SIXSA

Simulation-based Inference for X-ray Spectral Analysis - latest developments

Didier Barret (IRAP, Toulouse, France)

Simon Dupourqué (IRAP, Toulouse, France)

Lumière workshop - January 2026

Motivations

- Investigate Simulation Based Inference as a way to accurately fit complex X-ray spectra without requiring to too extensive computational resources

- As currently shown with XRISM-Resolve data, fitting spectra is rather challenging : takes ages and full Bayesian analysis is often not considered

- We need to prepare for data of even higher complexity and richness as expected from X-IFU (and data below 2 keV)

SIXSA

- In Paper 1 : Demonstration of working principles with CCD-like resolution— introduction of the PCA to reduce the dimensions of the spectra. Worked well in Gaussian and Poisson regimes (BD A&A 2024)

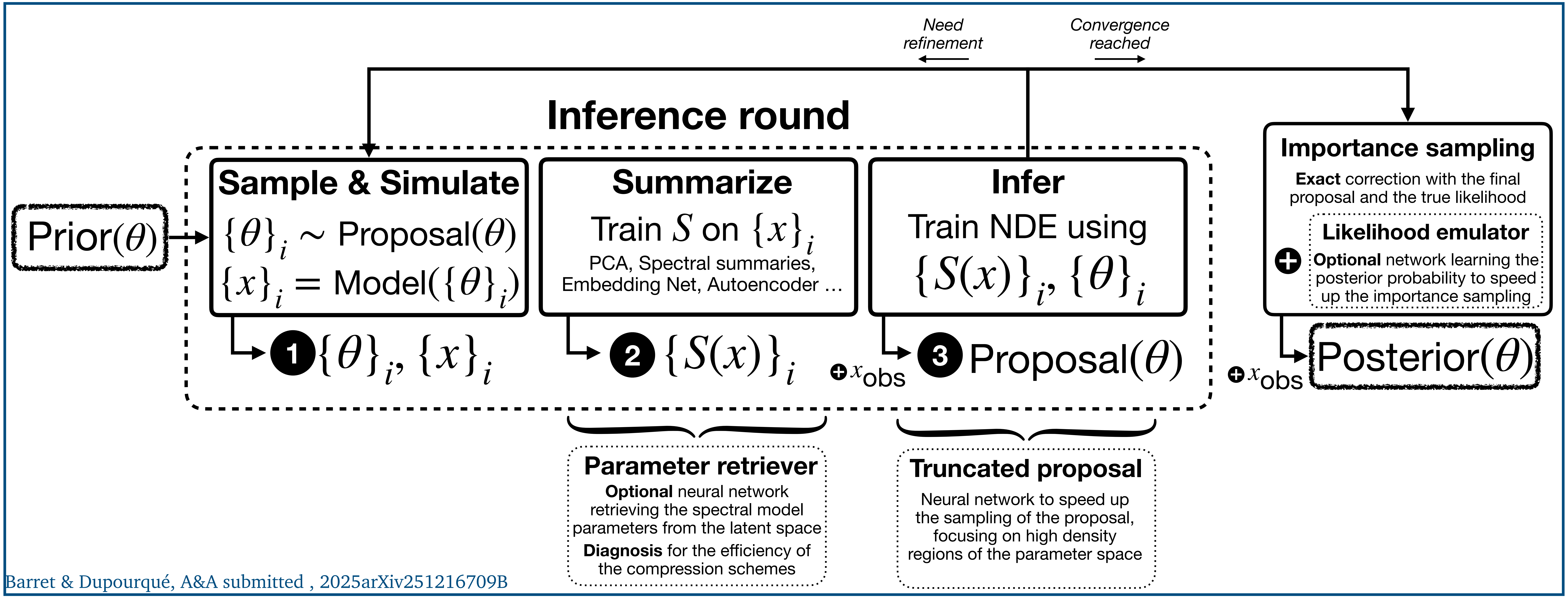
- In Paper 2: First application to X-IFU mock spectra with the introduction of summary statistics to reduce the dimension of the spectra (DB, A&A 2025)

- In Paper 3: Usage of an auto encoder to compress X-IFU like spectra and importance sampling for correcting the approximate posteriors with the known likelihood *— scope of my presentation (Barret & Dupourqué, A&A submitted , 2025arXiv251216709B)*

SIXSA : what it does and doesn’t (yet)

- Built upon the **sbi** package (Boelts 2025)
- Relies on PyXspec to enable the use of the suite of spectral models available in XSPEC
  - For X-IFU like spectra and complex models, simulations are the bottle neck (so is the evaluation of the likelihoods)
  - 10000s of simulations required : NICER/XMM-Newton up to 6000 sim./seconds — Resolve/X-IFU down to 50 sim./seconds
- Developed and tested on a Multi-core Mac-Power book (M3, 16 cores, 128G)
- Use of XSPEC MCMC (Arnaud 1996), BXA (Buchner+) posteriors (obtained through parallelization) and Jaxspec (Dupourqué+)
- Available on GitHub
- Limited to one spectrum at a time for non amortized inference
  - Joint fitting of multiple spectra with tied model parameters not yet implemented
  - Considering background is not an issue if a background spectrum exists (to be added to the source simulated spectrum) or a background model exists (to be folded simultaneously to the source model)

SIXSA pipeline — Multi-round inference



Need refinement

Convergence reached

Inference round

Prior(θ)

Sample & Simulate

$\{\theta\}_i \sim$ Proposal(θ)

$\{x\}_i =$ Model($\{\theta\}_i$)

❶ $\{\theta\}_i, \{x\}_i$

Summarize

Train S on $\{x\}_i$

PCA, Spectral summaries, Embedding Net, Autoencoder …

❷ $\{S(x)\}_i$

⊕ x_{obs}

Infer

Train NDE using $\{S(x)\}_i, \{\theta\}_i$

❸ Proposal(θ)

Importance sampling

Exact correction with the final proposal and the true likelihood

⊕ **Likelihood emulator**

Optional network learning the posterior probability to speed up the importance sampling

⊕ x_{obs}

Posterior(θ)

Parameter retriever

Optional neural network retrieving the spectral model parameters from the latent space

Diagnosis for the efficiency of the compression schemes

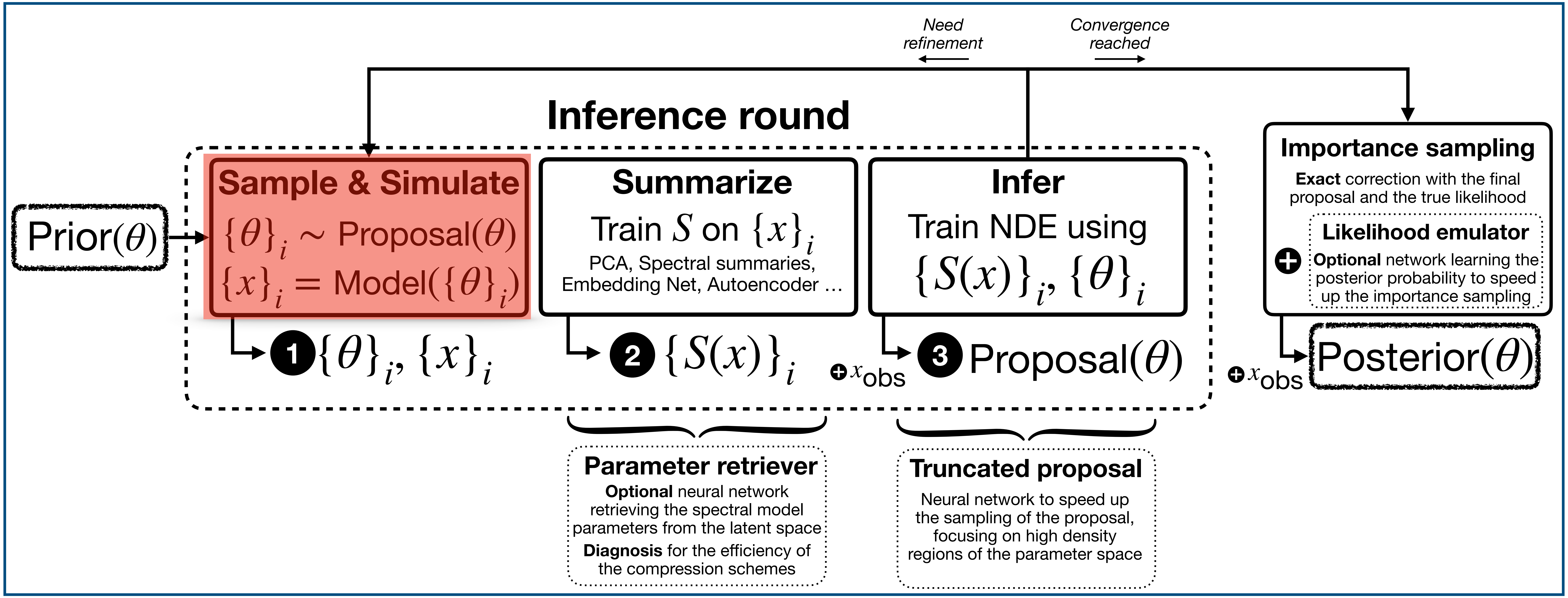
Truncated proposal

Neural network to speed up the sampling of the proposal, focusing on high density regions of the parameter space

Barret & Dupourqué, A&A submitted , 2025arXiv251216709B

- At each step, sanity checks are performed

SIXSA pipeline — Multi-round inference

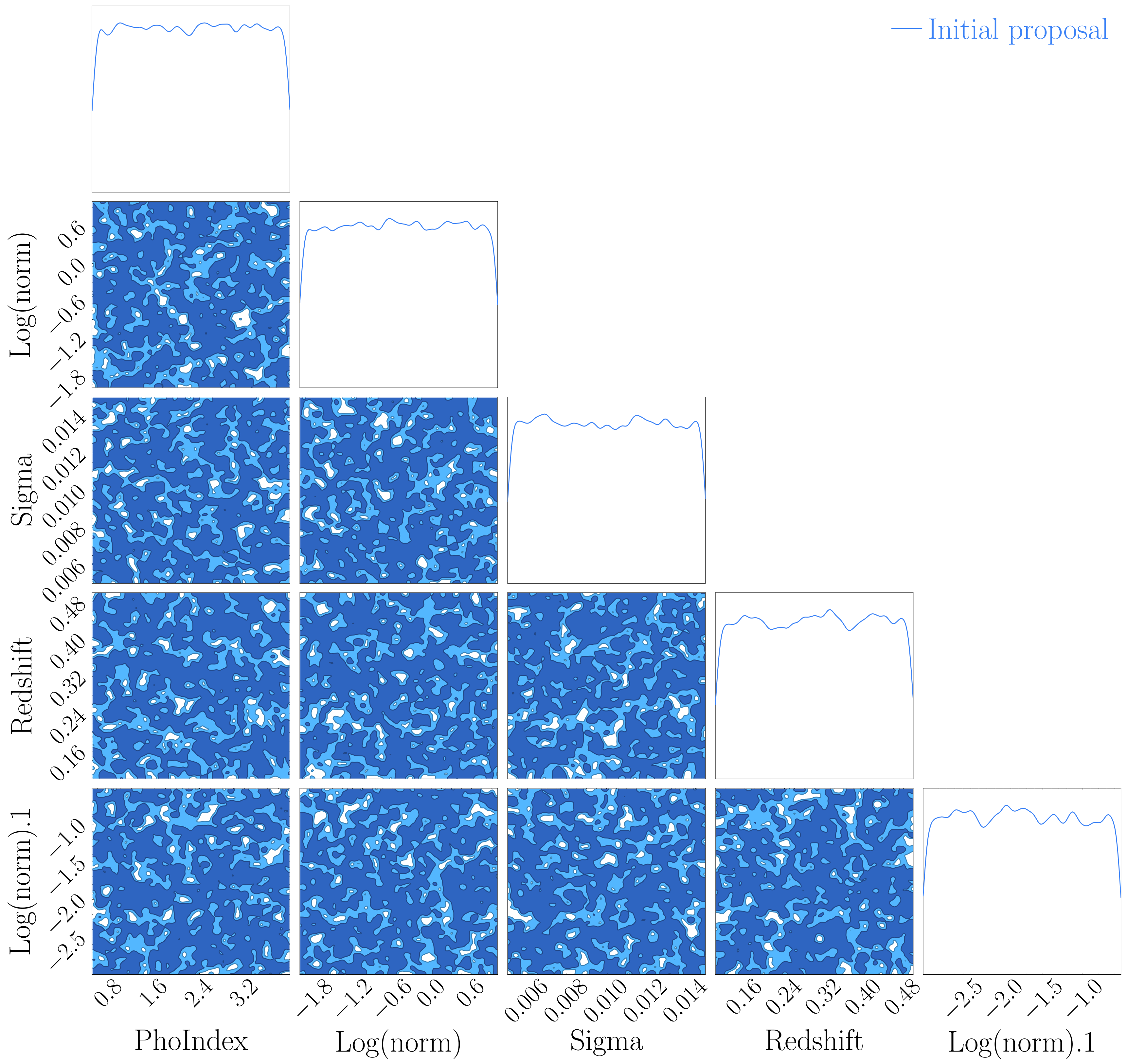


Need refinement
Convergence reached
Inference round
Prior(θ)
Sample & Simulate
$\{\theta\}_i \sim$ Proposal(θ)
$\{x\}_i =$ Model($\{\theta\}_i$)
Summarize
Train S on $\{x\}_i$
PCA, Spectral summaries, Embedding Net, Autoencoder …
Infer
Train NDE using
$\{S(x)\}_i, \{\theta\}_i$
Importance sampling
Exact correction with the final proposal and the true likelihood
Likelihood emulator
Optional network learning the posterior probability to speed up the importance sampling
1 $\{\theta\}_i, \{x\}_i$
2 $\{S(x)\}_i$
x_{obs}
3 Proposal(θ)
x_{obs}
Posterior(θ)
Parameter retriever
Optional neural network retrieving the spectral model parameters from the latent space
Diagnosis for the efficiency of the compression schemes
Truncated proposal
Neural network to speed up the sampling of the proposal, focusing on high density regions of the parameter space

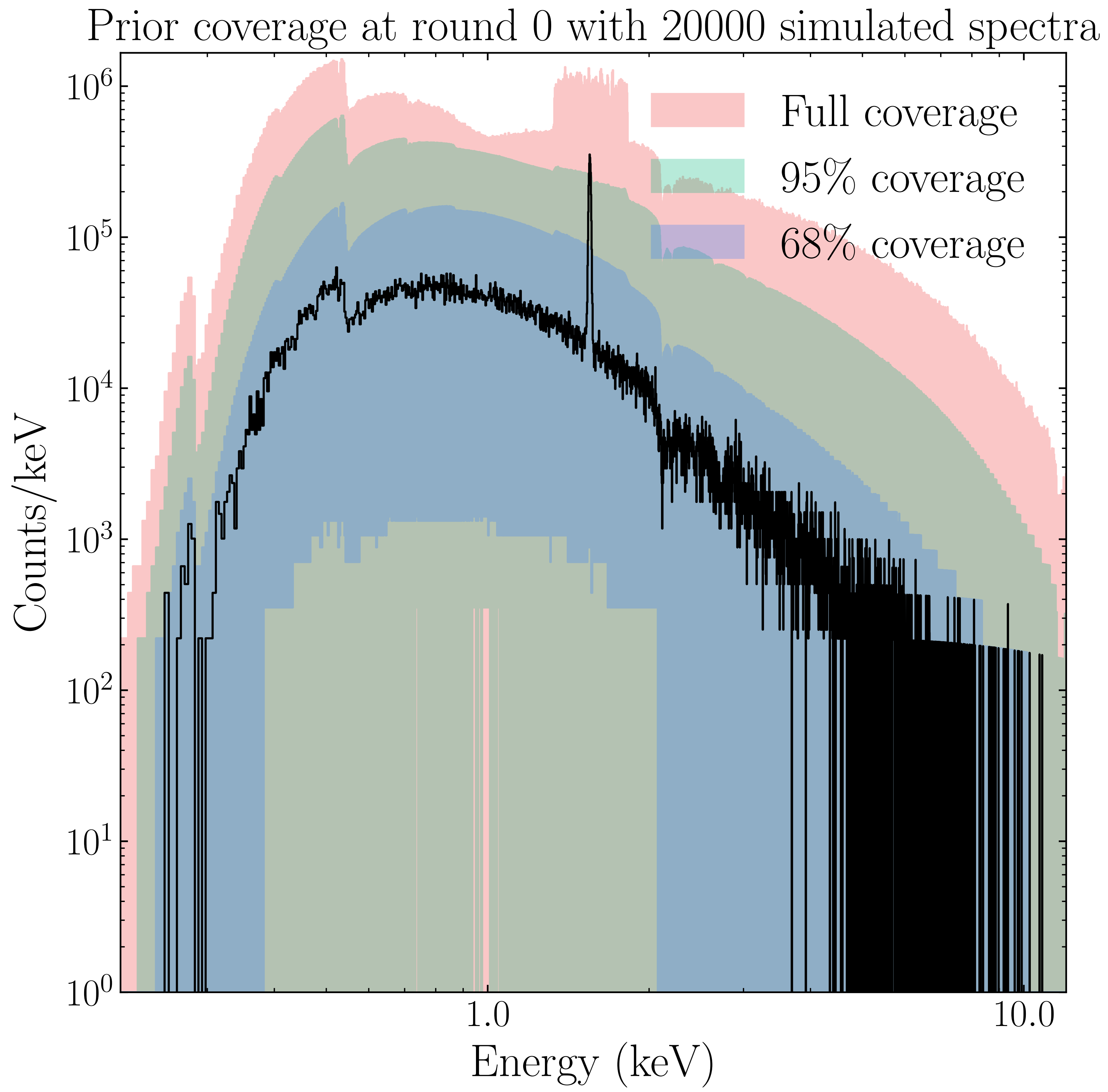
- At each step, sanity checks are performed

