# Machine Learning Deep Neural Networks and experimental likelihoods

2020 International Workshop on
the High Energy CEPC
26 October 2020

**Riccardo Torre**

INFN Genova

Istituto Nazionale di Fisica Nucleare

Based on arxiv: 1911:03305 [hep-ph]

## The DNNLikelihood: enhancing likelihood distribution with Deep Learning

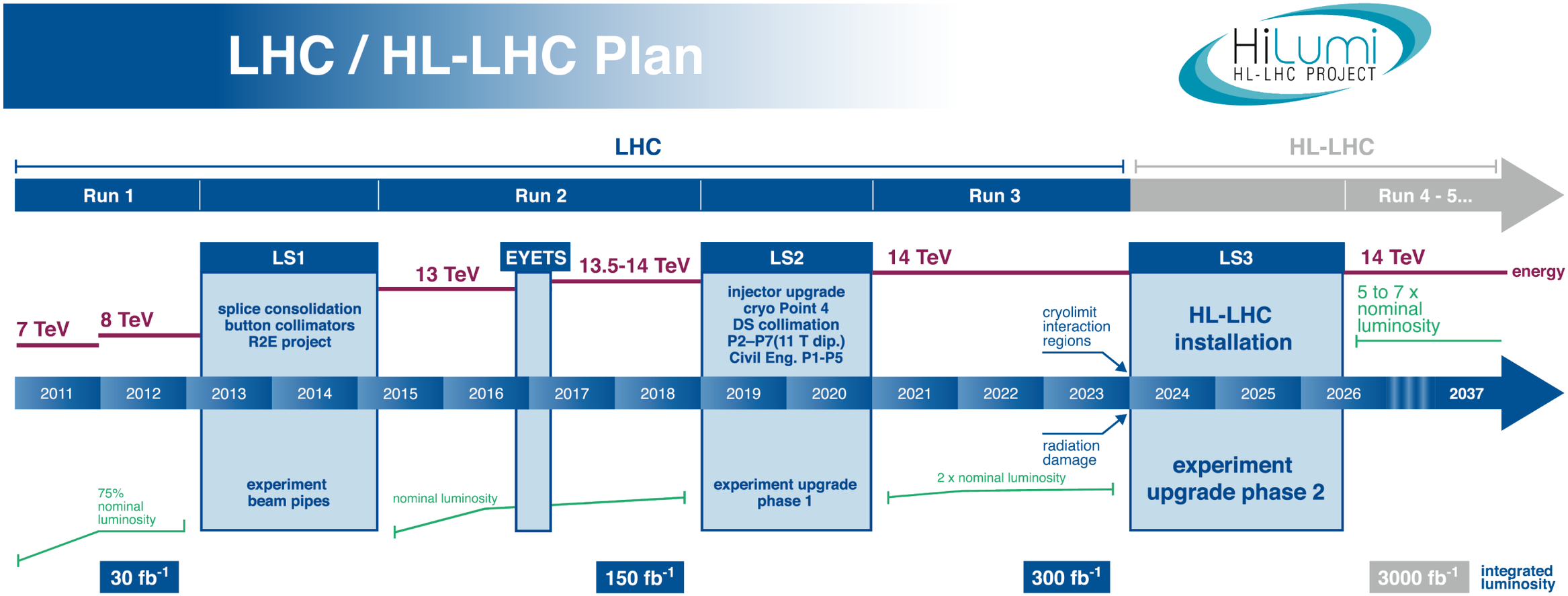
Andrea Coccaro[a], Maurizio Pierini[b], Luca Silvestrini[b,c], and Riccardo Torre[a,b]

[a] *INFN, Sezione di Genova, Via Dodecaneso 33, I-16146 Genova, Italy*

[b] *CERN, 1211 Geneva 23, Switzerland*

[c] *INFN, Sezione di Roma, P.le A. Moro, 2, I-00185 Roma, Italy*

# LHC physics program




Main goal: Find signs of New Physics

- directly: probing on-shell new physics → direct searches
- indirectly: probing the effect of new physics on SM observables → precision physics

# LHC legacy

- In case no new physics is found we have to ensure to take as much as possible anyway from the LHC
- After the HL-LHC it is important to catalog, combine (when possible) and preserve all physics results
- The parameter space of BSM physics explored at the LHC is huge and collecting results in the form of limits on cross-sections and/or 1D/2D plots is not enough
- The legacy of the LHC should consist of a collection of likelihood functions corresponding to the various channels and physics hypotheses
- The SMEFT is a primary candidate of such BSM parameter space. Hard to explore in details, many flat directions, etc.
- Need to develop proper statistical approaches to explore large parameter spaces, encode and preserve all information on theory/experimental uncertainties
- Machine Learning is gaining a leading role in this statistical approach since it offers both more power in exploring large parameter spaces and in encoding large amount of information in a standard and portable way

# The Likelihood Function

Bayes Theorem:

$$\mathcal{P}(\mathrm{data}|\mathrm{pars})\mathcal{P}(\mathrm{pars}) = \mathcal{P}(\mathrm{pars}|\mathrm{data})\mathcal{P}(\mathrm{data})$$

Likelihood function — Prior probability — Posterior probability — Bayesian evidence

**Frequentist inference**

e.g. Maximum Likelihood Estimation (MLE)

**Bayesian inference**

e.g. Maximum A Posteriori (MAP)

**The Likelihood Function (LF) is the central object in statistical inference!**

# Distributing likelihoods: existing proposals

Different approaches (and frameworks) in presenting and distributing experimental information

**Examples are:**

1. Present just the results of the analysis (this was the approach until recent years)

2. Cross section measurements with uncertainties and possibly correlations

3. Measurements in (possibly uncorrelated) bins for several different signal regions, as, for instance Higgs Simplified Template Cross Sections (STXS)

4. Simplified Likelihood: parametrize the likelihood in terms of “combined” nuisance parameters using Gaussian approximation up to 3$^{rd}$ moment

5. HistFactory framework (ATLAS): this is going towards publishing all information that allows to reconstruct the full likelihood

6. …?

# Distributing likelihoods: our approach

**Our approach**: encode the full likelihood with all the dependence on elementary nuisance parameters into a DNN function. This allows for:

1. Encoding also unbinned likelihoods (especially, but not only, used in Flavor physics)
2. Interpolation of different signal regions/hypotheses beyond simple approximations
3. Re-sampling with custom priors to study the impact of different hypotheses on systematic uncertainties
4. Efficient combination of different likelihoods (when correlations are known)
5. Interpretation of results within different statistical approaches (Frequentist vs Bayesian)
6. Simple framework independent distribution through the ONNX format (this allows inference in any software environment (Python, R, Matlab, Mathematica, etc..)

