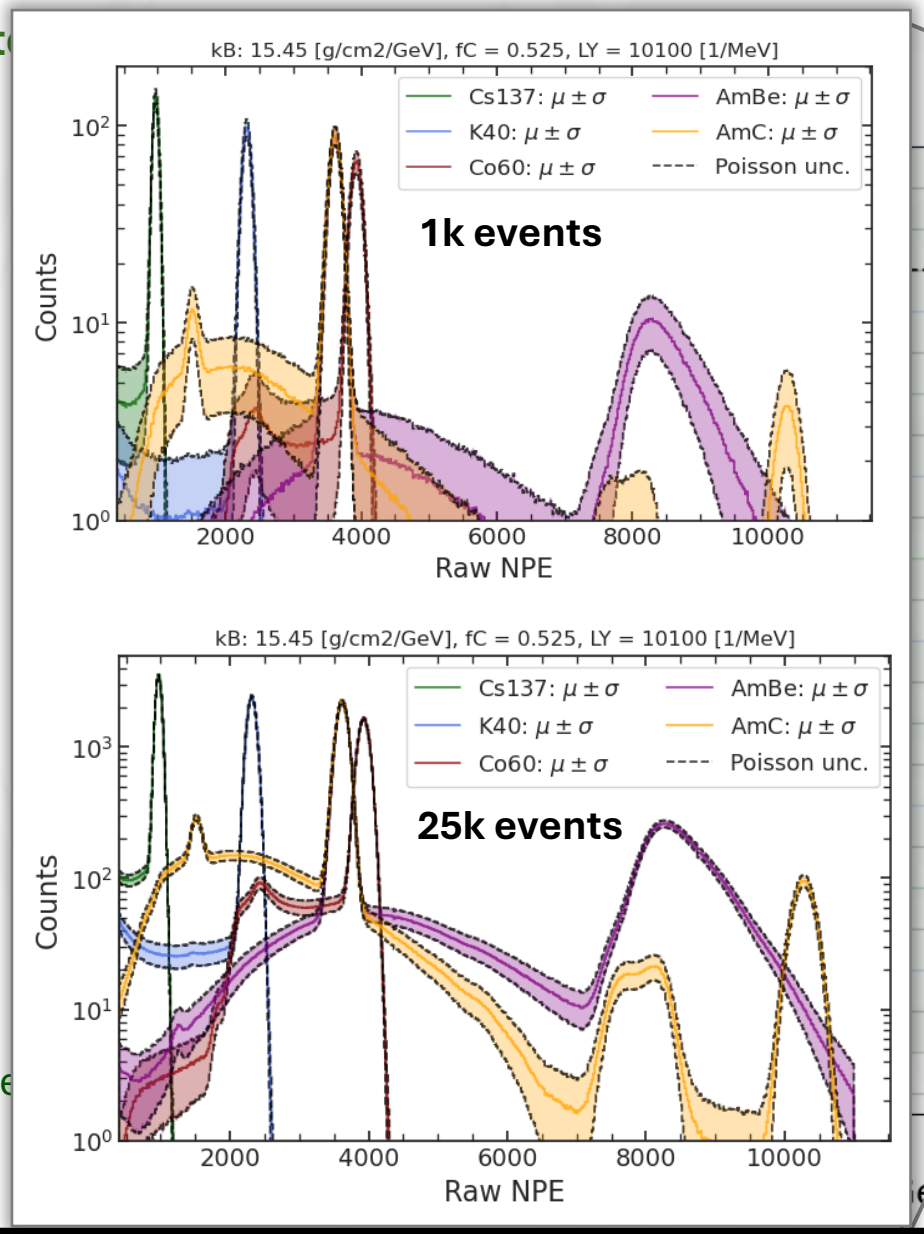
→ anchor PDF to high-stat spectra

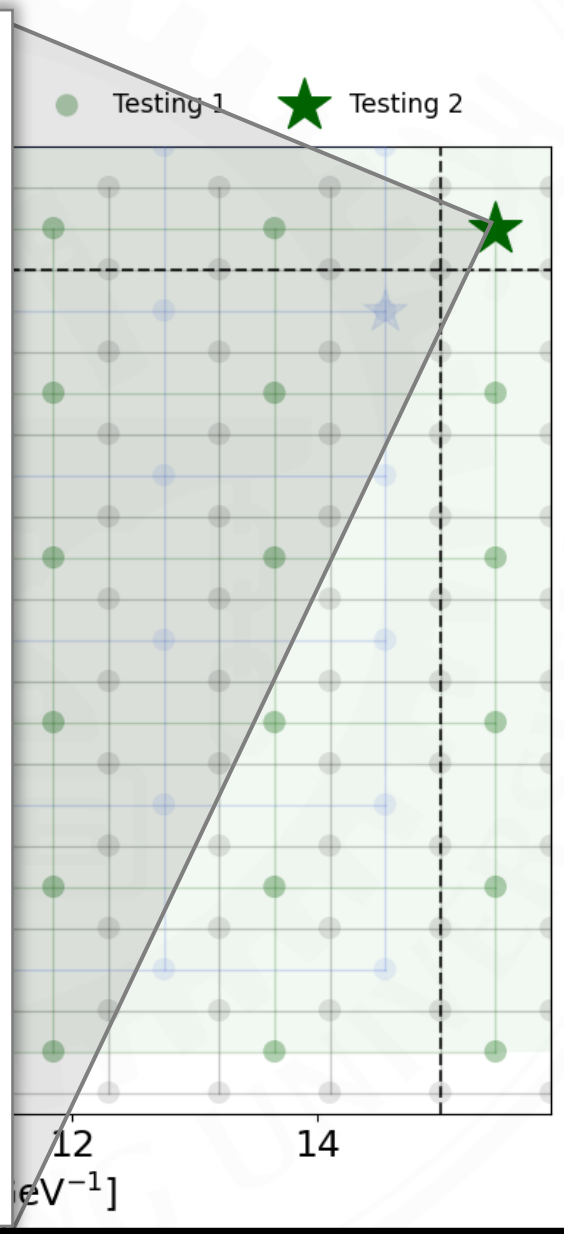
Testing data #1
- 10 points per param (10^3 combs)
- 10k x 5 events per point (50M events)

→ check possible **biases over parameter space**

Testing data #2
- 1 point
- 1000 x [1k; 2k; 5k; 10k; 25k] events x 5 with diff. see

→ Analysis of **systematic uncertainty** of the model





A. Gavrikov, A. Serafini, D. Dolzhikov et al., *Commun Phys* **9**, 63 (2026)

Datasets: training/validation strategy

Always in the center of the detector

Training data
- 21 points per param (21^3 combs)
- 10k events x 5 sources per point (~500M events)

Validation data #1
- 10 points per param (10^3 combs)
- 10k x 5 events per point (50M events)

→ cover most of parameter space

Validation data #2
- 3 points in total
- 10M x 5 events per point (150M events)

→ anchor PDF to high-stat spectra

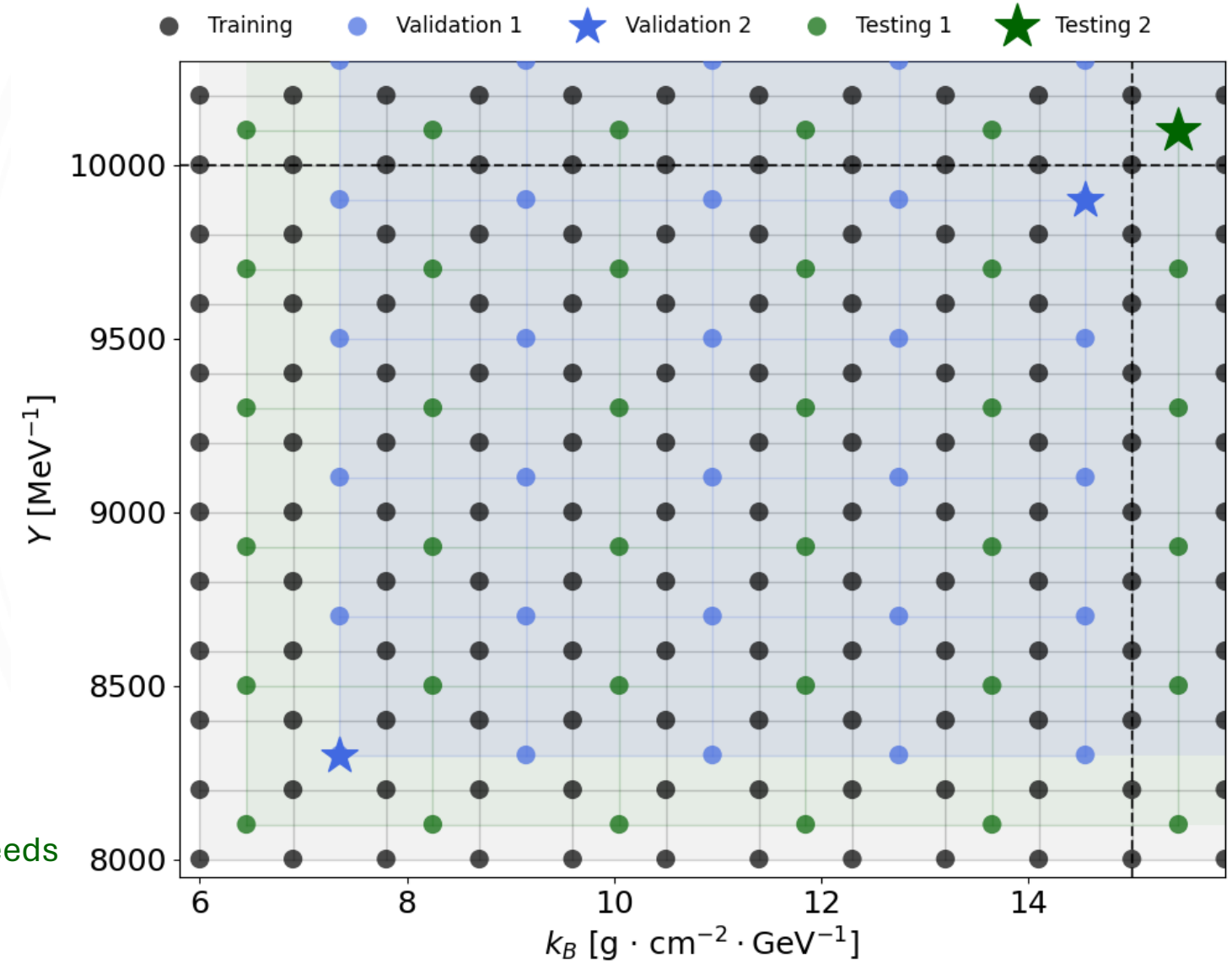
Testing data #1
- 10 points per param (10^3 combs)
- 10k x 5 events per point (50M events)

→ check possible **biases over parameter space**

Testing data #2
- 1 point
- 1000 x [1k; 2k; 5k; 10k; 25k] events x 5 with diff. seeds