Sample and simulate

- A power law with a narrow line of fixed energy with unknown redshift and width (Poisson statistics added)



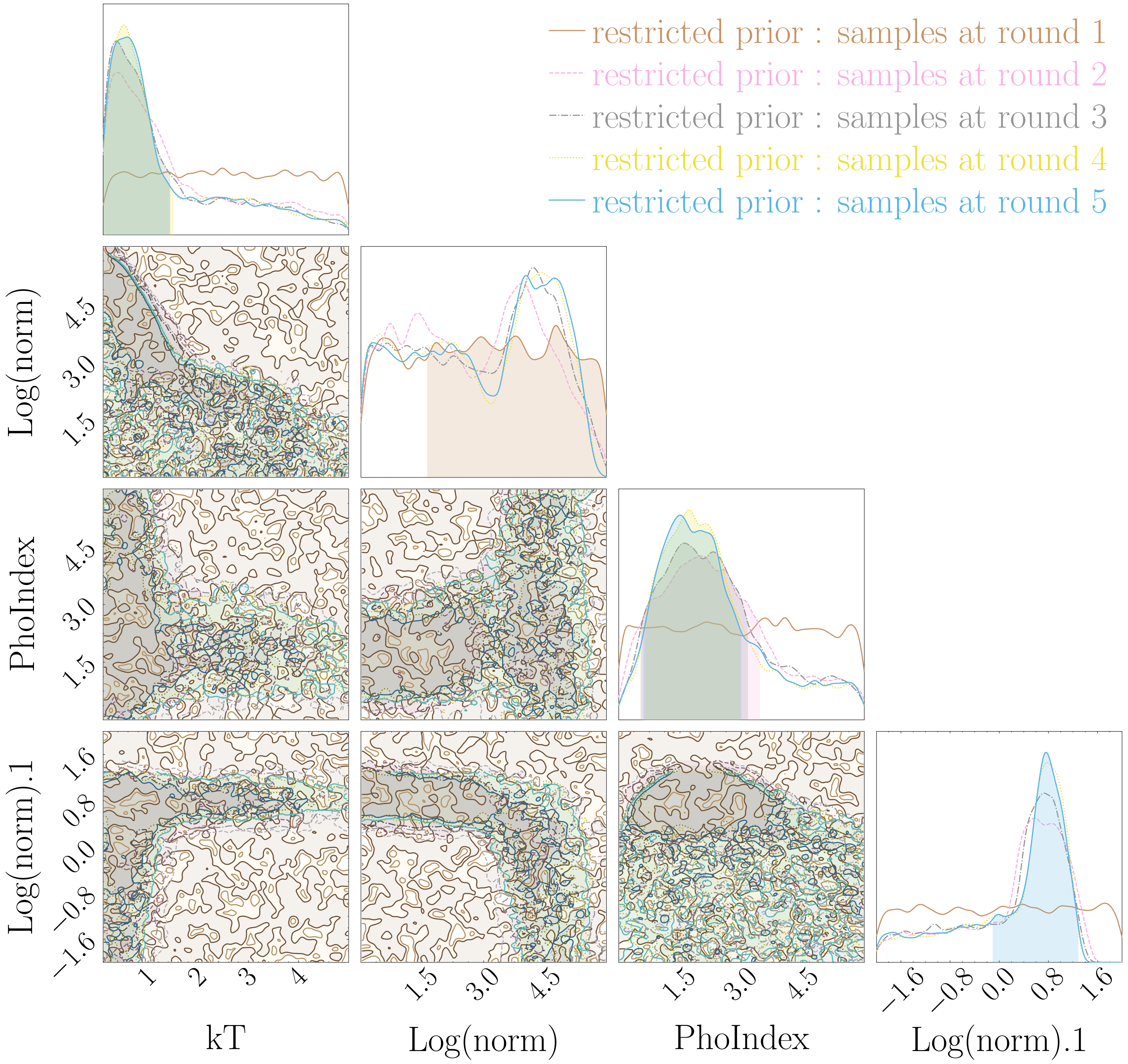
Proposal before first round of inference



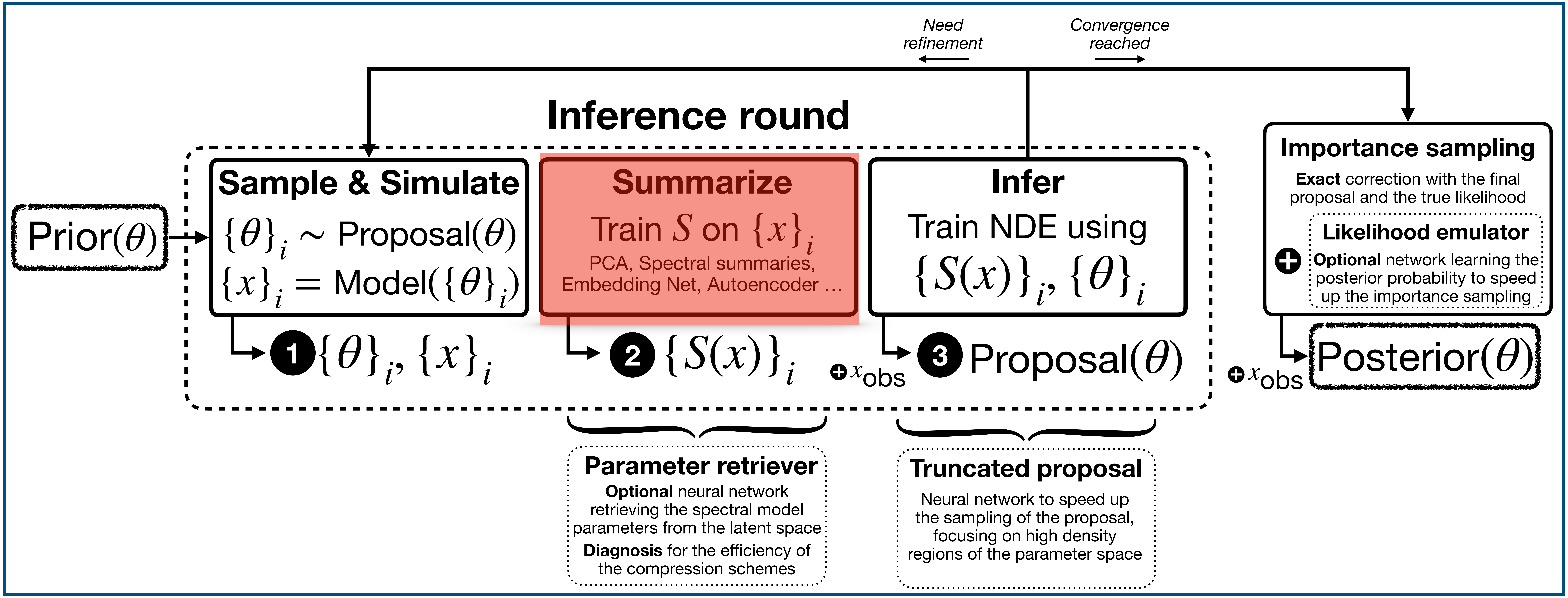
Prior coverage before the first round of inference

Restricting the priors through a classifier

- Train a classifier to restrict the parameter space, by keeping parameter samples associated with the lowest c-stat (a given fraction say 5-10% of the samples)
- Very fast — does not need large training samples
- The **sbi** package provides such a classifier (MLP, Resnet)
- Design your own with you own classification criteria



SIXSA pipeline — Multi-round inference



Need refinement

Convergence reached

Inference round

Prior(θ)

Sample & Simulate

$\{\theta\}_i \sim$ Proposal(θ)

$\{x\}_i =$ Model($\{\theta\}_i$)

Summarize

Train S on $\{x\}_i$

PCA, Spectral summaries, Embedding Net, Autoencoder …

Infer

Train NDE using

$\{S(x)\}_i, \{\theta\}_i$

1 $\{\theta\}_i, \{x\}_i$

2 $\{S(x)\}_i$

$+ x_{\text{obs}}$

3 Proposal(θ)

Importance sampling

Exact correction with the final proposal and the true likelihood

Likelihood emulator

Optional network learning the posterior probability to speed up the importance sampling

$+ x_{\text{obs}}$

Posterior(θ)

Parameter retriever

Optional neural network retrieving the spectral model parameters from the latent space

Diagnosis for the efficiency of the compression schemes

Truncated proposal

Neural network to speed up the sampling of the proposal, focusing on high density regions of the parameter space

- At each step, sanity checks can be performed

Why summarizing spectra?

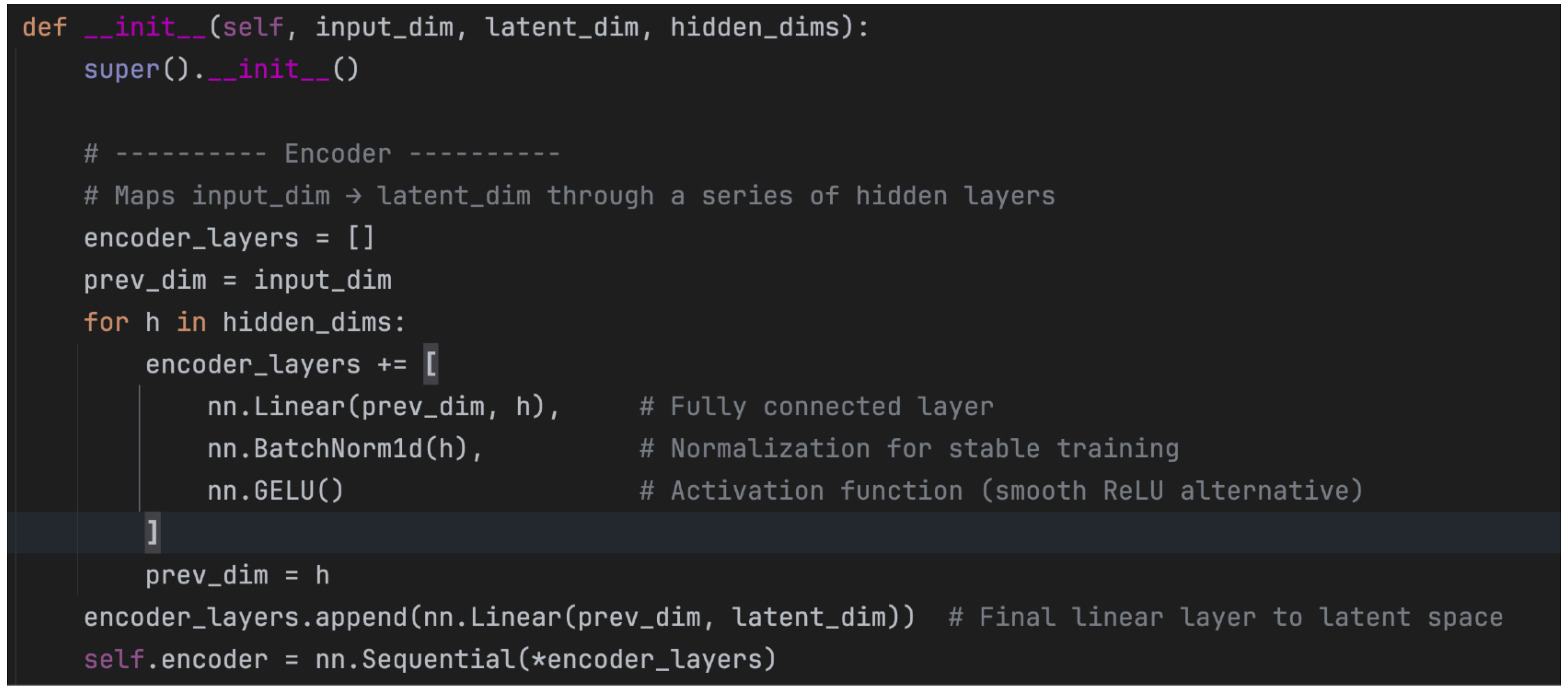
- Reduce the information to ease the neural density estimator learning of the likelihood and prevent overfitting — limit is around hundreds of channels hence not applicable to Resolve/X-IFU spectra
- Various reduction techniques:
  - Principal Component Analysis — not efficient when the spectra are similar
  - Summary statistics — feature engineering, best if physics informed — global ones such as harness ratios good for featureless spectra
  - Auto-encoders

Autoencoders to summarize spectra

- An autoencoder is a type of artificial neural network used primarily for unsupervised learning, especially in tasks like data compression, denoising, or dimensionality reduction.
- Autoencoders have a symmetrical architecture, usually composed of three main parts:
  - 1. Encoder
    - ➡ Transforms the input data into a compressed (lower-dimensional) representation, called the latent space or bottleneck.
    - ➡ Learns to capture the most important features of the input.
  - 2. Latent Space / Bottleneck
    - ➡ The compressed version of the input.
    - ➡ Contains the essential information the model thinks is necessary to reconstruct the input.
  - 3. Decoder
    - ➡ Reconstructs the original input from the latent representation.
    - ➡ Tries to make the output as close as possible to the original input.
- The network is trained to minimize the difference between the input and the output, usually using a loss function like the Mean Squared Error (MSE).