**Train with several examples**

$$(\vec{\mu}, \vec{\theta}) \rightarrow \mathcal{L}(\vec{\mu}, \vec{\theta})$$

**Obtain a DNN interpolator**

$$\mathcal{L}_{\mathrm{DNN}}(\vec{\mu}, \vec{\theta})$$

# A toy LHC-like New Physics search

Toy experiment already considered in the literature

**Buckley, Citron, Fichet, Kraml, Waltenberger, Wardle, 1809.05548 [hep-ph]**

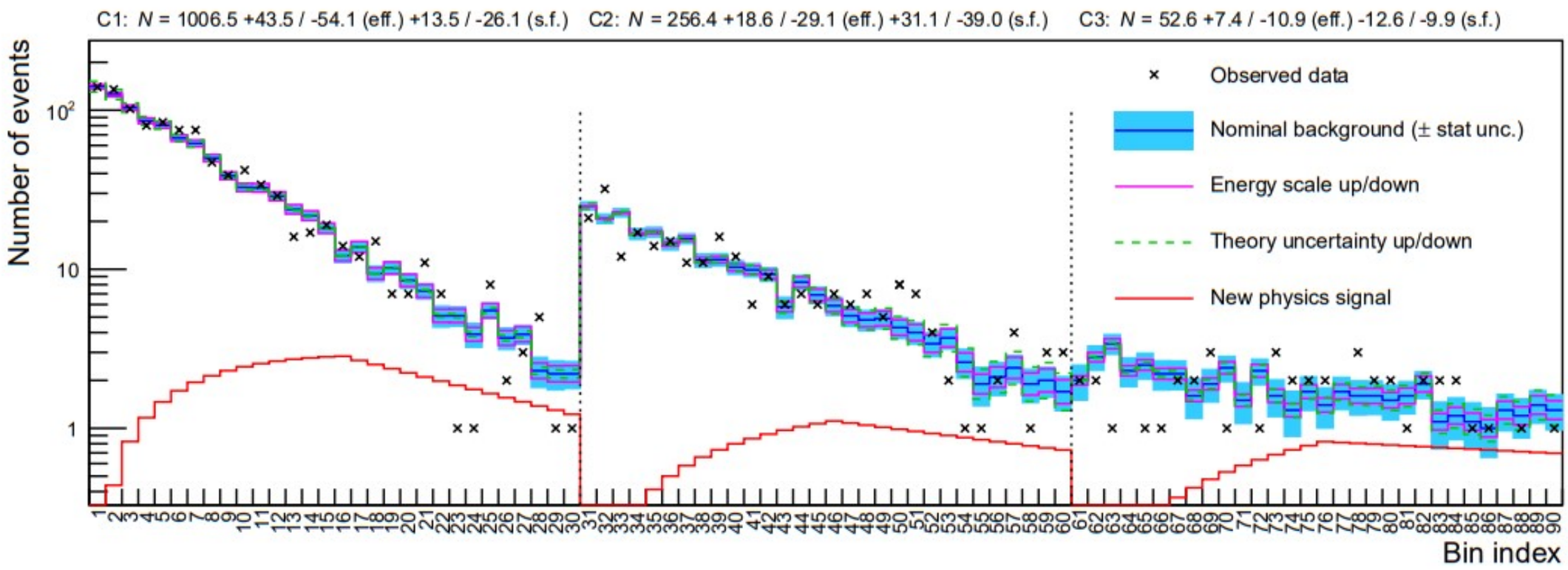



**Figure 2**. LHC-like search for new physics (mockup). The search is performed across three event categories, each divided into 30 bins to make a total of 90 search regions. The nominal expected contribution in each bin from the background and from the new physics signal is shown by the blue and red lines, respectively. The solid and dashed lines show the $\pm 1\sigma$ correlated variation in each bin expected due to an experimental and theoretical uncertainty while the blue shaded band shows the uncorrelated uncertainty in each bin due to limited MC simulation. The "observed" number of events in data in each bin is indicated by the black points.

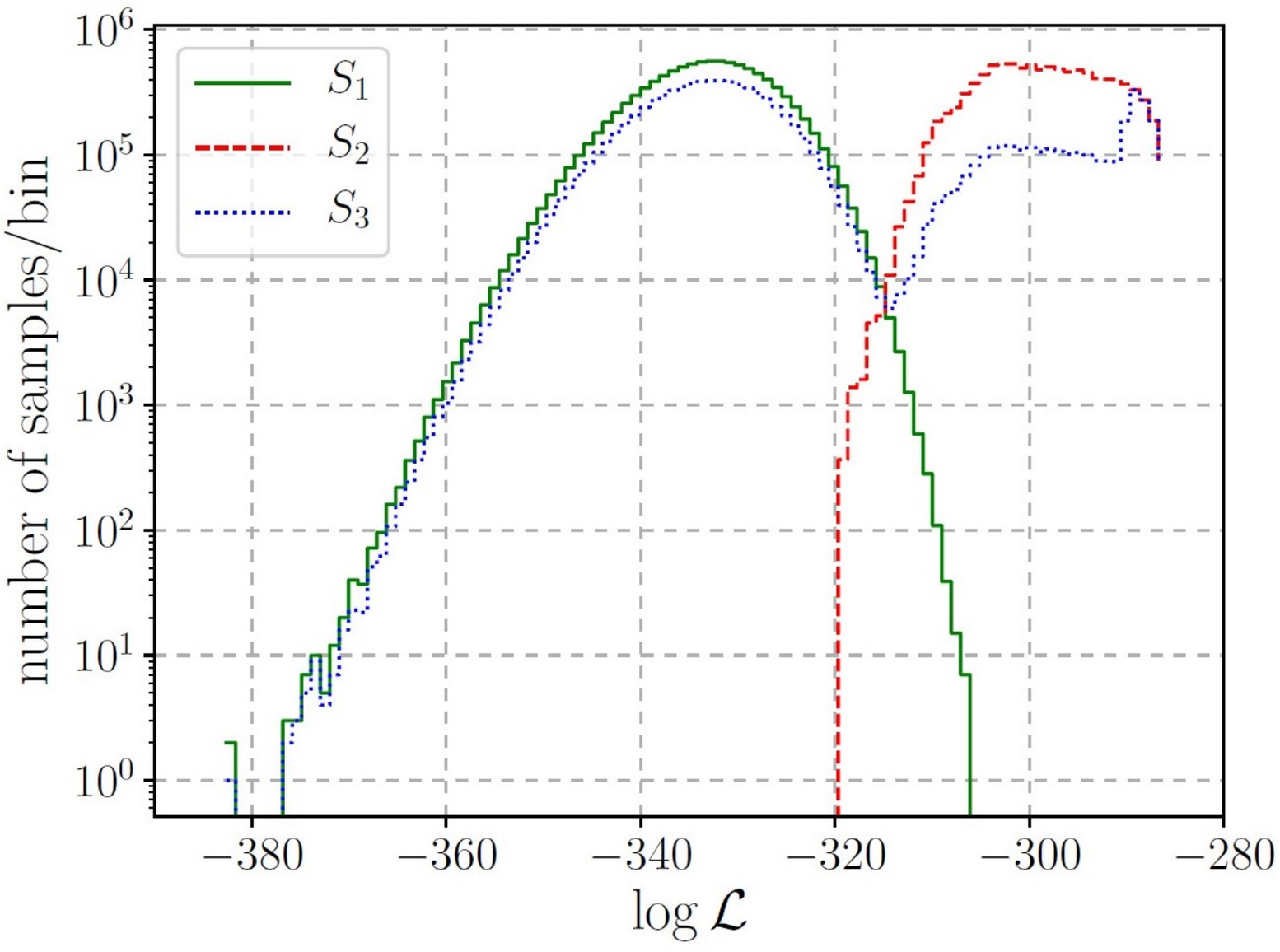
- One physical parameter (signal strength $\mu$)
- 94 nuisance parameters (90 fully uncorrelated, two fully correlated, two normalizations)
- Non Gaussian (and not satisfying Wilks' hypotheses!)

# Sampling

Supervised learning problem (interpolation) where high precision is needed

If we want to allow for both Frequentist and Bayesian inference, we need to know the LF well in very different regions (where prior volume is large and close to local maxima of the LF)

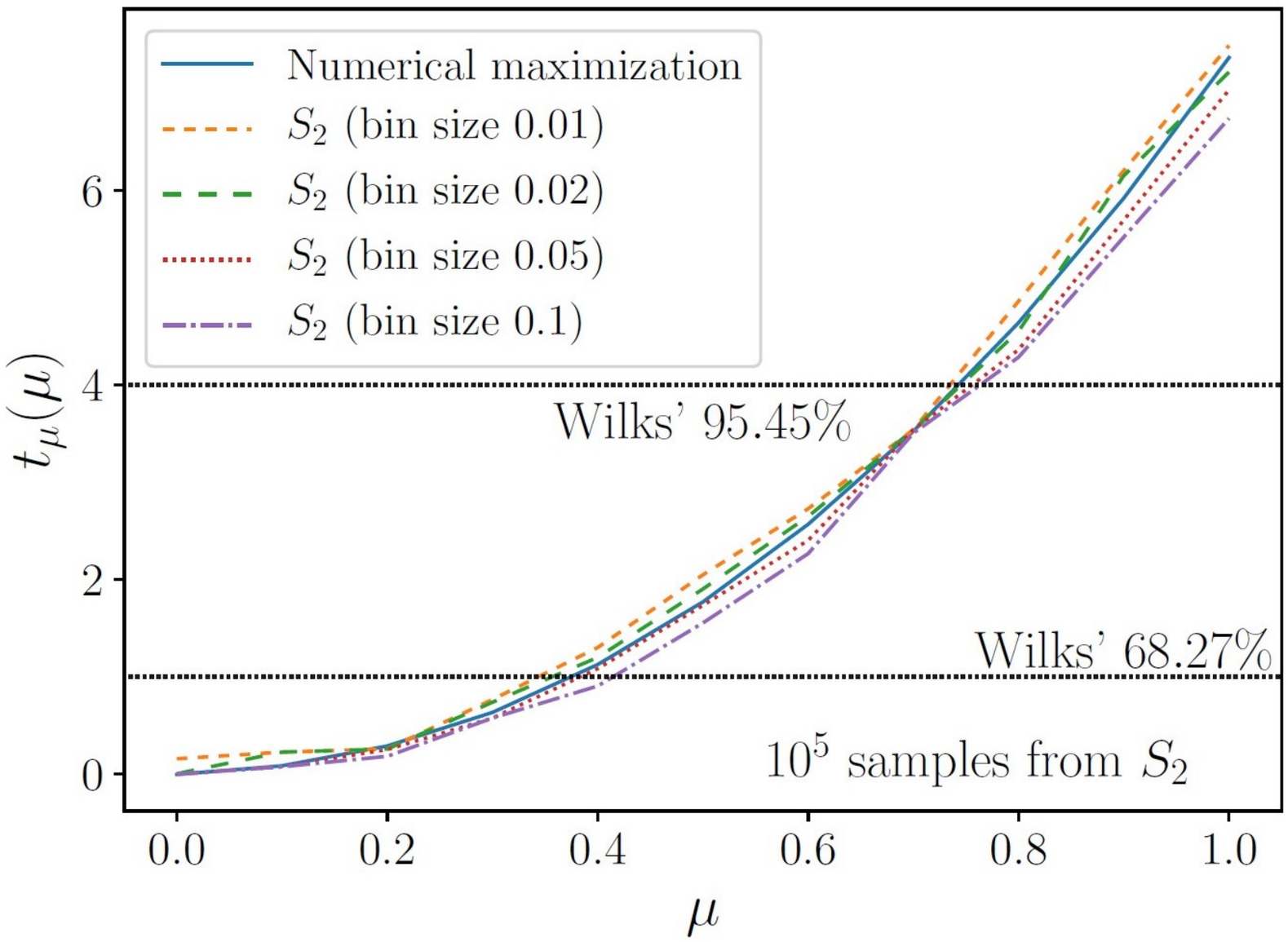
We sample with the emcee3 (ensemble sampling method) Python package (checking convergence with several different techniques)