→ Analysis of **systematic uncertainty** of the model

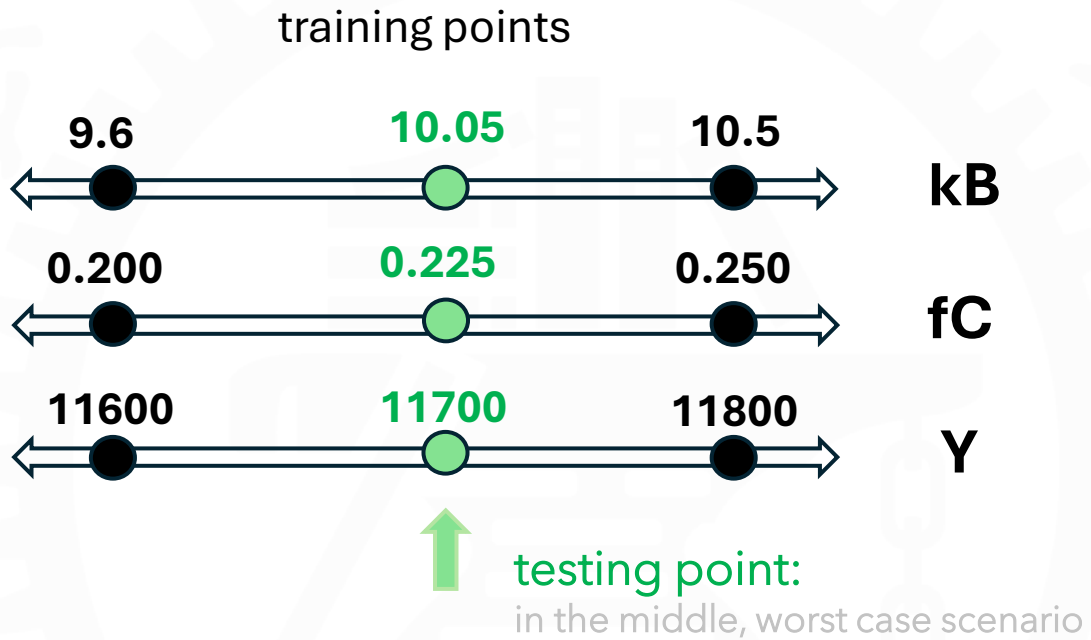
Y and k_B example



A. Gavrikov, A. Serafini, D. Dolzhikov et al., *Commun Phys* **9**, 63 (2026)

Some results: spectra modelling

Example with NFDE:



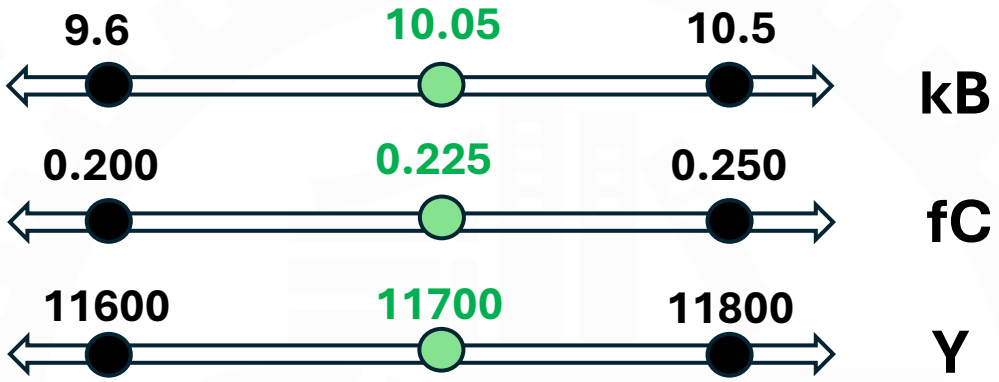
A. Gavrikov, A. Serafini, D. Dolzhikov et al., *Commun Phys* **9**, 63 (2026)

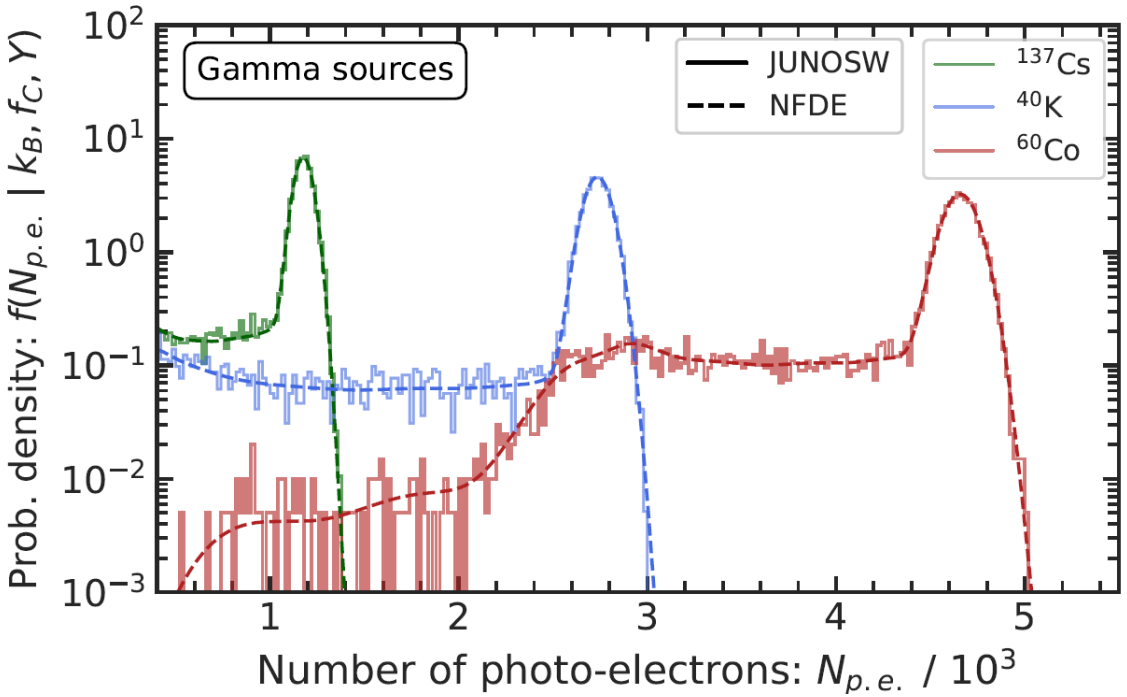
Some results: spectra modelling

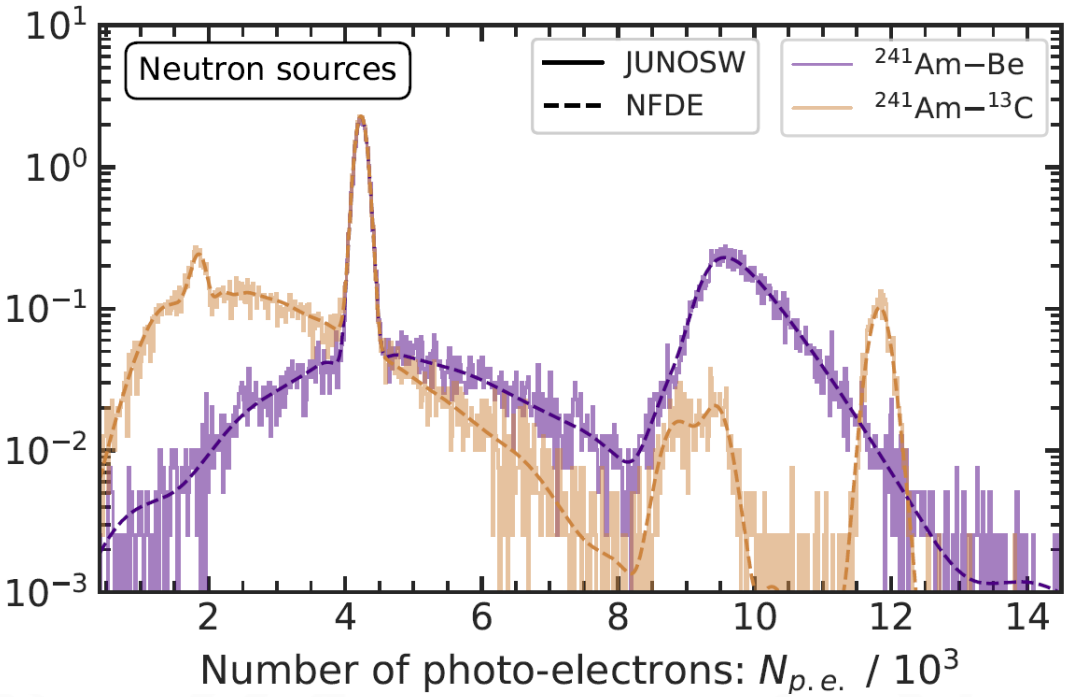
Example with NFDE:

Model provides spectra estimates for ***any continuous*** values of the parameters

Accurate interpolation. Model's outputs are smooth and denoised







A. Gavrikov, A. Serafini, D. Dolzhikov et al., *Commun Phys* **9**, 63 (2026)

Some results: parameter estimation

Example with NFDE:

- **Combined fit** on all sources together
- **Cost function:** Extended unbinned likelihood (NFDE)
- **Optimizer:** Nested Sampling (ultranest)*

→ Estimate the **kB, fC, LY best parameters**

→ Explores full phase space, **provides full posterior**

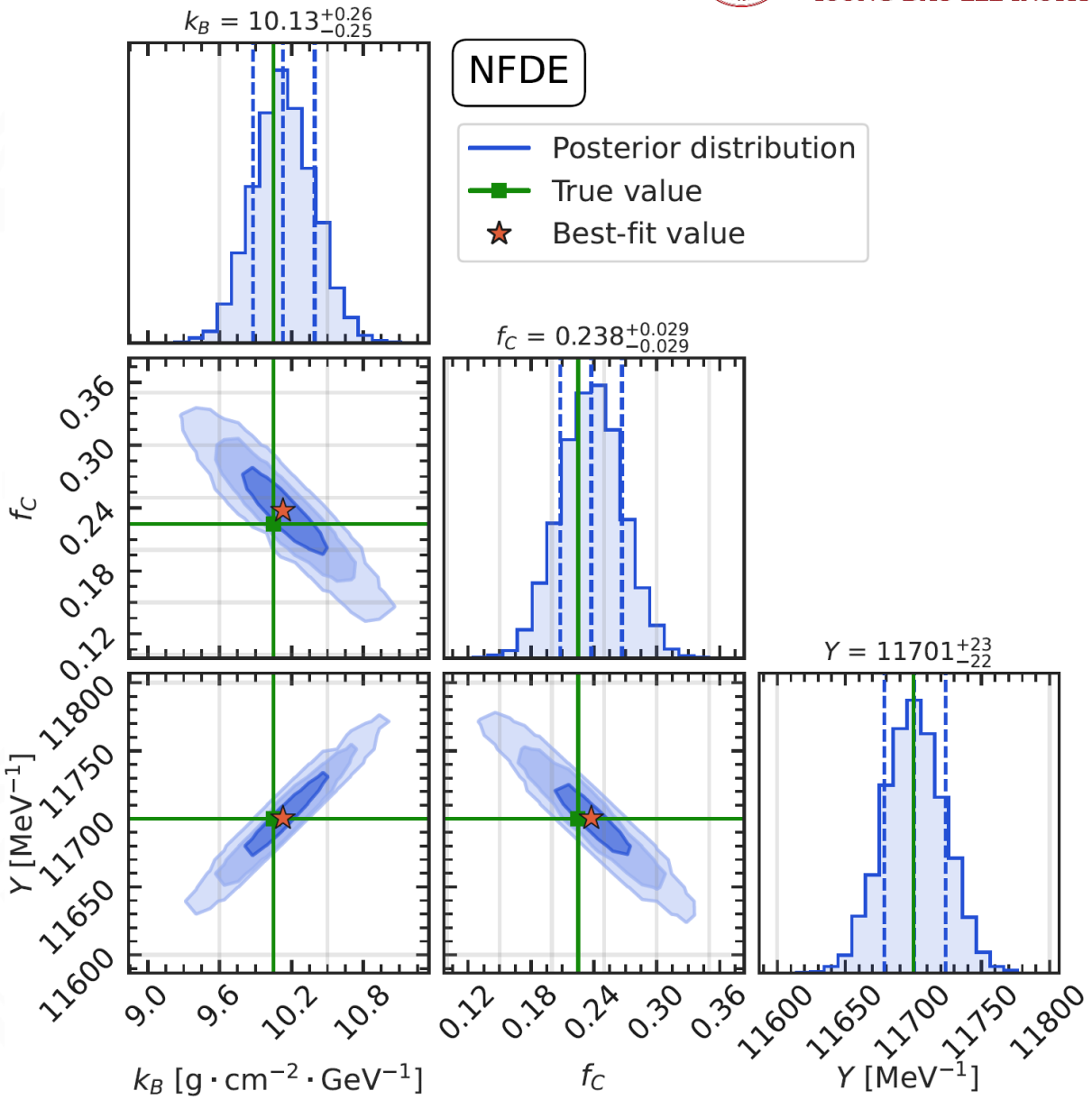
→ **Shows correlations** between the parameters