Autoencoders to summarize spectra

- Structure of the encoder:
  - A sequence of fully connected (nn.Linear) layers.
  - Each hidden layer is followed by:
    - ➡ Batch Normalization (nn.BatchNorm1d) to stabilize training.
    - ➡ GELU activation (nn.GELU) — a smooth, non-linear activation
  - The final encoder layer maps the last hidden layer to the latent space (latent_dim).
- The decoder mirrors the structure of the encoder
- Instead of using the MSE for the loss function, we consider the c-stat to account for the Poisson statistics of the simulated spectra
- Each spectrum (2-3000 bins) is reduced to an array of 64 elements (64 arbitrarily chosen)

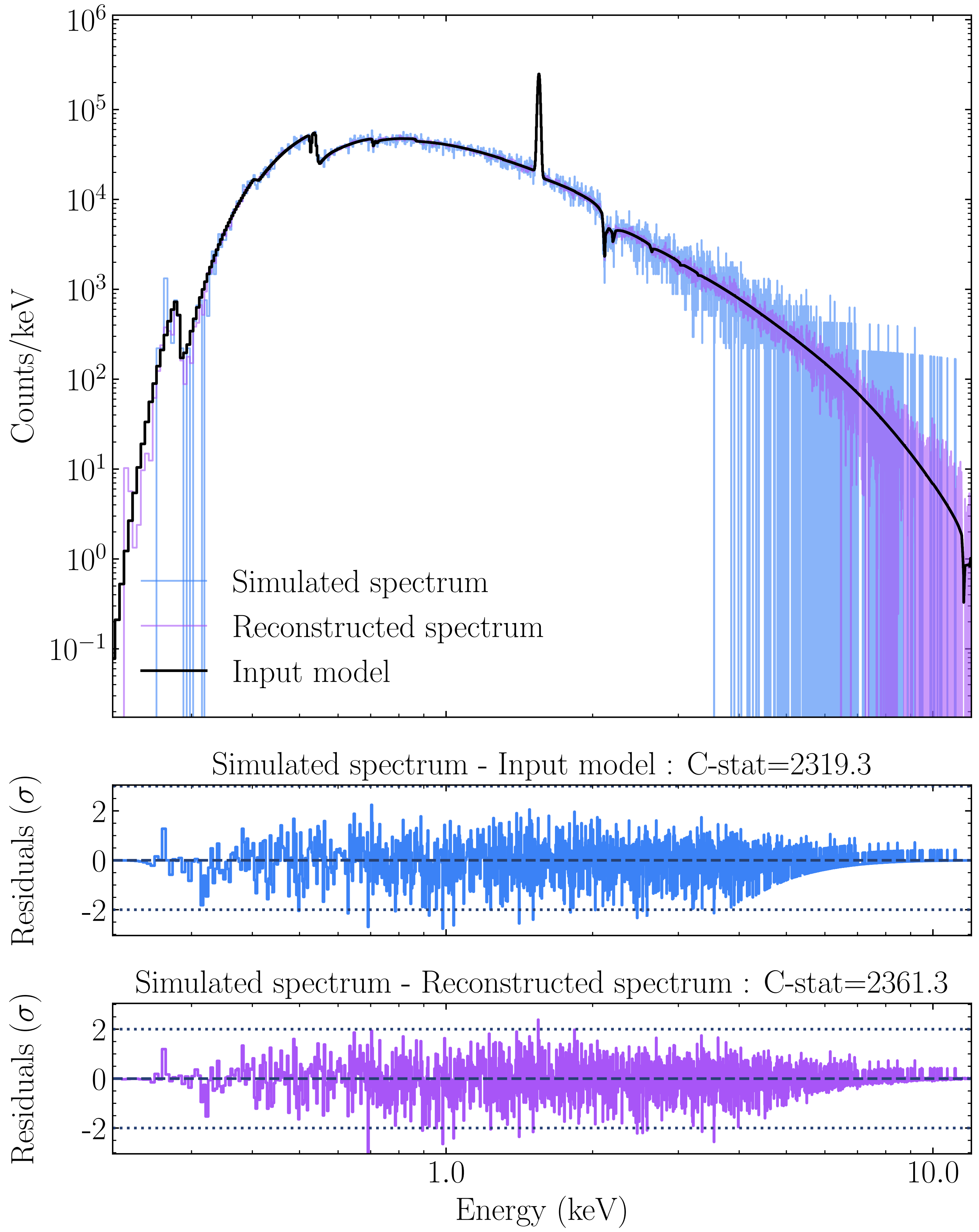


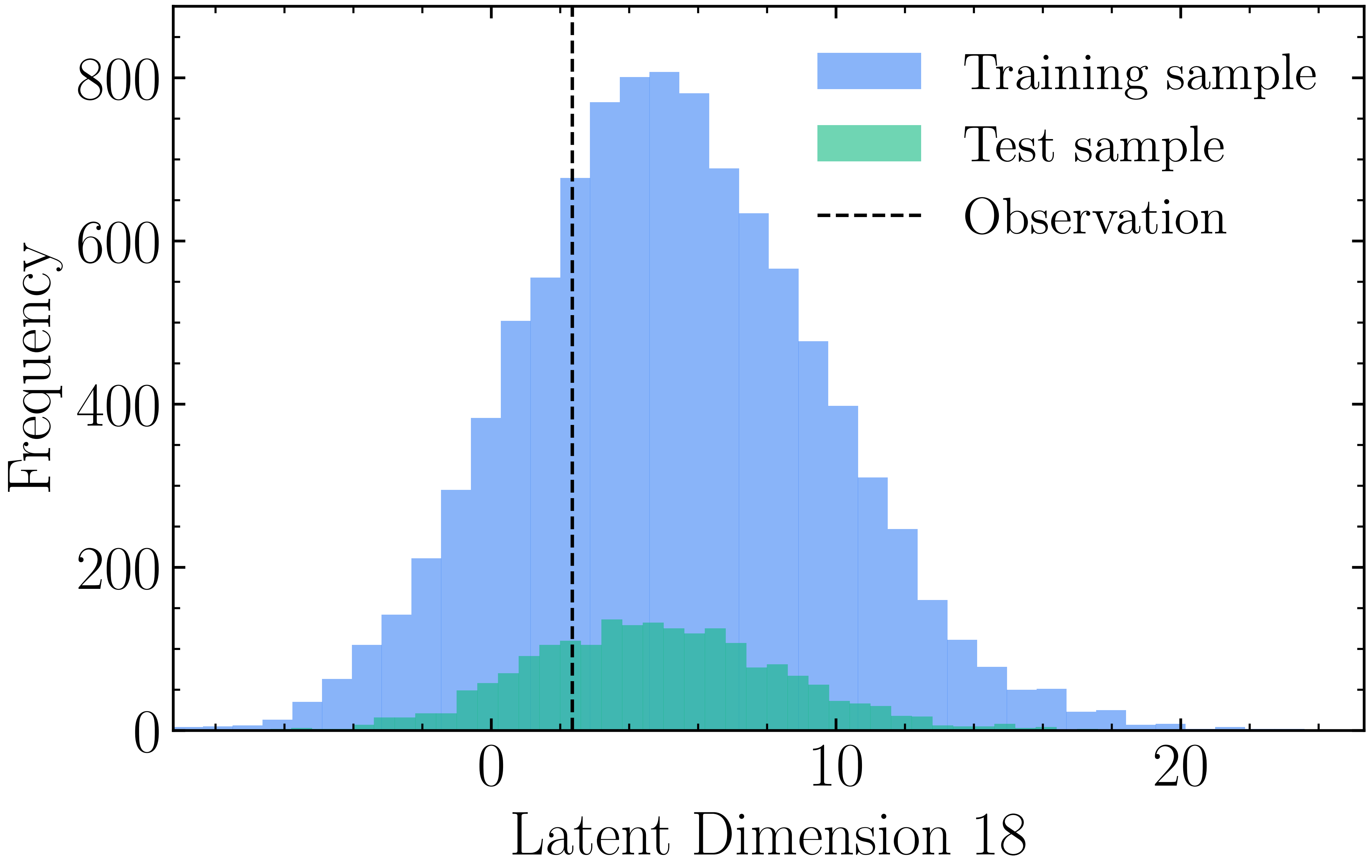
```python
def __init__(self, input_dim, latent_dim, hidden_dims):
    super().__init__()

    # ---------- Encoder ----------
    # Maps input_dim → latent_dim through a series of hidden layers
    encoder_layers = []
    prev_dim = input_dim
    for h in hidden_dims:
        encoder_layers += [
            nn.Linear(prev_dim, h),     # Fully connected layer
            nn.BatchNorm1d(h),          # Normalization for stable training
            nn.GELU()                   # Activation function (smooth ReLU alternative)
        ]
        prev_dim = h
    encoder_layers.append(nn.Linear(prev_dim, latent_dim))  # Final linear layer to latent space
    self.encoder = nn.Sequential(*encoder_layers)
```

Python code of the encoder (torch)

Autoencoders to summarize spectra

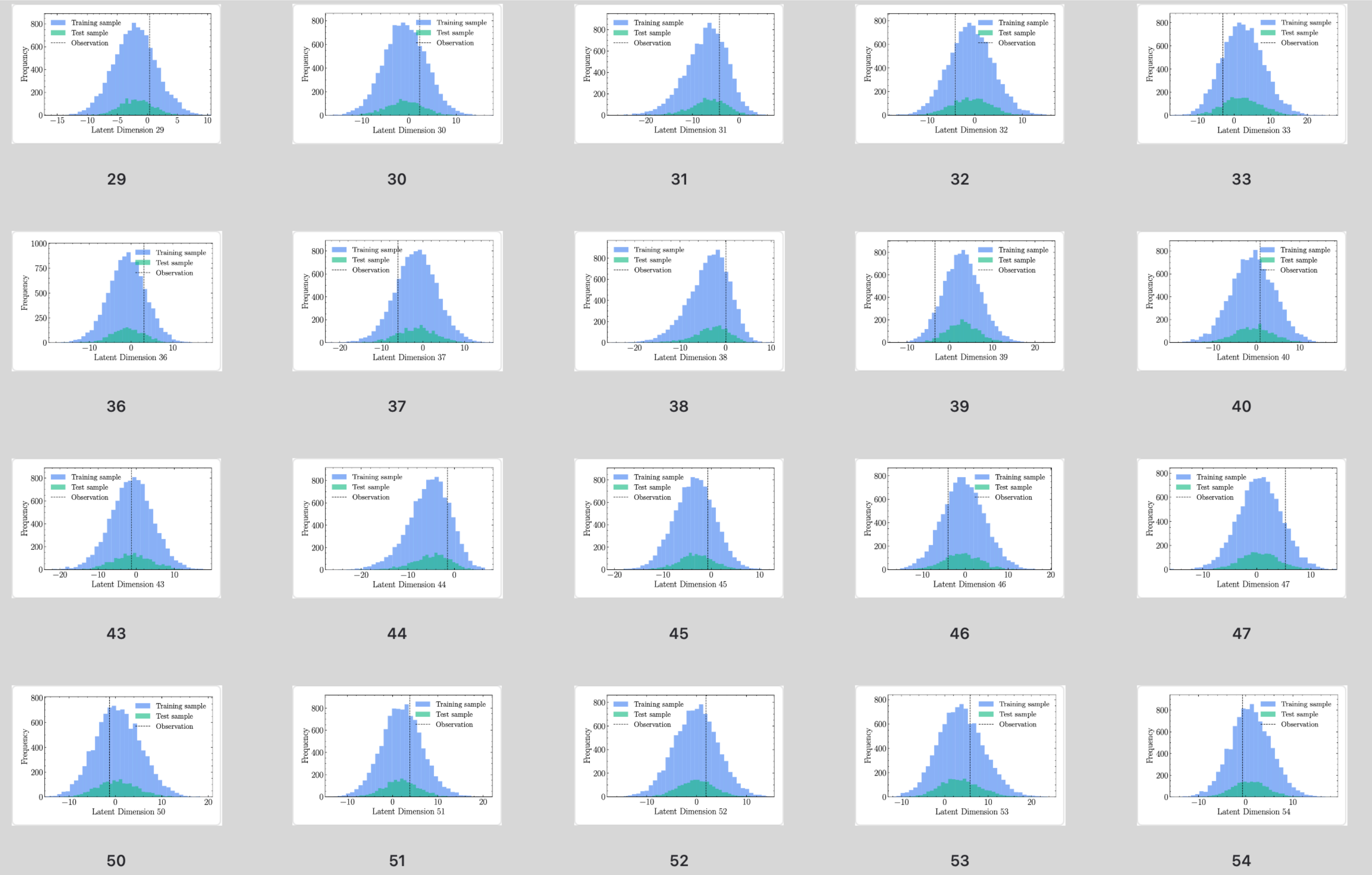




Simulated spectrum versus input model and reconstructed spectrum

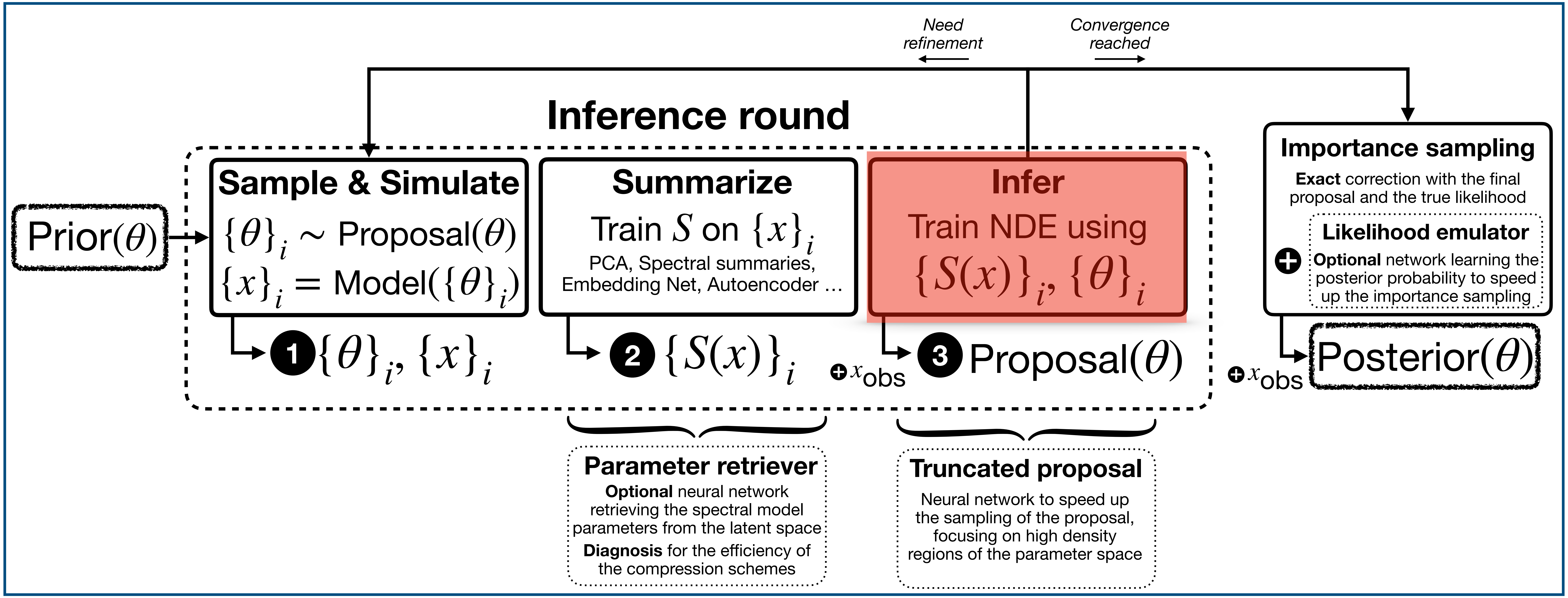
Histogram of the 18th latent space dimension

Autoencoders to summarize spectra



Latent space (more dimensions)