# Inference with analysis likelihood: Frequentist

For frequentist inference we construct the test statistics

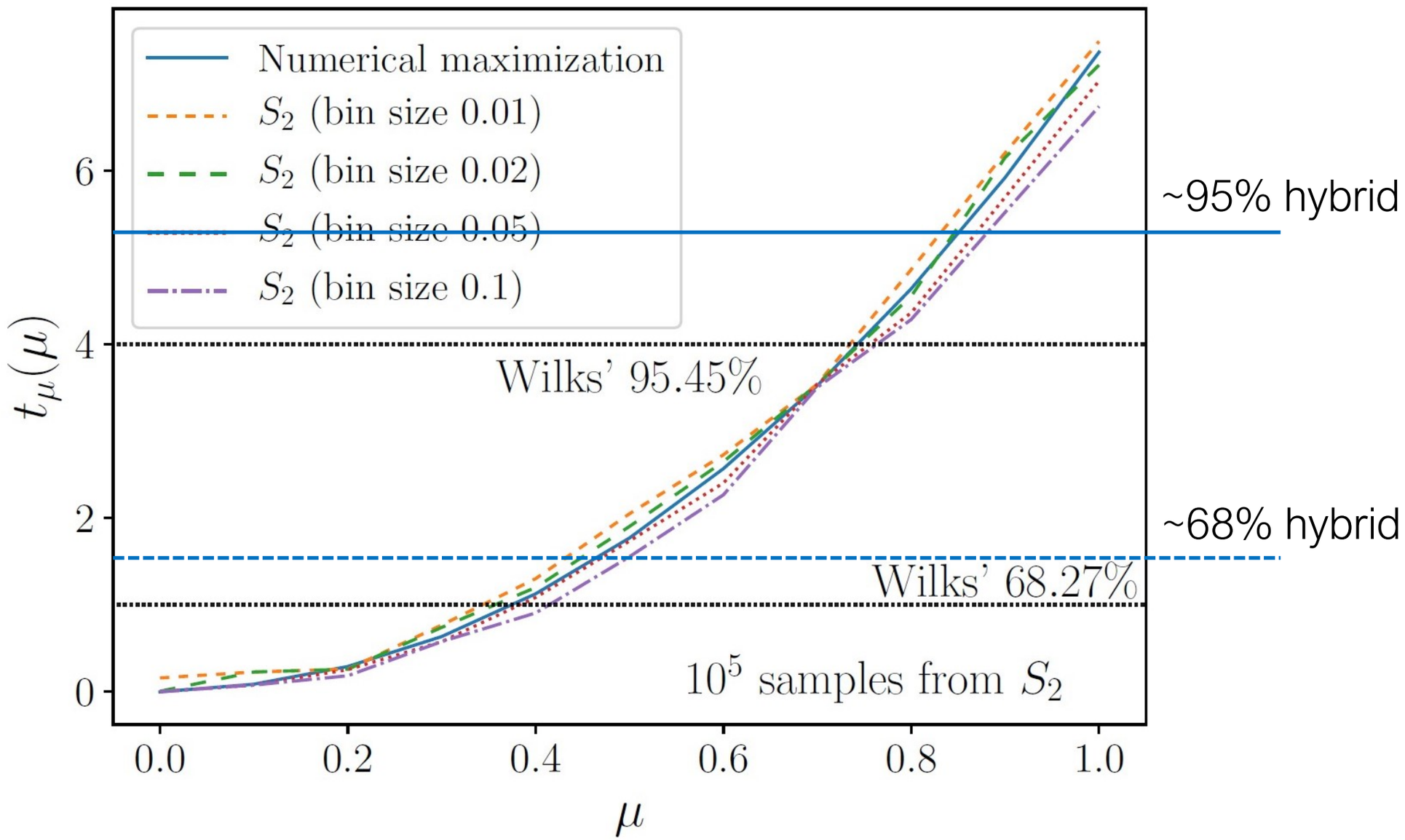
$$t_\mu(\mu) = -2\log\frac{\mathcal{L}_{\text{prof}}(\mu)}{\mathcal{L}_{\max}}$$



# Inference with analysis likelihood: Frequentist

For frequentist inference we construct the test statistics

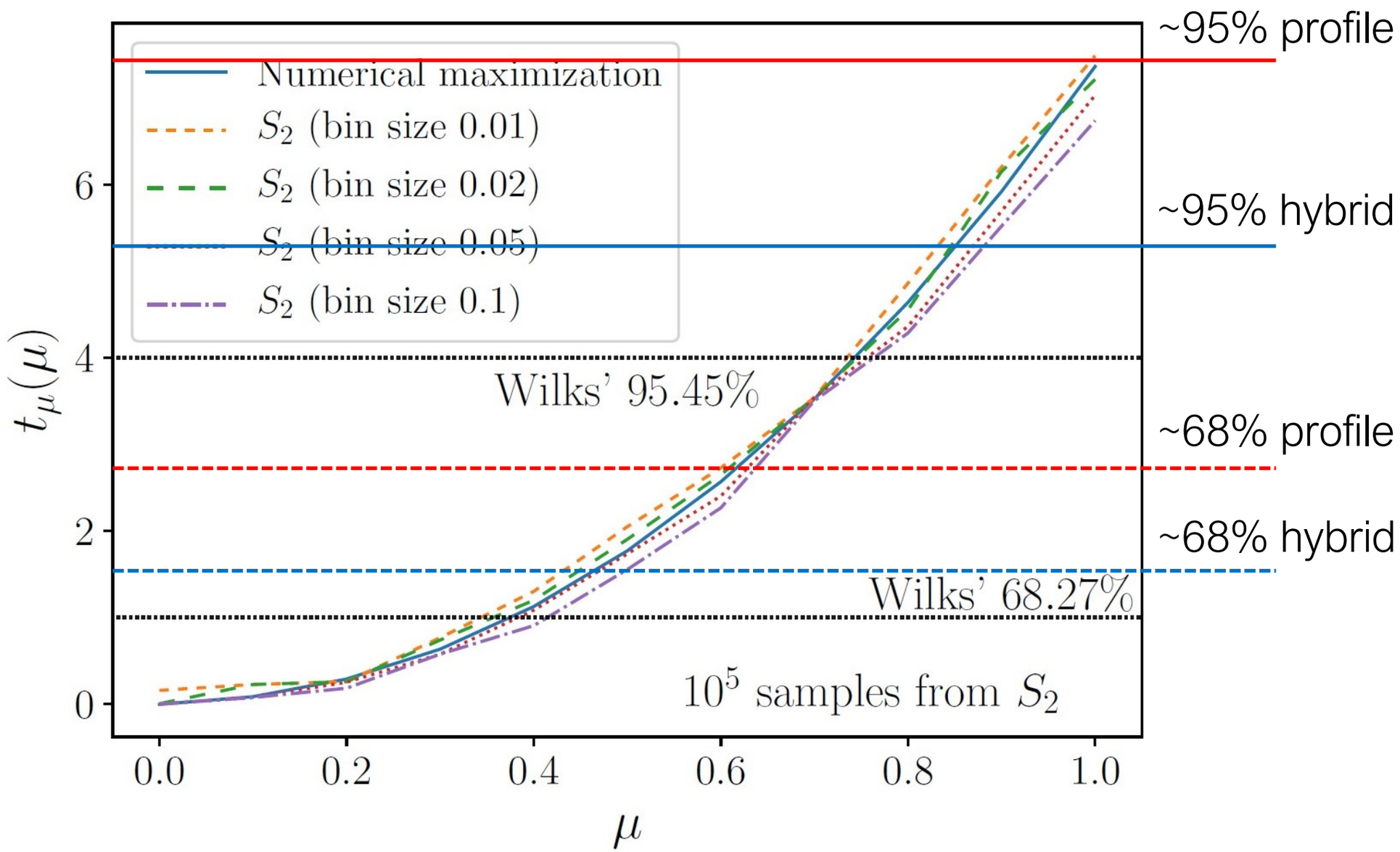
$$t_\mu(\mu) = -2\log\frac{\mathcal{L}_{\text{prof}}(\mu)}{\mathcal{L}_{\max}}$$