Parameters estimation combined:

kB: **10.13 ± 0.26**	10.05 [g/cm^2/GeV]
fC: **0.238 ± 0.029**	0.225
LY: **11701 ± 23**	11700 [1/MeV]

→ **well within 1σ** (fluctuations expected for 1 fit)

*all fits performed with Minuit, MCMC, Nested Sampling (ultranest) → consistent results



A. Gavrikov, A. Serafini, D. Dolzhikov et al., *Commun Phys* **9**, 63 (2026)

Some results: parameter estimation

Example with NFDE:

- **Combined fit** on all sources together
- **Cost function:** Extended unbinned likelihood (NFDE)
- **Optimizer:** Nested Sampling (ultranest)*

→ Estimate the **kB, fC, LY best parameters**

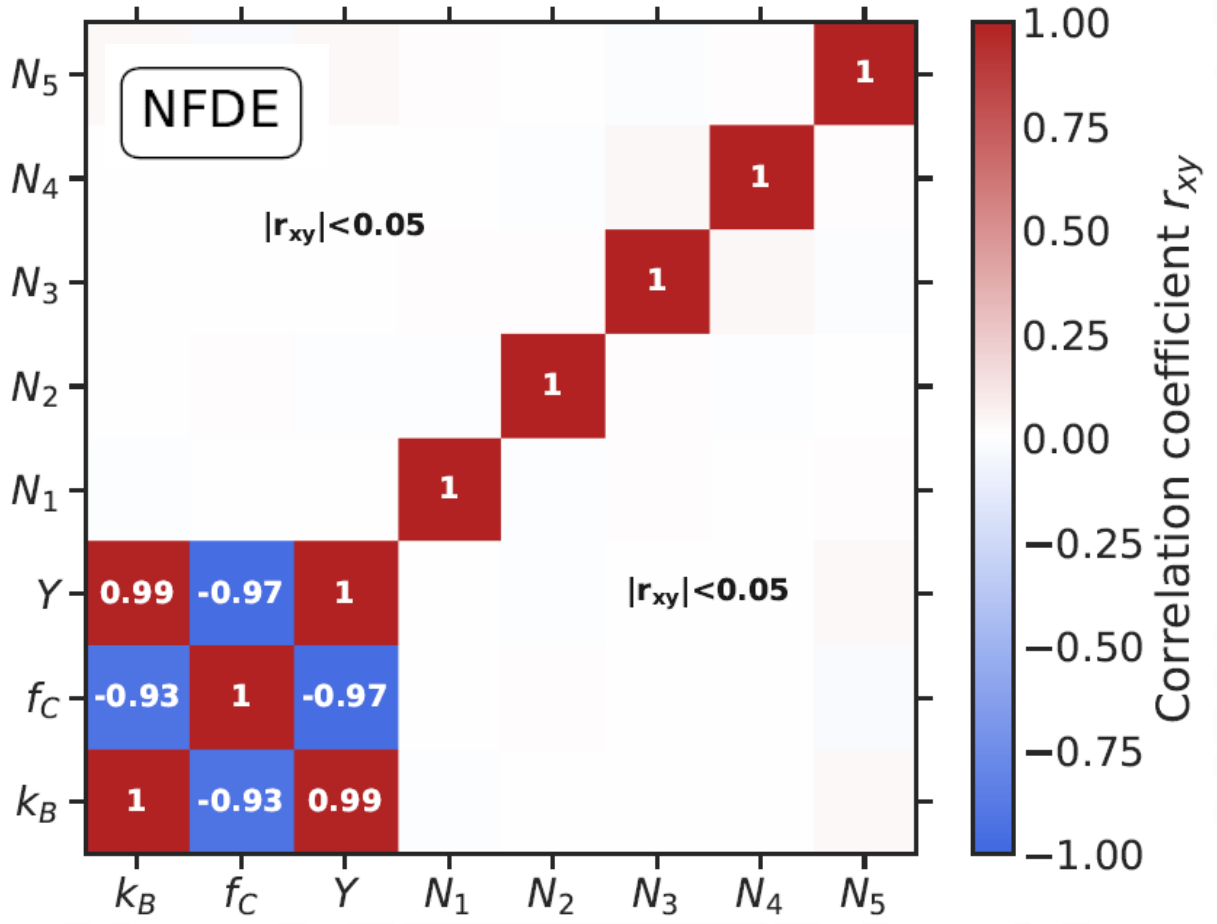
→ Explores full phase space, **provides full posterior**

→ **Shows correlations** between the parameters

Parameters estimation combined:

kB: **10.13 ± 0.26**	10.05 [g/cm^2/GeV]
fC: **0.238 ± 0.029**	0.225
LY: **11701 ± 23**	11700 [1/MeV]

→ **well within 1σ** (fluctuations expected for 1 fit)



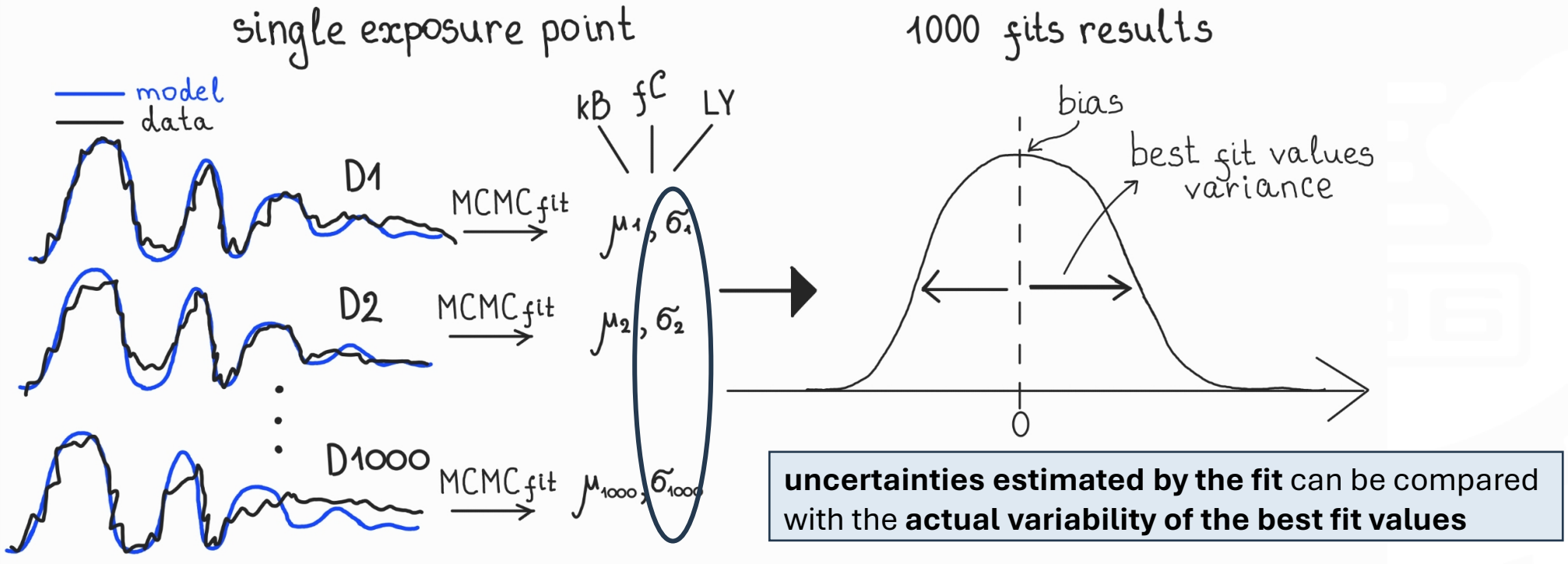
*all fits performed with Minuit, MCMC, Nested Sampling (ultranest) → consistent results

A. Gavrikov, A. Serafini, D. Dolzhikov et al., *Commun Phys* **9**, 63 (2026)

Evaluate ML-driven systematic uncertainty

To perform systematic uncertainty estimation analysis, we use the testing dataset 2:

- **Unseen during training point** in the parameter space: kB, fC, LY = (15.45, 0.525, 10100)
- **5 different exposures**: 1k, 2k, 5k, 10k, 25k events
- **1000 datasets with different MC generator seed** per each exposure



A. Gavrikov, A. Serafini, D. Dolzhikov et al., *Commun Phys* **9**, 63 (2026)

Evaluate ML-driven systematic uncertainty

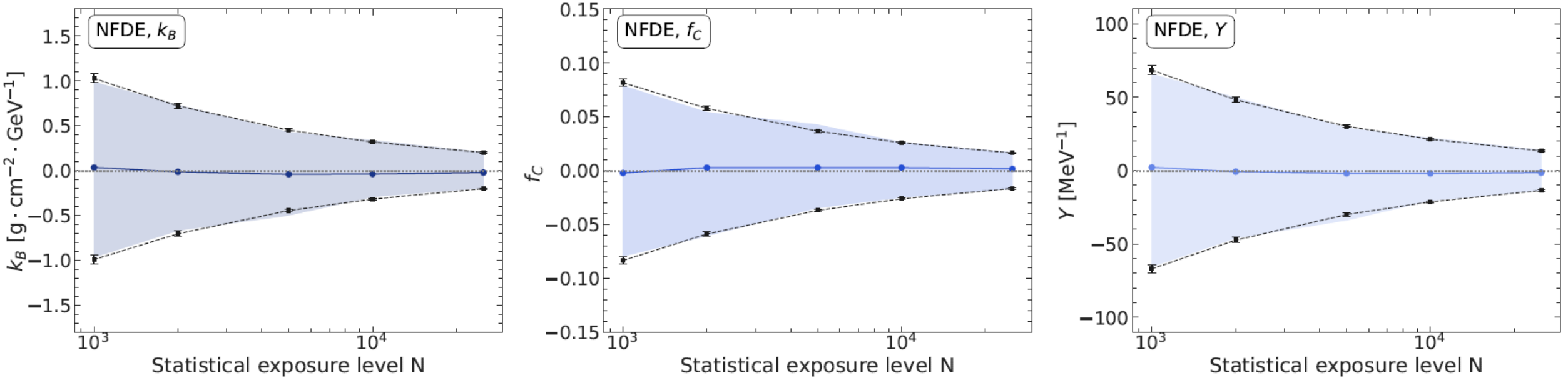
To perform systematic uncertainty estimation analysis, we use the testing dataset 2:

- **Unseen during training point** in the parameter space: kB, fC, LY = (15.45, 0.525, 10100)
- **5 different exposures**: 1k, 2k, 5k, 10k, 25k events
- **1000 datasets with different MC generator seed** per each exposure

→ **Estimated uncertainty matches observed variability**

Estimated Stat. Unc.