SIXSA pipeline — Multi-round inference

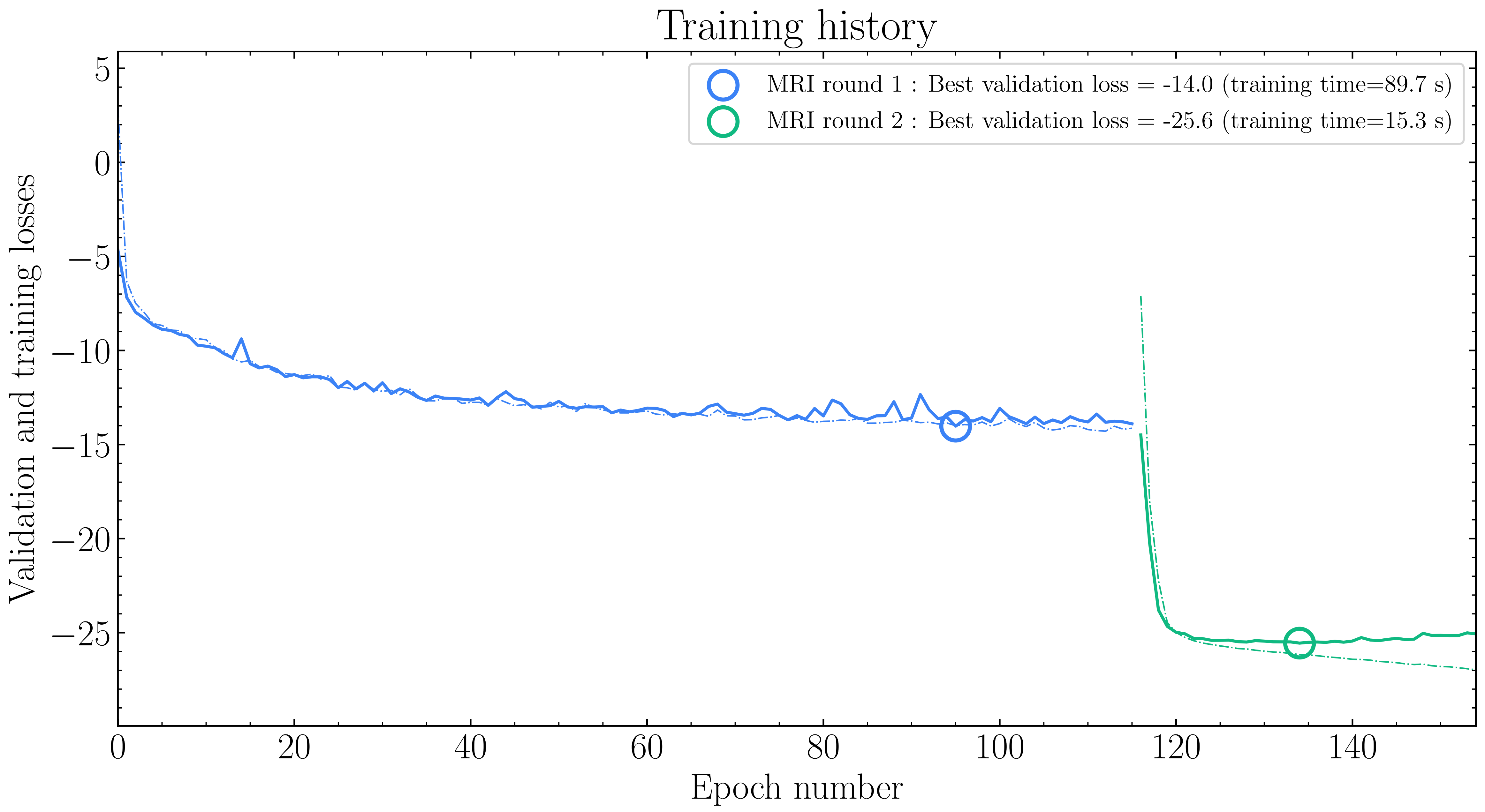


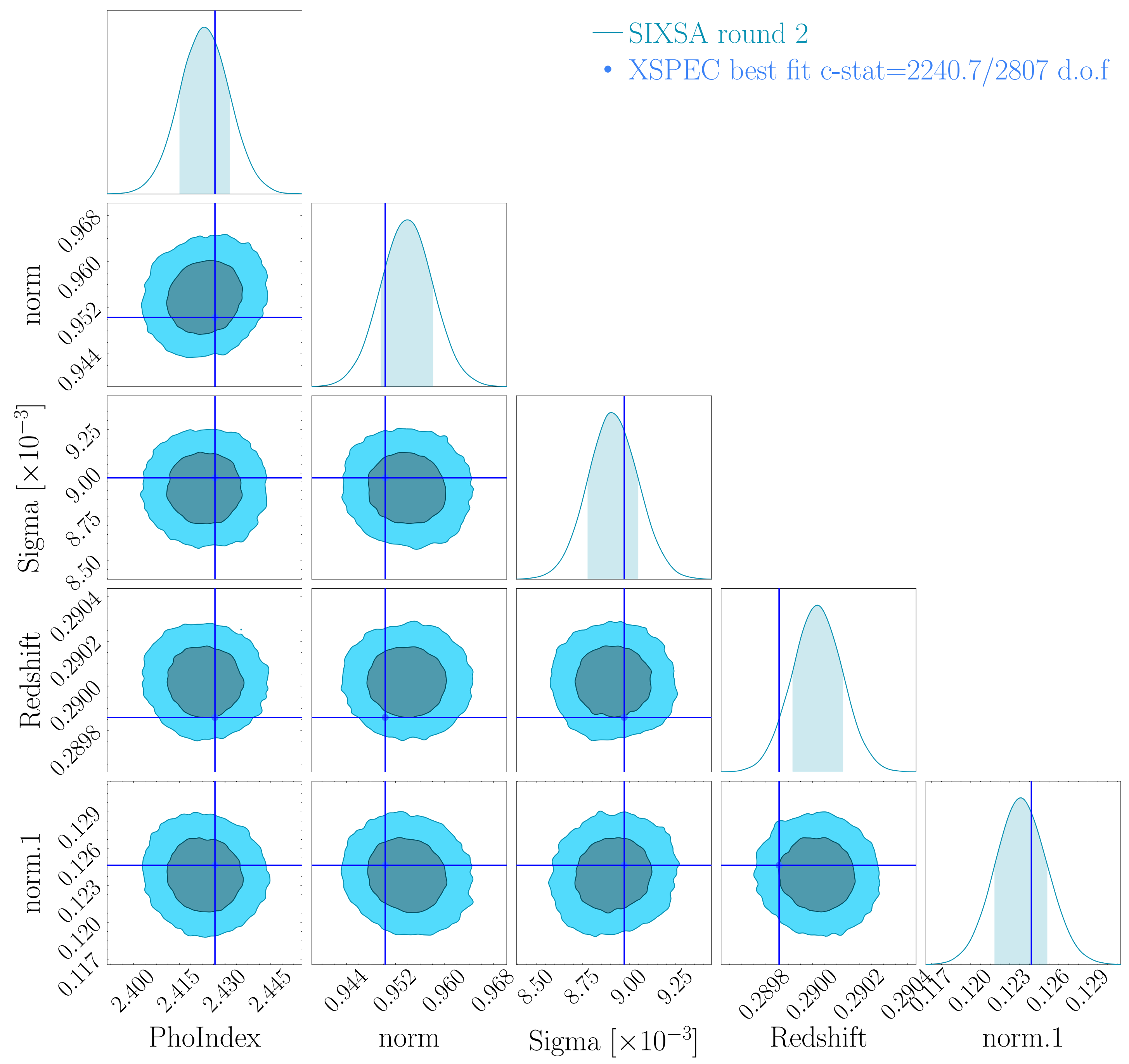
Need refinement
Convergence reached
Inference round
Prior(θ)
Sample & Simulate
$\{\theta\}_i \sim$ Proposal(θ)
$\{x\}_i =$ Model($\{\theta\}_i$)
Summarize
Train S on $\{x\}_i$
PCA, Spectral summaries, Embedding Net, Autoencoder …
Infer
Train NDE using
$\{S(x)\}_i, \{\theta\}_i$
1 $\{\theta\}_i, \{x\}_i$
2 $\{S(x)\}_i$
+ x_{obs}
3 Proposal(θ)
Importance sampling
Exact correction with the final proposal and the true likelihood
Likelihood emulator
+
Optional network learning the posterior probability to speed up the importance sampling
+ x_{obs}
Posterior(θ)
Parameter retriever
Optional neural network retrieving the spectral model parameters from the latent space
Diagnosis for the efficiency of the compression schemes
Truncated proposal
Neural network to speed up the sampling of the proposal, focusing on high density regions of the parameter space

- At each step, sanity checks are performed

Infer

- A neural network is then trained to learn the mapping between the model parameters and the compressed representations of the spectra as to approximate the likelihood

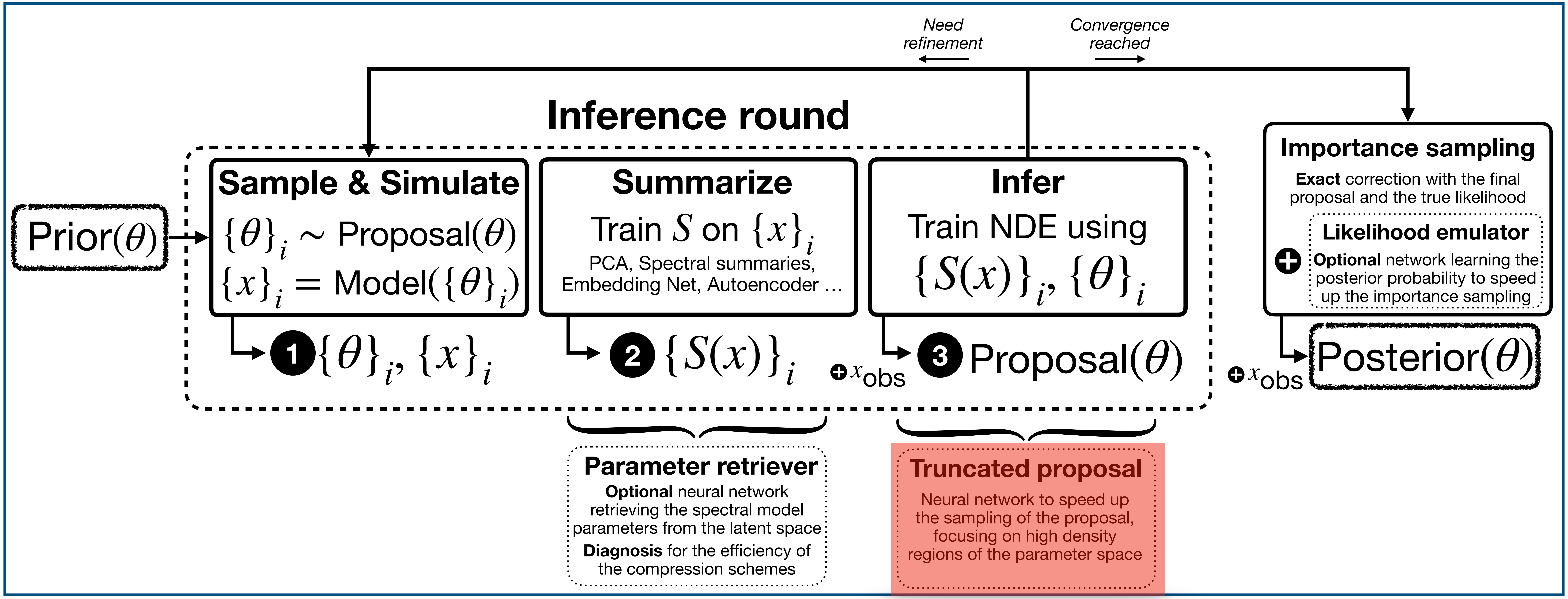




Validation and training losses

Posteriors at the second round of inference

SIXSA pipeline — Multi-round inference



Need refinement ←

Convergence reached →

Inference round

Prior(θ)

Sample & Simulate

$\{\theta\}_i \sim$ Proposal(θ)

$\{x\}_i =$ Model($\{\theta\}_i$)

❶ $\{\theta\}_i, \{x\}_i$

Summarize

Train S on $\{x\}_i$

PCA, Spectral summaries, Embedding Net, Autoencoder …

❷ $\{S(x)\}_i$

Infer

Train NDE using

$\{S(x)\}_i, \{\theta\}_i$

+ x_{obs} ❸ Proposal(θ)

Importance sampling

Exact correction with the final proposal and the true likelihood

+ **Likelihood emulator**

Optional network learning the posterior probability to speed up the importance sampling

+ x_{obs} Posterior(θ)

Parameter retriever

Optional neural network retrieving the spectral model parameters from the latent space

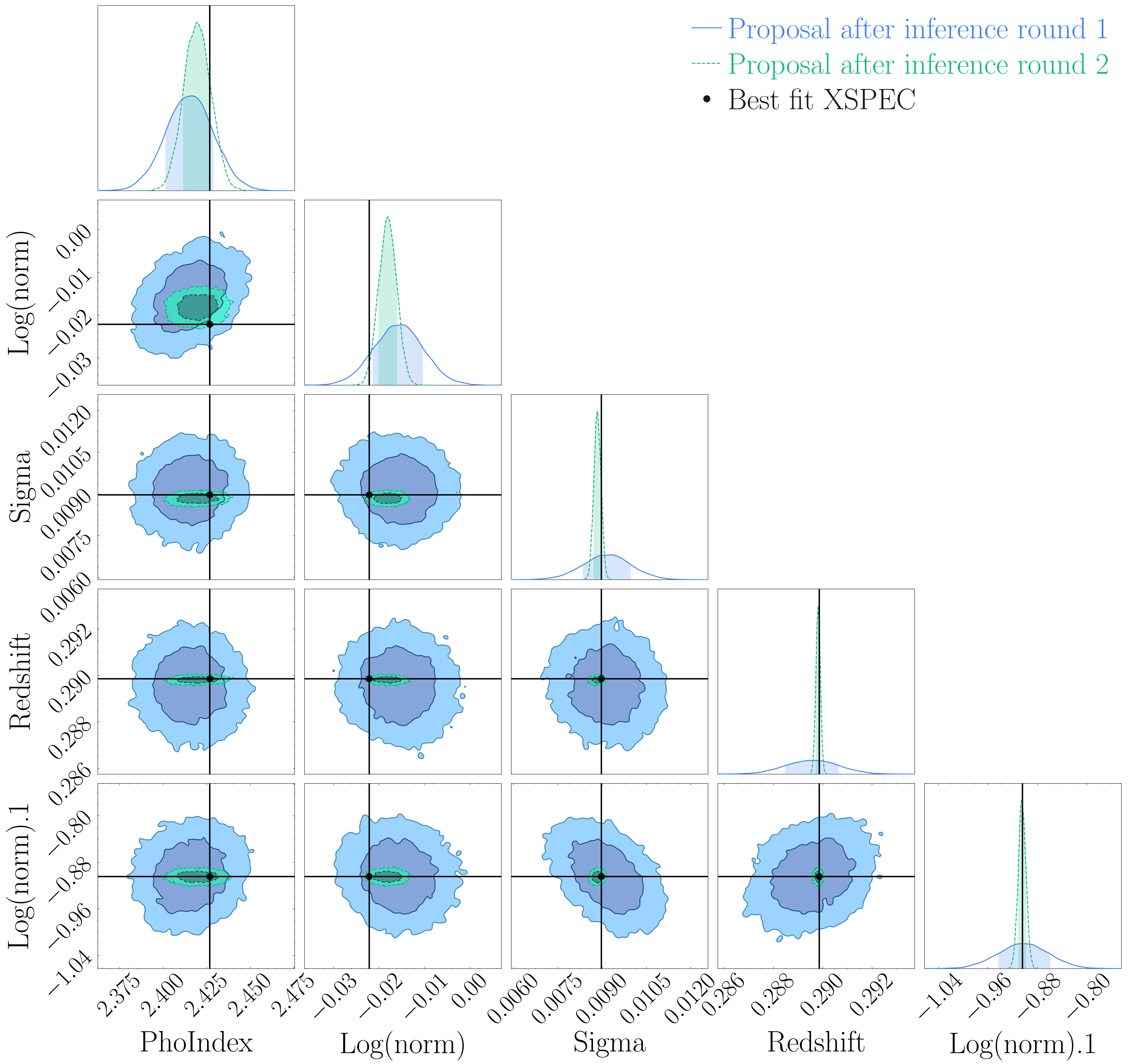
Diagnosis for the efficiency of the compression schemes