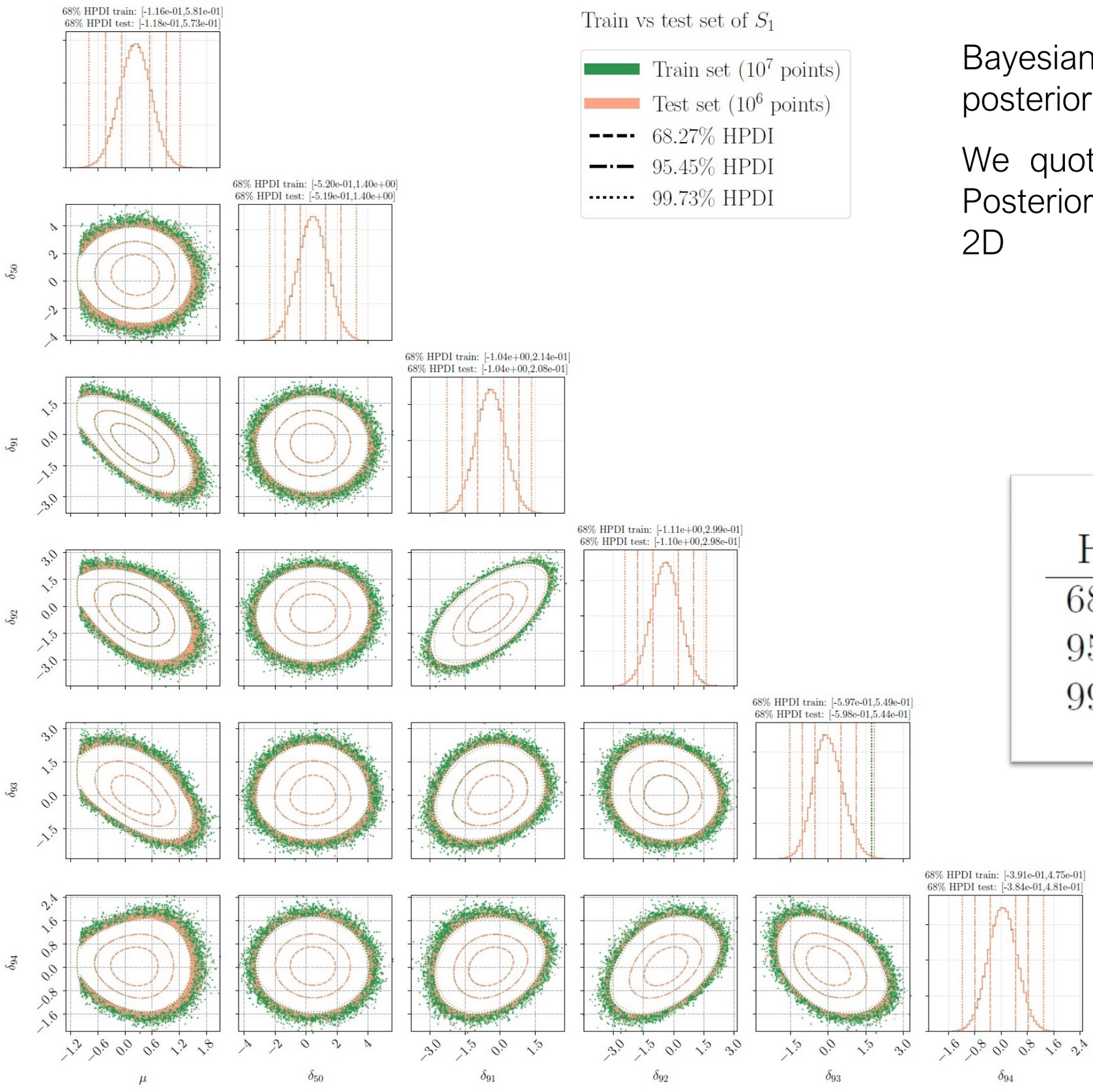
# Inference with analysis likelihood: Frequentist

For frequentist inference we construct the test statistics

$$t_\mu(\mu) = -2\log\frac{\mathcal{L}_{\text{prof}}(\mu)}{\mathcal{L}_{\max}}$$



# Inference with analysis likelihood: Bayesian




Bayesian inference is based on (marginal) posterior probability distributions

We quote results as marginalized Highest Posterior Density Intervals (HPDI) in 1D and 2D

| HPDI | $\mu > -1$ | $\mu > 0$ |
|---|---|---|
| 68.27% | $[-0.12, 0.58]$ | 0.48 |
| 95.45% | $[-0.47, 0.92]$ | 0.86 |
| 99.73% | $[-0.82, 1.26]$ | 1.22 |

# The DNNLikelihood

We set up a **supervised learning** problem for interpolation (regression) and build and train a simple multilayer perceptron (MLP) model with Keras+TensorFlow with the following procedure:

- From our sampling we know the LF (our output variable $y$) for several different values of the input parameters (parameter of interest $\mu$ and nuisance parameters $\boldsymbol{\theta}$, collectively denoted by the input vector $\mathbf{x}$).
- The nuisance parameters have already been standardized in the analysis, so that their distribution is Normal with zero mean and unit variance (they do not need preprocessing)
- The values of the LF vary largely, by almost 100 orders of magnitude, and it is better to formulate the learning problem in terms of the log-LF
- The log-LF still varies between ~-380 and ~-285, which is not a good range for a supervised learning problem, so we standardized the output variable (log-LF)
- Now the training dataset is constituted by a set of standardized input variables (parameters) with an associated output variable (scaled log-LF)
- When doing inference, the output values of the DNN are scaled back to their initial distribution
- Notice that our sampling arises from the analytic knowledge of the LF and, as such, does not contain any stochastic “noise”
- For this reason, our problem constitutes an interpolation more than a regression (generally, overfitting is not possible)

# The DNNLikelihood

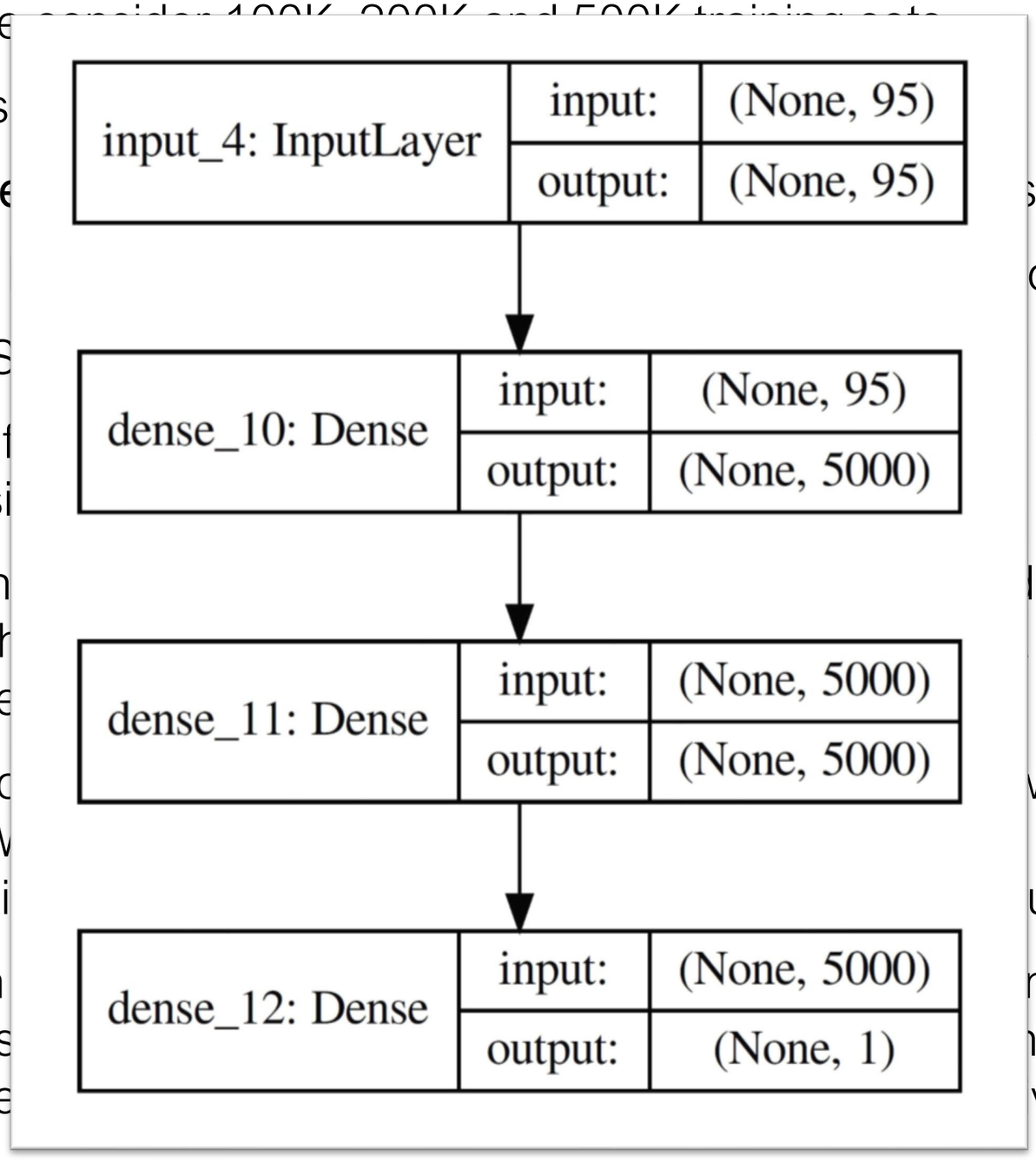
Optimization of the MLP has been carried out on the following hyperparameters:

- **Size of training set:** we consider 100K, 200K,and 500K training sets
- **Loss function:** mean squared error (mse)
- **Number of hidden layers:** usually 2 are enough, more complex problems can need more layers
- **Number of nodes per layer:** we consider 500, 1000, 2000, and 5000 nodes per layer
- **Activation functions:** Scaled Exponential Linear Unit (SELU)
- **Batch size:** we keep fixed the number of batches to around 200, and therefore vary batch size with training sample size: 512, 1024, 2048
- **Optimizer:** Adam with starting Learning Rate 0.001. Learning rate is decreased by a factor 0.2 every 40 epochs with no improvement in the validation loss within a tolerance of 1/N with N number of training events
- **Regularizer:** we do not need regularization! We cannot overfit, since we are doing interpolation and not regression. We just use early stopping to shorten training time. We stop after 48 epochs with no improvement in validation loss within a tolerance of 1/N with N number of training events
- **Ensembling:** for each architecture we train 5 identical models with randomly extracted training sets and take the best one to show results. We have experimented ensemble techniques, such as stacking, which are very promising but were not needed in this “relatively simple” case

# The DNNLikelihood

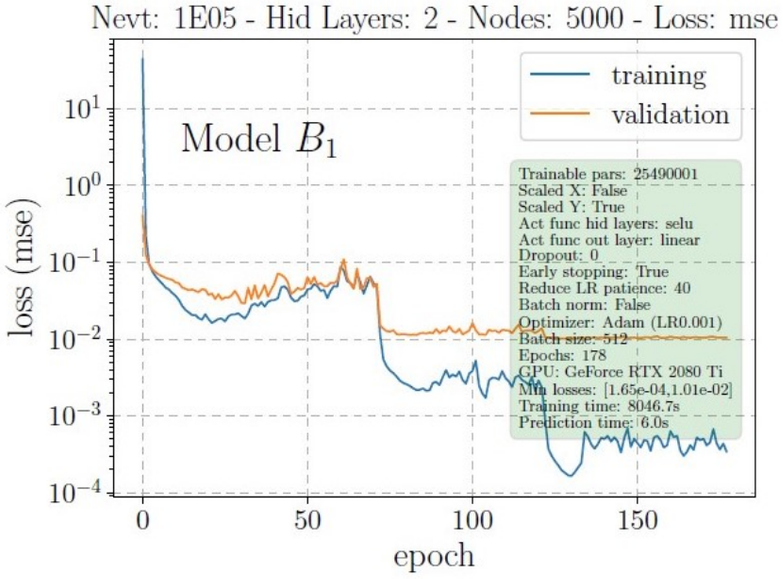
Optimization of the MLP has been carried out on the following hyperparameters:

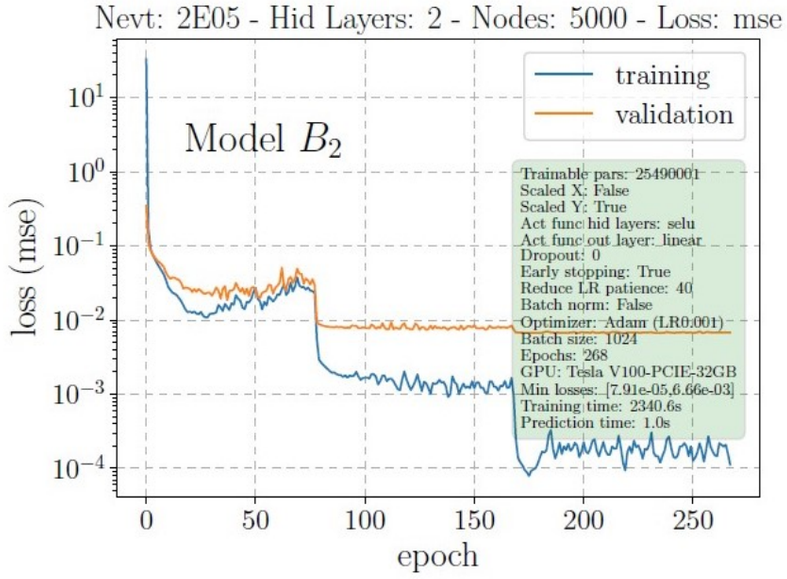
- **Size of training set:** we
- **Loss function:** mean s
- **Number of hidden laye** s can need more layers
- **Number of nodes per** des per layer
- **Activation functions:** S
- **Batch size:** we keep f therefore vary batch size with training sample si
- **Optimizer:** Adam with lecreased by a factor 0.2 every 40 epochs with tolerance of 1/N with N number of training eve
- **Regularizer:** we do nc ve are doing interpolation and not regression. W We stop after 48 epochs with no improvement i umber of training events
- **Ensembling:** for each ndomly extracted training sets and take the bes nsemble techniques, such as stacking, which are vely simple" case

