Reliable Likelihood Ratios for Neural Simulation-Based Inference in HEP

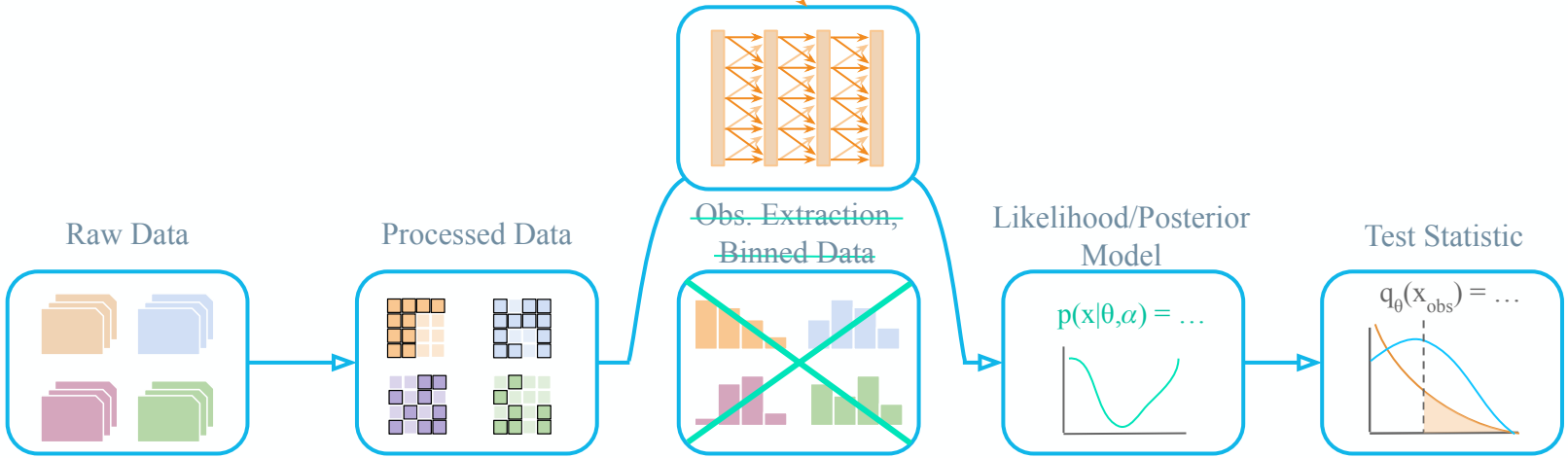
Nino Kovačić*, **Stephen Jiggins** & the NEEDLE team

*nkovacic.phy@pmf.hr , https://needle-sbi.github.io

ML4JETS 2026 Vienna

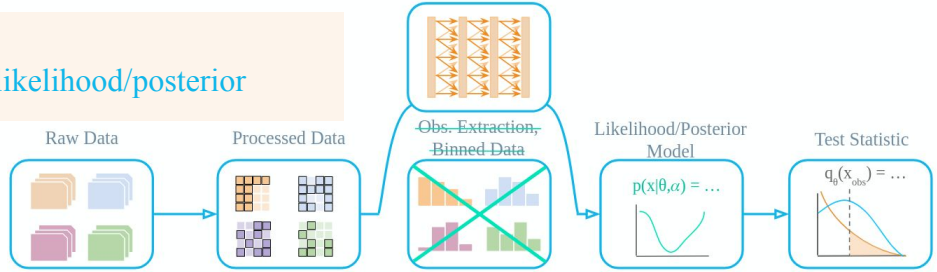
A brief introduction to NSBI and NLRE

- **What is Neural Simulation-Based Inference?**[1]
  - The core idea is to utilize NNs as estimators (surrogates) for the likelihood/posterior
  - How does that affect a typical workflow? (simplifying greatly)

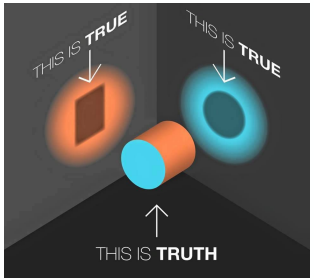


A brief introduction to NSBI and NLRE

- **What is Neural Simulation-Based Inference?[1]**
  - The core idea is to utilize NNs as estimators (surrogates) for the likelihood/posterior



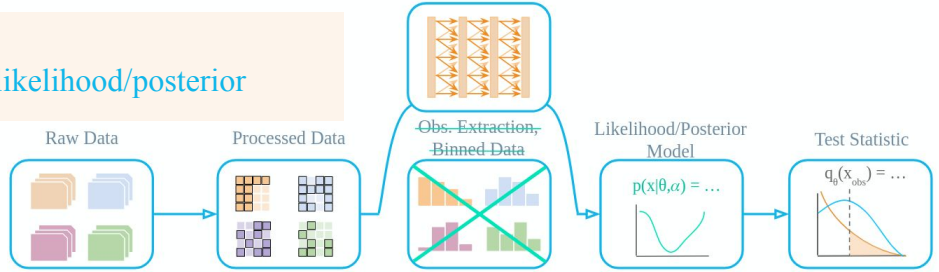
- **What does NSBI offer?**
  - Main point: increase in statistical power:
    - According to the Neyman–Pearson lemma, the likelihood ratio is the most powerful* test statistic (and that is what NSBI aims to obtain)
    - No dimensionality reduction of datasets required
    - No averaging of information across bins**



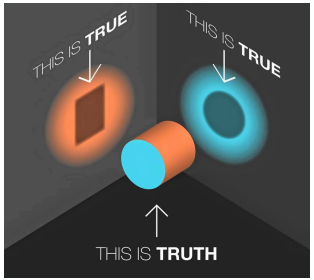
*Strictly speaking, we should be talking about sufficiency, this has been simplified in the interest of time **If performing unbinned fits

A brief introduction to NSBI and NLRE

- **What is Neural Simulation-Based Inference?[1]**
  - The core idea is to utilize NNs as estimators (surrogates) for the likelihood/posterior



- **What does NSBI offer?**
  - Main point: increase in statistical power:
    - According to the Neyman–Pearson lemma, the likelihood ratio is the most powerful* test statistic (and that is what NSBI aims to obtain)
    - No dimensionality reduction of datasets required
    - No averaging of information across bins**



- **What is Neural Likelihood Ratio Estimation (NLRE)?**
  - A flavor of NSBI where the probability ratios are estimated via NNs and used to form the likelihood model
  - In this talk we will refer to NLRE interchangeably with Calibrated Classifiers (CARL)[2] method
  - Here, NN binary classification scores are leveraged to extract probability ratios

*Strictly speaking, we should be talking about sufficiency, this has been simplified in the interest of time **If performing unbinned fits

NLRE in High Energy Physics

- **Why is NLRE well-suited for HEP applications:**
  - Easily constructed likelihoods for HEP inference problems
    - E.g. combining multiple classifiers to obtain a composite likelihood for EFT-style mixture fits

$$\mathcal{L}\left(\vec{\theta}\,|x_i\right)\frac{1}{\text{const.}} = p(x_i|\vec{\theta})\frac{1}{p_{\text{ref}}(x_i)} = \sum_{s\in\{S\}}\nu_s f_s(\vec{\theta})\frac{p_s(x_i|\vec{\theta})}{p_{\text{ref}}(x_i)} + \sum_{b\in\{B\}}\nu_s g_b(\vec{\theta})\frac{p_b(x_i|\vec{\theta})}{p_{\text{ref}}(x_i)} + \sum_{\tilde{s}\in\{\tilde{S}\}}\sum_{n}\nu_{\tilde{s}} I_{\tilde{s}n}(\vec{\theta})\frac{p_n(x_i|\vec{\theta})}{p_{\text{ref}}(x_i)}$$

NLRE in High Energy Physics

- **Why is NLRE well-suited for HEP applications:**
  - Easily constructed likelihoods for HEP inference problems
    - E.g. combining multiple classifiers to obtain a composite likelihood for EFT-style mixture fits
  - Well behaved training and an "easy" training objective:
    - Unlike NLE/NPE methods, binary classifiers do not need to learn a true density
      - Thus, there is no need to learn normalization which is the hardest component of density estimation

$$\mathcal{L}\left(\vec{\theta}\,|x_i\right)\frac{1}{\text{const.}} = p(x_i|\vec{\theta})\frac{1}{p_{\text{ref}}(x_i)} = \sum_{s\in\{S\}}\nu_s f_s(\vec{\theta})\frac{p_s(x_i|\vec{\theta})}{p_{\text{ref}}(x_i)} + \sum_{b\in\{B\}}\nu_s g_b(\vec{\theta})\frac{p_b(x_i|\vec{\theta})}{p_{\text{ref}}(x_i)} + \sum_{\tilde{s}\in\{\tilde{S}\}}\sum_{n}\nu_{\tilde{s}} I_{\tilde{s}n}(\vec{\theta})\cdot\frac{p_n(x_i|\vec{\theta})}{p_{\text{ref}}(x_i)}$$

NLRE in High Energy Physics

- **Why is NLRE well-suited for HEP applications:**
  - Easily constructed likelihoods for HEP inference problems
    - E.g. combining multiple classifiers to obtain a composite likelihood for EFT-style mixture fits
  - Well behaved training and an "easy" training objective:
    - Unlike NLE/NPE methods, binary classifiers do not need to learn a true density
      - Thus, there is no need to learn normalization which is the hardest component of density estimation
      - Practically, NLRE is "promoted" to NLE by the the common reference factoring out of the likelihood or its exact cancellation in profile likelihood fits

$$\mathcal{L}\left(\vec{\theta}\,|x_i\right)\frac{1}{\text{const.}} = p(x_i|\vec{\theta})\frac{1}{p_{\text{ref}}(x_i)} = \sum_{s\in\{S\}}\nu_s f_s(\vec{\theta})\frac{p_s(x_i|\vec{\theta})}{p_{\text{ref}}(x_i)} + \sum_{b\in\{B\}}\nu_s g_b(\vec{\theta})\frac{p_b(x_i|\vec{\theta})}{p_{\text{ref}}(x_i)} + \sum_{\tilde{s}\in\{\tilde{S}\}}\sum_{n}\nu_{\tilde{s}} I_{\tilde{s}n}(\vec{\theta})\frac{p_n(x_i|\vec{\theta})}{p_{\text{ref}}(x_i)}$$

NLRE in High Energy Physics

- **Why is NLRE well-suited for HEP applications:**
  - Easily constructed likelihoods for HEP inference problems
  - Well behaved training and an “easy” training objective
- **What are the challenges with NLRE?**

$$\mathcal{L}\left(\vec{\theta}\,|x_i\right)\frac{1}{\text{const.}} = p(x_i|\vec{\theta})\frac{1}{p_{\text{ref}}(x_i)} = \sum_{s\in\{S\}}\nu_s f_s(\vec{\theta})\frac{p_s(x_i|\vec{\theta})}{p_{\text{ref}}(x_i)} + \sum_{b\in\{B\}}\nu_s g_b(\vec{\theta})\frac{p_b(x_i|\vec{\theta})}{p_{\text{ref}}(x_i)} + \sum_{\tilde{s}\in\{\tilde{S}\}}\sum_{n}\nu_{\tilde{s}} I_{\tilde{s}n}(\vec{\theta})\cdot\frac{p_n(x_i|\vec{\theta})}{p_{\text{ref}}(x_i)}$$

NLRE in High Energy Physics

- **Why is NLRE well-suited for HEP applications:**
  - Easily constructed likelihoods for HEP inference problems
  - Well behaved training and an “easy” training objective
- **What are the challenges with NLRE?**
  - Exposed to any issues in estimator quality:
    - Derived ratios need to be of very high quality in order to perform an unbinned fit
      - Composite likelihoods mean that the errors compound!

$$\mathcal{L}\left(\vec{\theta}\,|x_i\right)\frac{1}{\text{const.}} = p(x_i|\vec{\theta})\frac{1}{p_{\text{ref}}(x_i)} = \sum_{s\in\{S\}}\nu_s f_s(\vec{\theta})\frac{p_s(x_i|\vec{\theta})}{p_{\text{ref}}(x_i)} + \sum_{b\in\{B\}}\nu_s g_b(\vec{\theta})\frac{p_b(x_i|\vec{\theta})}{p_{\text{ref}}(x_i)} + \sum_{\tilde{s}\in\{\tilde{S}\}}\sum_{n}\nu_{\tilde{s}} I_{\tilde{s}n}(\vec{\theta})\cdot\frac{p_n(x_i|\vec{\theta})}{p_{\text{ref}}(x_i)}$$

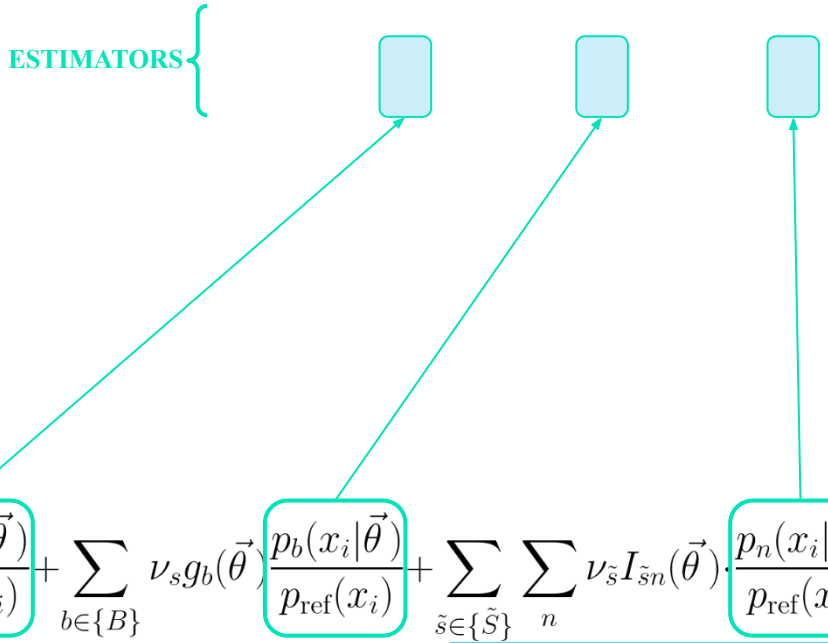
NLRE in High Energy Physics

- **Why is NLRE well-suited for HEP applications:**
  - Easily constructed likelihoods for HEP inference problems
  - Well behaved training and an “easy” training objective
- **What are the challenges with NLRE?**
  - Exposed to any issues in estimator quality:
    - Derived ratios need to be of very high quality in order to perform an unbinned fit
      - Composite likelihoods mean that the errors compound!
  - Handling training and bookkeeping:
    - Training and inference still need to happen on very large scales (of HEP datasets)

$$\mathcal{L}\left(\vec{\theta}\,|x_i\right)\frac{1}{\text{const.}} = p(x_i|\vec{\theta})\frac{1}{p_{\text{ref}}(x_i)} = \sum_{s\in\{S\}}\nu_s f_s(\vec{\theta})\frac{p_s(x_i|\vec{\theta})}{p_{\text{ref}}(x_i)} + \sum_{b\in\{B\}}\nu_s g_b(\vec{\theta})\frac{p_b(x_i|\vec{\theta})}{p_{\text{ref}}(x_i)} + \sum_{\tilde{s}\in\{\tilde{S}\}}\sum_{n}\nu_{\tilde{s}} I_{\tilde{s}n}(\vec{\theta})\cdot\frac{p_n(x_i|\vec{\theta})}{p_{\text{ref}}(x_i)}$$