Truncated proposal

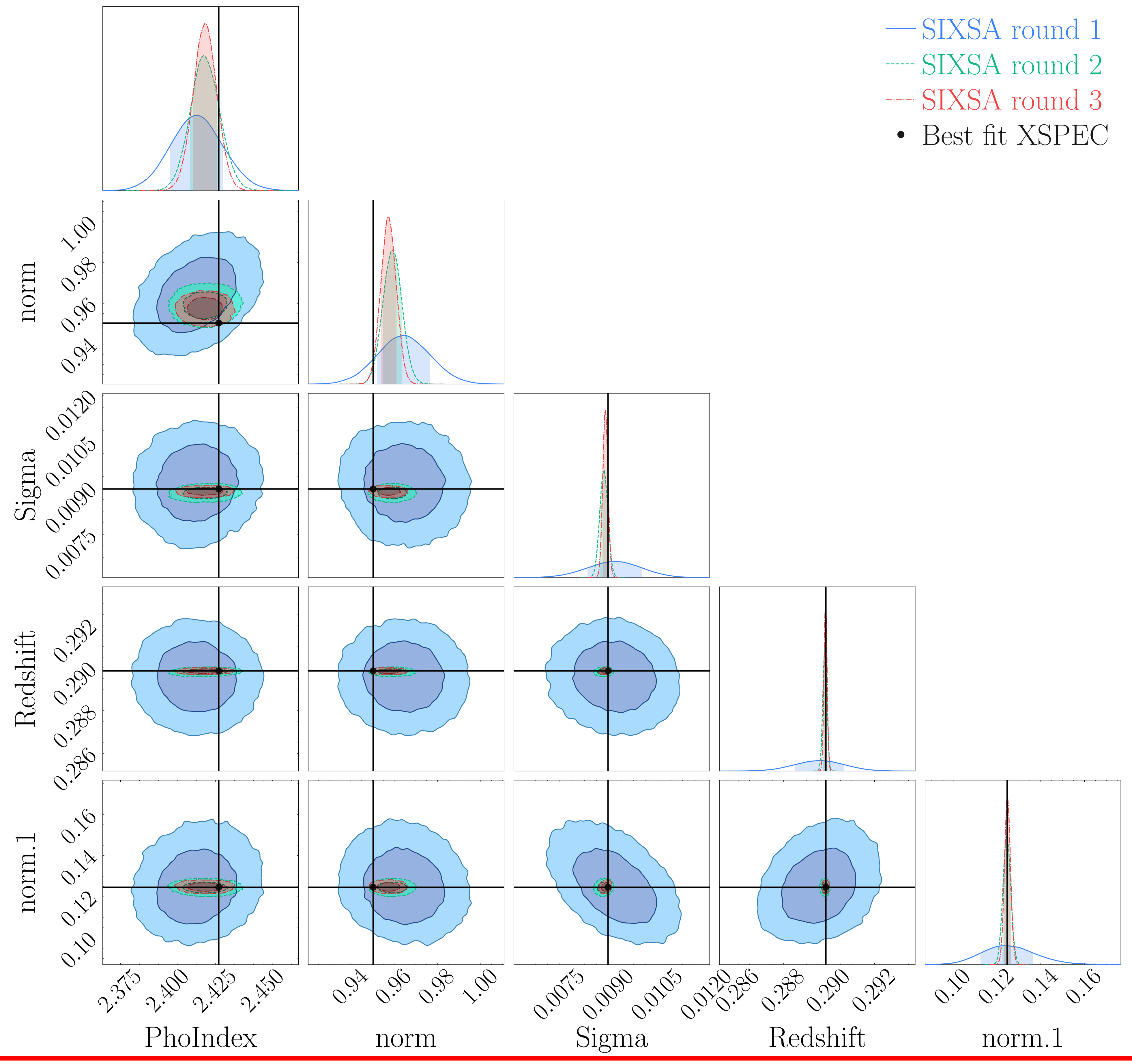
Neural network to speed up the sampling of the proposal, focusing on high density regions of the parameter space

- At each step, sanity checks are performed

Truncated proposal - useful for BXA

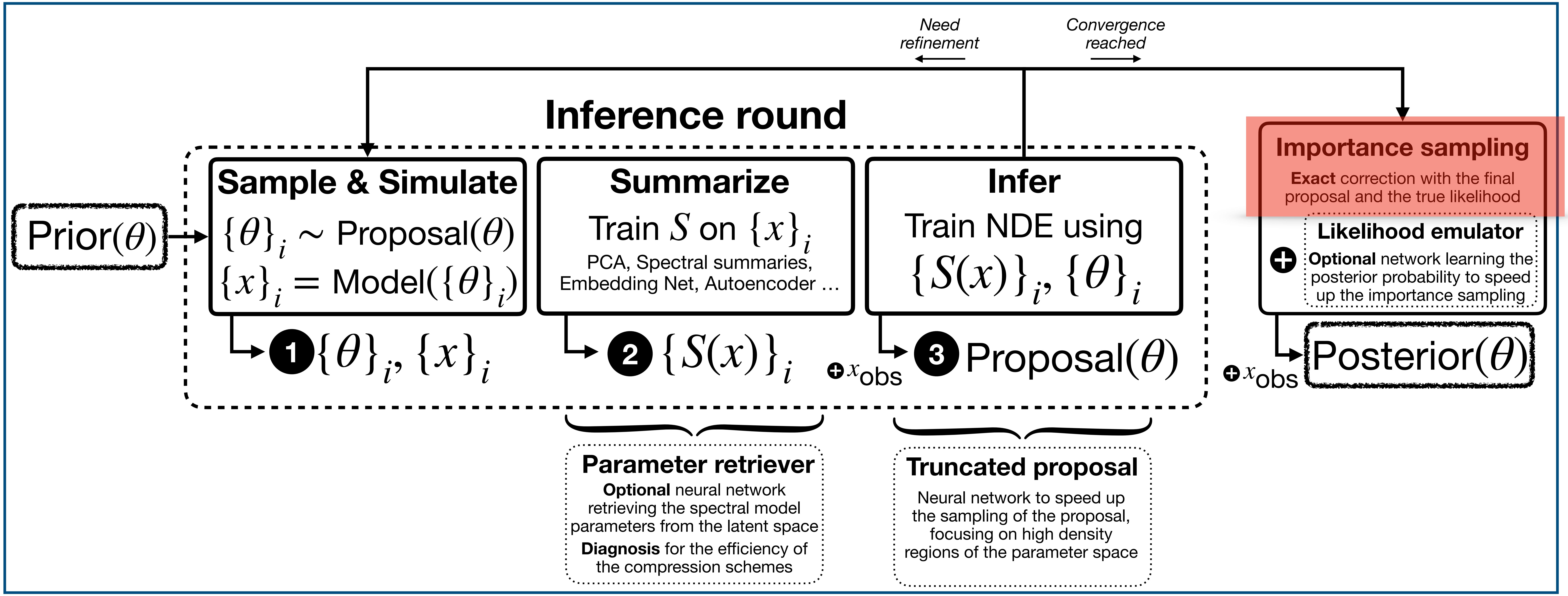


Truncated proposal after the first and second inference round



Posteriors after the first three rounds of inference

SIXSA pipeline — Multi-round inference



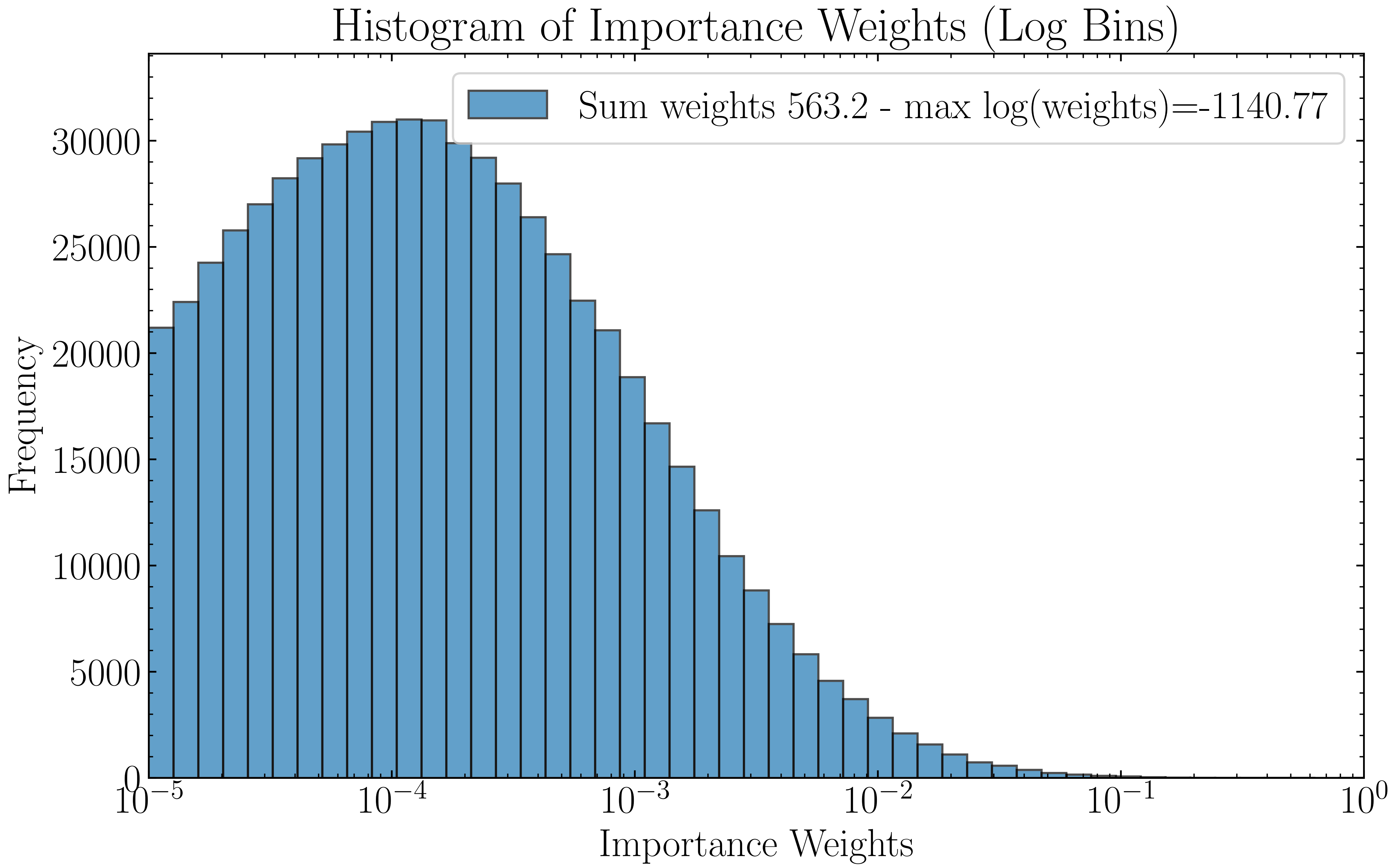
Need refinement
Convergence reached
Inference round
Prior(θ)
Sample & Simulate
$\{\theta\}_i \sim$ Proposal(θ)
$\{x\}_i =$ Model($\{\theta\}_i$)
Summarize
Train S on $\{x\}_i$
PCA, Spectral summaries, Embedding Net, Autoencoder …
Infer
Train NDE using
$\{S(x)\}_i, \{\theta\}_i$
1 $\{\theta\}_i, \{x\}_i$
2 $\{S(x)\}_i$
+ x_{obs}
3 Proposal(θ)
Importance sampling
Exact correction with the final proposal and the true likelihood
+
Likelihood emulator
Optional network learning the posterior probability to speed up the importance sampling
+ x_{obs}
Posterior(θ)
Parameter retriever
Optional neural network retrieving the spectral model parameters from the latent space
Diagnosis for the efficiency of the compression schemes
Truncated proposal
Neural network to speed up the sampling of the proposal, focusing on high density regions of the parameter space

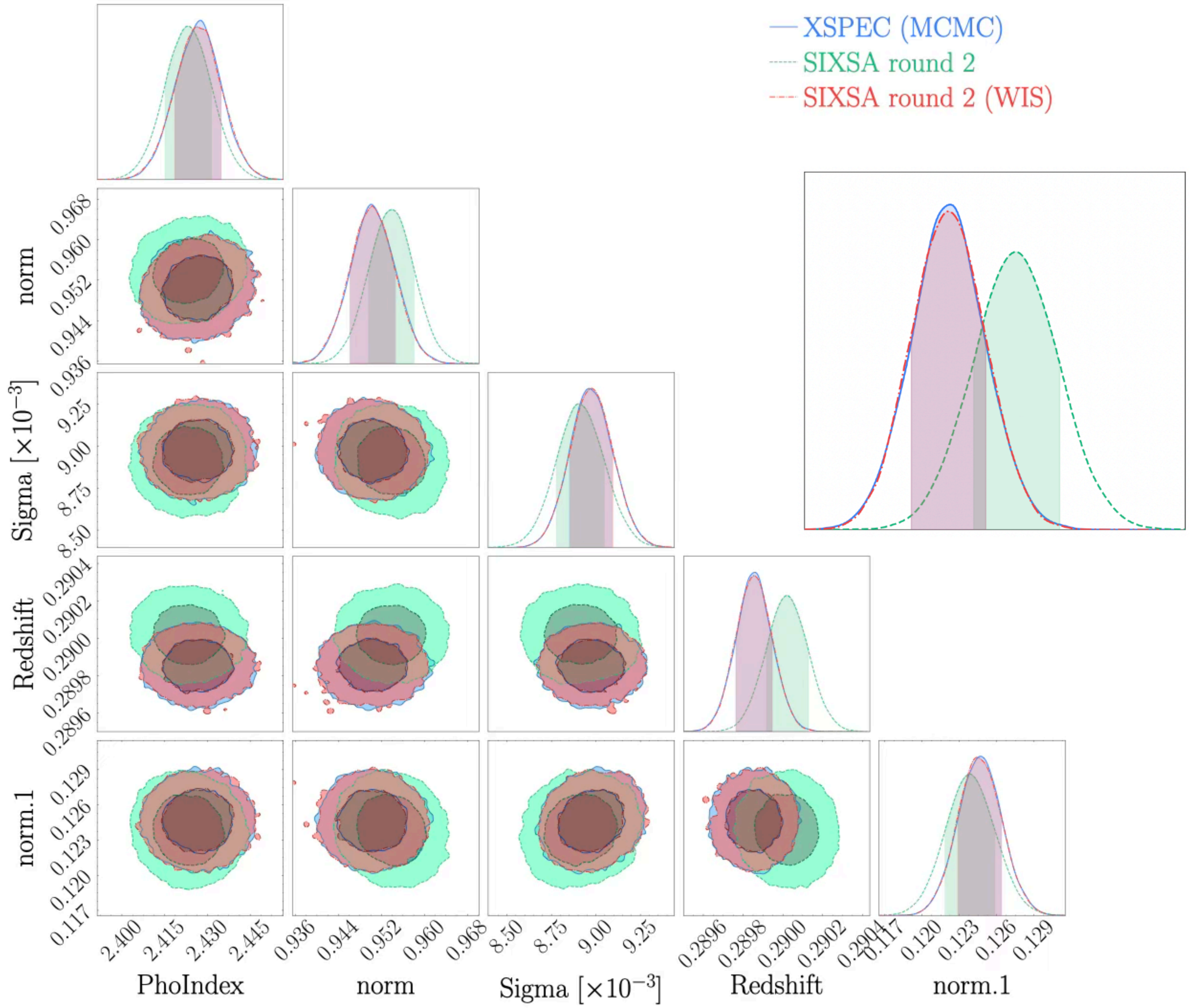
- At each step, sanity checks are performed

Importance sampling

- SIXSA does not require a likelihood to generate approximated posteriors.
- However, in our case, the likelihood is known and can be used to correct the approximate posteriors through importance sampling as to get asymptotically exact posteriors
- How do it works ?
  - The target posterior distribution is $p(\theta \mid x) \; \propto \; p(x \mid \theta)\, p(\theta)$,
  - The proposal distribution is $q(\theta \mid x)$
  - The importance weights are then : $w(\theta) \; \propto \; \dfrac{p(x \mid \theta) \;\; p(\theta)}{q(\theta \mid x)}$
  - We draw a large number of samples from the proposal distribution, compute their importance weights with the exact likelihoods