Observed Std. Dev.



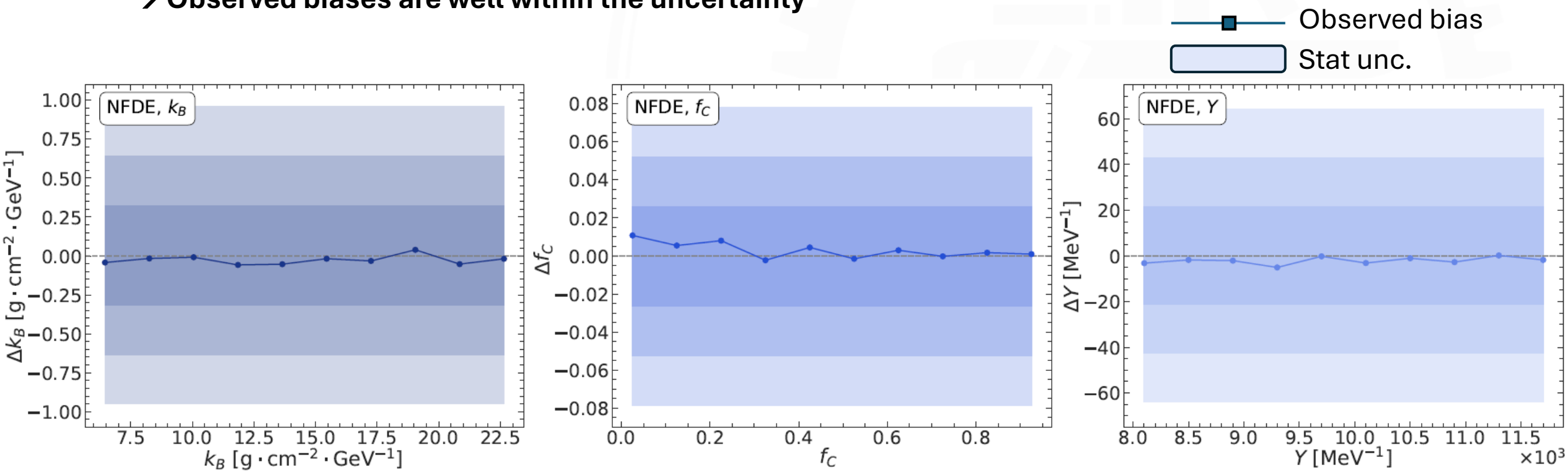
A. Gavrikov, A. Serafini, D. Dolzhikov et al., *Commun Phys* **9**, 63 (2026)

Estimation of ML-driven systematic bias

Using **the testing dataset 1,** one can check the bias across **different points**:

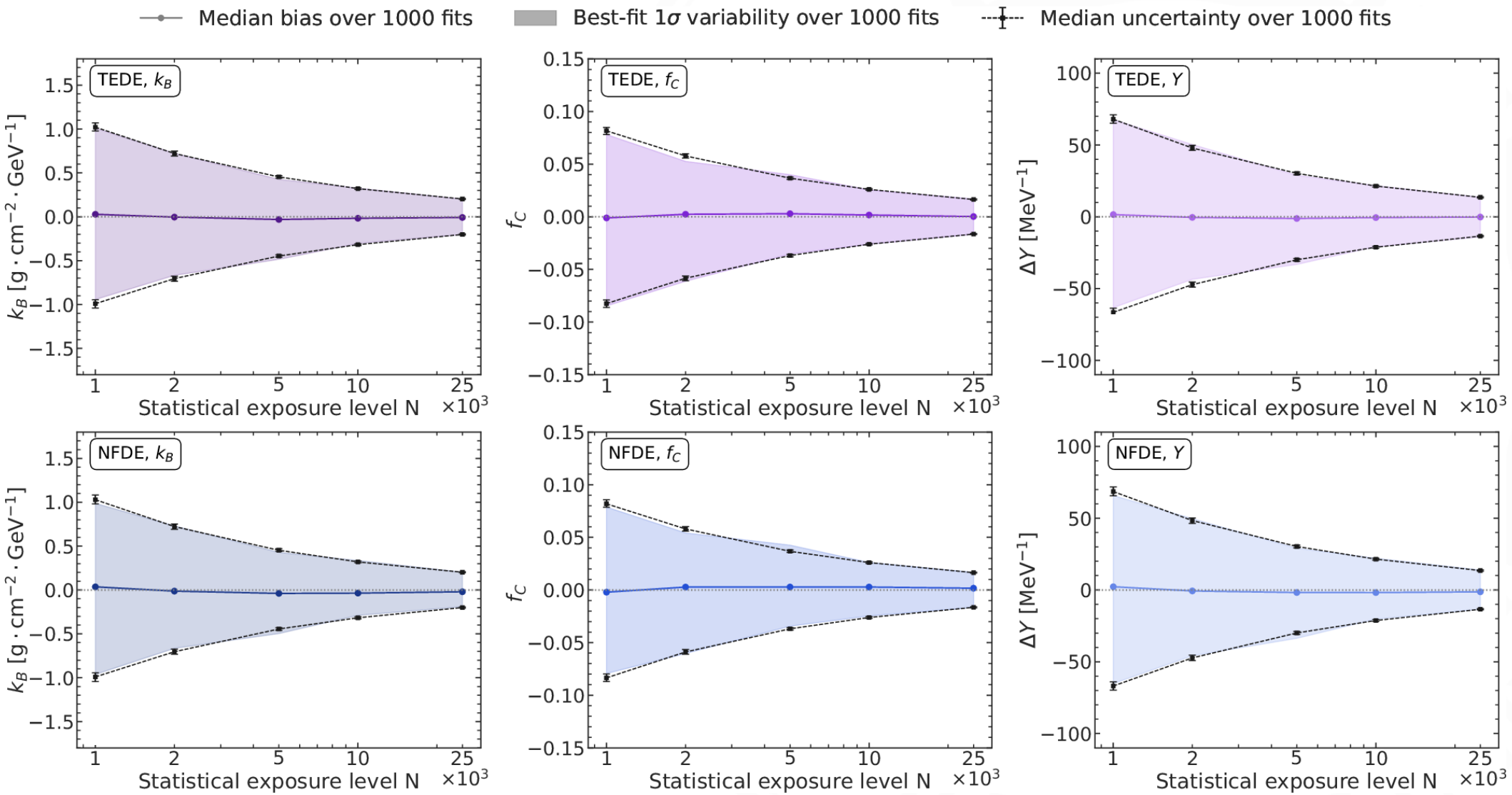
- ➢ Run MCMC fits per each testing point of the dataset
- ➢ Compare bias with the uncertainty **obtained by the previous analysis** for the 10k exposure point

→ **Observed biases are well within the uncertainty**



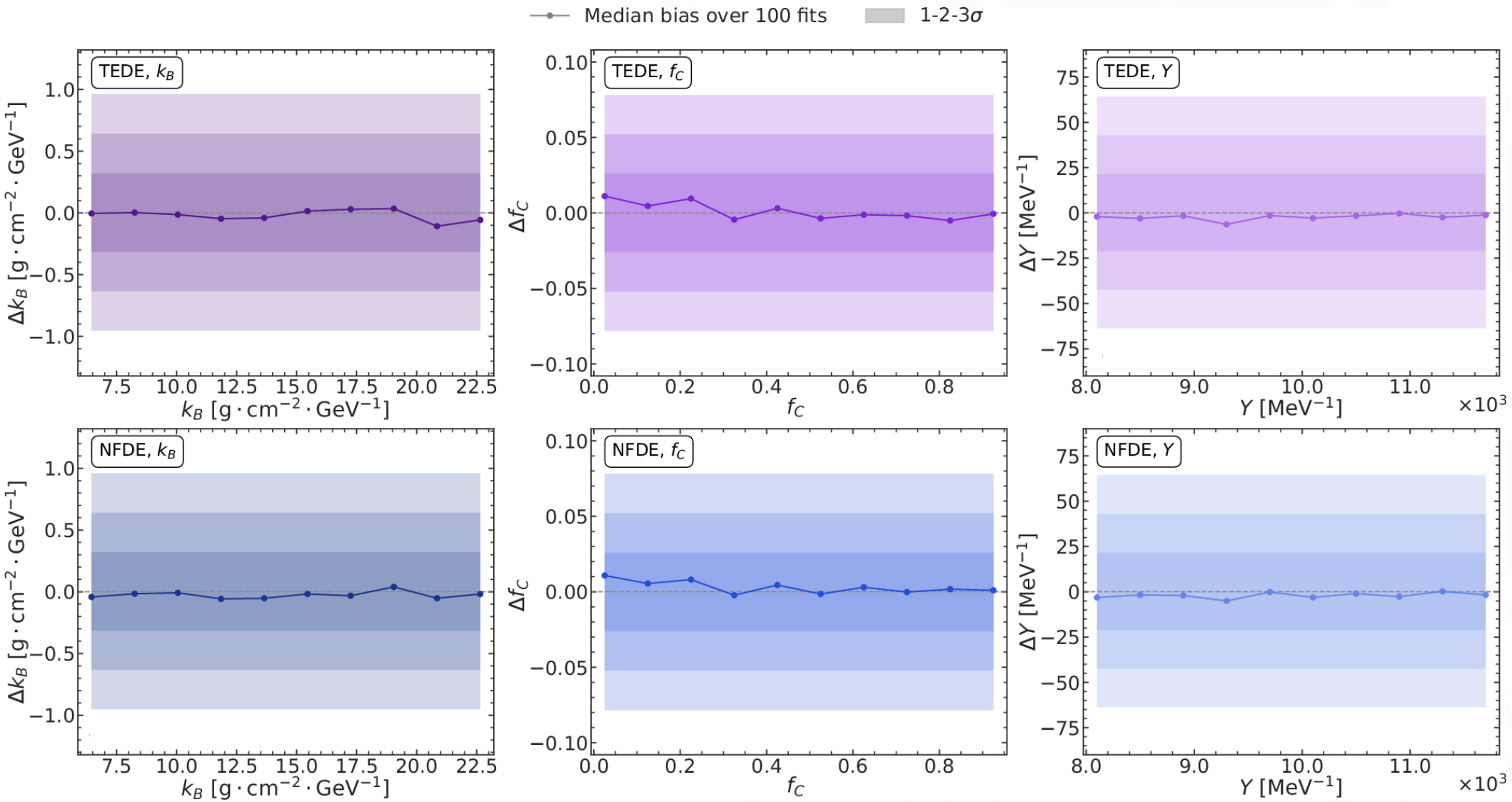
A. Gavrikov, A. Serafini, D. Dolzhikov et al., *Commun Phys* **9**, 63 (2026)

TEDE vs. NFDE (uncertainty)



A. Gavrikov, A. Serafini, D. Dolzhikov et al., *Commun Phys* **9**, 63 (2026)

TEDE vs. NFDE (bias)



A. Gavrikov, A. Serafini, D. Dolzhikov et al., *Commun Phys* **9**, 63 (2026)

Summary

Summary

- **Challenge: systematics investigation** for precision **neutrino physics with JUNO and** accurate energy calibration

  → requires an extremely **well tuned MC**, but the **traditional MC tuning** process is **computationally prohibitive**