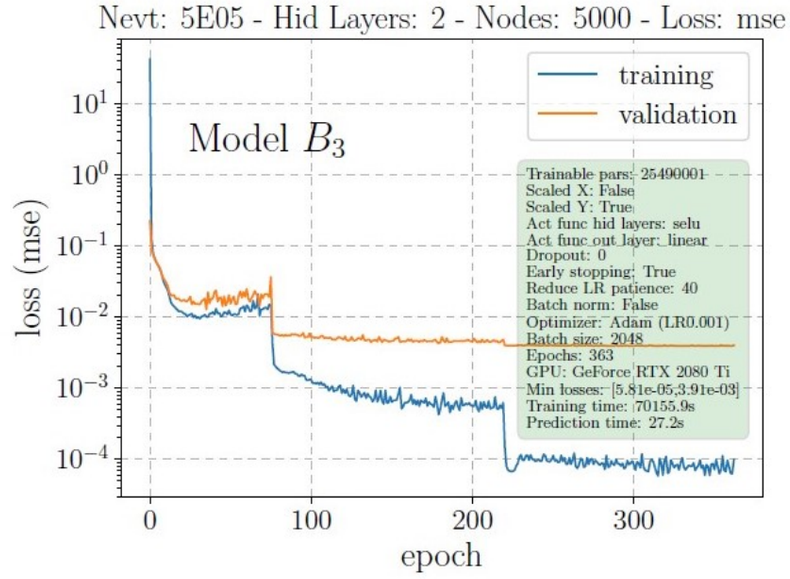


# Results: Bayesian DNNLikelihood

## Train with unbiased sampling S1



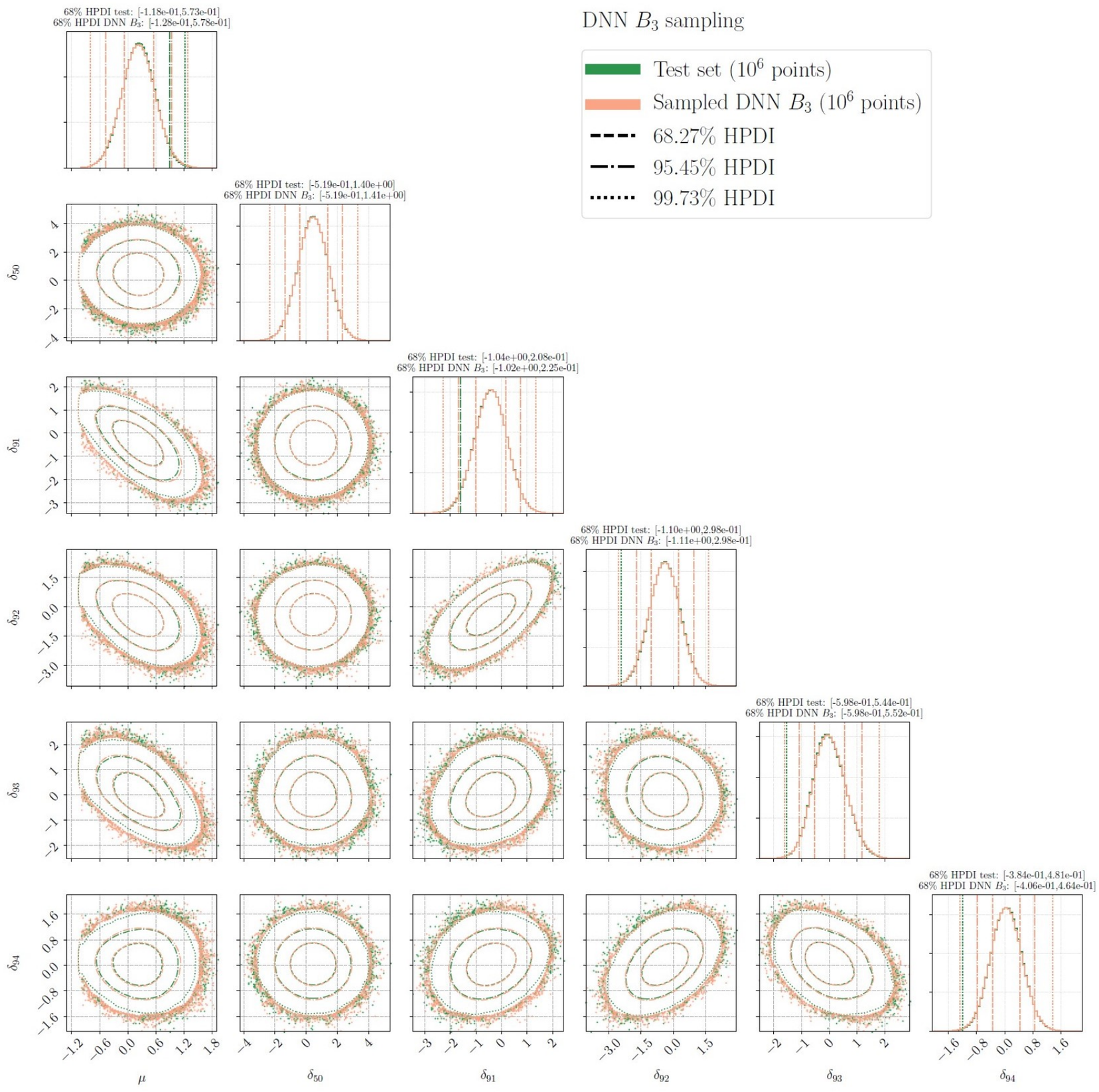

## Metrics

| Name | $B_1$ | $B_2$ | $B_3$ |
|---|---|---|---|
| Sample size ($\times 10^5$) | 1 | 2 | 5 |
| Epochs | 178 | 268 | 363 |
| Loss train (MSE) ($\times 10^{-3}$) | 0.14 | 0.088 | 0.054 |
| Loss val (MSE) ($\times 10^{-3}$) | 10.11 | 6.66 | 3.9 |
| Loss test (MSE) ($\times 10^{-3}$) | 10.02 | 6.64 | 3.9 |
| ME train ($\times 10^{-3}$) | 0.47 | 0.53 | 0.28 |
| ME val ($\times 10^{-3}$) | 5.44 | 2.58 | 1.76 |
| ME test ($\times 10^{-3}$) | 4.91 | 2.31 | 1.72 |
| Median $p$-value of 1D K-S test vs pred. on train | 0.41 | 0.46 | 0.39 |
| Median $p$-value of 1D K-S test vs. pred. on val. | 0.24 | 0.33 | 0.43 |
| Median $p$-value of 1D K-S val vs. pred. on test | 0.24 | 0.40 | 0.34 |
| Training time (s) | 1007 | 2341 | 8446 |
| Prediction time ($\mu$s/point) | 11.5 | 10.4 | 14.5 |

## Results

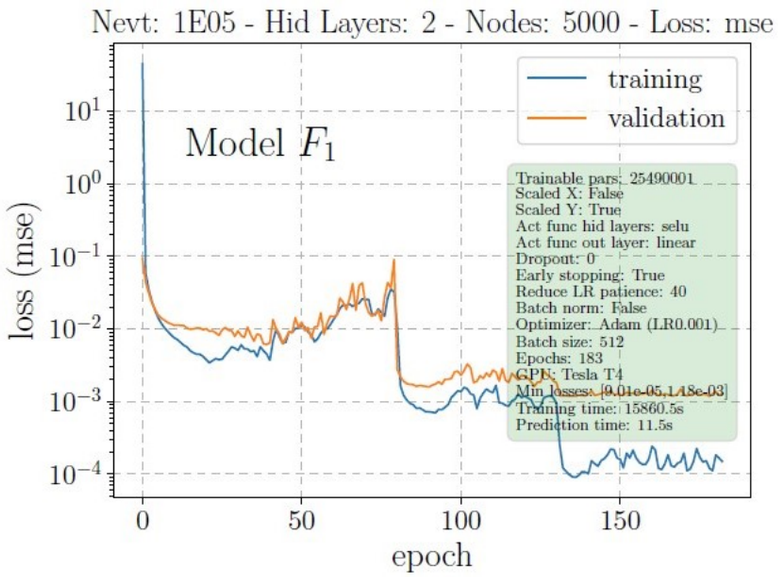
| HPDI | $\mu > -1$ | $\mu > 0$ | $B_1$ | $B_2$ | $B_3$ |
|---|---|---|---|---|---|
| 68.27% | $[-0.12, 0.58]$ | 0.48 | 0.49 | 0.49 | 0.49 |
| 95.45% | $[-0.47, 0.92]$ | 0.86 | 0.92 | 0.91 | 0.88 |
| 99.73% | $[-0.82, 1.26]$ | 1.22 | 1.35 | 1.34 | 1.29 |