NLRE in High Energy Physics

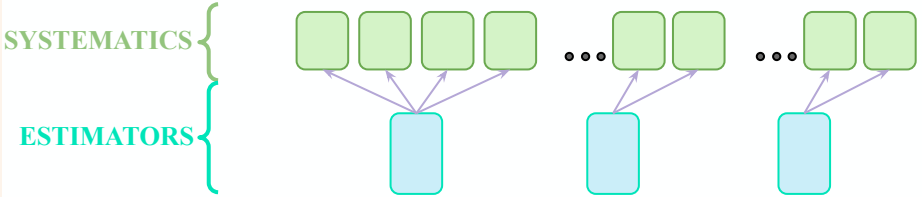
- **Why is NLRE well-suited for HEP applications:**
  - Easily constructed likelihoods for HEP inference problems
  - Well behaved training and an “easy” training objective
- **What are the challenges with NLRE?**
  - Exposed to any issues in estimator quality:
    - Derived ratios need to be of very high quality in order to perform an unbinned fit
      - Composite likelihoods mean that the errors compound!
  - Handling training and bookkeeping:
    - Training and inference still need to happen on very large scales (of HEP datasets)
    - We need to train multiple estimators



$$\mathcal{L}\left(\vec{\theta}\,|x_i\right)\frac{1}{\text{const.}} = p(x_i|\vec{\theta})\frac{1}{p_{\text{ref}}(x_i)} = \sum_{s\in\{S\}}\nu_s f_s(\vec{\theta})\frac{p_s(x_i|\vec{\theta})}{p_{\text{ref}}(x_i)} + \sum_{b\in\{B\}}\nu_s g_b(\vec{\theta})\frac{p_b(x_i|\vec{\theta})}{p_{\text{ref}}(x_i)} + \sum_{\tilde{s}\in\{\tilde{S}\}}\sum_{n}\nu_{\tilde{s}} I_{\tilde{s}n}(\vec{\theta})\frac{p_n(x_i|\vec{\theta})}{p_{\text{ref}}(x_i)}$$

NLRE in High Energy Physics

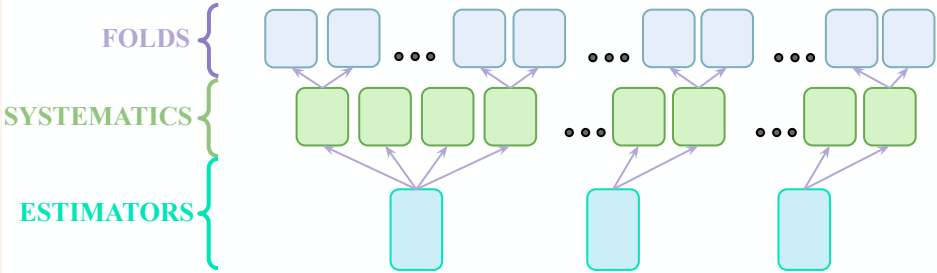
- **Why is NLRE well-suited for HEP applications:**
  - Easily constructed likelihoods for HEP inference problems
  - Well behaved training and an “easy” training objective
- **What are the challenges with NLRE?**
  - Exposed to any issues in estimator quality:
    - Derived ratios need to be of very high quality in order to perform an unbinned fit
      - Composite likelihoods mean that the errors compound!
  - Handling training and bookkeeping:
    - Training and inference still need to happen on very large scales (of HEP datasets)
    - We need to train multiple estimators, learn the systematics



$$\mathcal{L}\left(\vec{\theta}\,|x_i\right)\frac{1}{\text{const.}} = p(x_i|\vec{\theta})\frac{1}{p_{\text{ref}}(x_i)} = \sum_{s\in\{S\}}\nu_s f_s(\vec{\theta})\frac{p_s(x_i|\vec{\theta})}{p_{\text{ref}}(x_i)} + \sum_{b\in\{B\}}\nu_s g_b(\vec{\theta})\frac{p_b(x_i|\vec{\theta})}{p_{\text{ref}}(x_i)} + \sum_{\tilde{s}\in\{\tilde{S}\}}\sum_{n}\nu_{\tilde{s}} I_{\tilde{s}n}(\vec{\theta})\frac{p_n(x_i|\vec{\theta})}{p_{\text{ref}}(x_i)}$$

NLRE in High Energy Physics

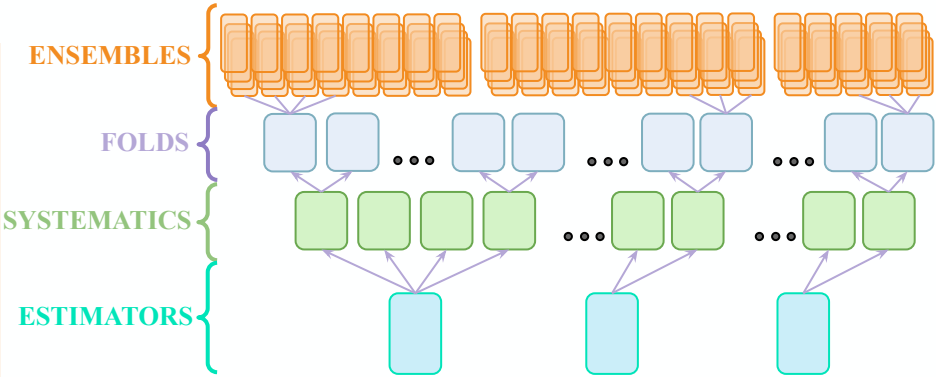
- **Why is NLRE well-suited for HEP applications:**
  - Easily constructed likelihoods for HEP inference problems
  - Well behaved training and an “easy” training objective
- **What are the challenges with NLRE?**
  - Exposed to any issues in estimator quality:
    - Derived ratios need to be of very high quality in order to perform an unbinned fit
      - Composite likelihoods mean that the errors compound!
  - Handling training and bookkeeping:
    - Training and inference still need to happen on very large scales (of HEP datasets)
    - We need to train multiple estimators, learn the systematics, train via k-fold cross-validation



$$\mathcal{L}\left(\vec{\theta}\,|x_i\right)\frac{1}{\text{const.}} = p(x_i|\vec{\theta})\frac{1}{p_{\text{ref}}(x_i)} = \sum_{s\in\{S\}}\nu_s f_s(\vec{\theta})\frac{p_s(x_i|\vec{\theta})}{p_{\text{ref}}(x_i)} + \sum_{b\in\{B\}}\nu_s g_b(\vec{\theta})\frac{p_b(x_i|\vec{\theta})}{p_{\text{ref}}(x_i)} + \sum_{\tilde{s}\in\{\tilde{S}\}}\sum_{n}\nu_{\tilde{s}} I_{\tilde{s}n}(\vec{\theta})\frac{p_n(x_i|\vec{\theta})}{p_{\text{ref}}(x_i)}$$

NLRE in High Energy Physics

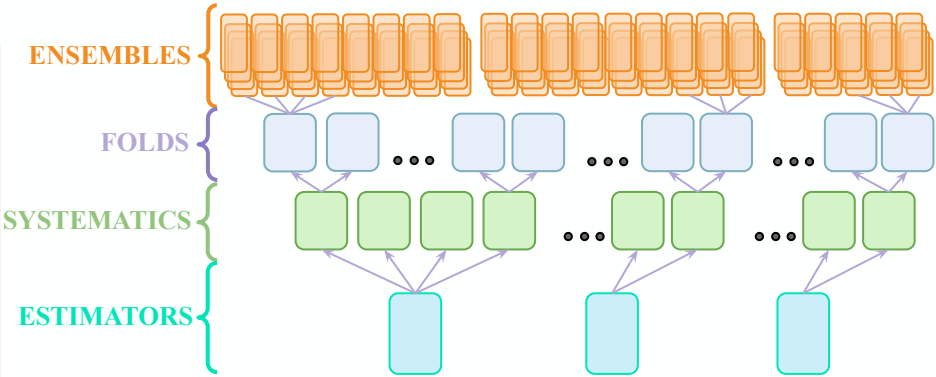
- **Why is NLRE well-suited for HEP applications:**
  - Easily constructed likelihoods for HEP inference problems
  - Well behaved training and an “easy” training objective
- **What are the challenges with NLRE?**
  - Exposed to any issues in estimator quality:
    - Derived ratios need to be of very high quality in order to perform an unbinned fit
      - Composite likelihoods mean that the errors compound!
  - Handling training and bookkeeping:
    - Training and inference still need to happen on very large scales (of HEP datasets)
    - We need to train multiple estimators, learn the systematics, train via k-fold cross-validation, and ensemble



$$\mathcal{L}\left(\vec{\theta}\,|x_i\right)\frac{1}{\text{const.}} = p(x_i|\vec{\theta})\frac{1}{p_{\text{ref}}(x_i)} = \sum_{s\in\{S\}}\nu_s f_s(\vec{\theta})\frac{p_s(x_i|\vec{\theta})}{p_{\text{ref}}(x_i)} + \sum_{b\in\{B\}}\nu_s g_b(\vec{\theta})\frac{p_b(x_i|\vec{\theta})}{p_{\text{ref}}(x_i)} + \sum_{\tilde{s}\in\{\tilde{S}\}}\sum_{n}\nu_{\tilde{s}} I_{\tilde{s}n}(\vec{\theta})\frac{p_n(x_i|\vec{\theta})}{p_{\text{ref}}(x_i)}$$

NLRE in High Energy Physics

- **Why is NLRE well-suited for HEP applications:**
  - Easily constructed likelihoods for HEP inference problems
  - Well behaved training and an "easy" training objective
- **What are the challenges with NLRE?**
  - Exposed to any issues in estimator quality:
    - Derived ratios need to be of very high quality in order to perform an unbinned fit
      - Composite likelihoods mean that the errors compound!
  - Handling training and bookkeeping:
    - Training and inference still need to happen on very large scales (of HEP datasets)
    - We need to train multiple estimators, learn the systematics, train via k-fold cross-validation, and ensemble
      - These costs are multiplicative, but parallelizable!

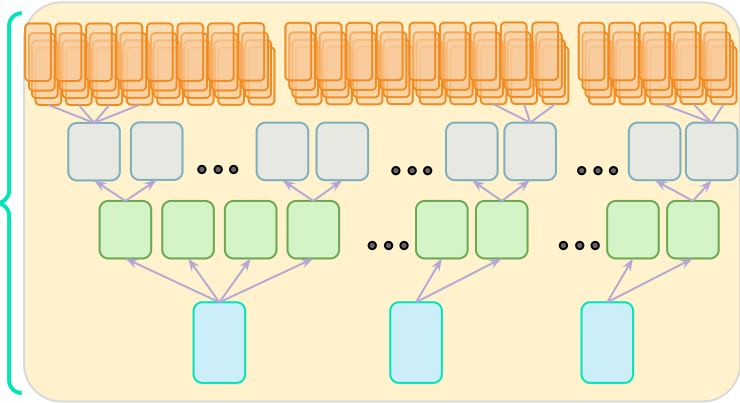


- Finally, for inference the flow through this DAG needs to "reverse" as it requires aggregation over all these NNs!

$$\mathcal{L}\left(\vec{\theta}\,|x_i\right)\frac{1}{\text{const.}} = p(x_i|\vec{\theta})\frac{1}{p_{\text{ref}}(x_i)} = \sum_{s\in\{S\}}\nu_s f_s(\vec{\theta})\frac{p_s(x_i|\vec{\theta})}{p_{\text{ref}}(x_i)} + \sum_{b\in\{B\}}\nu_s g_b(\vec{\theta})\frac{p_b(x_i|\vec{\theta})}{p_{\text{ref}}(x_i)} + \sum_{\tilde{s}\in\{\tilde{S}\}}\sum_{n}\nu_{\tilde{s}} I_{\tilde{s}n}(\vec{\theta})\frac{p_n(x_i|\vec{\theta})}{p_{\text{ref}}(x_i)}$$

needle-sbi: solving the practical deployment of NSBI in HEP

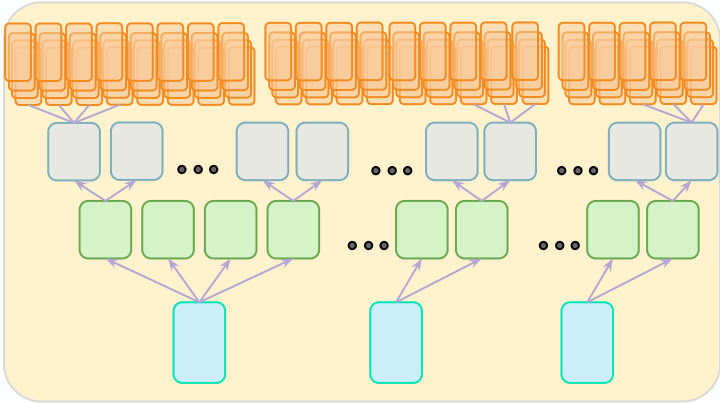
- **needle-sbi* is a workflow orchestrator for large-scale NSBI in HEP:**
  - Unified DAG execution with either LAW[3] or b2luigi[4] :
    it stitches the models together!

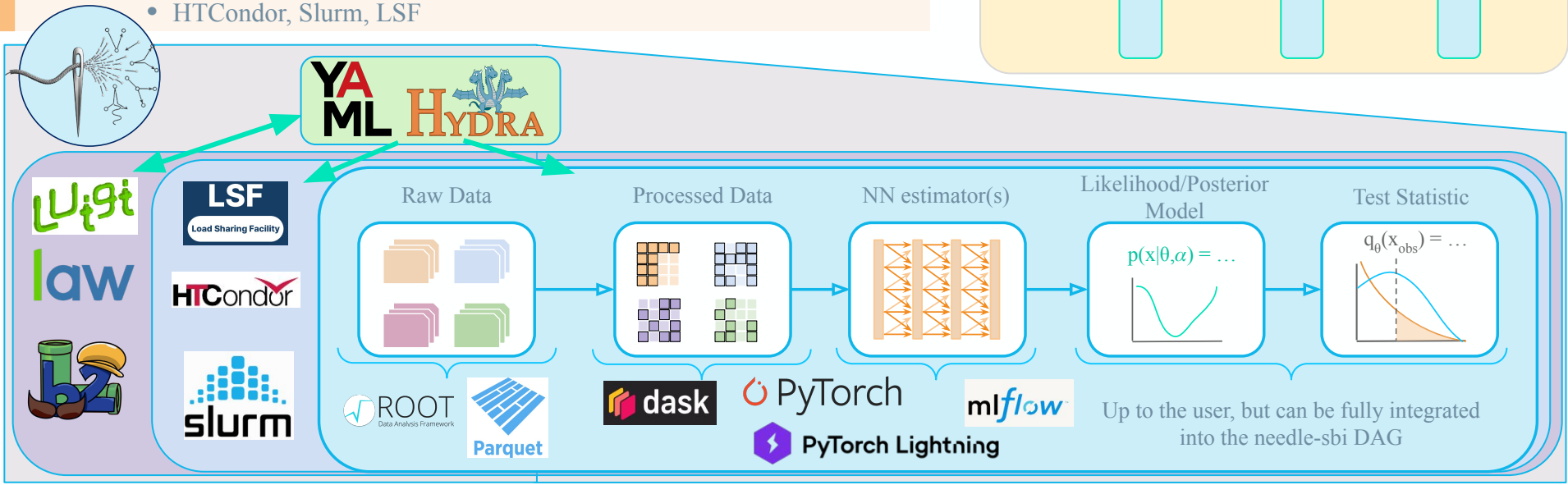


*First shown at this year's CHEP 2026 conference! Indico link

needle-sbi: solving the practical deployment of NSBI in HEP

- **needle-sbi is a workflow orchestrator for large-scale NSBI in HEP:**
  - Unified DAG execution with either LAW[3] or b2luigi[4] : it stitches the models together!
  - It combines efficient memory usage and data loading with dask/PyTorch
  - Allows for parallel and vectorized NN model evaluation
  - Supports the common workload managers for HTC/HPC:
    - HTCondor, Slurm, LSF