Summary

- **Challenge: systematics investigation** for precision **neutrino physics with JUNO and** accurate energy calibration
  → requires an extremely **well tuned MC**, but the **traditional MC tuning** process is **computationally prohibitive**
- **Solution:** we developed a **Simulation-Based Inference framework** using two **neural likelihood estimators** (TEDE and NFDE) that replace the slow MC simulation with a **fast, accurate surrogate** models

Summary

- **Challenge: systematics investigation** for precision **neutrino physics with JUNO and** accurate energy calibration

  → requires an extremely **well tuned MC**, but the **traditional MC tuning** process is **computationally prohibitive**

- **Solution:** we developed a **Simulation-Based Inference framework** using two **neural likelihood estimators** (TEDE and NFDE) that replace the slow MC simulation with a **fast, accurate surrogate** models

- **Key Achievements:**
  - our **models successfully learn** the complex, **non-linear energy response of the JUNO detector MC**, including the strong correlations between k_B, f_C and Y parameters in the MC
  - when integrated with **Bayesian inference**, our method recovers all parameters with **near-zero systematic bias**
  - parameter **uncertainties are limited purely by statistics** and scale correctly as $1/\sqrt{N}$

    → possible to achieve <1% systematics

Summary

- **Challenge: systematics investigation** for precision **neutrino physics with JUNO and** accurate energy calibration

  → requires an extremely **well tuned MC**, but the **traditional MC tuning** process is **computationally prohibitive**

- **Solution:** we developed a **Simulation-Based Inference framework** using two **neural likelihood estimators** (TEDE and NFDE) that replace the slow MC simulation with a **fast, accurate surrogate** models

- **Key Achievements:**
  - our **models successfully learn** the complex, **non-linear energy response of the JUNO detector MC**, including the strong correlations between k_B, f_C and Y parameters in the MC
  - when integrated with **Bayesian inference**, our method recovers all parameters with **near-zero systematic bias**
  - parameter **uncertainties are limited purely by statistics** and scale correctly as $1/\sqrt{N}$

    → possible to achieve <1% systematics

- **The approach itself is detector-agnostic** and can be applied in other experiments to solve similar problems

- **Ongoing work:** neural scaling laws for optimizing the simulation budgets & implementation and validation of the approach within the experiment

Thank you!

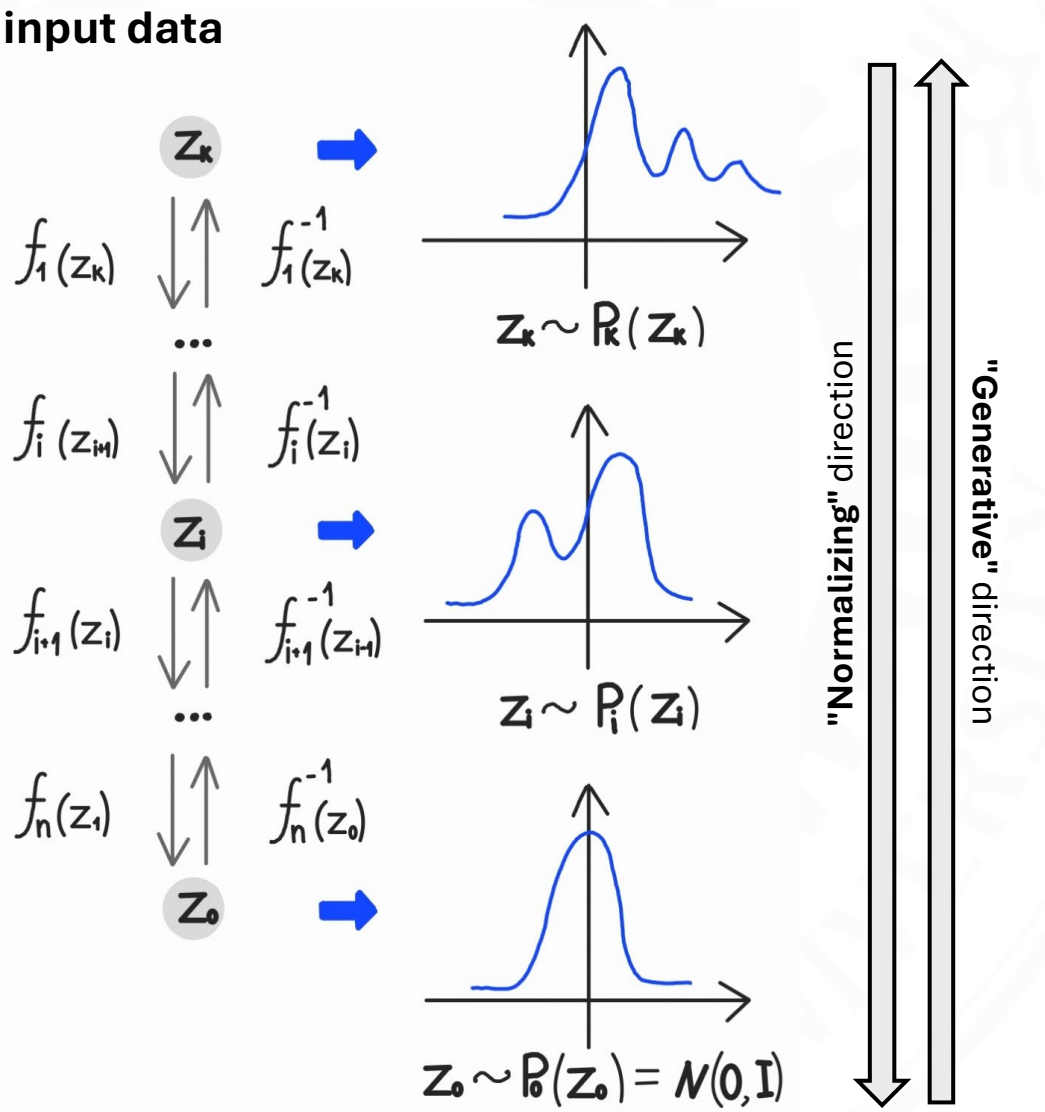
Repo

Paper

Backup

Normalizing flows-based density estimation

- **Normalizing flows [1]:**
  - **Generative** machine learning **model**
  - Aims to **explicitly model a density estimation**
  - Can be generalized for modeling **a conditional density estimation** as well
  - To obtain a value from the target distribution:
    - sample a value from the base distribution and passing it through all the flows

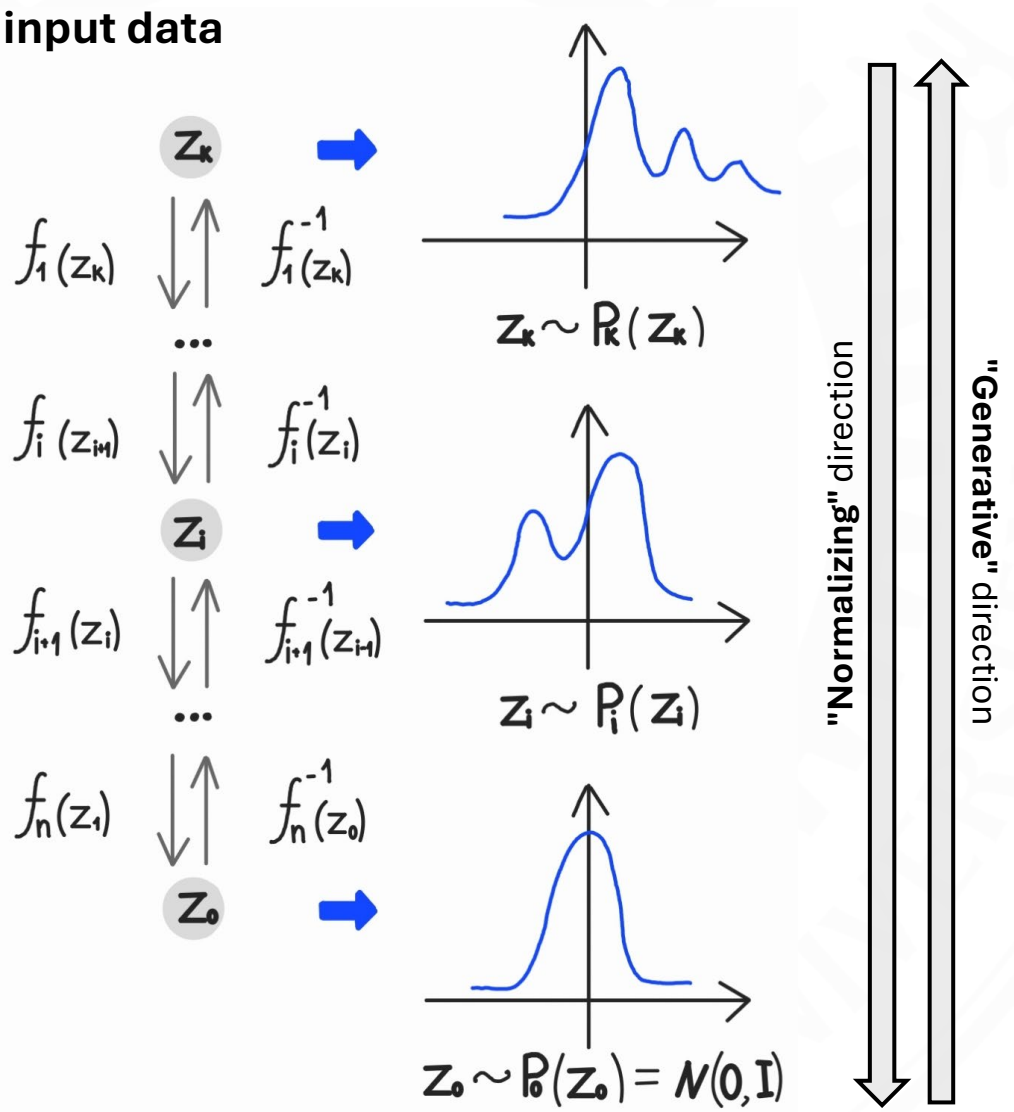


[1] R. J. Rezende et al, arxiv: 1505.05770

Normalizing flows-based density estimation

- **Normalizing flows [1]:**
  - **Generative** machine learning **model**
  - Aims to **explicitly model a density estimation**
  - Can be generalized for modeling **a conditional density estimation** as well
  - To obtain a value from the target distribution:
    - sample a value from the base distribution and passing it through all the flows