Importance sampling correction

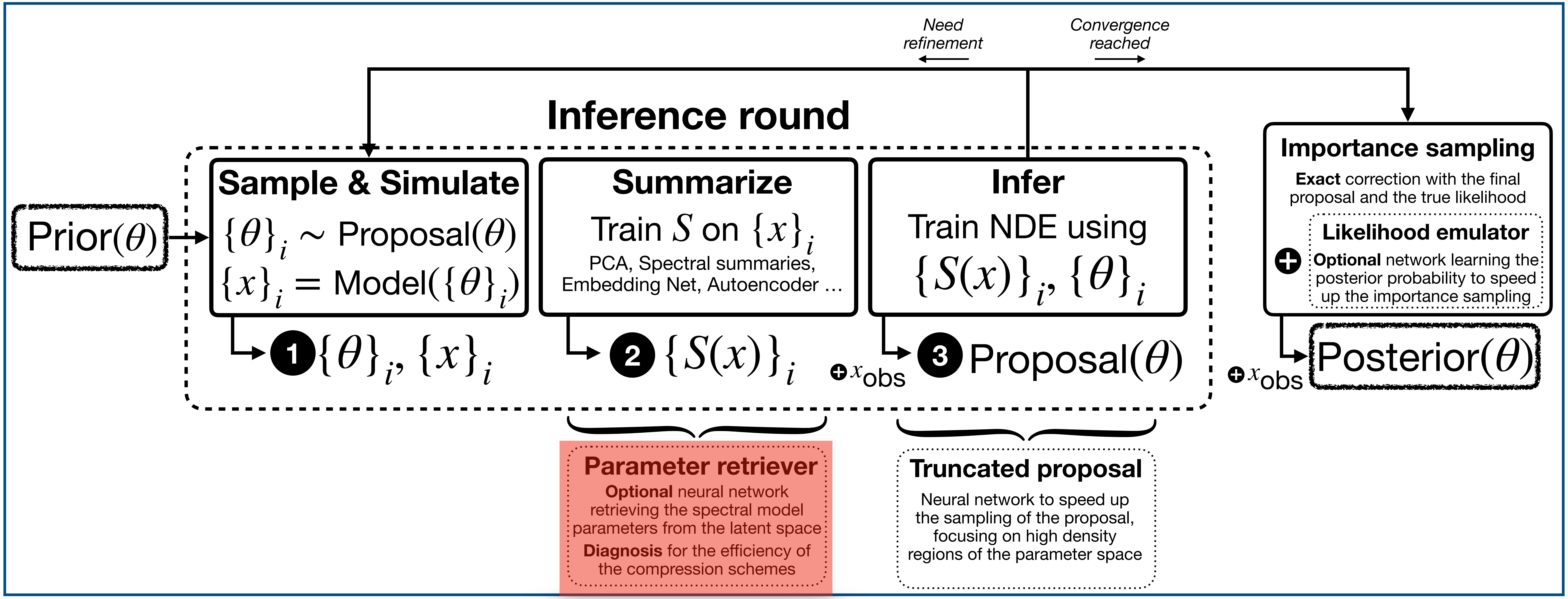




Histogram of importance weights

Importance sampling corrected posteriors with XSPEC MCMC

SIXSA pipeline — Multi-round inference



Need refinement
Convergence reached
Inference round
Prior(θ)
Sample & Simulate
$\{\theta\}_i \sim$ Proposal(θ)
$\{x\}_i =$ Model($\{\theta\}_i$)
Summarize
Train S on $\{x\}_i$
PCA, Spectral summaries, Embedding Net, Autoencoder …
Infer
Train NDE using
$\{S(x)\}_i, \{\theta\}_i$
1 $\{\theta\}_i, \{x\}_i$
2 $\{S(x)\}_i$
+ x_{obs}
3 Proposal(θ)
Importance sampling
Exact correction with the final proposal and the true likelihood
Likelihood emulator
+
Optional network learning the posterior probability to speed up the importance sampling
+ x_{obs}
Posterior(θ)
Parameter retriever
Optional neural network retrieving the spectral model parameters from the latent space
Diagnosis for the efficiency of the compression schemes
Truncated proposal
Neural network to speed up the sampling of the proposal, focusing on high density regions of the parameter space

- At each step, sanity checks are performed

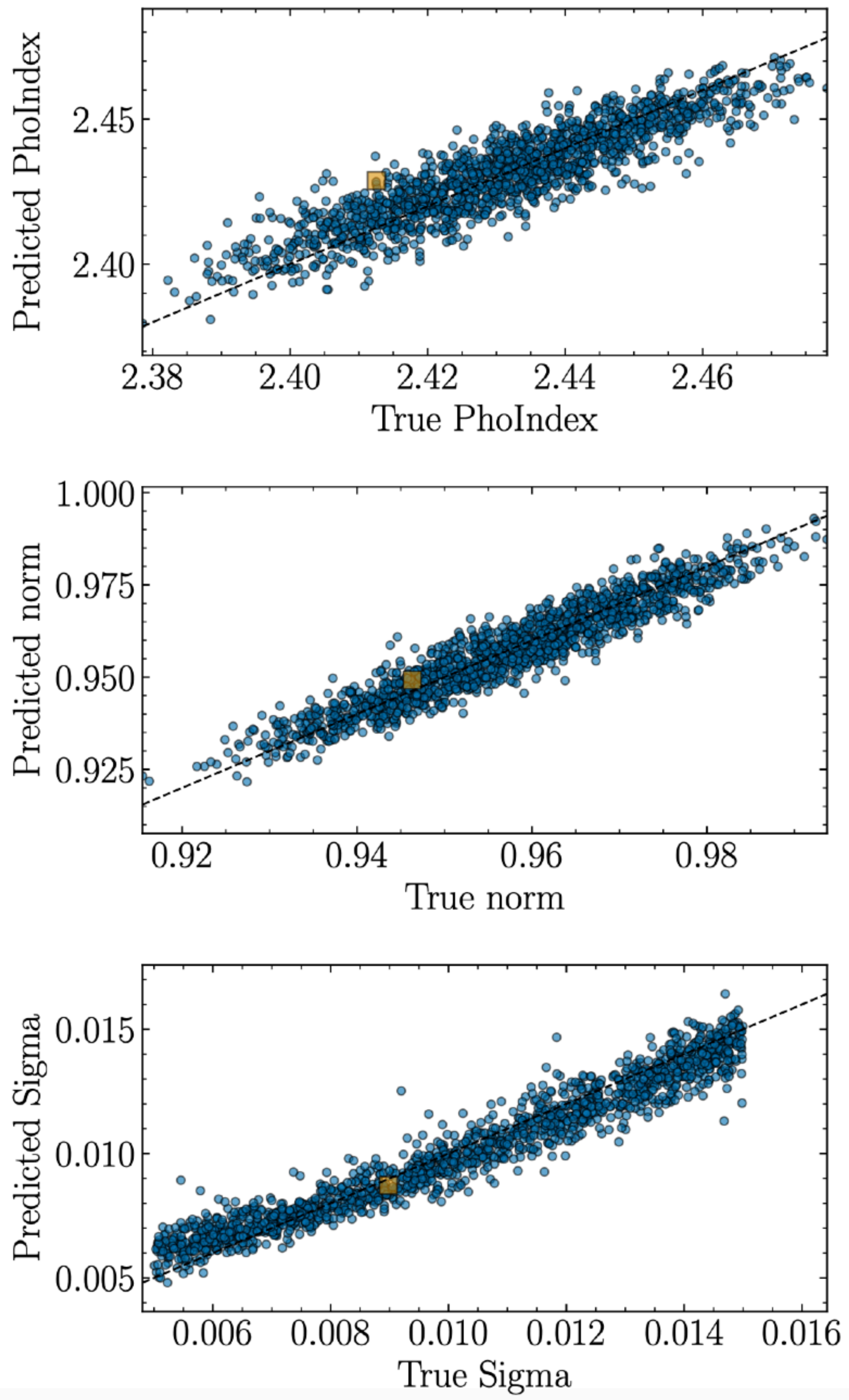
Parameter retriever

- Before using compressed spectral data in a neural density estimator, one can test whether a neural network can reliably map model parameters to the compressed spectra.
- This step validates the effectiveness of the compression method in preserving relevant information from the original high-dimensional spectra.
- Why Compression First?
  - Raw spectra contain thousands of bins—too complex for direct parameter mapping.
  - Compression reduces dimensionality while aiming to retain essential features.
- Testing this mapping helps determine:
  - Whether the compression retains the key physical features.
  - Which model parameters are best constrained by the data.
- Neural Network Architecture: Parameter_retriever
  - Type: Multi-Layer Perceptron (MLP)

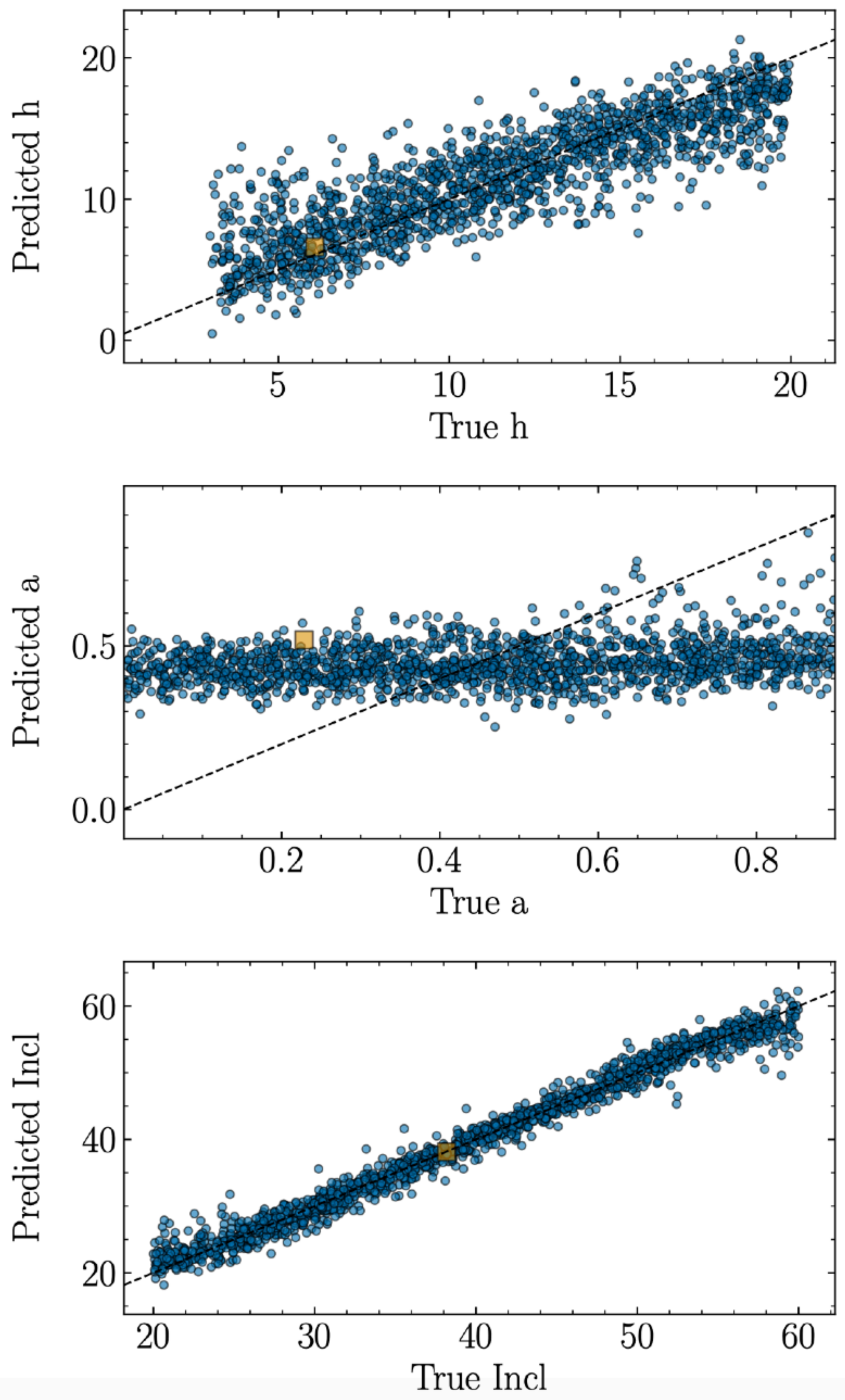
Parameter retriever

- Before using compressed spectral data in a neural density estimator, one can test whether a neural network can reliably map model parameters to the compressed spectra.
- This step validates the effectiveness of the compression method in preserving relevant information from the original high-dimensional spectra.
- Why Compression First?
  - Raw spectra contain thousands of bins—too complex for direct parameter mapping.
  - Compression reduces dimensionality while aiming to retain essential features.
- Testing this mapping helps determine:
  - Whether the compression retains the key physical features.
  - Which model parameters are best constrained by the data.
- Neural Network Architecture: Parameter_retriever
  - Type: Multi-Layer Perceptron (MLP)

Parameter retriever

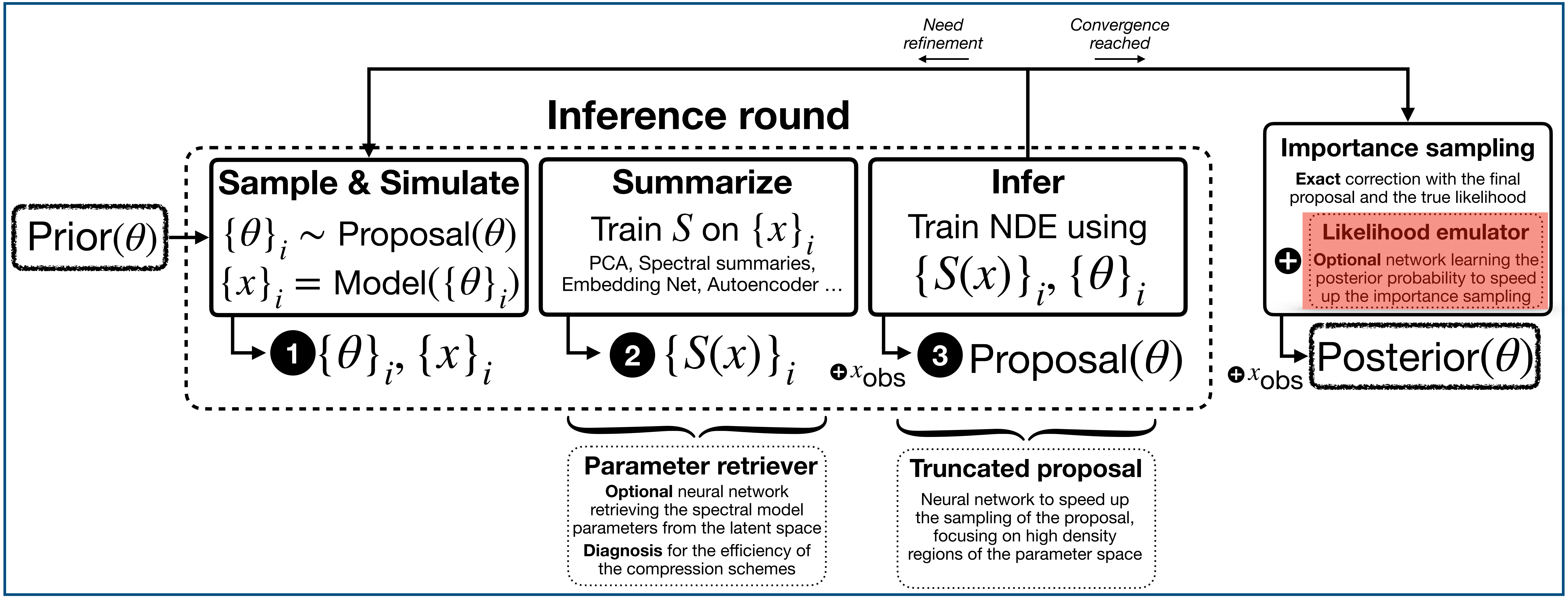


When things go right



When things go wrong (spin is not constrained)

SIXSA pipeline — Multi-round inference


Need refinement
Convergence reached
Inference round
Prior(θ)
Sample & Simulate
$\{\theta\}_i \sim$ Proposal(θ)
$\{x\}_i =$ Model($\{\theta\}_i$)
Summarize
Train S on $\{x\}_i$
PCA, Spectral summaries, Embedding Net, Autoencoder …
Infer
Train NDE using
$\{S(x)\}_i, \{\theta\}_i$
1 $\{\theta\}_i, \{x\}_i$
2 $\{S(x)\}_i$
+ x_{obs}
3 Proposal(θ)
Importance sampling
Exact correction with the final proposal and the true likelihood
+
Likelihood emulator
Optional network learning the posterior probability to speed up the importance sampling
+ x_{obs}
Posterior(θ)
Parameter retriever
Optional neural network retrieving the spectral model parameters from the latent space
Diagnosis for the efficiency of the compression schemes
Truncated proposal
Neural network to speed up the sampling of the proposal, focusing on high density regions of the parameter space