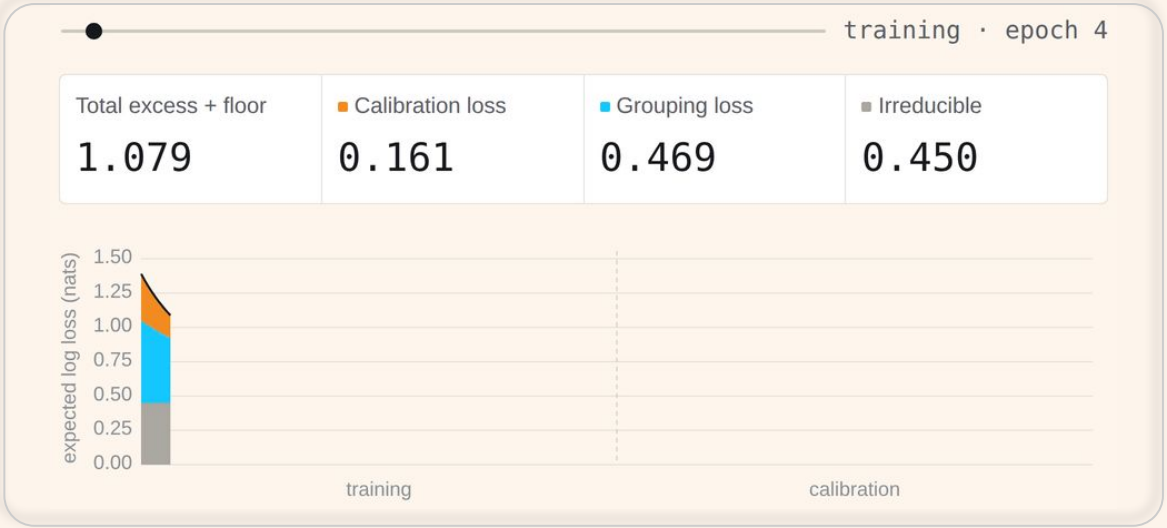


Calibration and classifier error decomposition

- **Classifier error/loss decomposes into 3 terms:**

$$\mathbb{E}\big[\mathrm{KL}(Y \,\|\, S)\big] = \underbrace{\mathbb{E}\big[\mathrm{KL}(Y \,\|\, S^\star)\big]}_{\text{irreducible}} + \underbrace{\mathbb{E}\big[\mathrm{KL}(S^\star \,\|\, C)\big]}_{\text{grouping,GL}} + \underbrace{\mathbb{E}\big[\mathrm{KL}(C \,\|\, S)\big]}_{\text{calibration,CL}}$$

  - Grouping loss GL[6] is the consequence of the classifier mapping two events epsilon-close in their scores when they shouldn't be



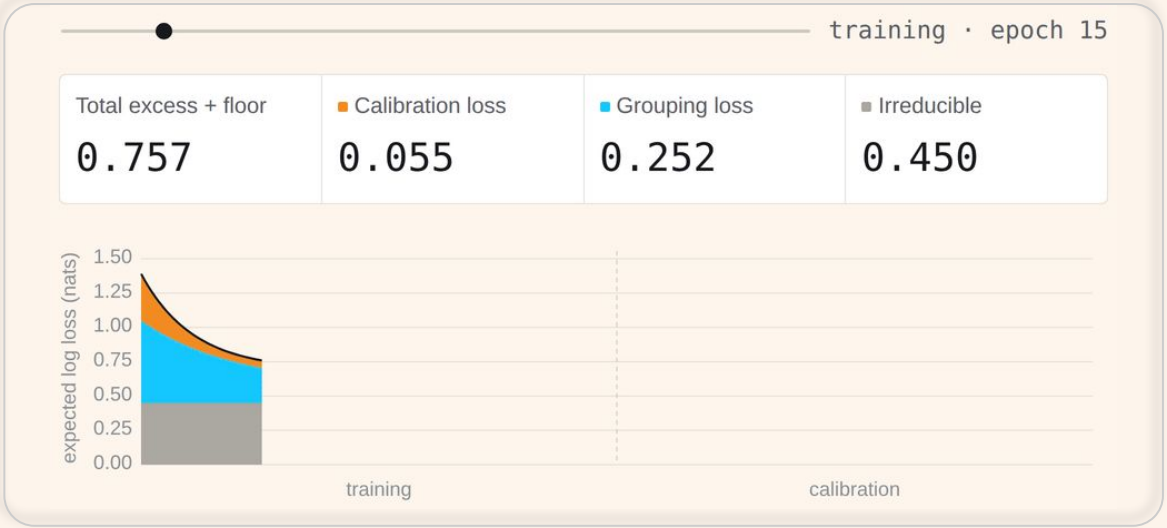
  - Above KL is the Kullback–Leibler divergence, $\mathcal{L}$ the loss (e.g. BCE or Brier), $C(x) = \mathbb{E}[Y|S(X)]$
    $S,\ f(x),\ s(x) = \sigma[f(x)]$ the classifier, its logit and score output, $S^\star$ Bayes-optimal classifier and $\mathcal{L}^\star$ its loss
    $g(f(x)),\ \bar{s}(x) = \sigma[g(f(x))]$ post-hoc (via calibration map g) calibrated classifier

Calibration and classifier error decomposition

- **Classifier error/loss decomposes into 3 terms:**

$$\mathbb{E}\big[\mathrm{KL}(Y \,\|\, S)\big] = \underbrace{\mathbb{E}\big[\mathrm{KL}(Y \,\|\, S^\star)\big]}_{\text{irreducible}} + \underbrace{\mathbb{E}\big[\mathrm{KL}(S^\star \,\|\, C)\big]}_{\text{grouping,GL}} + \underbrace{\mathbb{E}\big[\mathrm{KL}(C \,\|\, S)\big]}_{\text{calibration,CL}}$$

  - Grouping loss GL[6] is the consequence of the classifier mapping two events epsilon-close in their scores when they shouldn't be



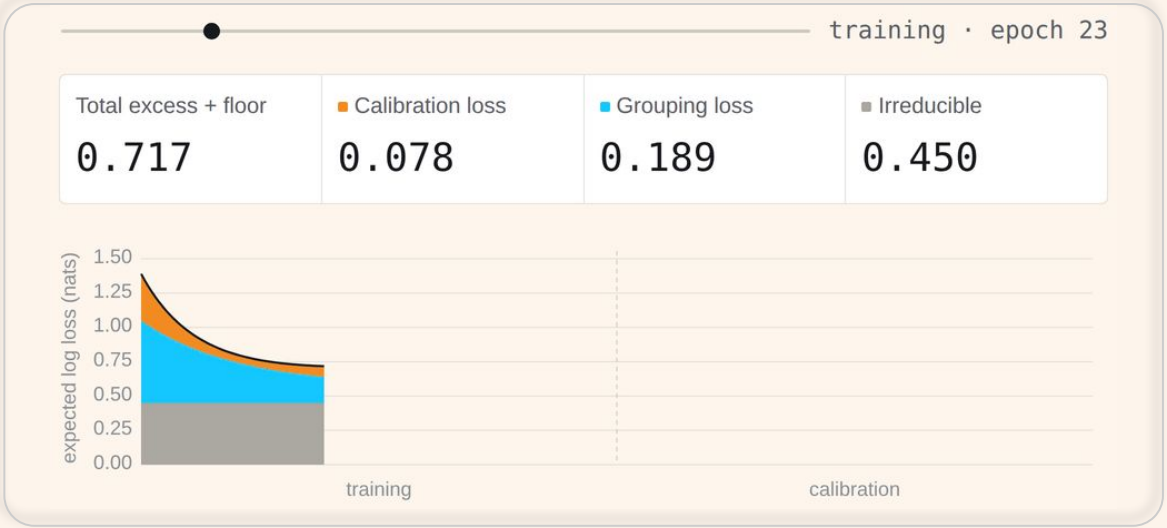
  - Above KL is the Kullback–Leibler divergence, $\mathcal{L}$ the loss (e.g. BCE or Brier), $C(x) = \mathbb{E}[Y|S(X)]$
    $S,\ f(x),\ s(x) = \sigma[f(x)]$ the classifier, its logit and score output, $S^\star$ Bayes-optimal classifier and $\mathcal{L}^\star$ its loss
    $g(f(x)),\ \bar{s}(x) = \sigma[g(f(x))]$ post-hoc (via calibration map g) calibrated classifier

Calibration and classifier error decomposition

- **Classifier error/loss decomposes into 3 terms:**

$$\mathbb{E}\big[\mathrm{KL}(Y \,\|\, S)\big] = \underbrace{\mathbb{E}\big[\mathrm{KL}(Y \,\|\, S^\star)\big]}_{\text{irreducible}} + \underbrace{\mathbb{E}\big[\mathrm{KL}(S^\star \,\|\, C)\big]}_{\text{grouping,GL}} + \underbrace{\mathbb{E}\big[\mathrm{KL}(C \,\|\, S)\big]}_{\text{calibration,CL}}$$

  - Grouping loss GL[6] is the consequence of the classifier mapping two events epsilon-close in their scores when they shouldn't be



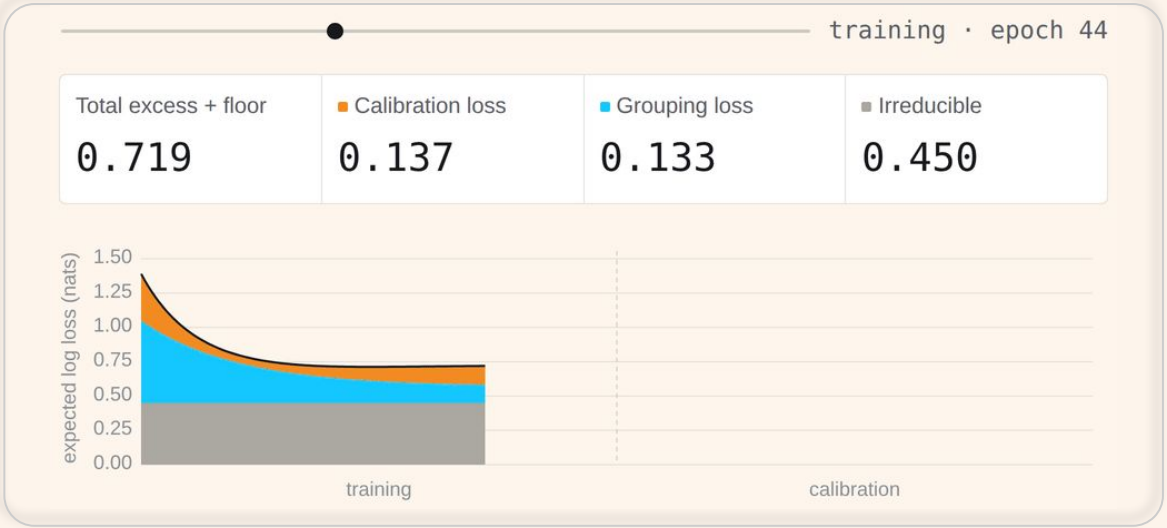
  - Above KL is the Kullback–Leibler divergence, $\mathcal{L}$ the loss (e.g. BCE or Brier), $C(x) = \mathbb{E}[Y|S(X)]$
    $S,\ f(x),\ s(x) = \sigma[f(x)]$ the classifier, its logit and score output, $S^\star$ Bayes-optimal classifier and $\mathcal{L}^\star$ its loss
    $g(f(x)),\ \bar{s}(x) = \sigma[g(f(x))]$ post-hoc (via calibration map g) calibrated classifier

Calibration and classifier error decomposition

- **Classifier error/loss decomposes into 3 terms:**

$$\mathbb{E}\big[\mathrm{KL}(Y \,\|\, S)\big] = \underbrace{\mathbb{E}\big[\mathrm{KL}(Y \,\|\, S^\star)\big]}_{\text{irreducible}} + \underbrace{\mathbb{E}\big[\mathrm{KL}(S^\star \,\|\, C)\big]}_{\text{grouping,GL}} + \underbrace{\mathbb{E}\big[\mathrm{KL}(C \,\|\, S)\big]}_{\text{calibration,CL}}$$

  - Grouping loss GL[6] is the consequence of the classifier mapping two events epsilon-close in their scores when they shouldn't be



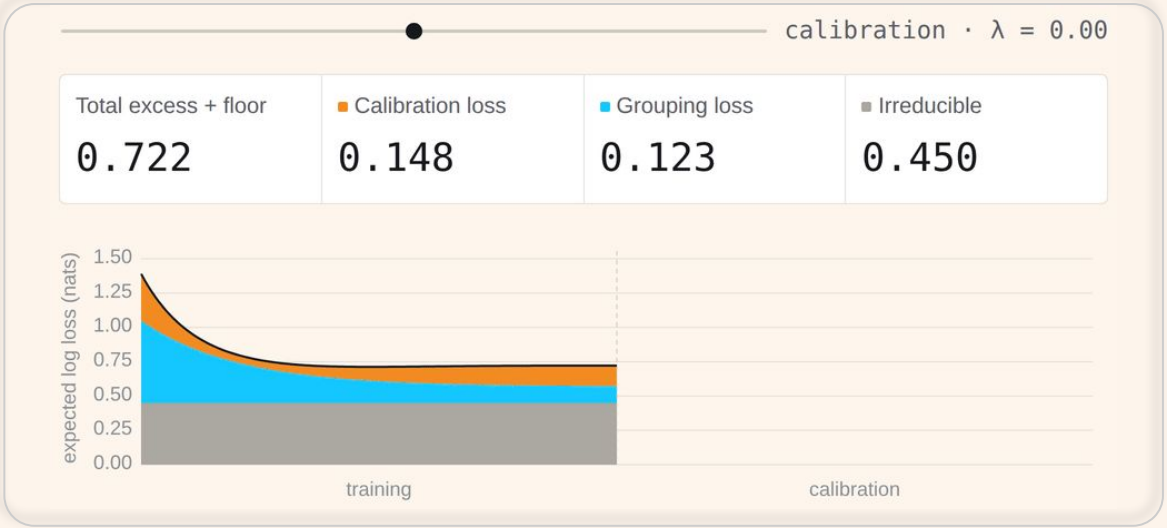
  - Above KL is the Kullback–Leibler divergence, $\mathcal{L}$ the loss (e.g. BCE or Brier), $C(x) = \mathbb{E}[Y|S(X)]$
    $S,\ f(x),\ s(x) = \sigma[f(x)]$ the classifier, its logit and score output, $S^\star$ Bayes-optimal classifier and $\mathcal{L}^\star$ its loss
    $g(f(x)),\ \bar{s}(x) = \sigma[g(f(x))]$ post-hoc (via calibration map g) calibrated classifier

Calibration and classifier error decomposition

- **Classifier error/loss decomposes into 3 terms:**

$$\mathbb{E}\big[\mathrm{KL}(Y \,\|\, S)\big] = \underbrace{\mathbb{E}\big[\mathrm{KL}(Y \,\|\, S^\star)\big]}_{\text{irreducible}} + \underbrace{\mathbb{E}\big[\mathrm{KL}(S^\star \,\|\, C)\big]}_{\text{grouping,GL}} + \underbrace{\mathbb{E}\big[\mathrm{KL}(C \,\|\, S)\big]}_{\text{calibration,CL}}$$

  - Grouping loss GL[6] is the consequence of the classifier mapping two events epsilon-close in their scores when they shouldn't be



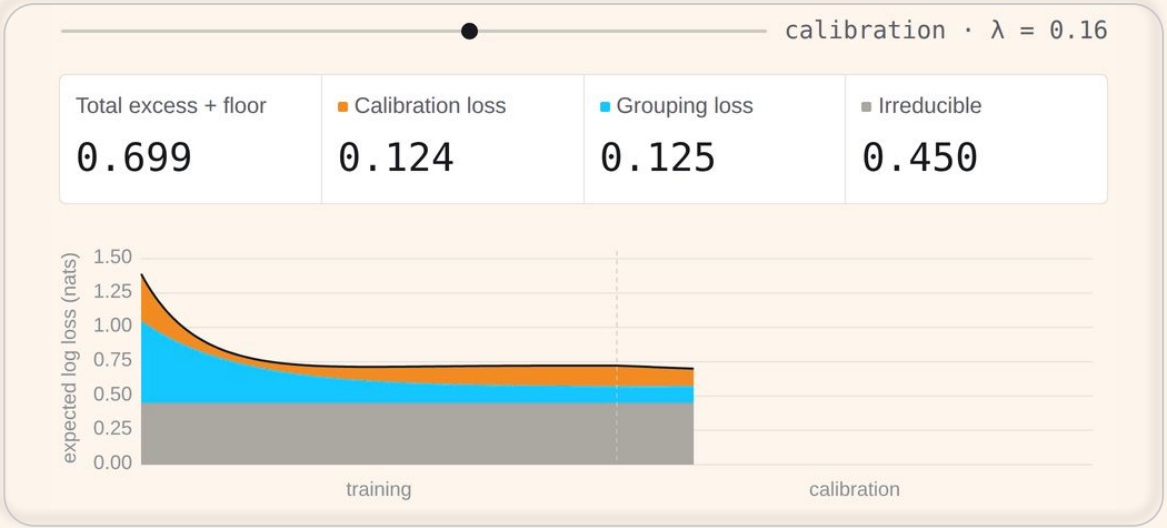
  - Above KL is the Kullback–Leibler divergence, $\mathcal{L}$ the loss (e.g. BCE or Brier), $C(x) = \mathbb{E}[Y|S(X)]$
    $S,\ f(x),\ s(x) = \sigma[f(x)]$ the classifier, its logit and score output, $S^\star$ Bayes-optimal classifier and $\mathcal{L}^\star$ its loss
    $g(f(x)),\ \bar{s}(x) = \sigma[g(f(x))]$ post-hoc (via calibration map g) calibrated classifier