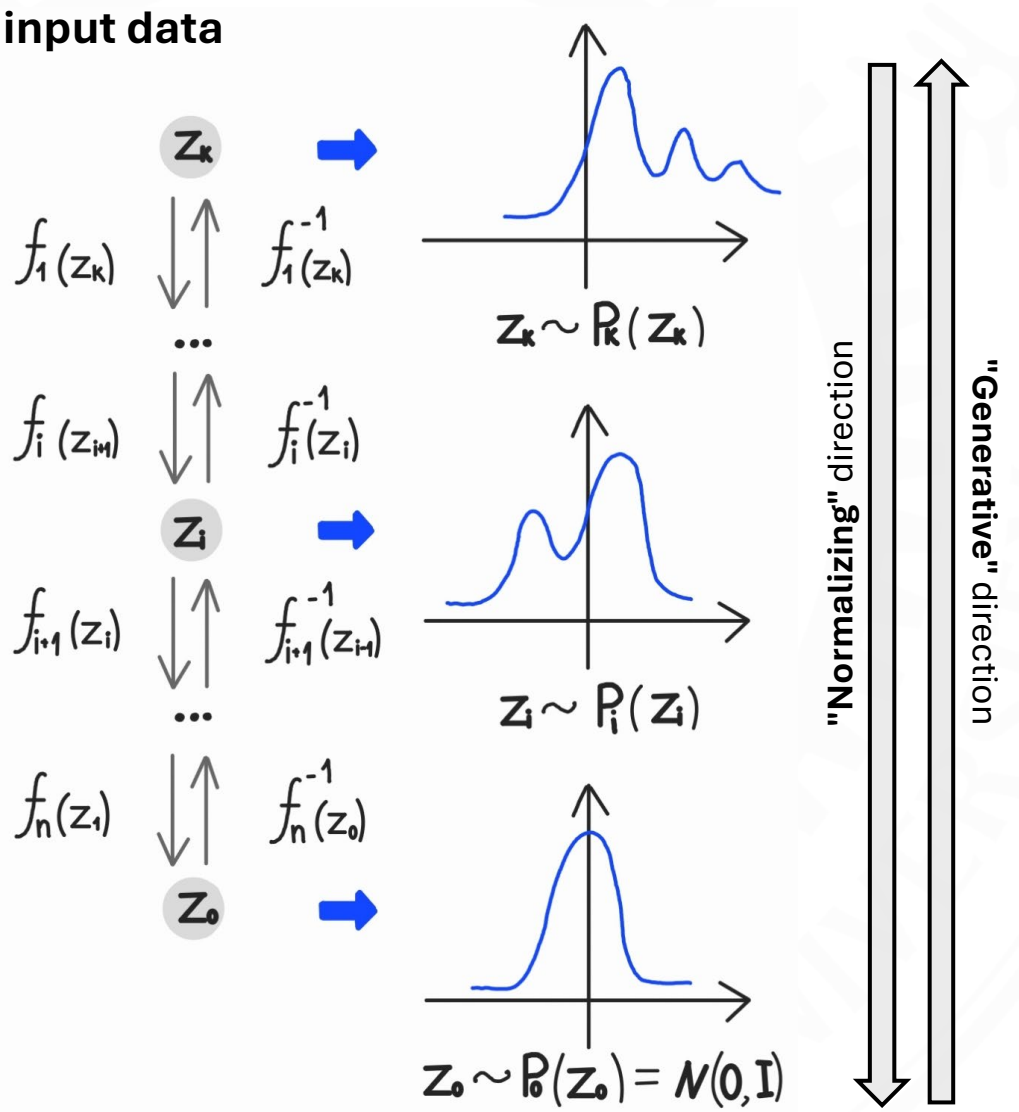
- Normalizing flows are based on an **idea of transforming a simple known distribution** (*e.g. standard Gaussian*) **to the complex, desired distribution** by applying multiple learnable (*using data*) transformations
- **Each transformation** is called Flow and **must be invertible**



[1] R. J. Rezende et al, arxiv: 1505.05770

Normalizing flows-based density estimation

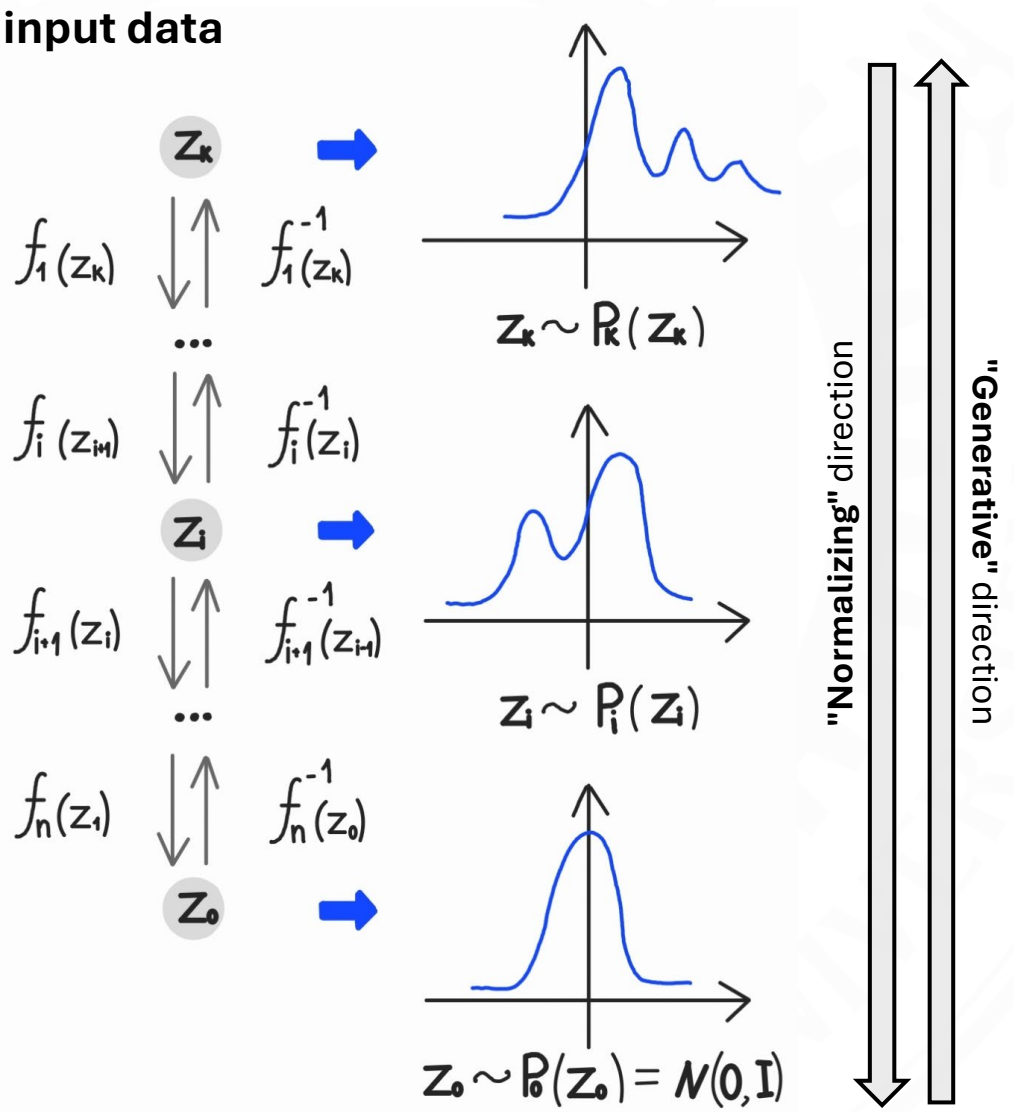
- **Normalizing flows [1]:**
  - Transformations can be very diverse as long as they follow **the invertibility condition**
  - **Planar flow**:
    - $f_\theta(x) = x + u_\theta \tanh\left(x w_\theta + b_\theta\right)$



[1] R. J. Rezende et al, arxiv: 1505.05770

Normalizing flows-based density estimation

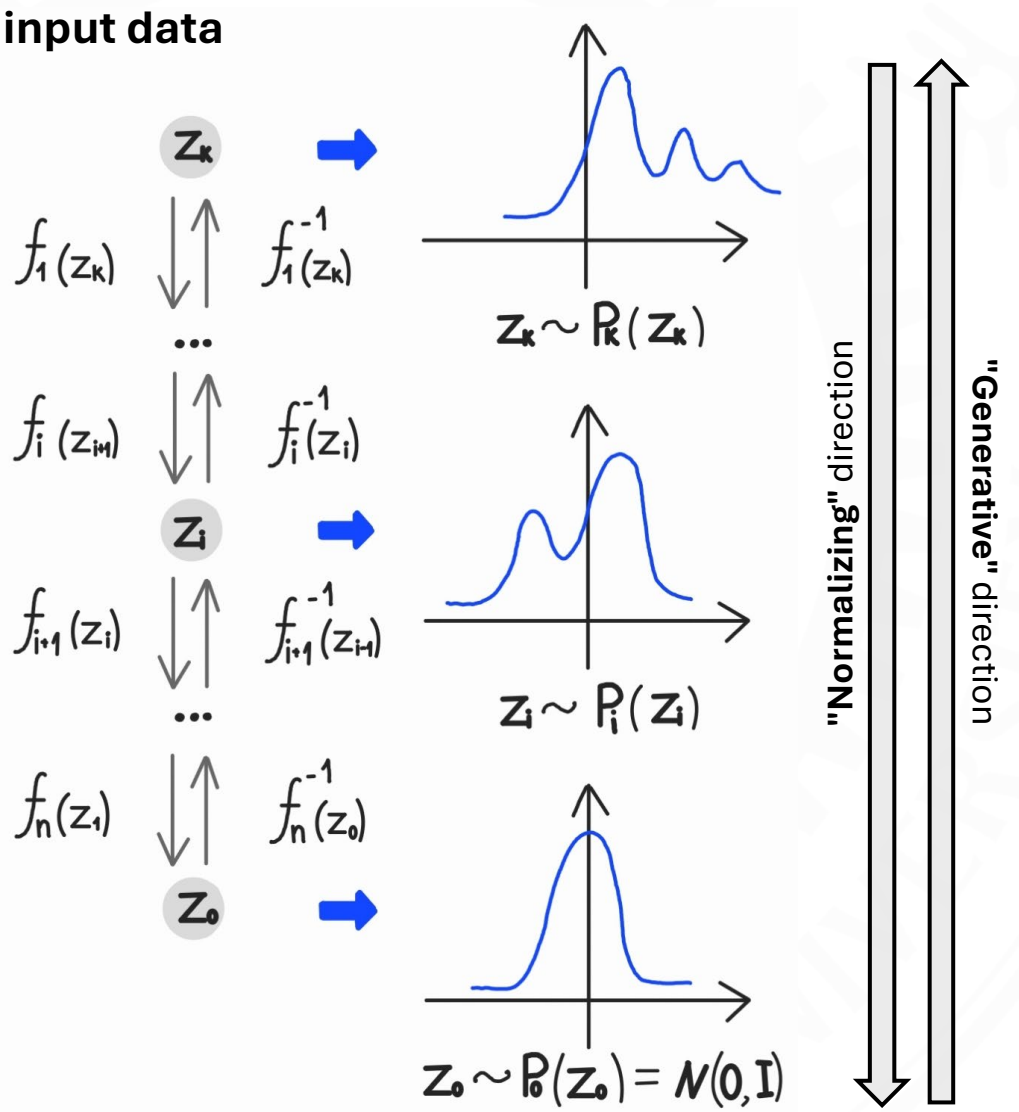
- **Normalizing flows [1]:**
  - Transformations can be very diverse as long as they follow **the invertibility condition**
  - **Planar flow**:
    - $f_\theta(x) = x + u_\theta \tanh\left(x w_\theta + b_\theta\right)$
    - where parameters $u_\theta, w_\theta, b_\theta$ are parametrized by neural networks to include the conditions (MC parameters + source type): **conditional probability**



[1] R. J. Rezende et al, arxiv: 1505.05770

Normalizing flows-based density estimation

- **Normalizing flows [1]:**
  - Transformations can be very diverse as long as they follow **the invertibility condition**
  - **Planar flow**:
    - $f_\theta(x) = x + u_\theta \tanh\left(x w_\theta + b_\theta\right)$
    - where parameters $u_\theta, w_\theta, b_\theta$ are parametrized by neural networks to include the conditions (MC parameters + source type): **conditional probability**
- How to perform the inference with the model?



[1] R. J. Rezende et al, arxiv: 1505.05770

Normalizing flows-based density estimation

- **Normalizing flows [1]:**
  - Transformations can be very diverse as long as they follow **the invertibility condition**
  - **Planar flow**:
    - $f_\theta(x) = x + u_\theta \tanh(xw_\theta + b_\theta)$
    - where parameters $u_\theta, w_\theta, b_\theta$ are parametrized by neural networks to include the conditions (MC parameters + source type): **conditional probability**
- How to perform the inference with the model?