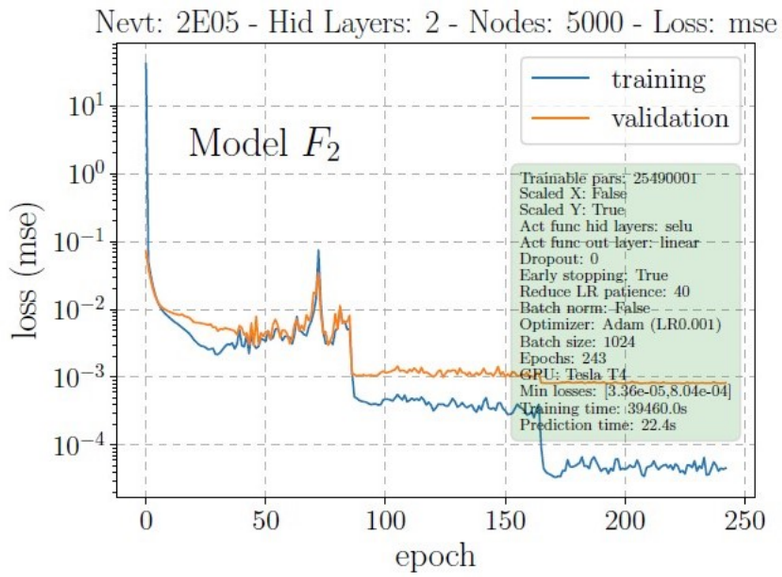
# Results: Bayesian DNNLikelihood

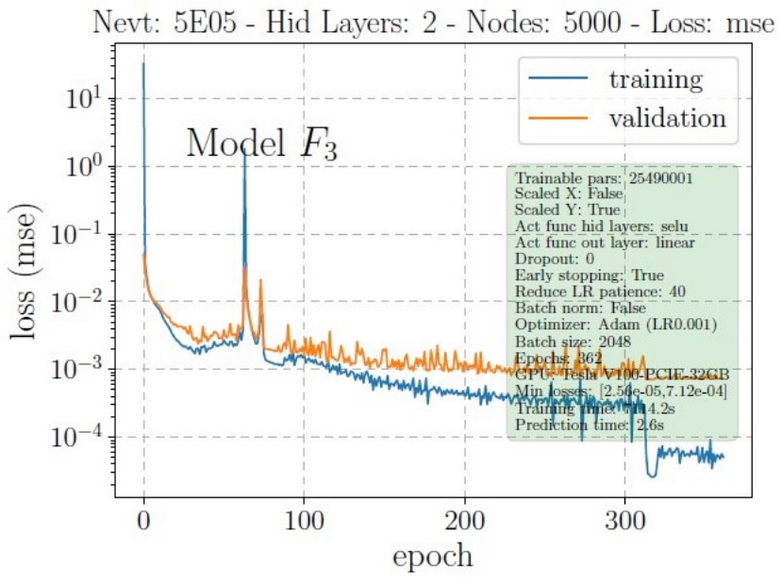


# Results: full DNNLikelihood

## Train with mixed sampling S3




### Metrics

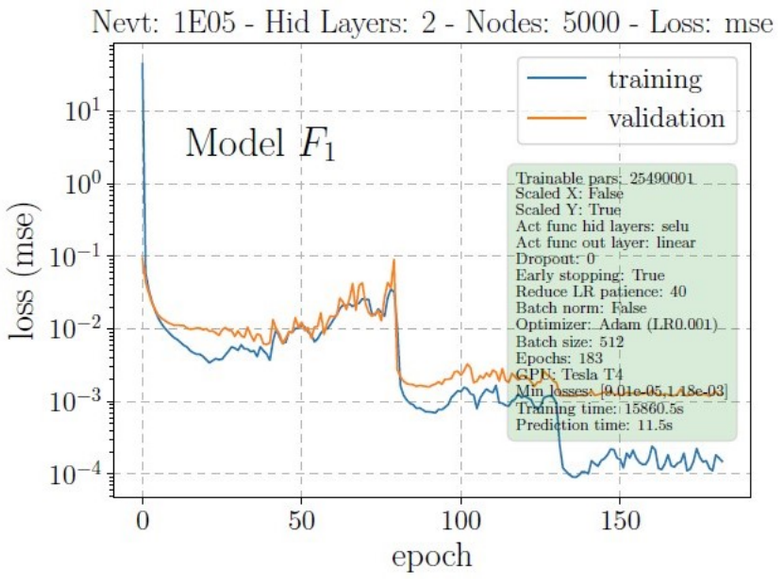
| Name | $F_1$ | $F_2$ | $F_3$ |
|---|---|---|---|
| Sample size ($\times 10^5$) | 1 | 2 | 5 |
| Epochs | 183 | 243 | 362 |
| Loss train (MSE) ($\times 10^{-3}$) | 0.092 | 0.026 | 0.030 |
| Loss val (MSE) ($\times 10^{-3}$) | 1.18 | 0.80 | 0.71 |
| Loss test (MSE) ($\times 10^{-3}$) | 1.17 | 0.80 | 0.72 |
| ME train ($\times 10^{-3}$) | 3.07 | 0.47 | 1.1 |
| ME val ($\times 10^{-3}$) | 1.78 | 0.87 | 0.82 |
| ME test ($\times 10^{-3}$) | 1.50 | 0.68 | 0.86 |
| Median $p$-value of 1D K-S test/pred-train | 0.53 | 0.48 | 0.44 |
| Median $p$-value of 1D K-S test/pred-val | 0.15 | 0.27 | 0.20 |
| Median $p$-value of 1D K-S val/pred-test | 0.13 | 0.31 | 0.33 |
| Mean error on $t_\mu(\mu)$ | 0.11 | 0.12 | 0.032 |
| Training time (s) | 1236 | 2819 | 7114 |
| Prediction time ($\mu$s/point) | 11.1 | 10.8 | 10.5 |

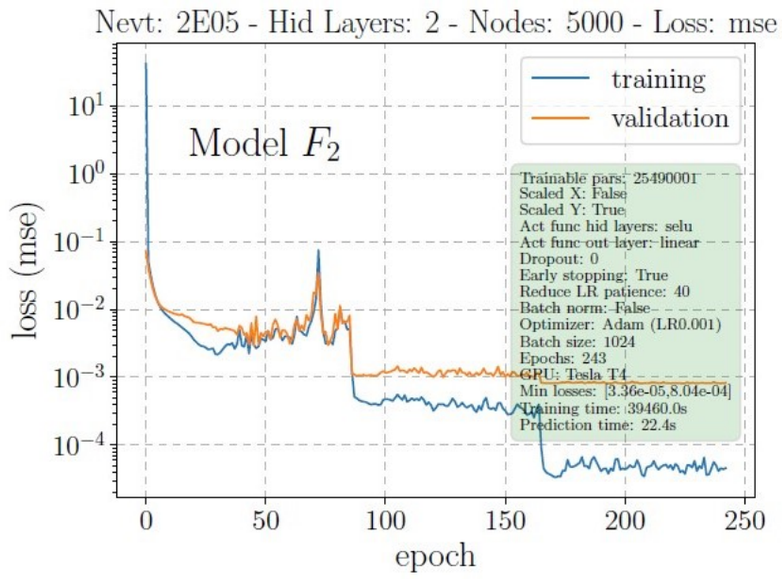
### Results

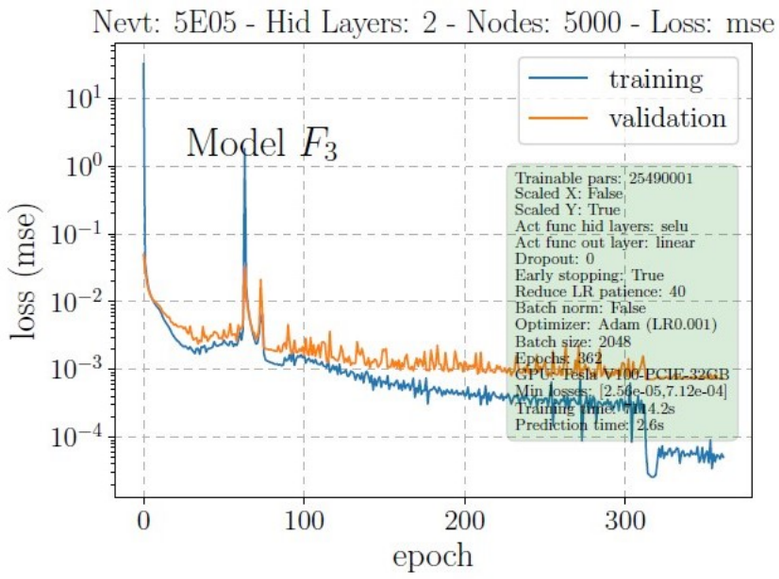
| HPDI | $\mu > -1$ | $\mu > 0$ | $F_1$ | $F_2$ | $F_3$ |
|---|---|---|---|---|---|
| 68.27% | $[-0.12, 0.58]$ | 0.48 | 0.50 | 0.50 | 0.48 |
| 95.45% | $[-0.47, 0.92]$ | 0.86 | 0.93 | 0.93 | 0.88 |
| 99.73% | $[-0.82, 1.26]$ | 1.22 | 1.35 | 1.37 | 1.28 |