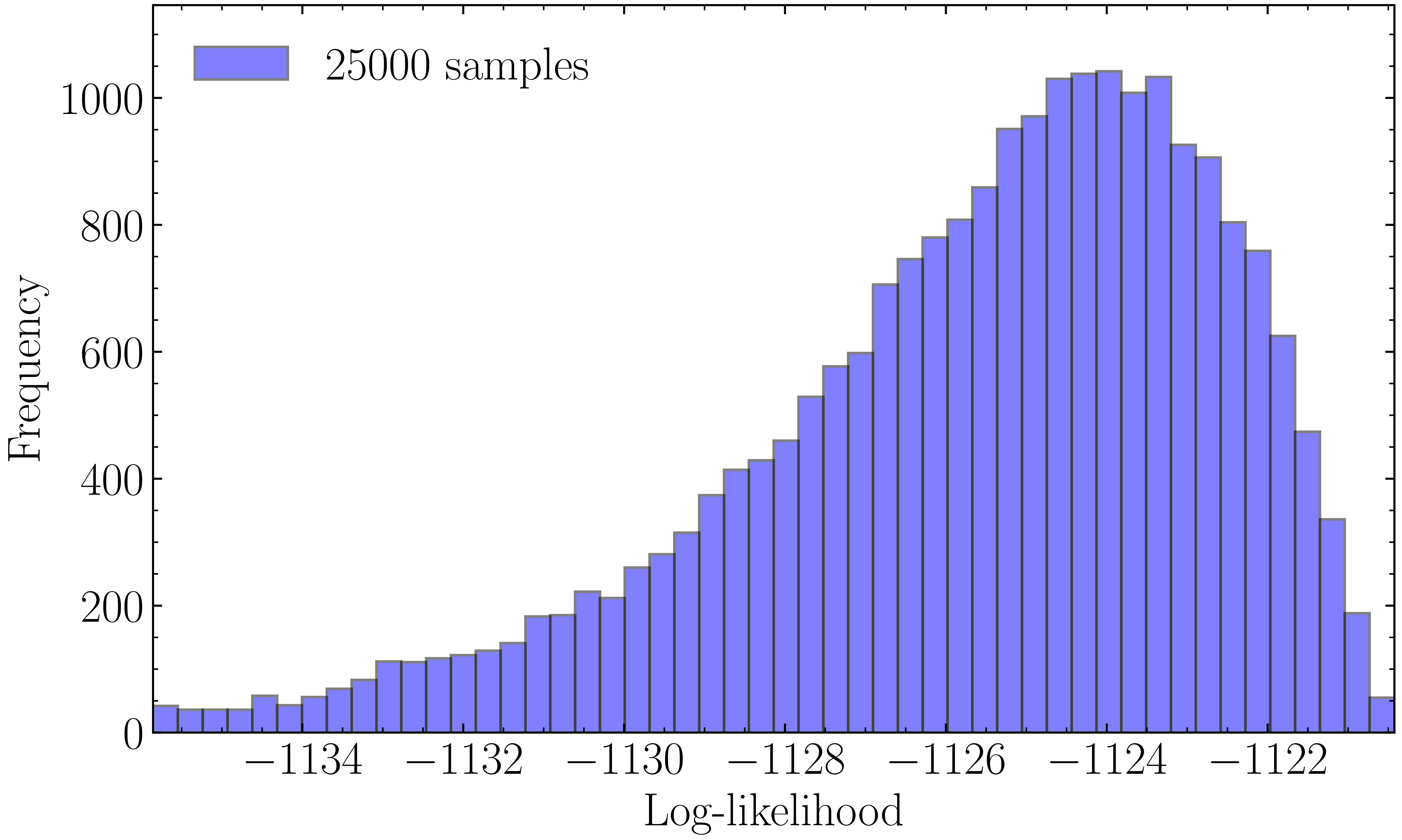
- At each step, sanity checks are performed

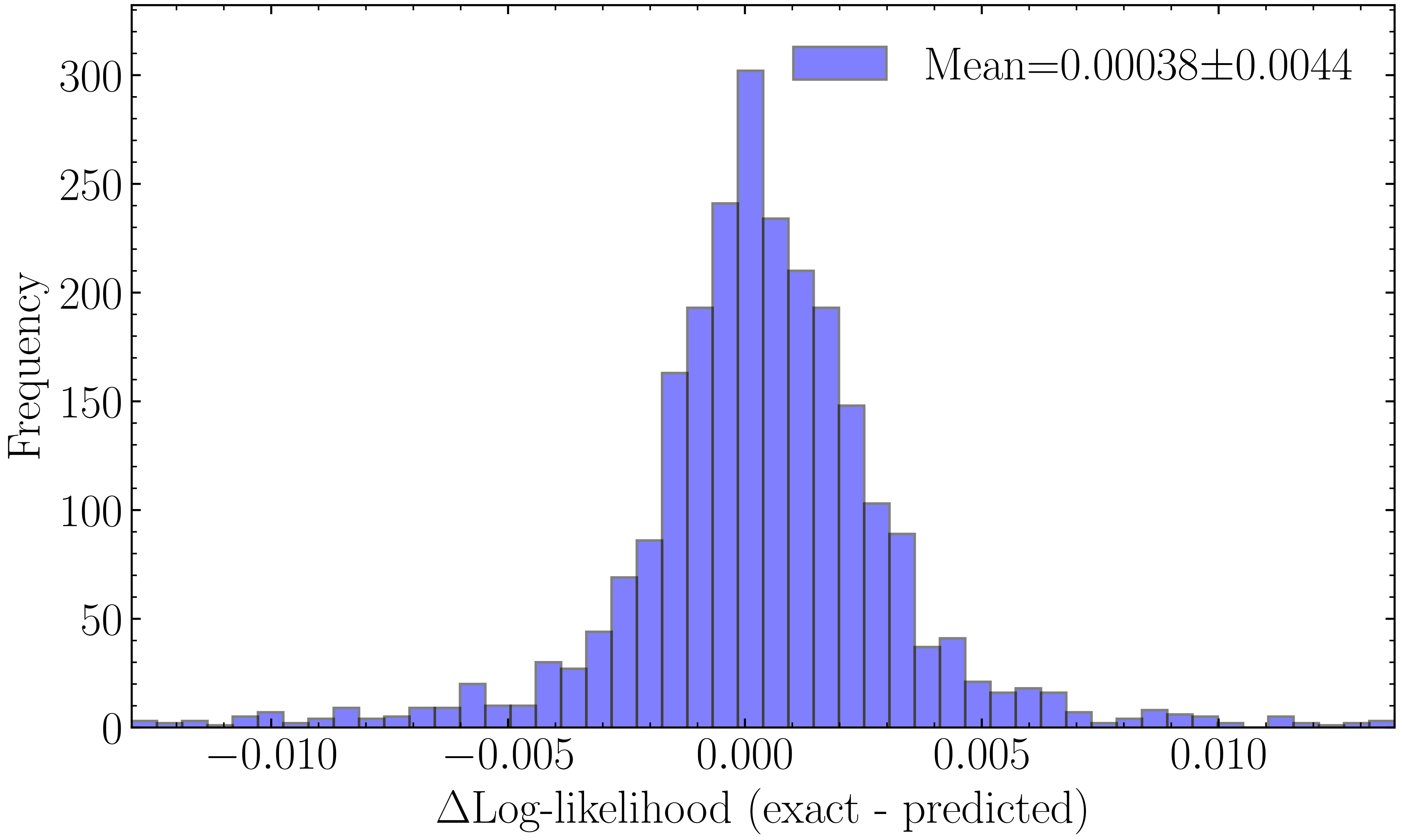
A likelihood emulator

- To reduce the computational burden of importance sampling, we use a neural network—Likelihood_emulator—to approximate the likelihood function. This allows for fast and efficient evaluation across large parameter spaces.
- Importance sampling requires a large number of likelihood evaluations.
- Neural networks can learn the mapping from spectral model parameters to the likelihood function using a moderate training set (~tens of thousands of samples).
- Once trained, the model dramatically speeds up evaluation without significantly sacrificing accuracy.

A likelihood emulator

- Once trained the NN can predict likelihoods from new model parameters in no time (importance sampling)

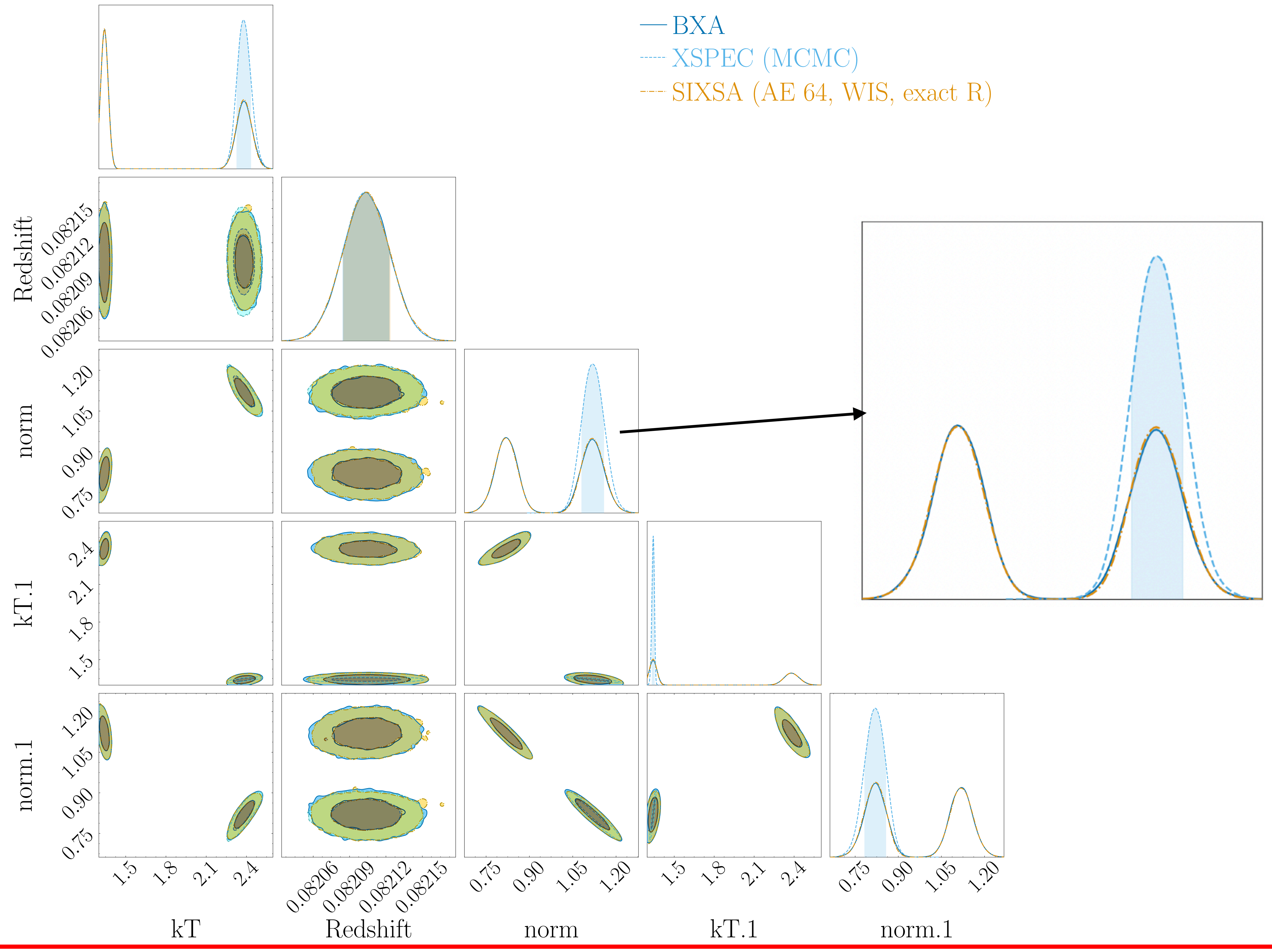




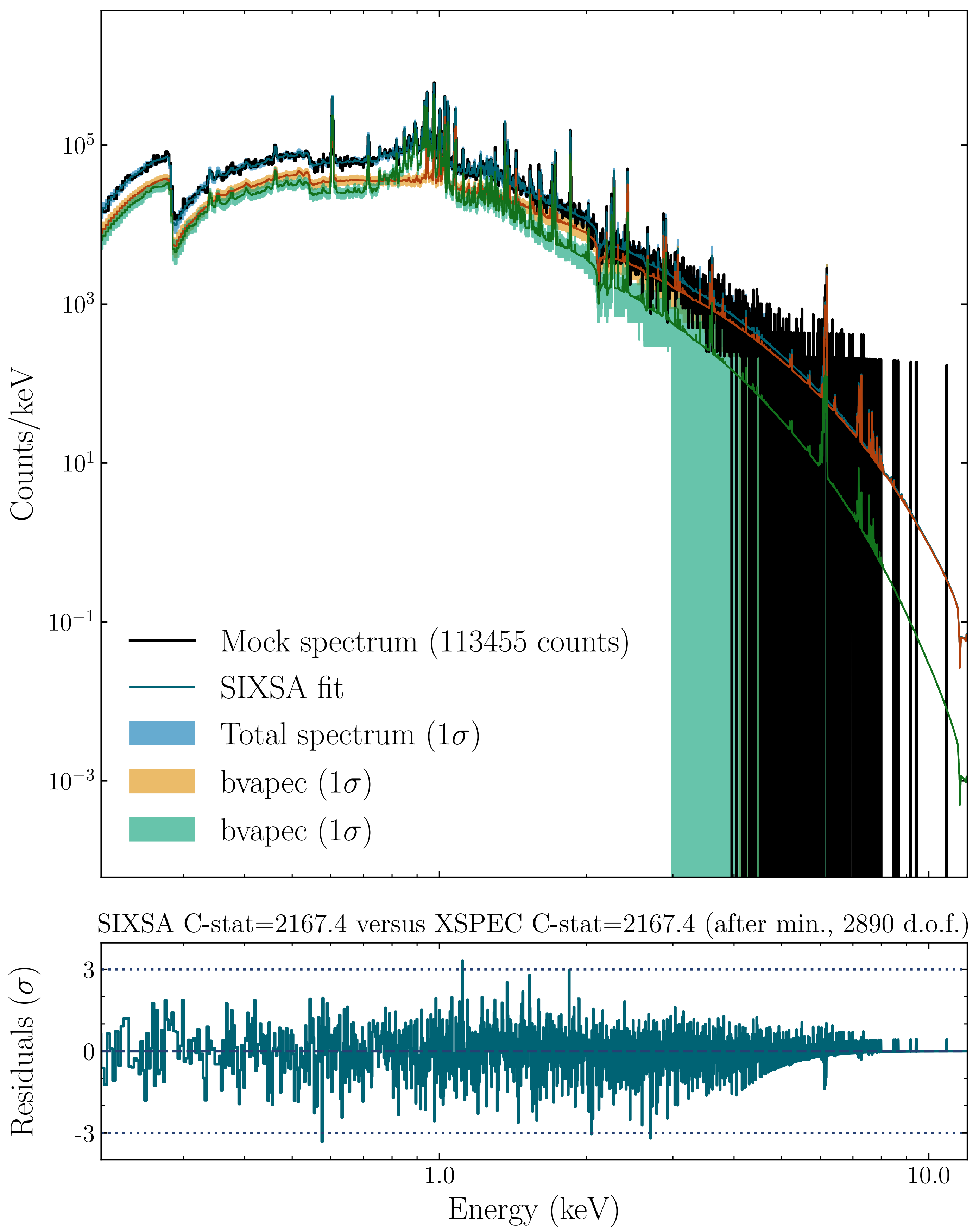
The training sample of the likelihood emulator