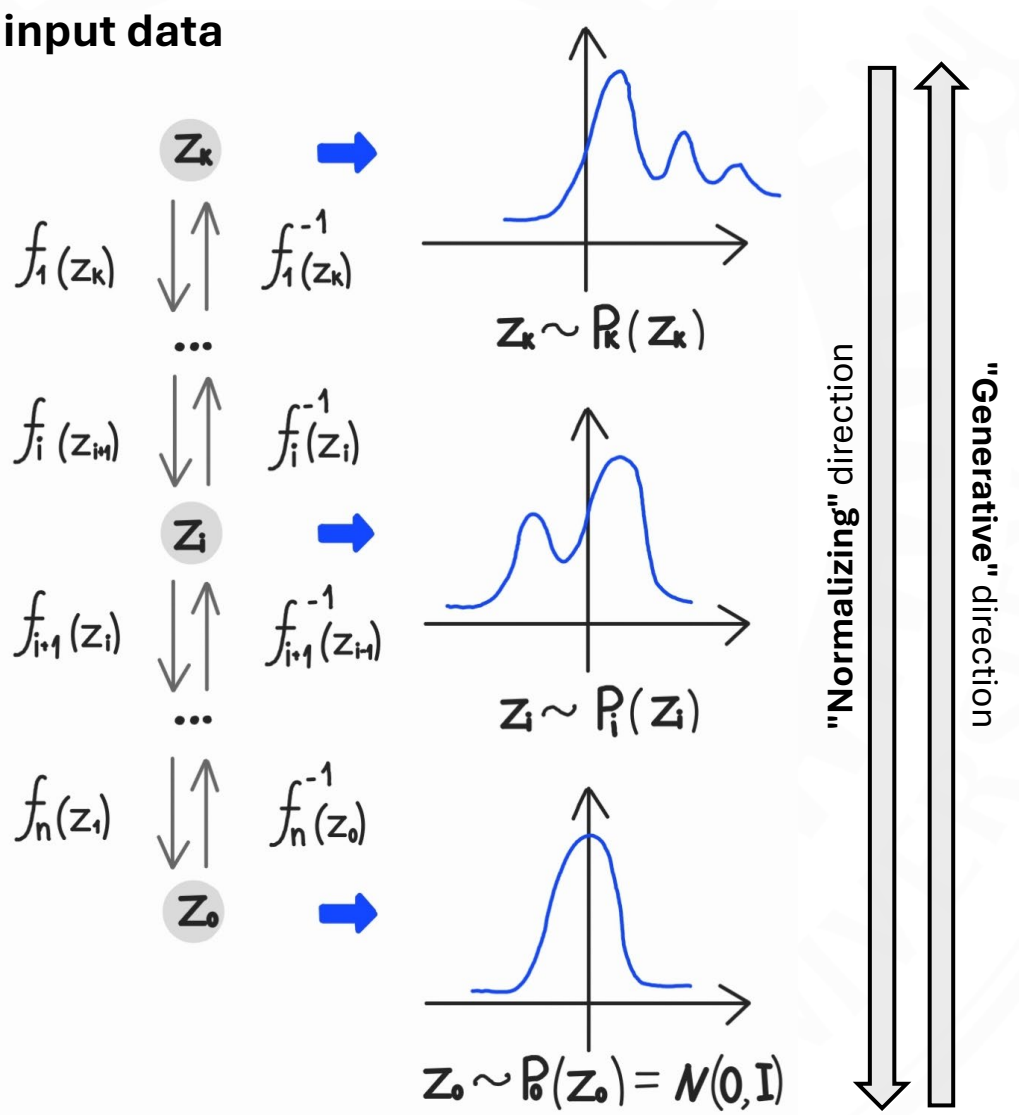
$$\log p(x|y) = \log p(z_0) + \sum_{k=1}^{K-1} \log \frac{df_{k,\theta_k}(z_k)}{dz_k} + \log \frac{df_{K,\theta_K}(x)}{dx}$$

Data

MC parameters + source type

well-known (e.g. standard Gaussian)

Computed based on the learned transformations



[1] R. J. Rezende et al, arxiv: 1505.05770

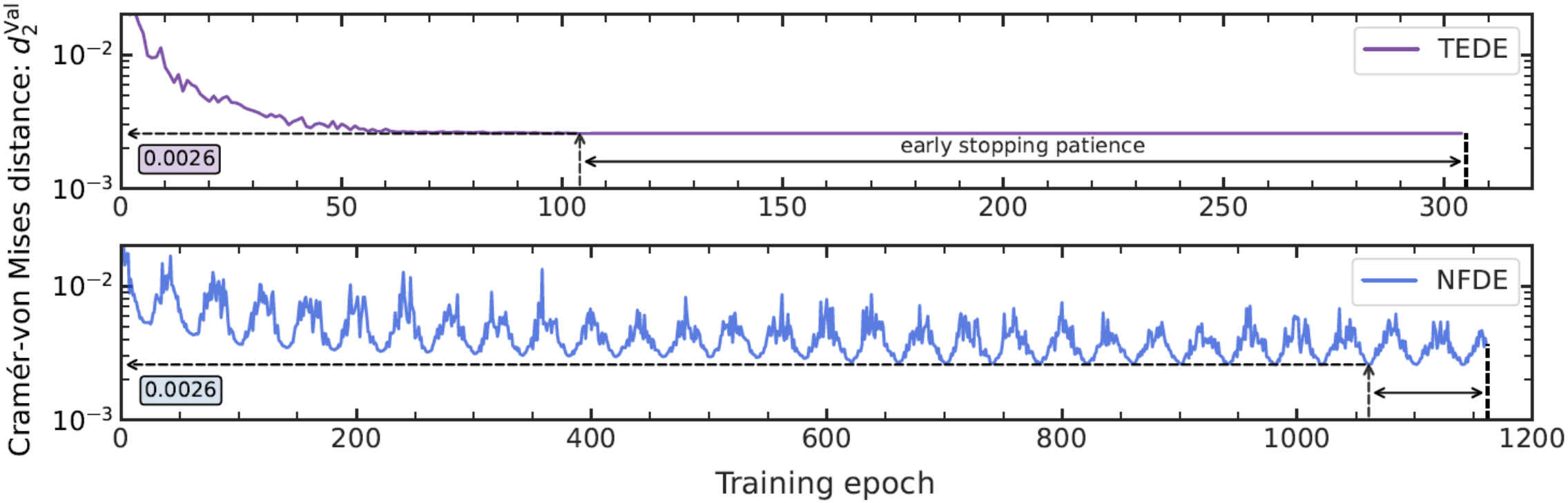
Training and validation of the models

- For both models we use **Kullback–Leibler divergence as the loss function** for training:
- As validation metrics we use p-norm distances between distributions' CDFs:

$$d_p(F_u, F_v) = \left(\int_{-\infty}^{+\infty} |F_u(x) - F_v(x)|^p \, dx \right)^{\frac{1}{p}}$$

as it is applicable for both TEDE and NFDE cases

p = 1: *Wasserstein distance*
p = 2: *Cramer von Mises distance*
p -> ∞: *Kolmogorov-Smirnov distance*

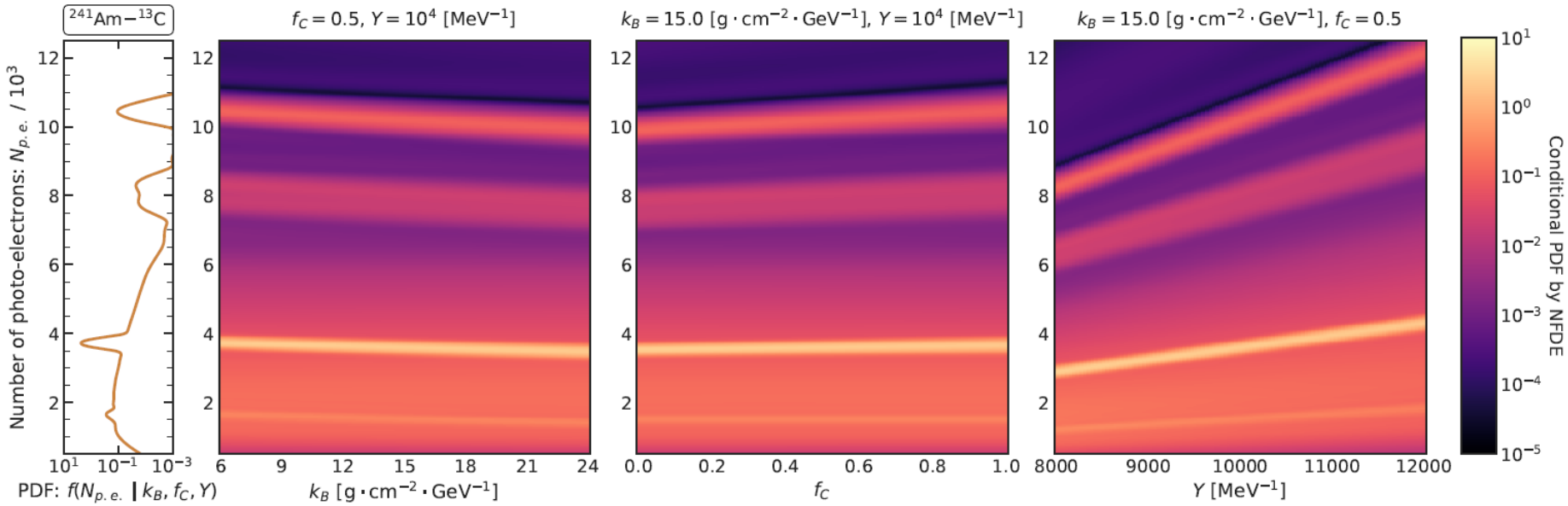


A. Gavrikov, A. Serafini, D. Dolzhikov et al., *Commun Phys* **9**, 63 (2026)

Learned dependences

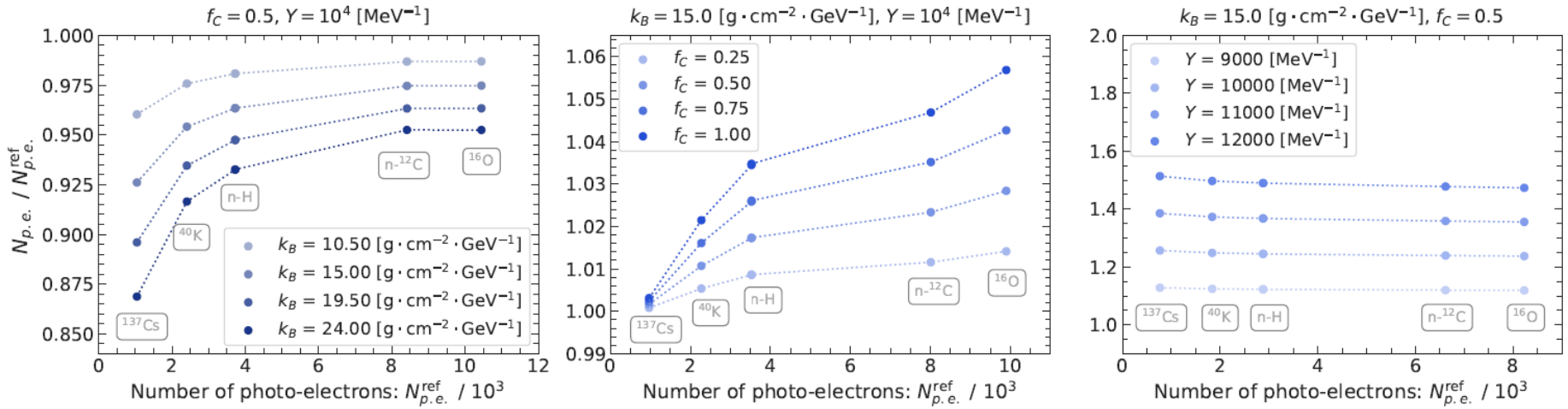
1) PDF vs. parameter

(almost linear)