Calibration and classifier error decomposition

- **Classifier error/loss decomposes into 3 terms:**

$$\mathbb{E}\big[\mathrm{KL}(Y \,\|\, S)\big] = \underbrace{\mathbb{E}\big[\mathrm{KL}(Y \,\|\, S^\star)\big]}_{\text{irreducible}} + \underbrace{\mathbb{E}\big[\mathrm{KL}(S^\star \,\|\, C)\big]}_{\text{grouping,GL}} + \underbrace{\mathbb{E}\big[\mathrm{KL}(C \,\|\, S)\big]}_{\text{calibration,CL}}$$

  - Grouping loss GL[6] is the consequence of the classifier mapping two events epsilon-close in their scores when they shouldn't be



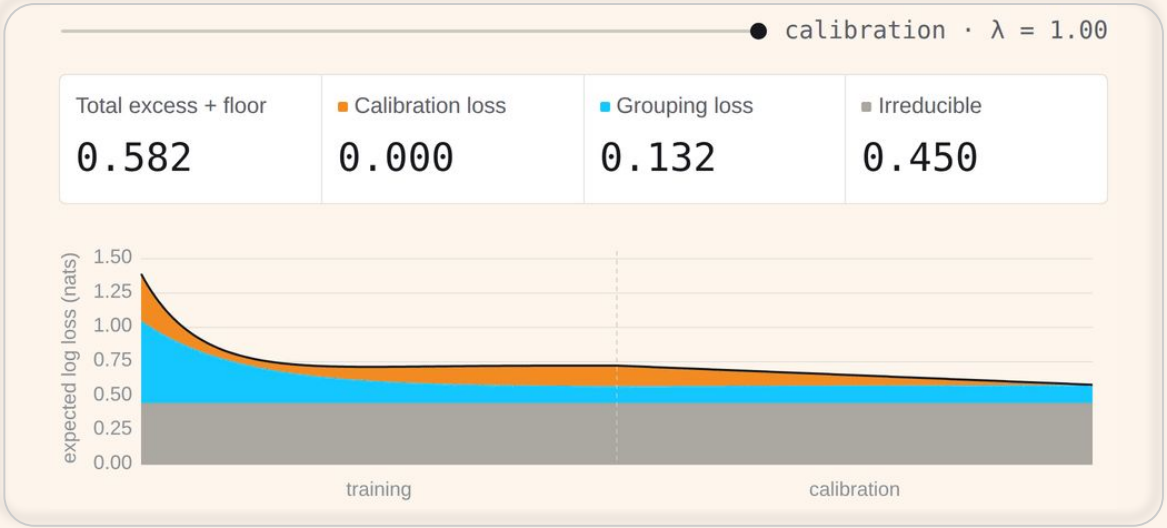
  - Above KL is the Kullback–Leibler divergence, $\mathcal{L}$ the loss (e.g. BCE or Brier), $C(x) = \mathbb{E}[Y|S(X)]$
    $S,\ f(x),\ s(x) = \sigma[f(x)]$ the classifier, its logit and score output, $S^\star$ Bayes-optimal classifier and $\mathcal{L}^\star$ its loss
    $g(f(x)),\ \bar{s}(x) = \sigma[g(f(x))]$ post-hoc (via calibration map g) calibrated classifier

Calibration and classifier error decomposition

- **Classifier error/loss decomposes into 3 terms:**

$$\mathbb{E}\big[\mathrm{KL}(Y \,\|\, S)\big] = \underbrace{\mathbb{E}\big[\mathrm{KL}(Y \,\|\, S^\star)\big]}_{\text{irreducible}} + \underbrace{\mathbb{E}\big[\mathrm{KL}(S^\star \,\|\, C)\big]}_{\text{grouping,GL}} + \underbrace{\mathbb{E}\big[\mathrm{KL}(C \,\|\, S)\big]}_{\text{calibration,CL}}$$

  - Grouping loss GL[6] is the consequence of the classifier mapping two events epsilon-close in their scores when they shouldn't be



  - Above KL is the Kullback–Leibler divergence, $\mathcal{L}$ the loss (e.g. BCE or Brier), $C(x) = \mathbb{E}[Y|S(X)]$
    $S,\ f(x),\ s(x) = \sigma[f(x)]$ the classifier, its logit and score output, $S^\star$ Bayes-optimal classifier and $\mathcal{L}^\star$ its loss
    $g(f(x)),\ \bar{s}(x) = \sigma[g(f(x))]$ post-hoc (via calibration map g) calibrated classifier

Calibration and classifier error decomposition

- **Classifier error/loss decomposes into 3 terms:**

$$\mathbb{E}\big[\mathrm{KL}(Y \,\|\, S)\big] \;=\; \underbrace{\mathbb{E}\big[\mathrm{KL}(Y \,\|\, S^\star)\big]}_{\text{irreducible}} \;+\; \underbrace{\mathbb{E}\big[\mathrm{KL}(S^\star \,\|\, C)\big]}_{\text{grouping,GL}} \;+\; \underbrace{\mathbb{E}\big[\mathrm{KL}(C \,\|\, S)\big]}_{\text{calibration,CL}}$$

  - Grouping loss GL[6] is the consequence of the classifier mapping two events epsilon-close in their scores when they shouldn't be

- **For a given post-hoc calibration map *g* that acts on the classifier outputs alone:**

$$g\circ f \text{ is a function of } f \quad\Longrightarrow\quad \mathrm{GL}\big(g\circ f\big) \;\geq\; \mathrm{GL}(f)$$

  - Above KL is the Kullback–Leibler divergence, $\mathcal{L}$ the loss (e.g. BCE or Brier), $C(x) = \mathbb{E}\big[Y|S(X)\big]$
    $S,\ f(x),\ s(x) = \sigma\big[f(x)\big]$ the classifier, its logit and score output, $S^\star$ Bayes-optimal classifier and $\mathcal{L}^\star$ its loss
    $g(f(x)),\ \bar{s}(x) = \sigma\big[g(f(x))\big]$ post-hoc (via calibration map *g*) calibrated classifier

Calibration and classifier error decomposition

- **Classifier error/loss decomposes into 3 terms:**

$$\mathbb{E}\big[\mathrm{KL}(Y \,\|\, S)\big] \;=\; \underbrace{\mathbb{E}\big[\mathrm{KL}(Y \,\|\, S^\star)\big]}_{\text{irreducible}} \;+\; \underbrace{\mathbb{E}\big[\mathrm{KL}(S^\star \,\|\, C)\big]}_{\text{grouping,GL}} \;+\; \underbrace{\mathbb{E}\big[\mathrm{KL}(C \,\|\, S)\big]}_{\text{calibration,CL}}$$

  - Grouping loss GL[6] is the consequence of the classifier mapping two events epsilon-close in their scores when they shouldn't be

- **For a given post-hoc calibration map *g* that acts on the classifier outputs alone:**

$$g\circ f \text{ is a function of } f \quad\Longrightarrow\quad \mathrm{GL}\big(g\circ f\big) \;\geq\; \mathrm{GL}(f)$$

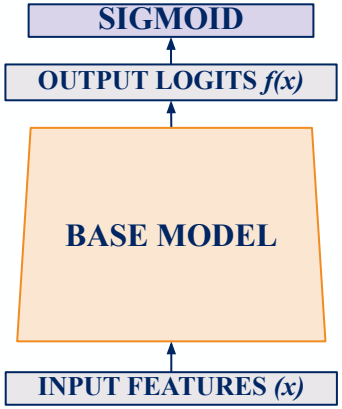
- **Even an ideal post-hoc calibration map that fully eliminates the calibration loss (CL), leaves GL is at best unchanged**

$$\mathrm{CL} \to 0 \text{ (ideal post-hoc } g) \quad\Longrightarrow\quad \mathcal{L}(g\circ f) - \mathcal{L}^\star \;\longrightarrow\; \mathrm{GL}(f)$$

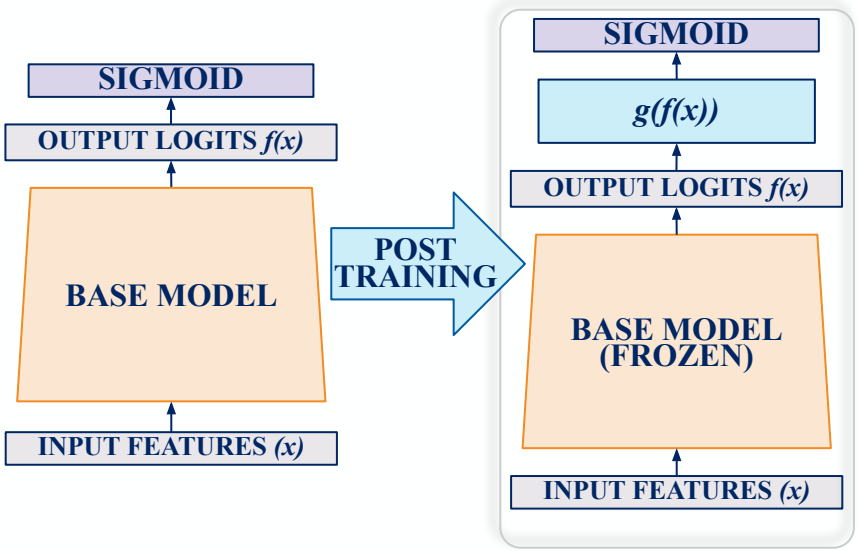
  - **This means that if GL is present, the Neyman-Pearson lemma optimality argument (or sufficiency of the probability ratio) collapses and this classifier sub-optimality cannot be removed by post-hoc calibration**

  - Above KL is the Kullback–Leibler divergence, $\mathcal{L}$ the loss (e.g. BCE or Brier), $C(x) = \mathbb{E}\big[Y|S(X)\big]$
    $S,\ f(x),\ s(x) = \sigma[f(x)]$ the classifier, its logit and score output, $S^\star$ Bayes-optimal classifier and $\mathcal{L}^\star$ its loss
    $g(f(x)),\ \bar{s}(x) = \sigma[g(f(x))]$ post-hoc (via calibration map *g*) calibrated classifier

Post-hoc Calibration vs Fine-tuning



Post-hoc Calibration vs Fine-tuning

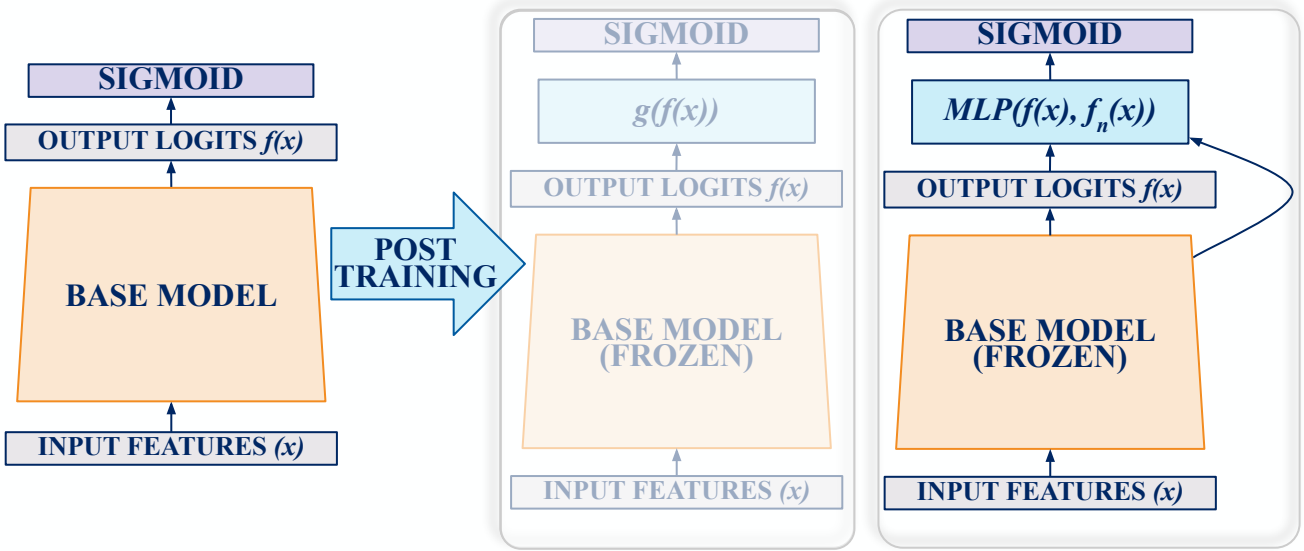


"Traditional" post-hoc methods:

- Platt scaling[6]
- Isotonic Regression[7]
- G-layers[8]
- …

They only transform models' outputs

Post-hoc Calibration vs Fine-tuning



"Traditional" post-hoc methods:

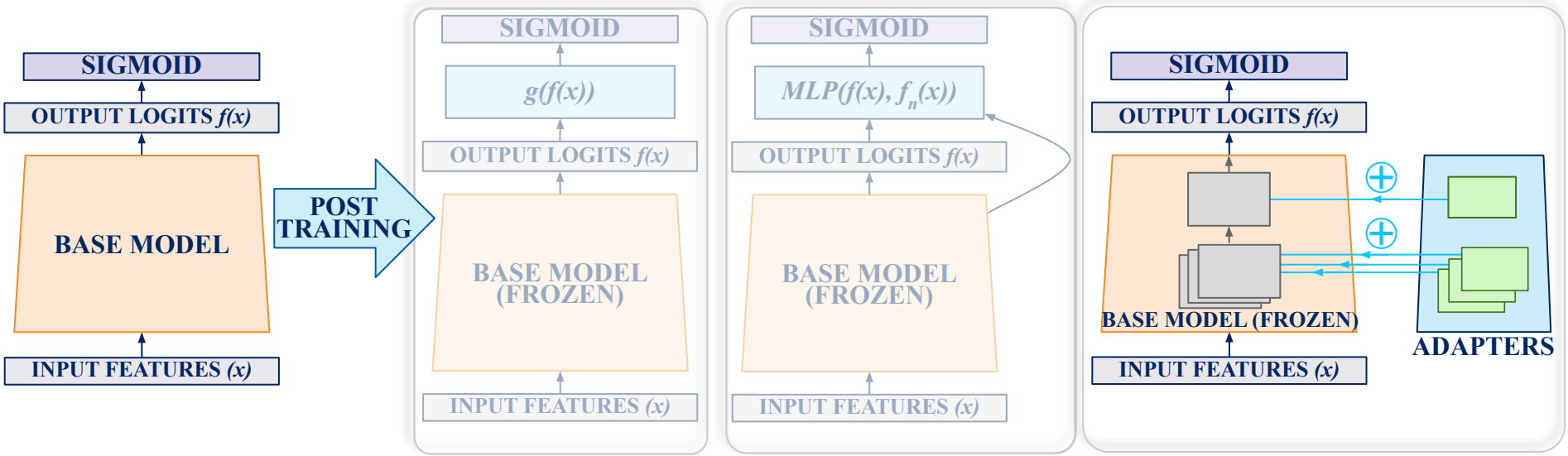
- Platt scaling[6]
- Isotonic Regression[7]
- G-layers[8]
- …

They only transform models' outputs

"Generalized" g-layers:

MLP now also hooks into base model's classification head

Post-hoc Calibration vs Fine-tuning



"Traditional" post-hoc methods:

- Platt scaling[6]
- Isotonic Regression[7]
- G-layers[8]
- …

They only transform models' outputs

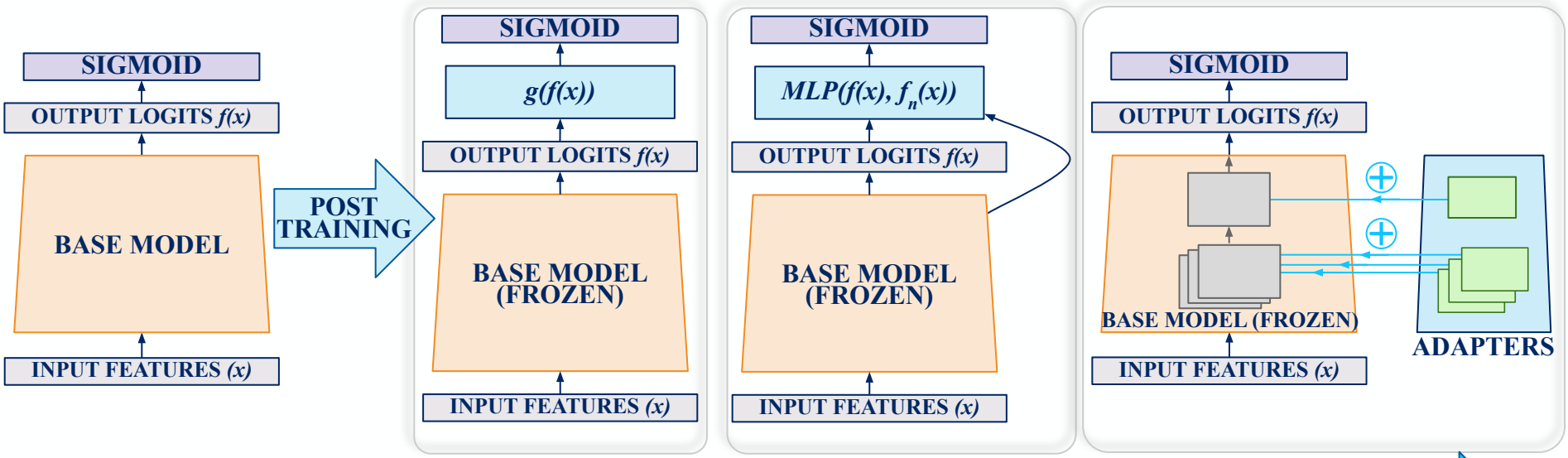
"Generalized" g-layers:

MLP now also hooks into base model's classification head

Adapter-based model modification:

Low-rank adapters[9] (LoRA) are injected into base model's linear transformations' matrices

Post-hoc Calibration vs Fine-tuning



METHOD CAPACITY AND BASE MODEL CORRECTION CEILING

"Traditional" post-hoc methods:

- Platt scaling[6]
- Isotonic Regression[7]
- G-layers[8]
- …

They only transform models' outputs

"Generalized" g-layers:

MLP now also hooks into base model's classification head

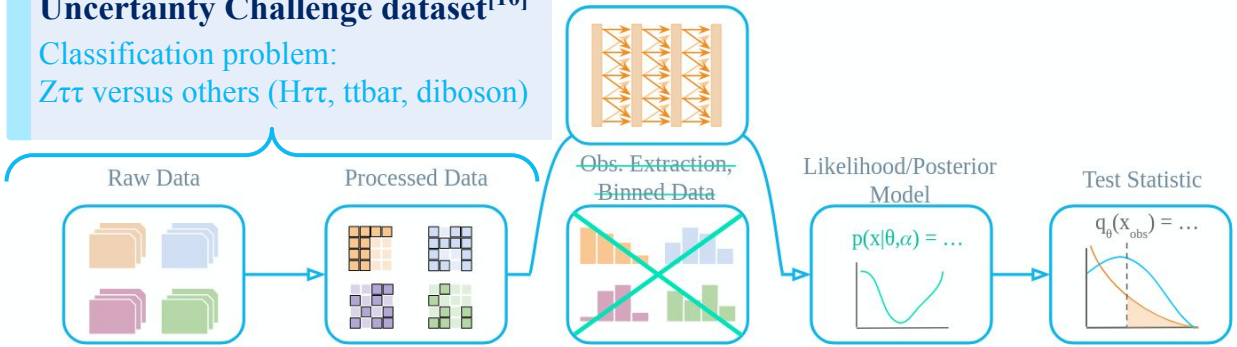
Adapter-based model modification:

Low-rank adapters[9] (LoRA) are injected into base model's linear transformations' matrices

Experiment setup

We used FAIR Universe HiggsML Uncertainty Challenge dataset[10]

Classification problem:
Zττ versus others (Hττ, ttbar, diboson)



Experiment setup

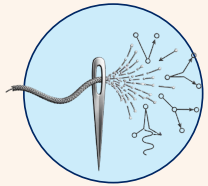
We used FAIR Universe HiggsML Uncertainty Challenge dataset[10]

Classification problem:
Zττ versus others (Hττ, ttbar, diboson)

Base model setups:

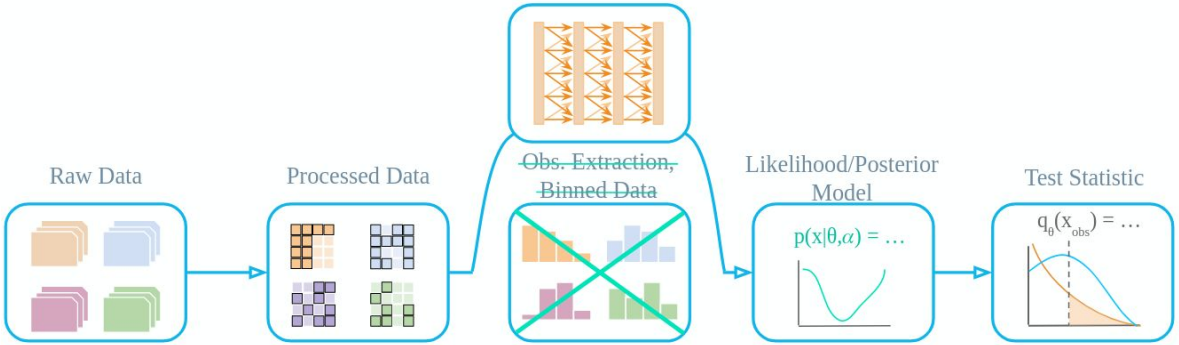
- Permutationally equivariant transformer-based models w.r.t particle ordering
- Use only kinematic features as inputs

We run everything fully automated by needle-sbi!



Experiment setup

- **We track a large number of relevant metrics*,[11] for each NN and each calibration method:**

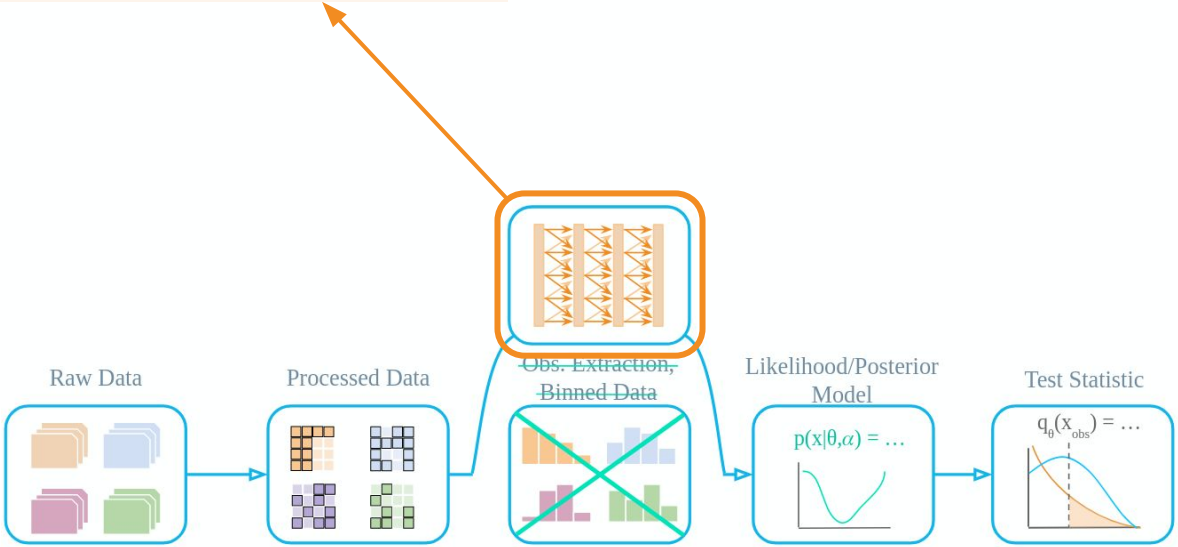


Experiment setup

- **We track a large number of relevant metrics*,[11] for each NN and each calibration method:**

Classifier quality: Brier Score

$$BS = \frac{1}{\sum_i w_i} \sum_i w_i \left(s(x_i) - y_i\right)^2$$



*listed here are only the metrics directly relevant to this talk's content

Experiment setup

- **We track a large number of relevant metrics*,[11] for each NN and each calibration method:**

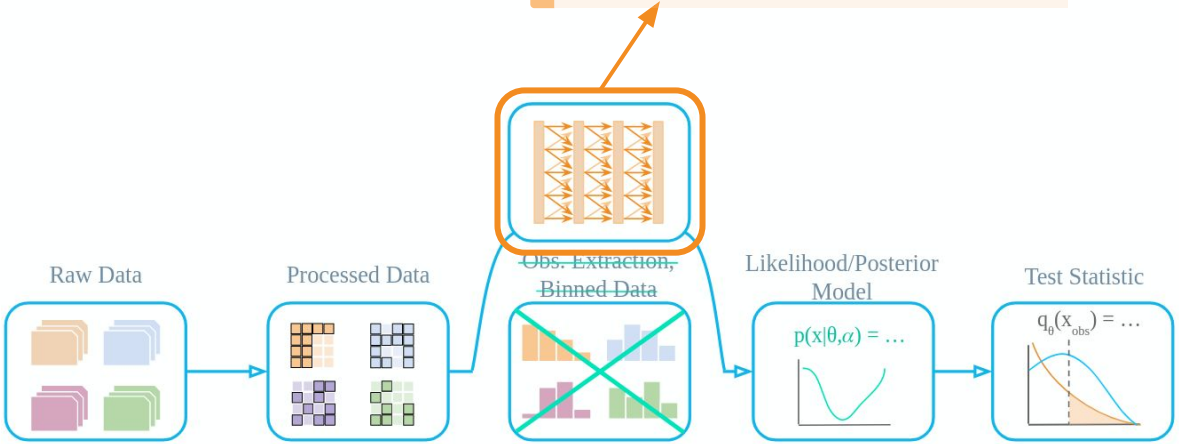
Classifier quality: Brier Score

$$BS = \frac{1}{\sum_i w_i} \sum_i w_i \left(s(x_i) - y_i\right)^2$$

Calibration quality:

(Adaptive/Averaged) Expected calibration error

$$\mathrm{ECE} = \frac{\sum_{b \in \mathrm{bins}} w_b \left| y_b^{\mathrm{total}} - s_b^{\mathrm{total}} \right|}{\sum_{b \in \mathrm{bins}} w_b}$$



*listed here are only the metrics directly relevant to this talk's content

Experiment setup

- **We track a large number of relevant metrics*,[11] for each NN and each calibration method:**

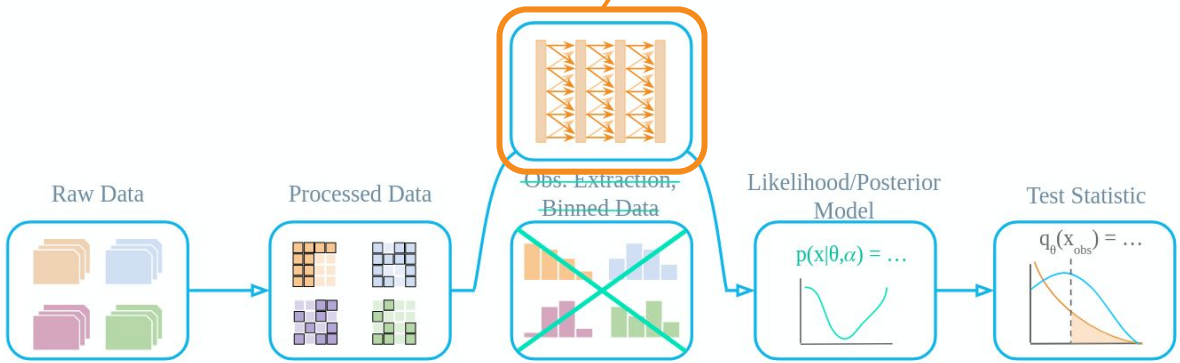
Classifier quality: Brier Score

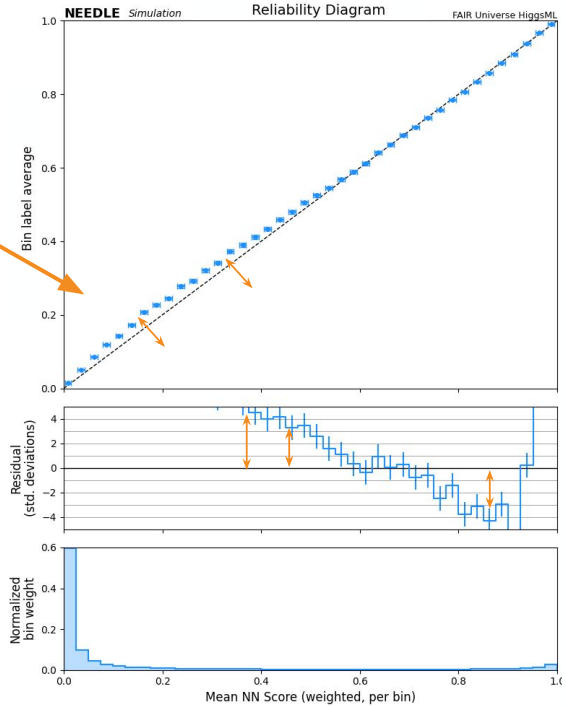
$$BS = \frac{1}{\sum_i w_i} \sum_i w_i \left(s(x_i) - y_i\right)^2$$

Calibration quality:

(Adaptive/Averaged) Expected calibration error

$$\mathrm{ECE} = \frac{\sum_{b \in \mathrm{bins}} w_b \left| y_b^{\mathrm{total}} - s_b^{\mathrm{total}} \right|}{\sum_{b \in \mathrm{bins}} w_b}$$





*listed here are only the metrics directly relevant to this talk's content

Experiment setup

- **We track a large number of relevant metrics*,[11] for each NN and each calibration method:**

Classifier quality: Brier Score

$$BS = \frac{1}{\sum_i w_i} \sum_i w_i \left(s(x_i) - y_i\right)^2$$

Calibration quality:

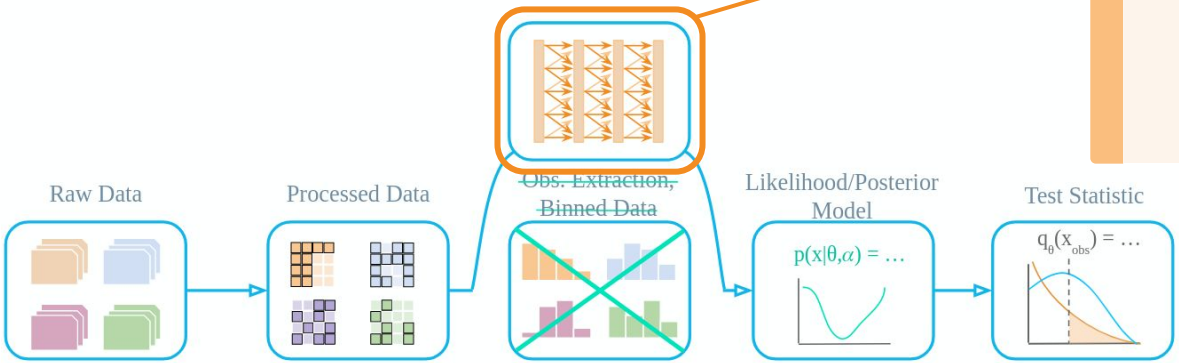
(Adaptive/Averaged) Expected calibration error

$$\mathrm{ECE} = \frac{\sum_{b \in \mathrm{bins}} w_b \left| y_b^{\mathrm{total}} - s_b^{\mathrm{total}} \right|}{\sum_{b \in \mathrm{bins}} w_b}$$

Ratio Estimation quality:

Reweighting Tests

For each feature: Wasserstein-1 distance (EMD) between distributions of the target and source after reweighting by the estimated prob. ratio