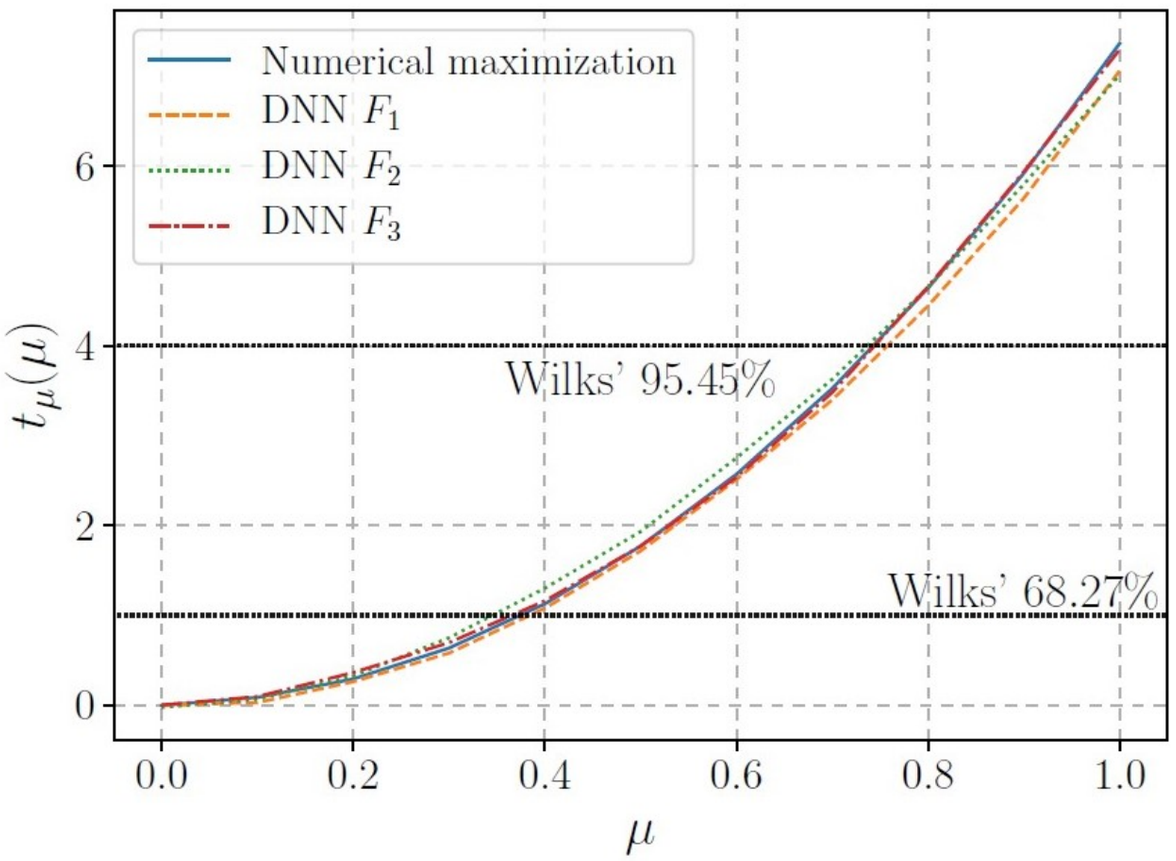
# Results: full DNNLikelihood

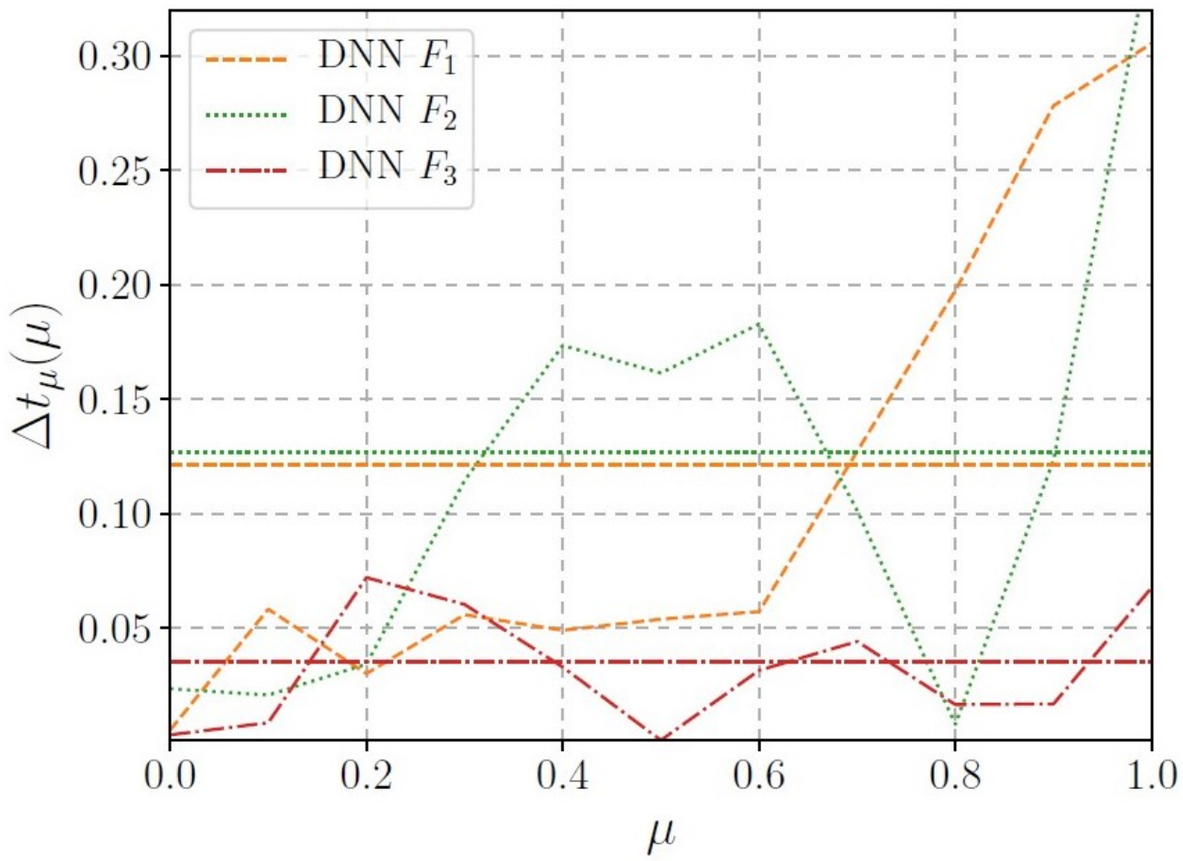
## Train with mixed sampling S3



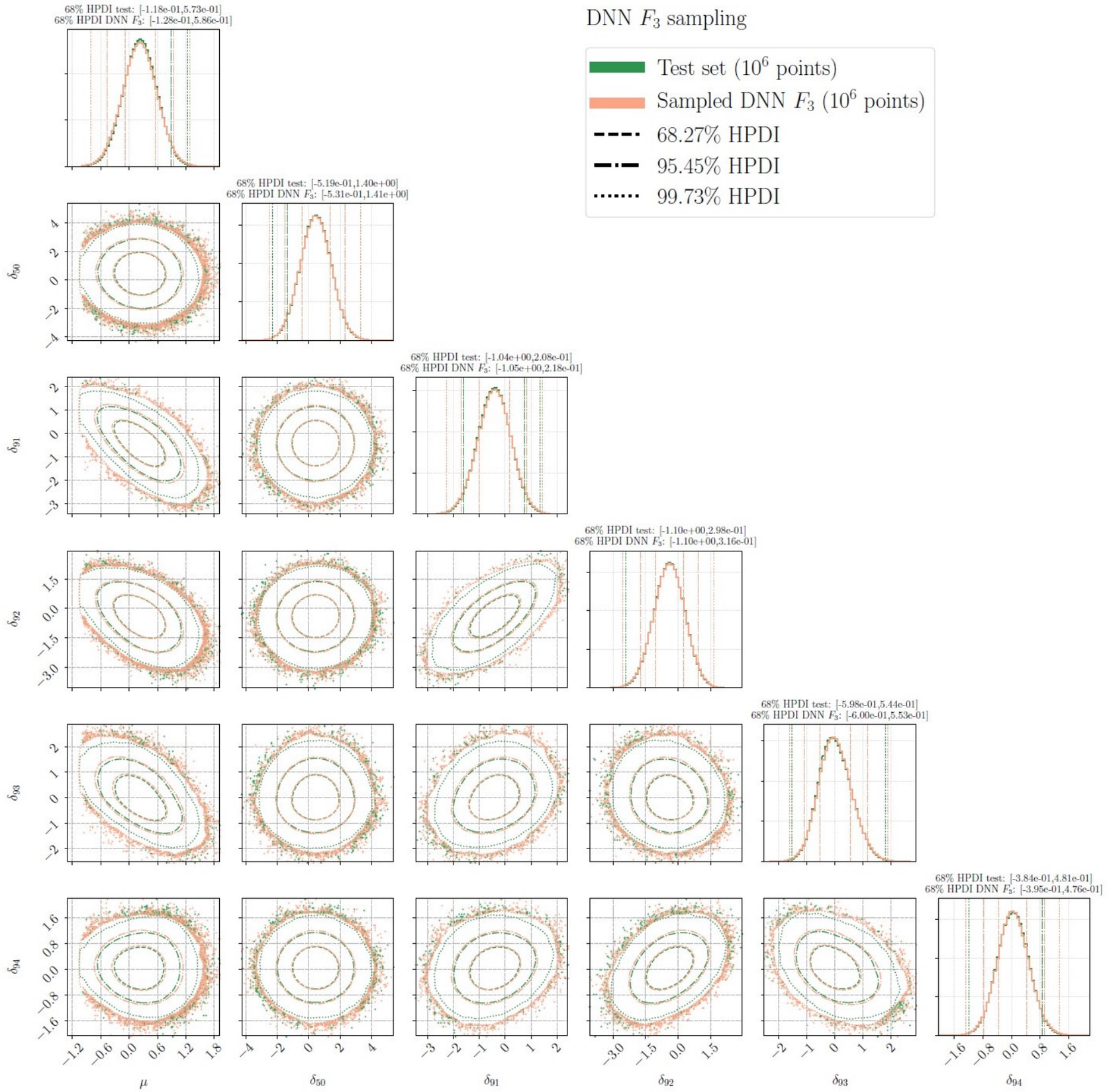



# Results: full DNNLikelihood



# Conclusions

- We introduced the DNNLikelihood, a framework to encode, distribute, combine, and analyze likelihood functions
- In the realistic example we studied it seems to work extremely well without the need of too much hyperparameters tuning or advanced techniques (which may be necessary for very complicated multimodal functions)
- All code used to produce the paper and all results are available on GitHub
- Together with the models our code always produces many auxiliary files keeping track of all parameters, metrics, results, etc. so that each model is carefully self documented
- A comprehensive Python module that allows to sample likelihoods, build models (and ensembles of models), optimize, and analyze the results within different statistical frameworks is in the last stage of development. It can be found on GitHub and is documented here.
- Real world examples are under study (ATLAS HistFactory Likelihoods, HepFit, Drell-Yan W/Y fit, etc.)
- A standardized procedure to preserve all DNNLikelihood and auxiliary material in the Zenodo database is also being developed

# THANK YOU!