2) PDFs vs. energy (NPE)

(energy non-linearity)



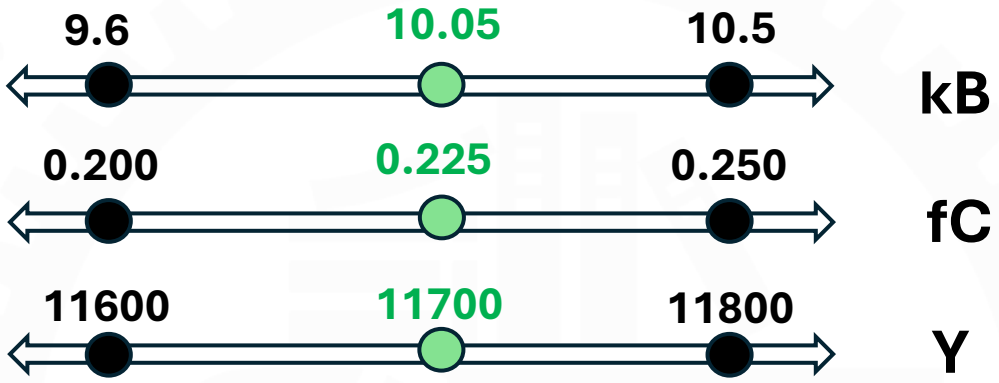
A. Gavrikov, A. Serafini, D. Dolzhikov et al., *Commun Phys* **9**, 63 (2026)

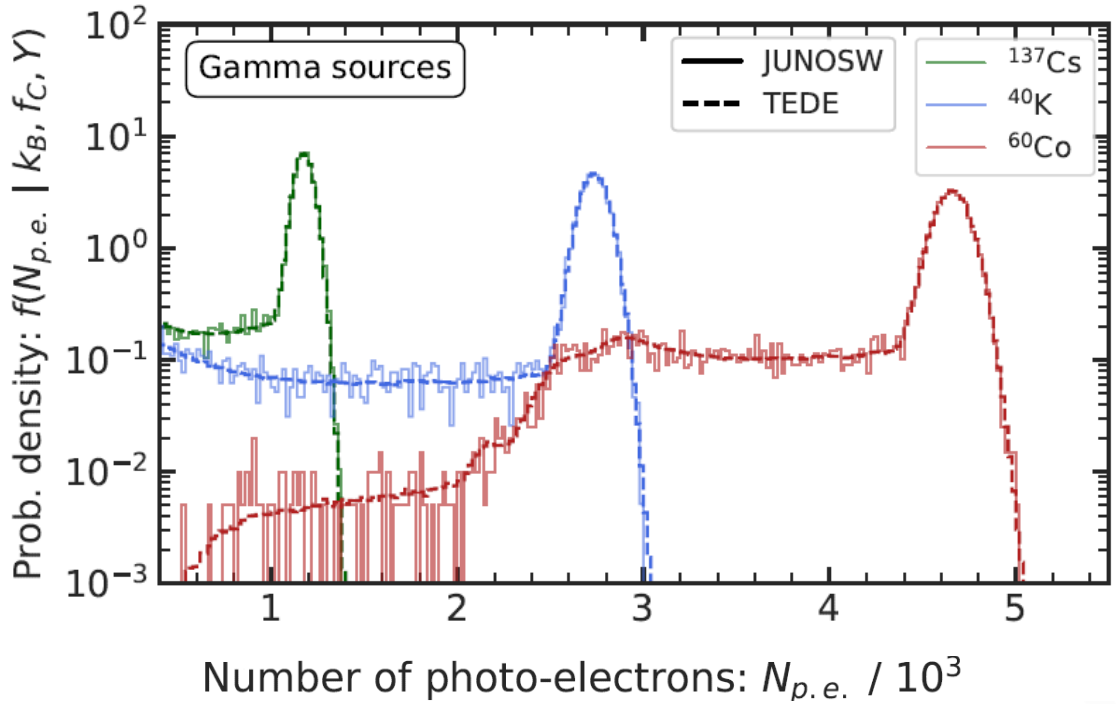
Some results: spectra modelling

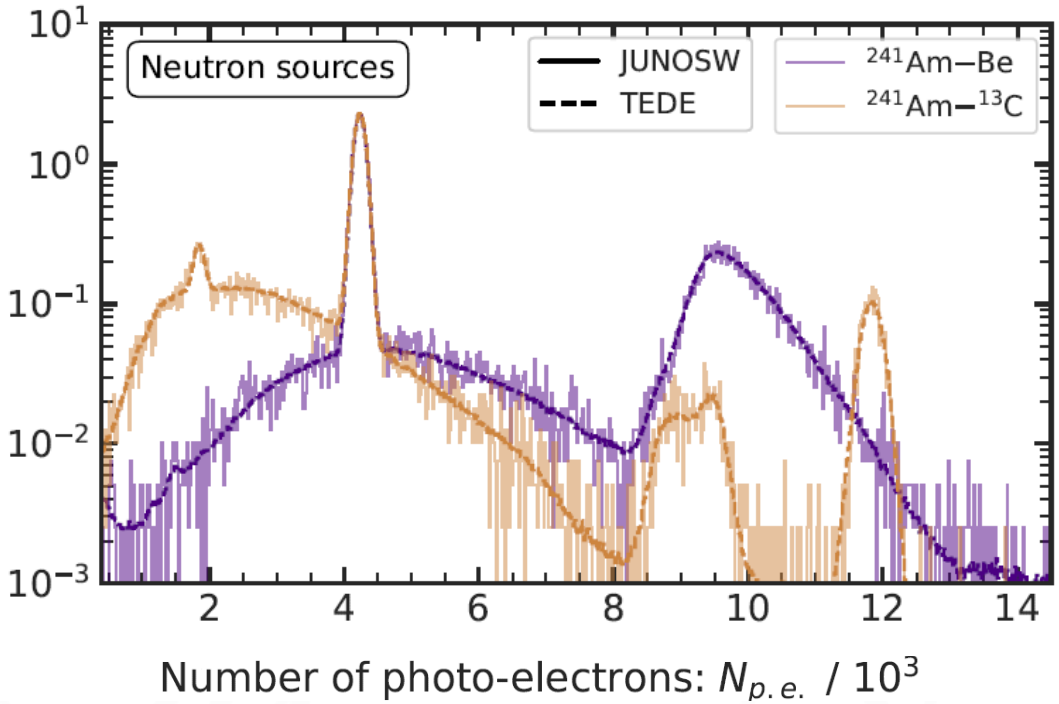
Example with TEDE:

Model provides spectra estimates for ***any continuous*** values of the parameters

Accurate interpolation. Model's outputs are smooth and denoised







A. Gavrikov, A. Serafini, D. Dolzhikov et al., *Commun Phys* **9**, 63 (2026)

Hyperparameter optimization

Hyperparameters	TEDE
Model architecture hyperparameters	
n_{tokens}	$\{k \mid k \in \mathbb{N}, 1 \leq k \leq 5\}$: **1**
d_{model}	$\{50k \mid k \in \mathbb{N}, 1 \leq k \leq 10\}$: **100**
n_{head}	$\{5, 10, 25\}$: **10**
n_{layers}	$\{k \mid k \in \mathbb{N}, 1 \leq k \leq 5\}$: **4**
d_{ff}	$\{32k \mid k \in \mathbb{N}, 1 \leq k \leq 15\}$: **384**
Dropout rate	$\{p_{\text{drop}} \mid 0.0 \leq p_{\text{drop}} \leq 0.5\}$: **0.0**
Activation function [114, 199–201]	`ReLU`, **`GeLU`**
Softmax temperature	$\{T \mid 0.25 \leq T \leq 3.0\}$: **2.03**
General training hyperparameters	
Optimizer [115, 202]	**`AdamW`**, `RMSprop`
Learning rate	$\{\eta \mid 10^{-5} \leq \eta \leq 10^{-2}\}$: $\mathbf{1.9 \times 10^{-3}}$
Scheduler type [203, 204]	**`Exponential`**, `ReduceOnPlateau`, `CosineAnnealing`, `None`
Weight decay	$\{\lambda \mid 10^{-6} \leq \lambda \leq 10^{-1}\}$: $\mathbf{8 \times 10^{-5}}$
Batch size	$\{2^k \mid k \in \mathbb{N}, 4 \leq k \leq 9\}$: **64**
Optimizer-specific hyperparameters	
`AdamW` decay rate β_1	$\{\beta_1 \mid 0.5 \leq \beta_1 \leq 0.95\}$: **0.930**
`AdamW` decay rate β_2	$\{\beta_2 \mid 0.9 \leq \beta_2 \leq 0.999\}$: **0.929**
`RMSprop` parameter α	$\{\alpha \mid 0.9 \leq \alpha \leq 0.999\}$: –
Scheduler-specific hyperparameters	
Cosine annealing period $T_{\max}$	$\{T_{\max} \mid 5 \leq T_{\max} \leq 75\}$: –
Exponential rate τ	$\{\tau \mid 0.8 \leq \tau \leq 0.99\}$: **0.92**
Reduce on plateau factor γ	$\{\gamma \mid 0.8 \leq \gamma \leq 0.99\}$: –
Optimization procedure setting	
Number of trials	**250**
Accelerator	GPU

Hyperparameters	NFDE
Model architecture hyperparameters	
n_{flows}	$\{100 + 5k \mid k \in \mathbb{Z}, 0 \leq k \leq 20\}$: **120**
n_{units}	$\{10 + 5k \mid k \in \mathbb{Z}, 0 \leq k \leq 8\}$: **30**
Flow type	**`Planar`** (fixed)
Activation function [114, 199–201]	`ReLU`, **`GeLU`**, `Tanh`, `SiLU`
General training hyperparameters	
Optimizer [115, 202]	**`AdamW`** (fixed)
Learning rate	$\mathbf{10^{-4}}$ (fixed)
Scheduler type [203, 204]	`ReduceOnPlateau`, **`CosineAnnealing`**
Weight decay	$\mathbf{10^{-4}}$ (fixed)
Batch size	**50** (fixed, 1 per CPU unit)
Optimizer-specific hyperparameters	
`AdamW` decay rate β_1	$\{\beta_1 \mid 0.5 \leq \beta_1 \leq 0.95\}$: **0.87**
`AdamW` decay rate β_2	$\{\beta_2 \mid 0.9 \leq \beta_2 \leq 0.999\}$: **0.901**
Scheduler-specific hyperparameters	
Cosine annealing period $T_{\max}$	$\{T_{\max} \mid 5 \leq T_{\max} \leq 75\}$: **50**
Reduce on plateau factor γ	$\{\gamma \mid 0.8 \leq \gamma \leq 0.99\}$: –
Optimization procedure setting	
Number of trials	**25**
Accelerator	50 parallel CPUs

A. Gavrikov, A. Serafini, D. Dolzhikov et al., *Commun Phys* **9**, 63 (2026)