*listed here are only the metrics directly relevant to this talk's content

Experiment setup

- **We track a large number of relevant metrics*,[11] for each NN and each calibration method:**

Classifier quality: Brier Score

$$BS = \frac{1}{\sum_i w_i} \sum_i w_i \left(s(x_i) - y_i\right)^2$$

Calibration quality:

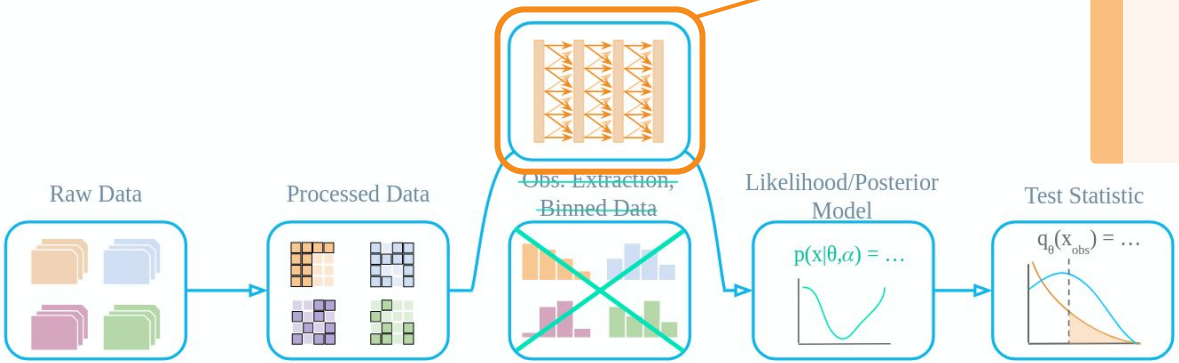
(Adaptive/Averaged) Expected calibration error

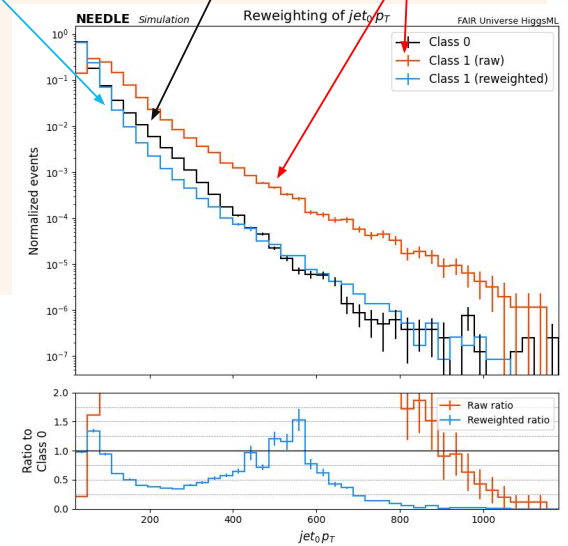
$$\mathrm{ECE} = \frac{\sum_{b \in \mathrm{bins}} w_b \left| y_b^{\mathrm{total}} - s_b^{\mathrm{total}} \right|}{\sum_{b \in \mathrm{bins}} w_b}$$

Ratio Estimation quality:

Reweighting Tests

For each feature: Wasserstein-1 distance (EMD) between distributions of the target and source after reweighting by the estimated prob. ratio





*listed here are only the metrics directly relevant to this talk's content

Experiment setup

- **We track a large number of relevant metrics*,[11] for each NN and each calibration method:**

Classifier quality: Brier Score

$$BS = \frac{1}{\sum_i w_i} \sum_i w_i \left(s(x_i) - y_i\right)^2$$

Calibration quality:

(Adaptive/Averaged) Expected calibration error

$$\mathrm{ECE} = \frac{\sum_{b \in \mathrm{bins}} w_b \left| y_b^{\mathrm{total}} - s_b^{\mathrm{total}} \right|}{\sum_{b \in \mathrm{bins}} w_b}$$

Ratio Estimation quality:

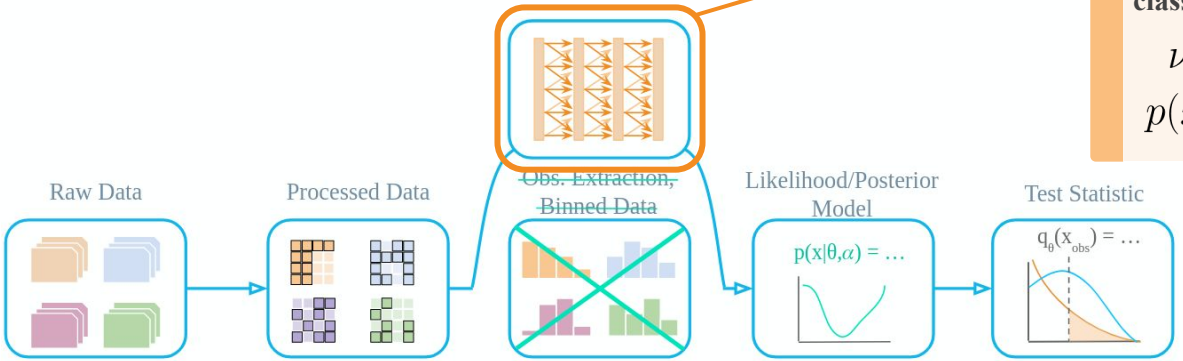
Reweighting Tests

For each feature: Wasserstein-1 distance (EMD) between distributions of the target and source after reweighting by the estimated prob. ratio

Asimov Fit Tests:

Simple 1-parameter signal injection test between 2 classes:

$$\nu_{\mathrm{total}} = c \cdot \nu_1 + \nu_0$$
$$p(x|c) = c \cdot p(x|y=1) + p(x|y=0)$$

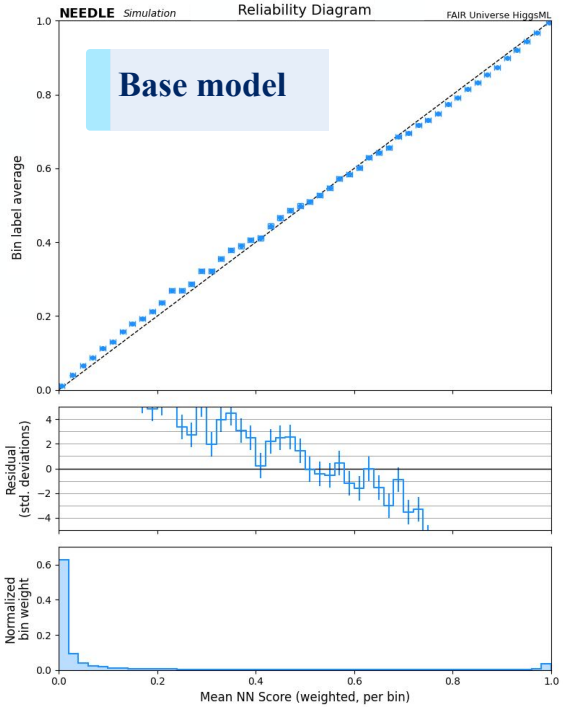


*listed here are only the metrics directly relevant to this talk's content

Results: calibration versus asimov fits

We compare: base model versus several calibrated/fine tuned variants

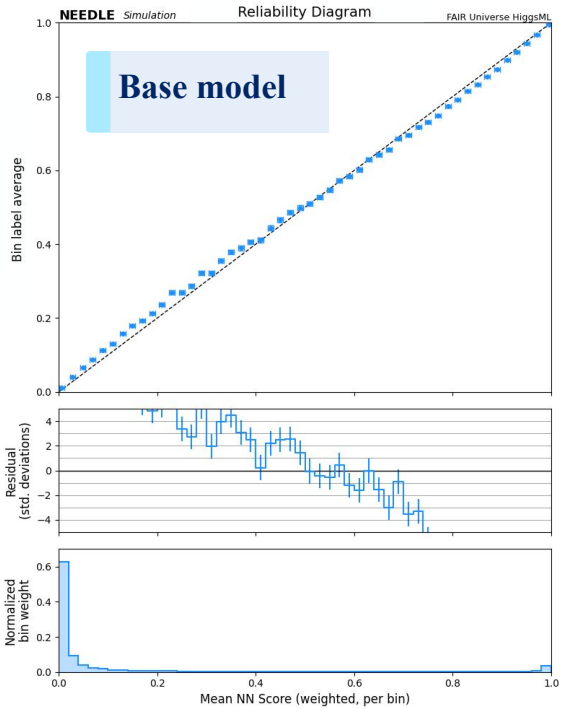
Shown are aggregated results over ensembles x folds

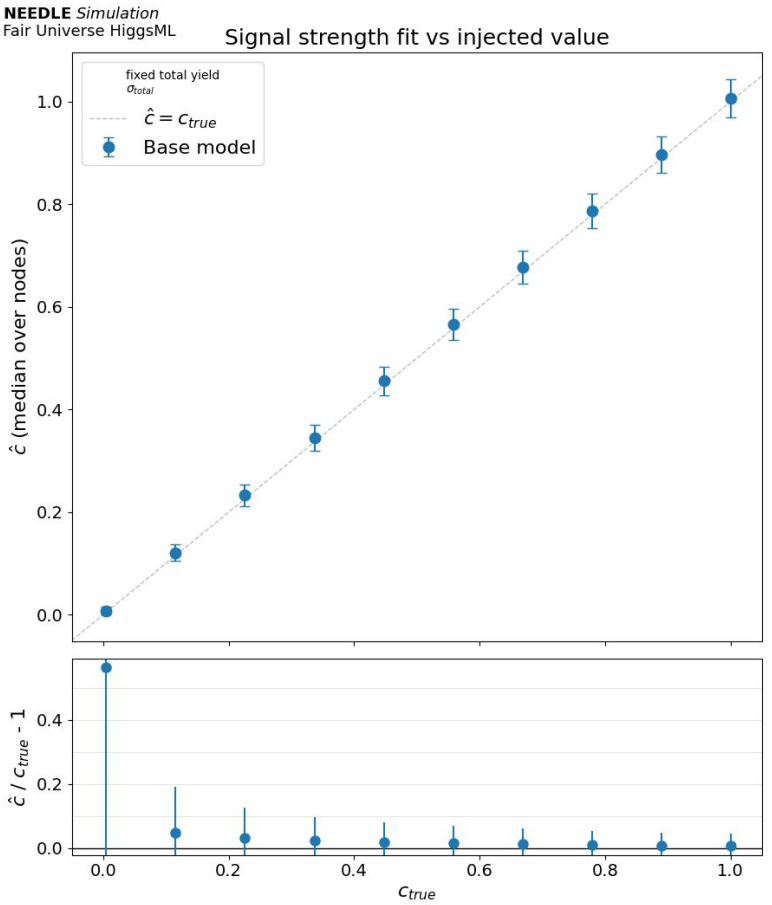


Results: calibration versus asimov fits

We compare: base model versus several calibrated/fine tuned variants

Shown are aggregated results over ensembles x folds

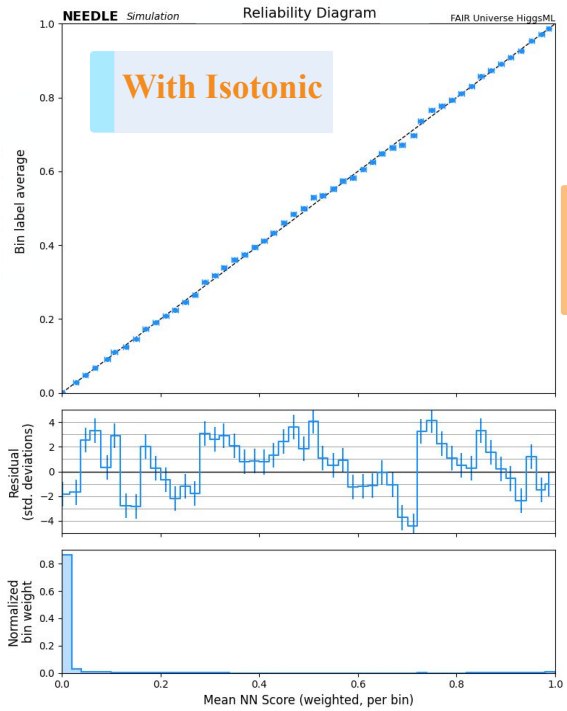




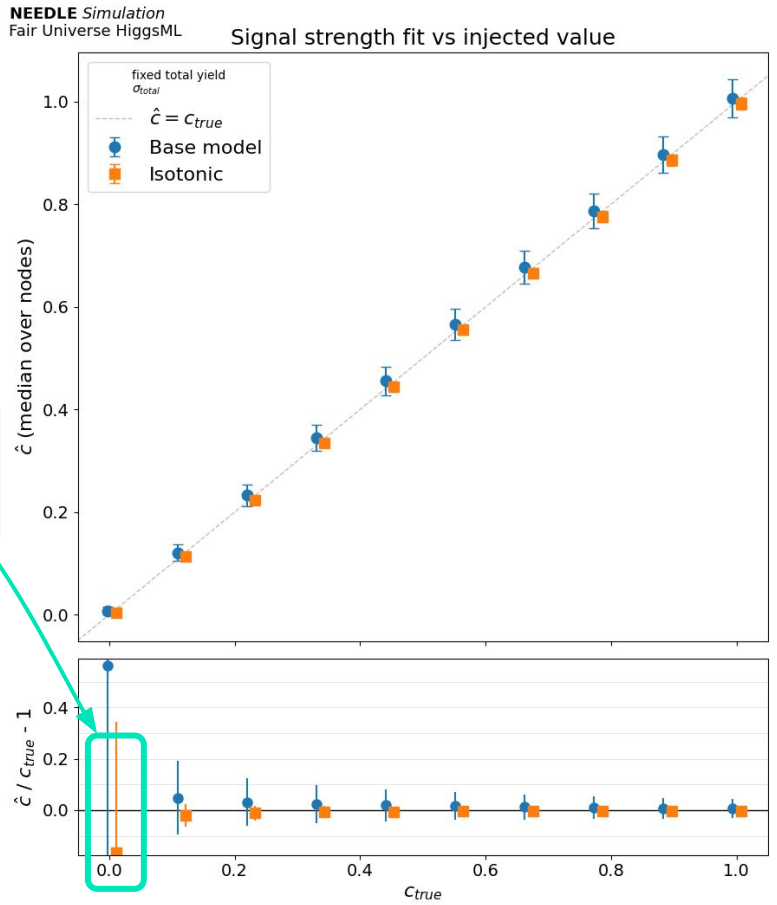
Results: calibration versus asimov fits

We compare: base model versus several calibrated/fine tuned variants

Shown are aggregated results over ensembles x folds



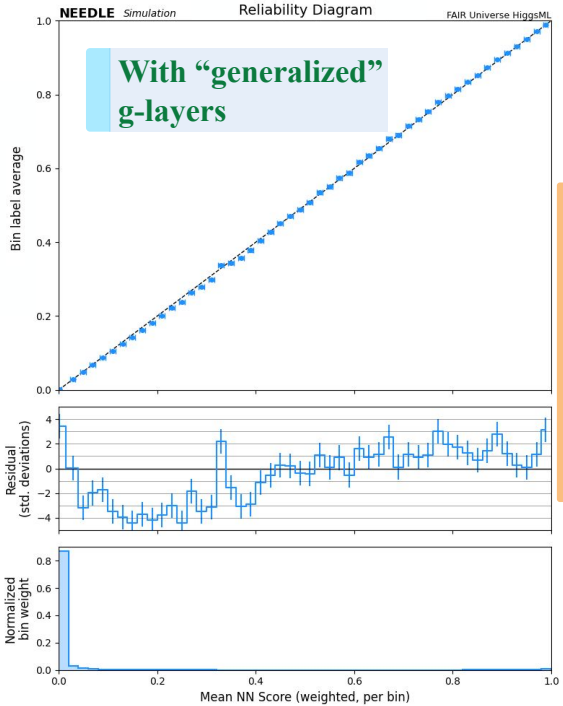
~17% residue at the lowest signal strength
Can this residue be eliminated?