The accuracy to predict new likelihood with the emulator

Further demonstration with 2 bvapec

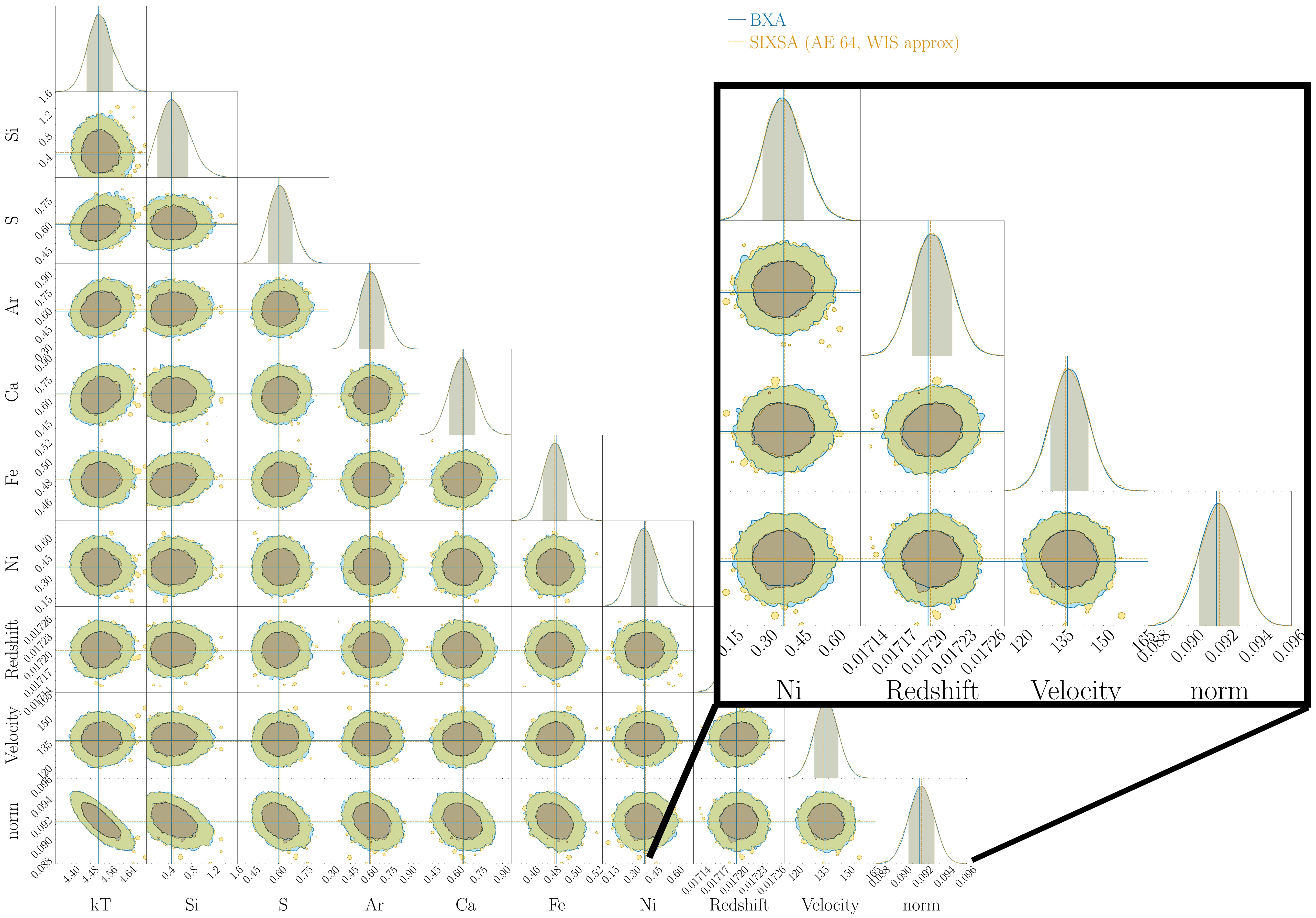


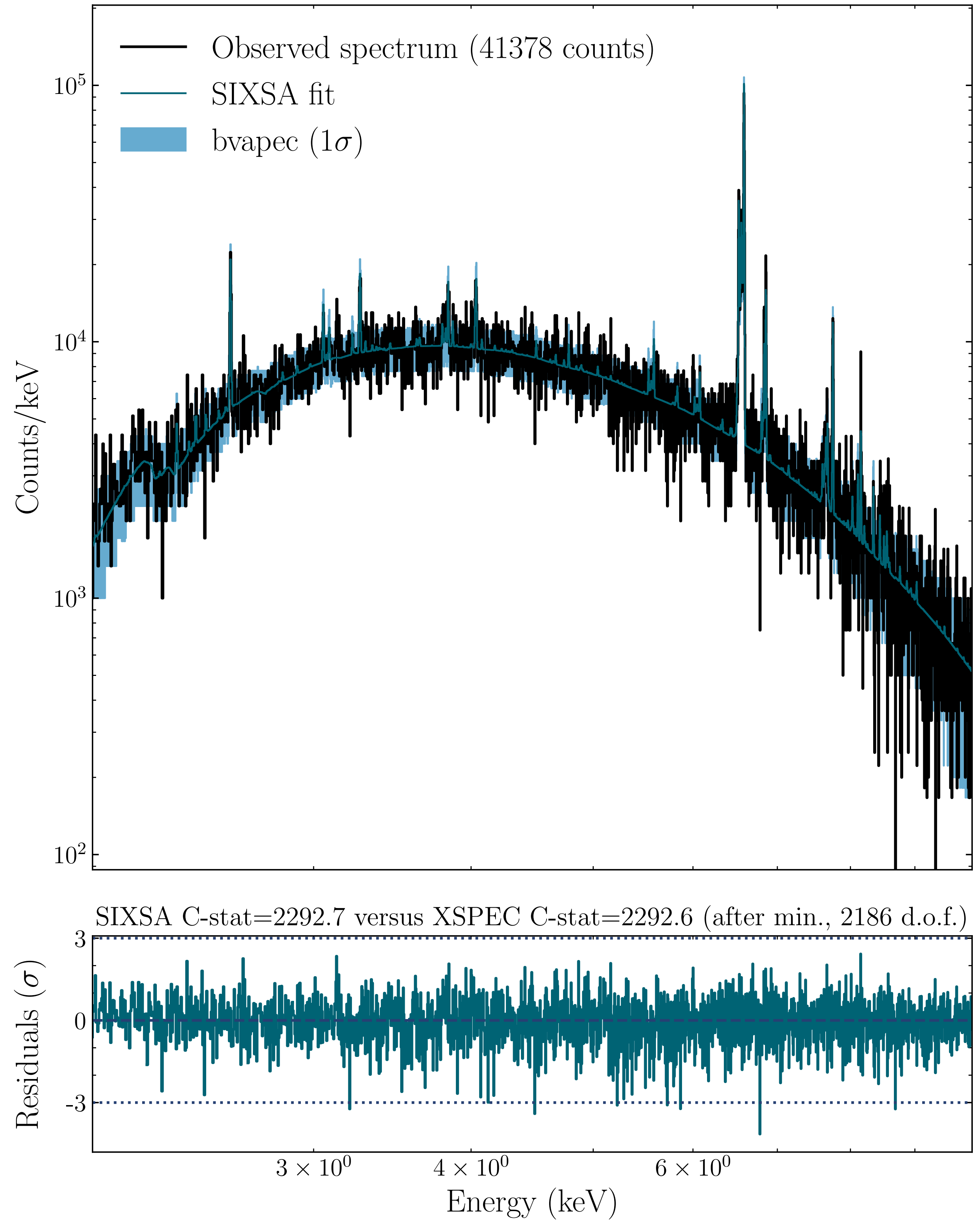
Posteriors for the two bvapec model



Folded spectrum with the two bvapec components

Application to real XRISM data - Perseus



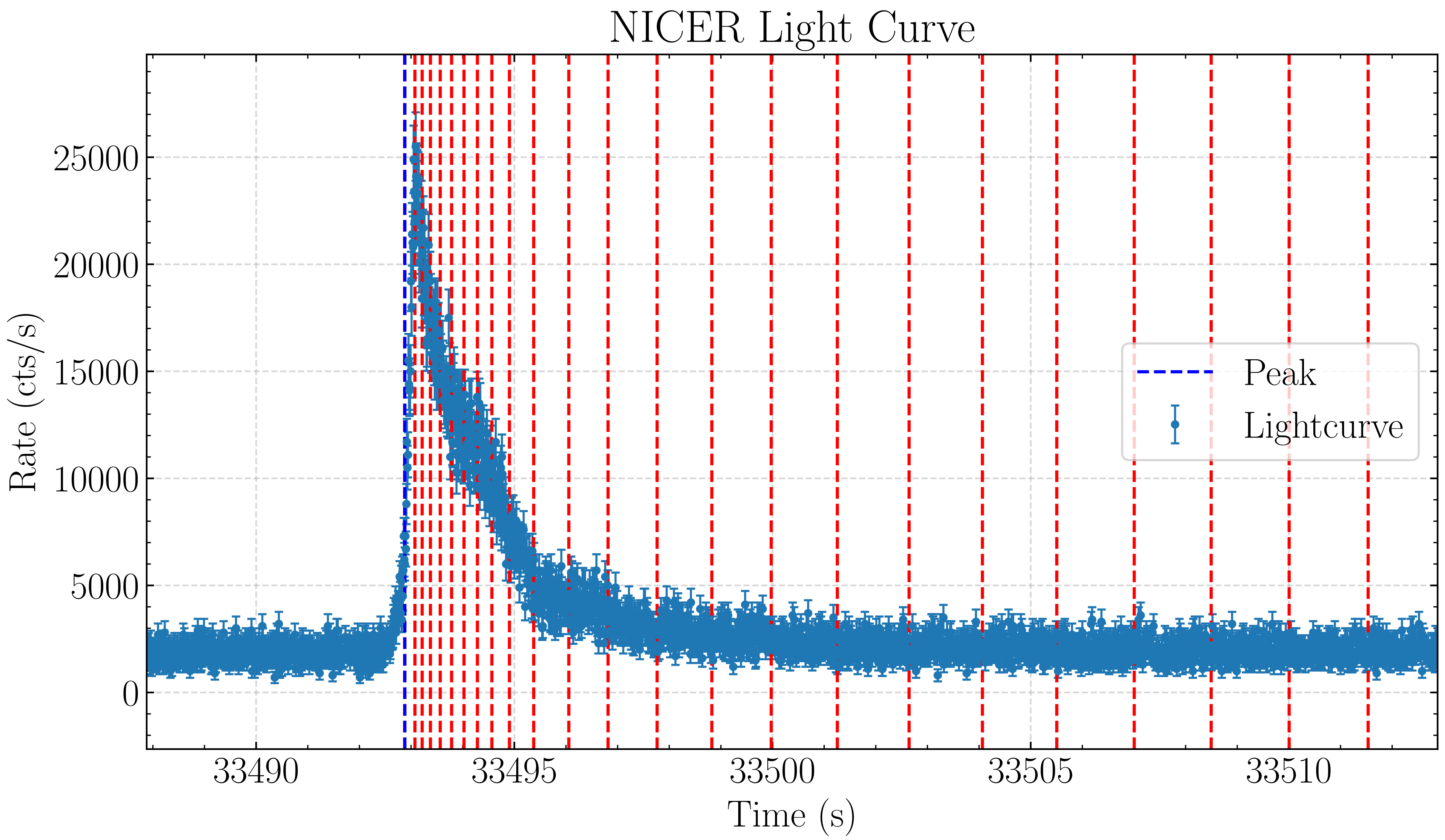


Posterior for a bvapec model with 6 metal abundance free

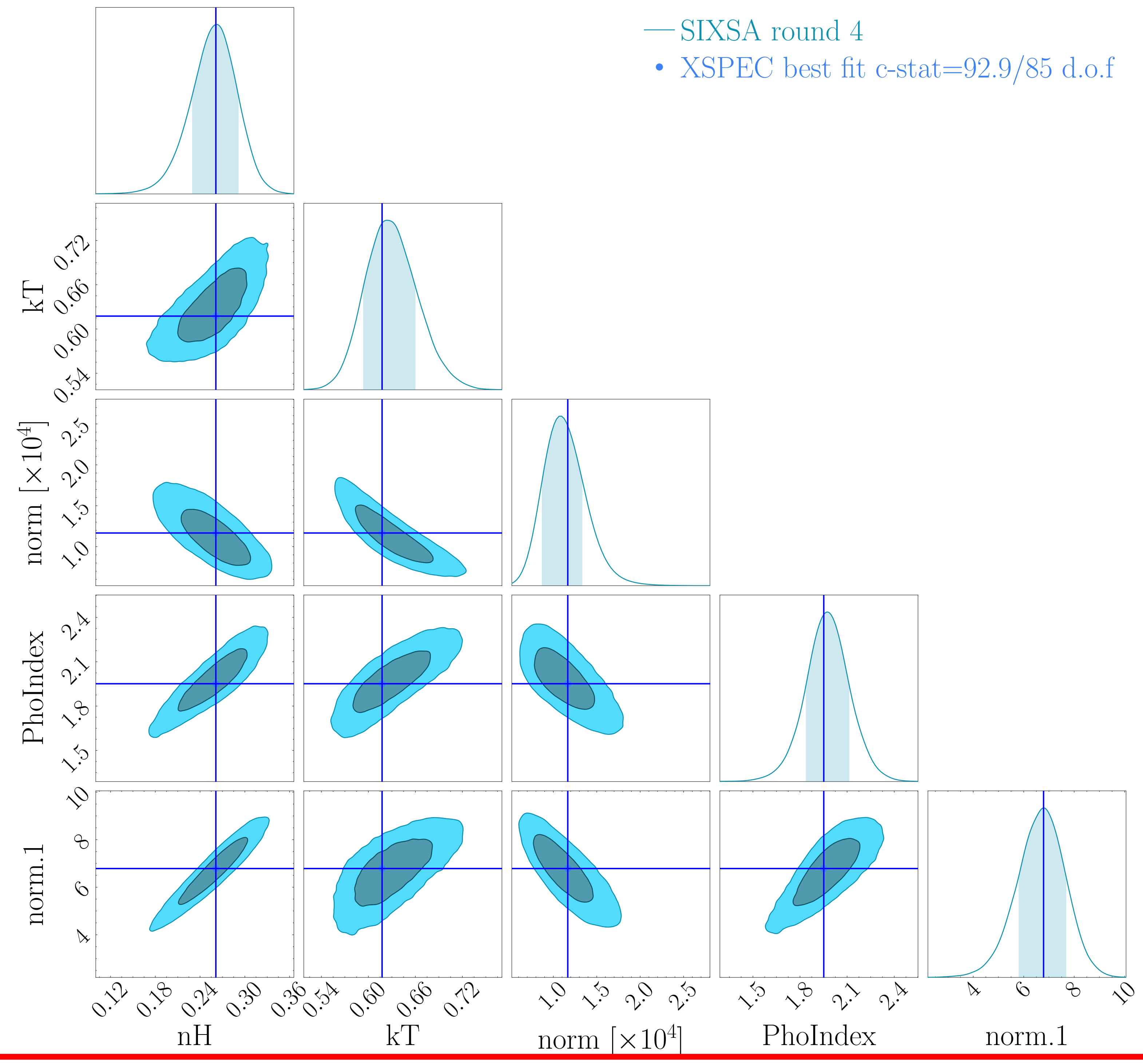
Folded spectrum with a bvapec model

Scientific application - Poisson regime

- Tracking the emission of the burst accounting for its interaction with the (blown) accretion disk emission !



A Type I X-ray burst observed by NICER



A posterior for one spectrum (2 minutes in MRI)

Conclusions

- Simulation-based inference with neural posterior estimation is a novel technique for X-ray spectra fitting
- SIXSA delivers exact posteriors, comparable to MCMC and BXA (compression and importance sampling)
- Applicable a wide range of spectral data from moderate to high spectral resolution data, biggest challenge is XRISM so far
- Adding new features, such as for detecting model mis-specifications, is planned an over the long run, investigate how to deal with systematics in the data
- Scientific applications to further increase its robustness. Nothing more efficient than real data.
- Catching up with latest developments in the field of SBI. Building blaock
- We want to make the building blocks available to the broader community ! — stay tuned !

BD, 2025arXiv251216709B