Results: calibration versus asimov fits

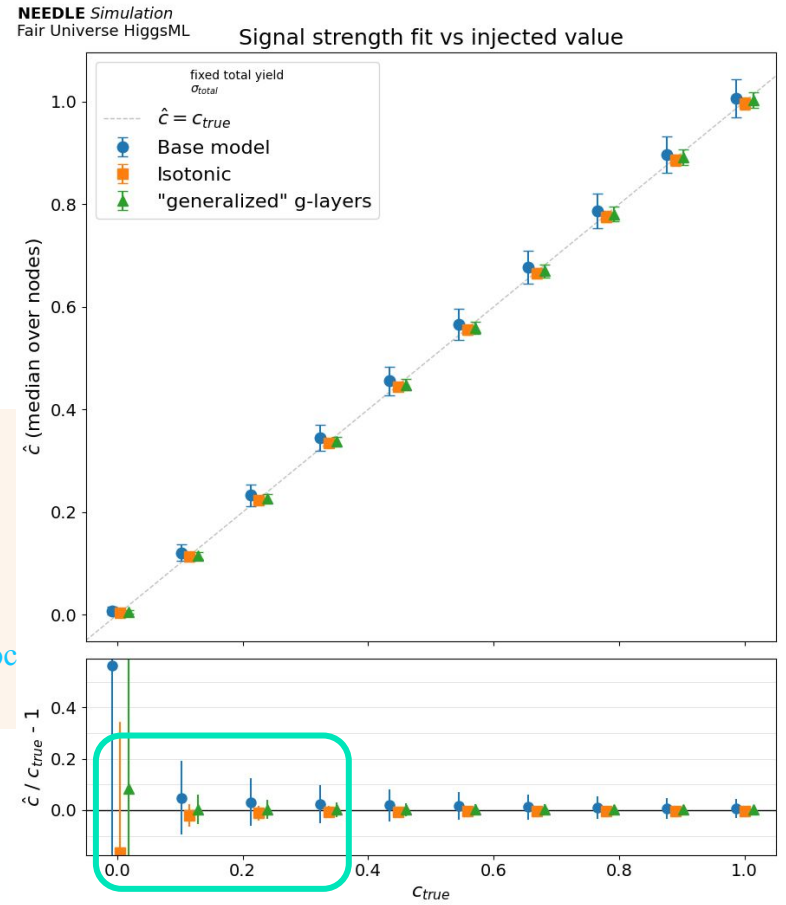
We compare: base model versus several calibrated/fine tuned variants

Shown are aggregated results over ensembles x folds



~17% residue at the lowest signal strength (with Isotonic)
Can this residue be eliminated?

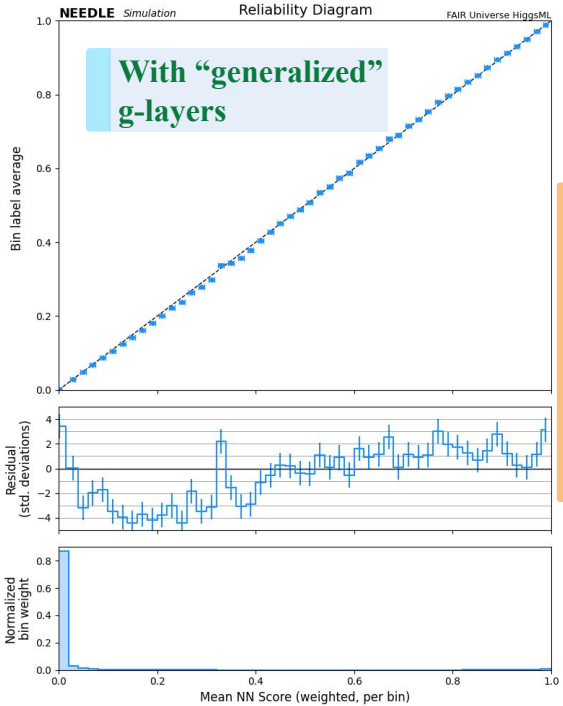
For low values of injected signal, the fit model becomes sensitive to the high-values of the probability ratio, which is exactly the place where post-hoc calibration won't help



Results: calibration versus asimov fits

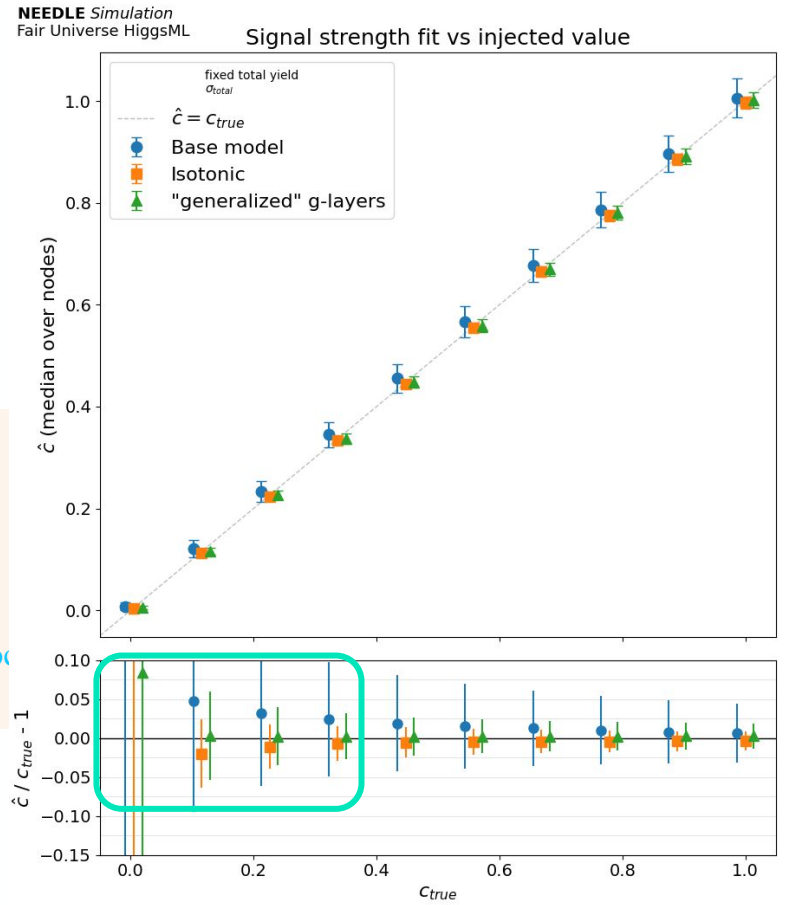
We compare: base model versus several calibrated/fine tuned variants

Shown are aggregated results over ensembles x folds



~17% residue at the lowest signal strength (with Isotonic)
Can this residue be eliminated?

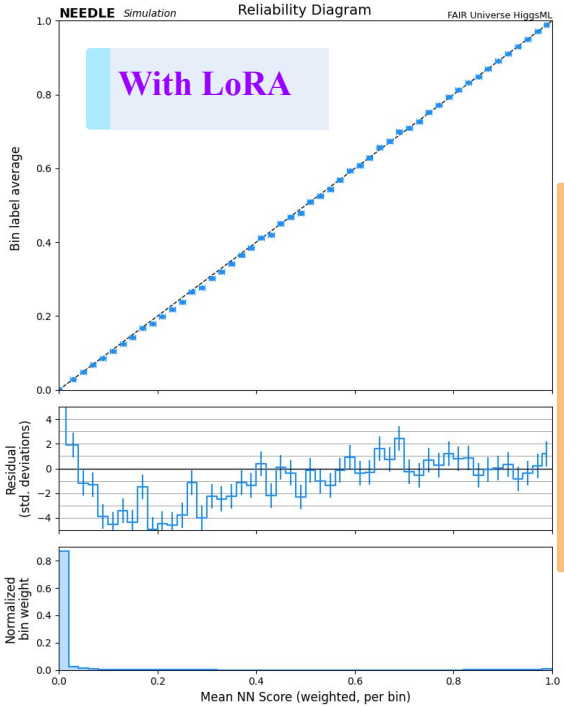
For low values of injected signal, the fit model becomes sensitive to the high-values of the probability ratio, which is exactly the place where post-hoc calibration won't help



Results: calibration versus asimov fits

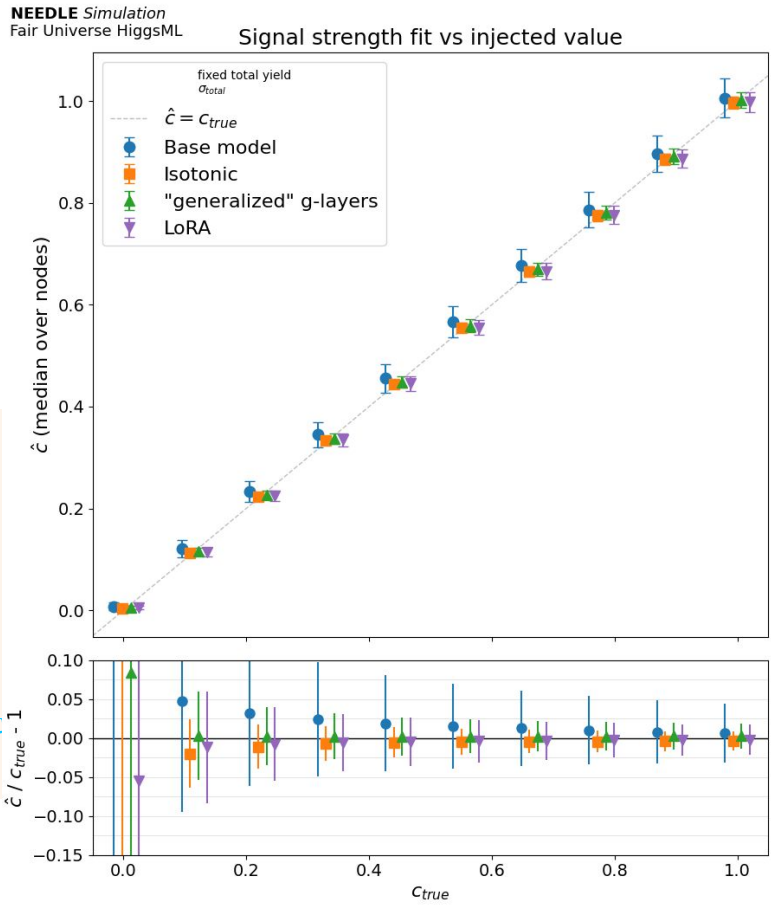
We compare: base model versus several calibrated/fine tuned variants

Shown are aggregated results over ensembles x folds



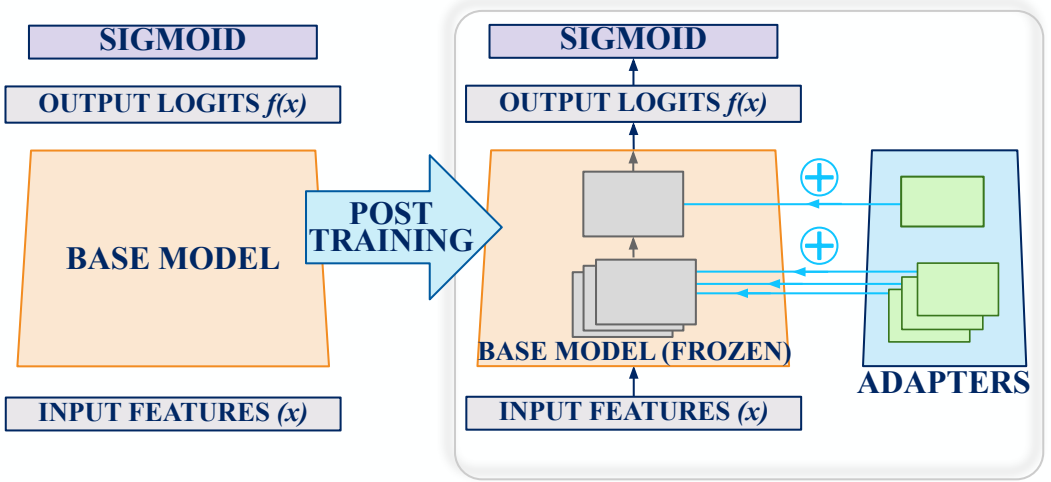
~17% residue at the lowest signal strength (with Isotonic)
Can this residue be eliminated?
Yes. And at no cost to the calibration performance

For low values of injected signal, the fit model becomes sensitive to the high-values of the probability ratio, which is exactly the place where post-hoc calibration won't help

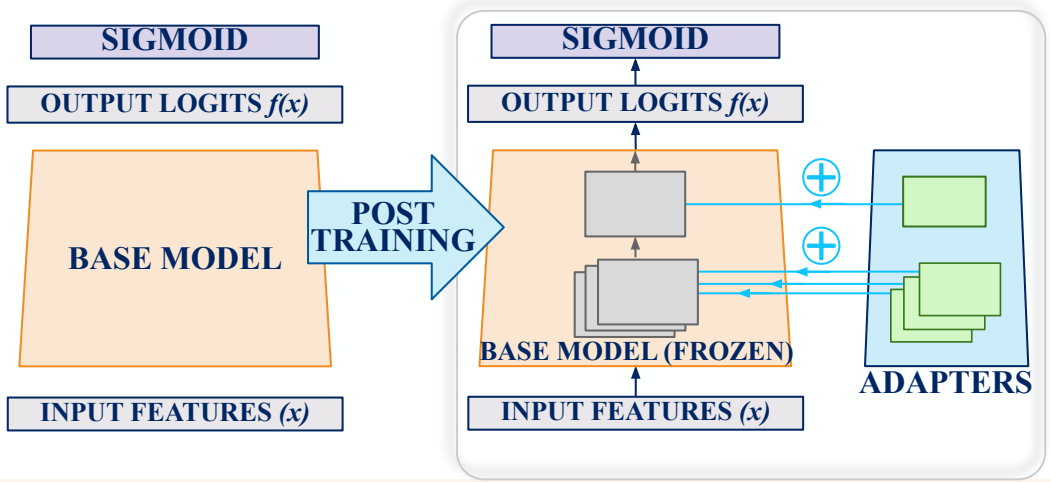


In-depth exploration of the adapter-based method

If this method can improve models after training, can it also make general classifier training more practical?



In-depth exploration of the adapter-based method

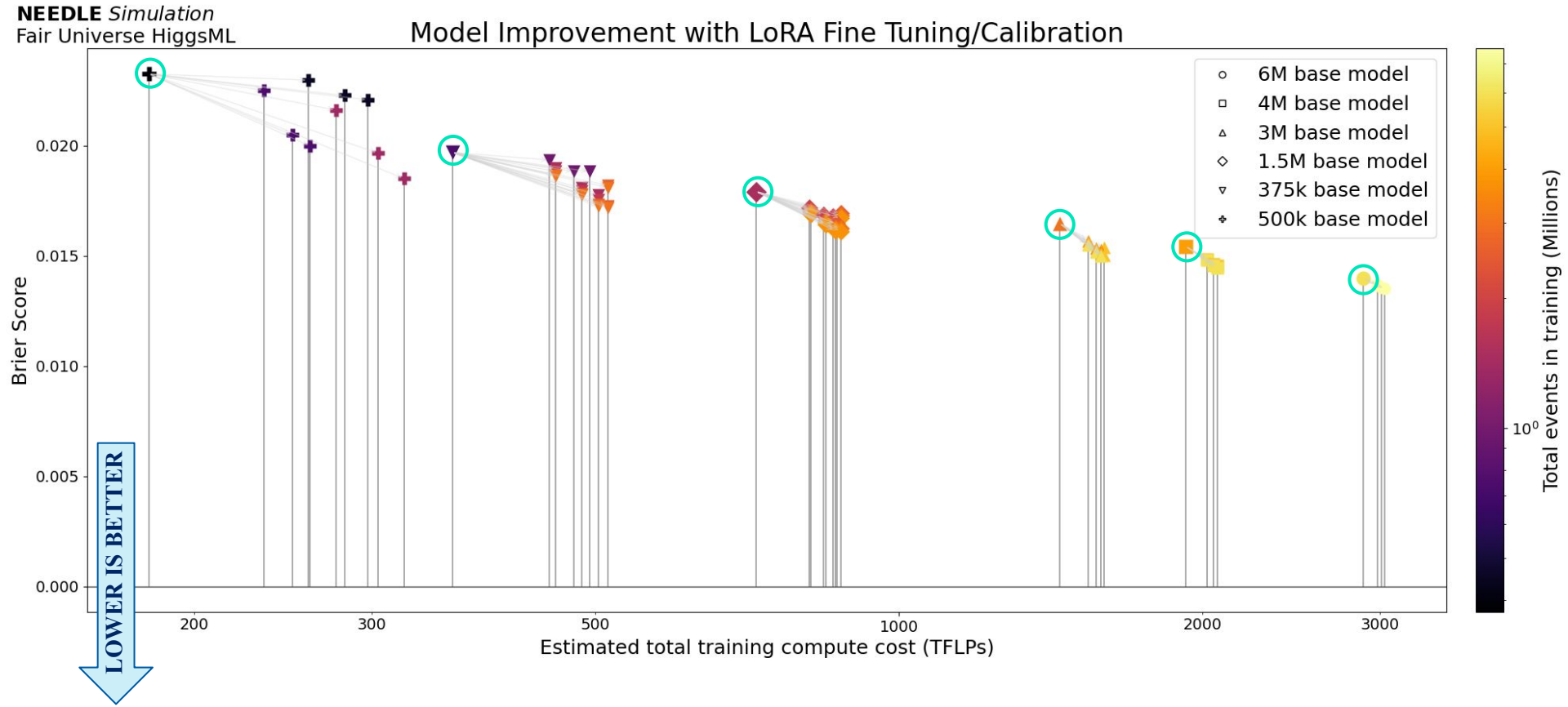


If this method can improve models after training, can it also make general classifier training more practical?

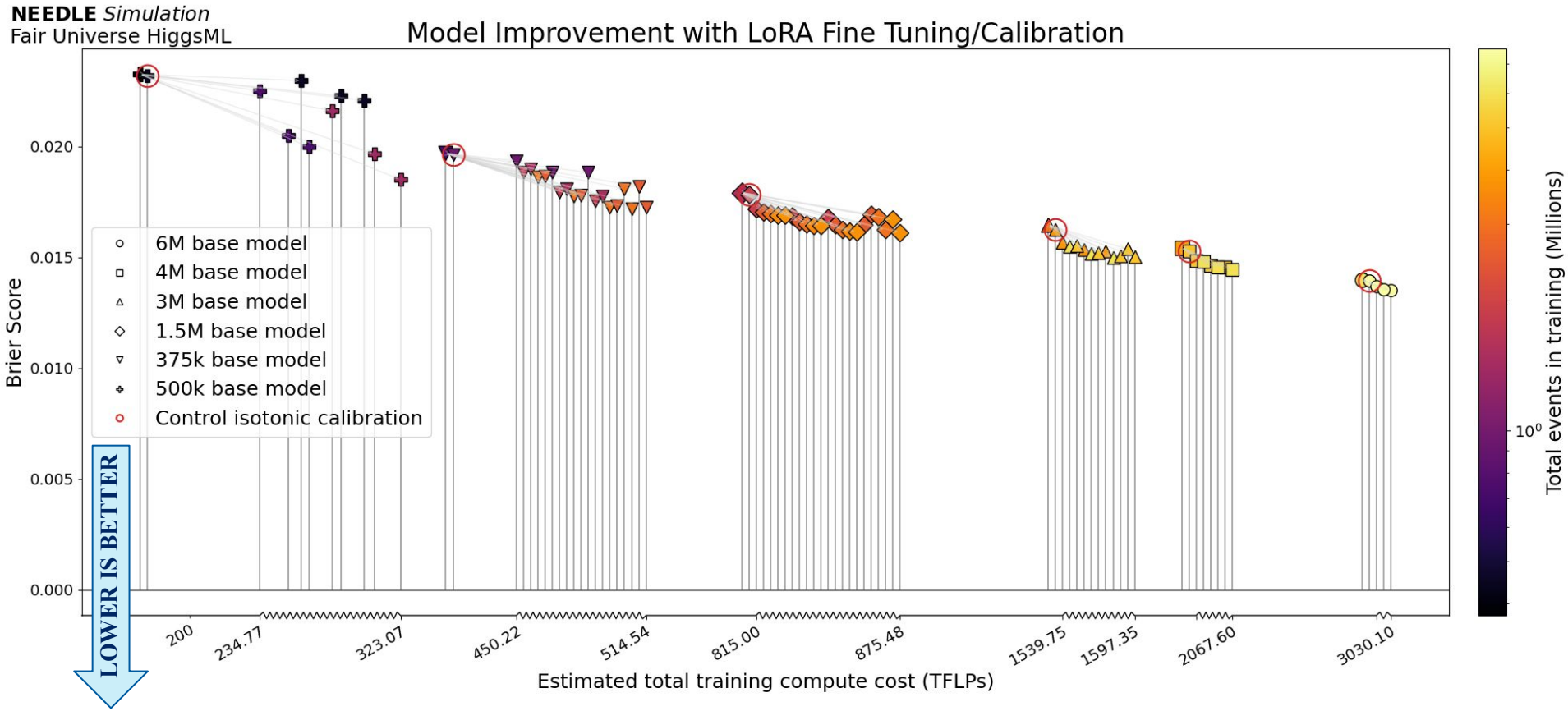
Test setup:

- Base models trained on different sized datasets, keeping their architecture constant
- For each NN, a set of adapters was fitted to it
  - We kept the adapter architecture and total training cost constant to better control the test
  - The total dataset size adapters were allowed to train was gradually increased
- As control we calibrated each model with isotonic regression
  - Isotonic regression was the best performing of the post-hoc methods we explored
  - The dataset size isotonic was fitted with is the same as the lowest-dataset size adapter
- The plots that follow show the results of aggregated metrics across all NNs (over folds x ensemble members)

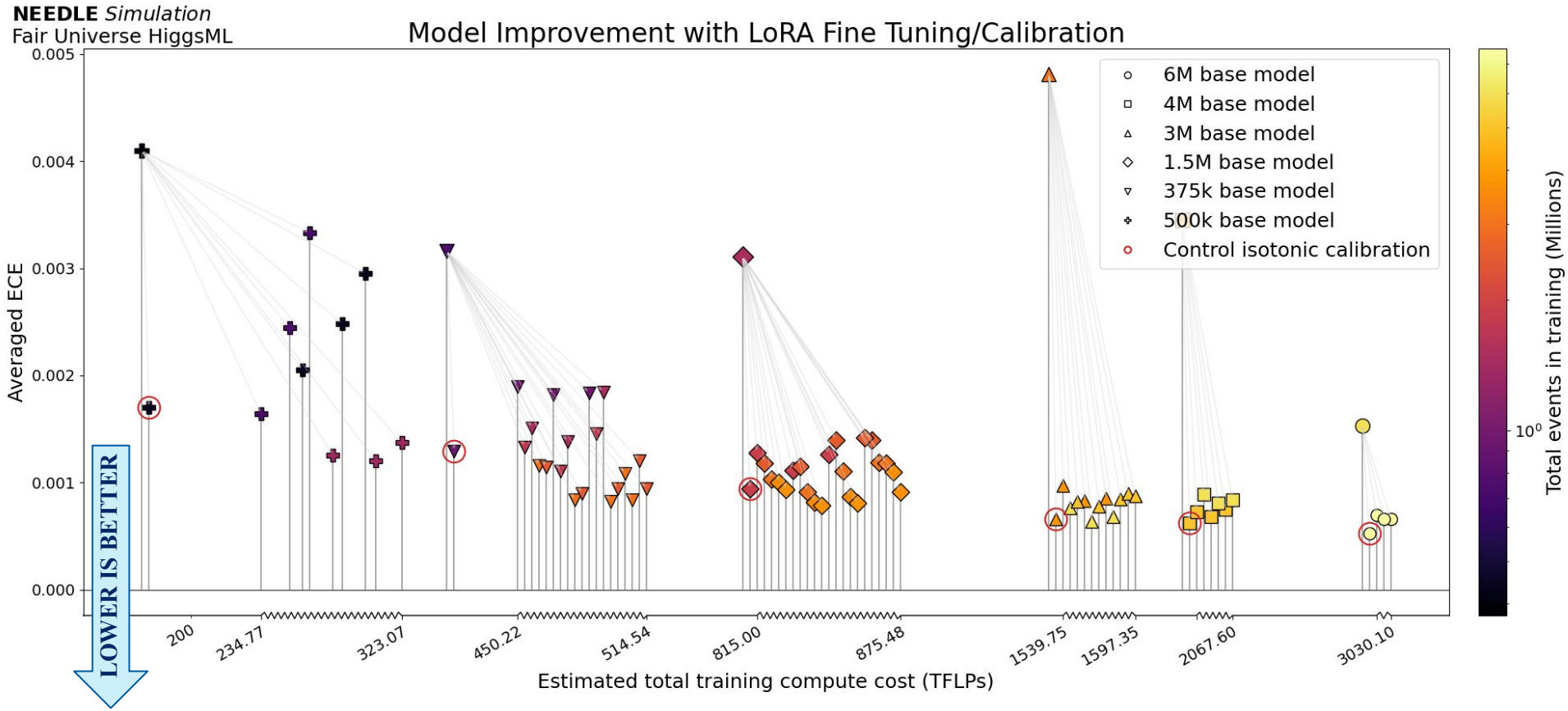
Results: base models versus calibrated/refined variants (Brier score)
True x-axis positions



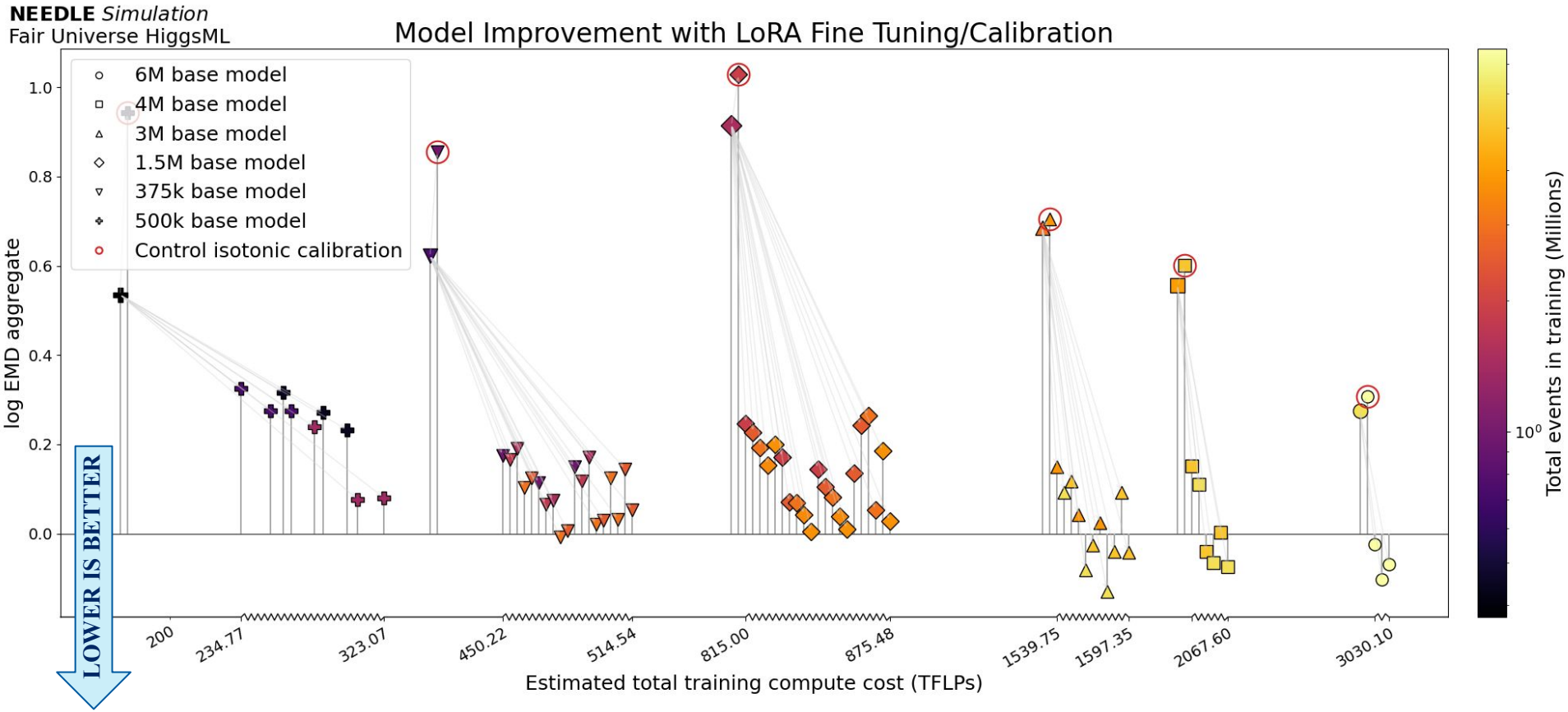
Results: base models versus calibrated/refined variants (Brier score)



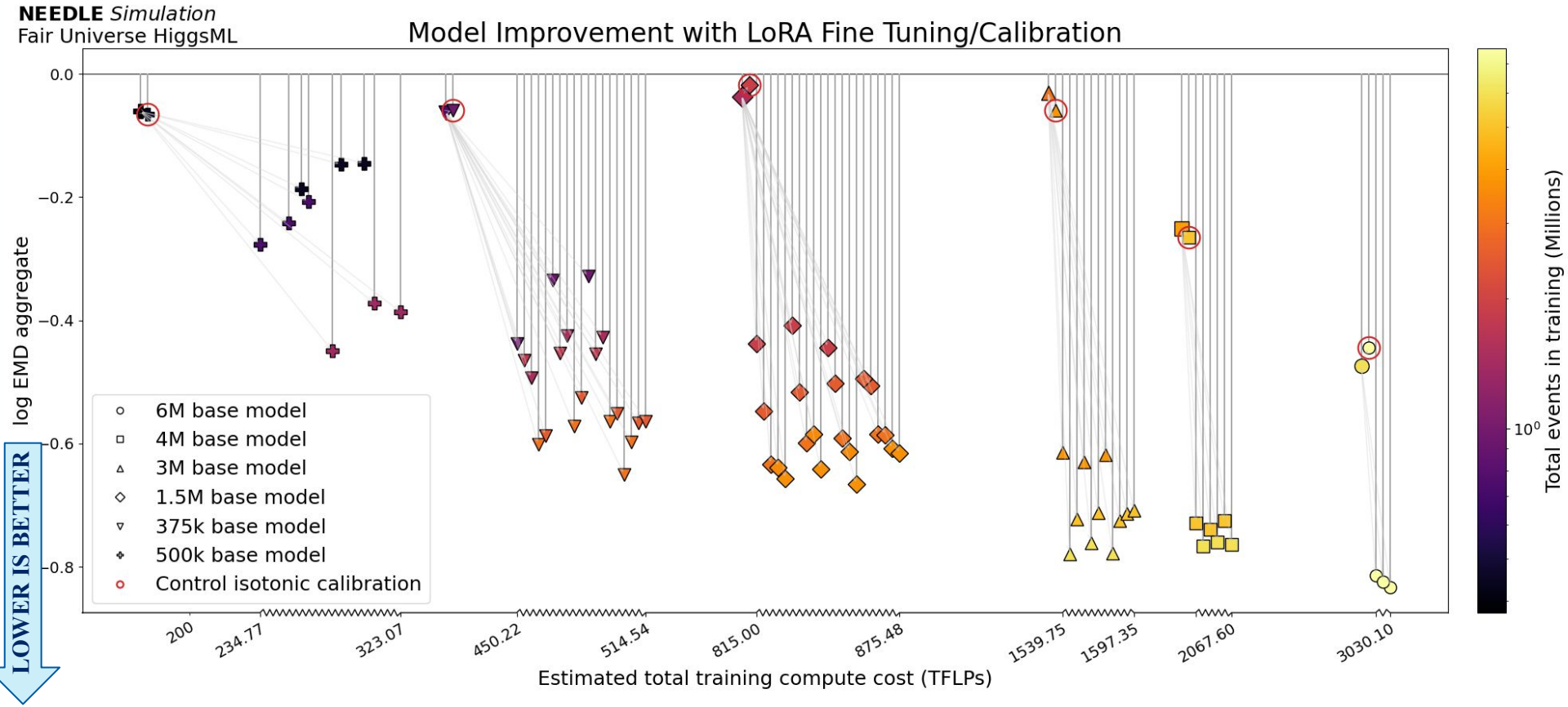
Results: base models versus calibrated/refined variants (AECE)



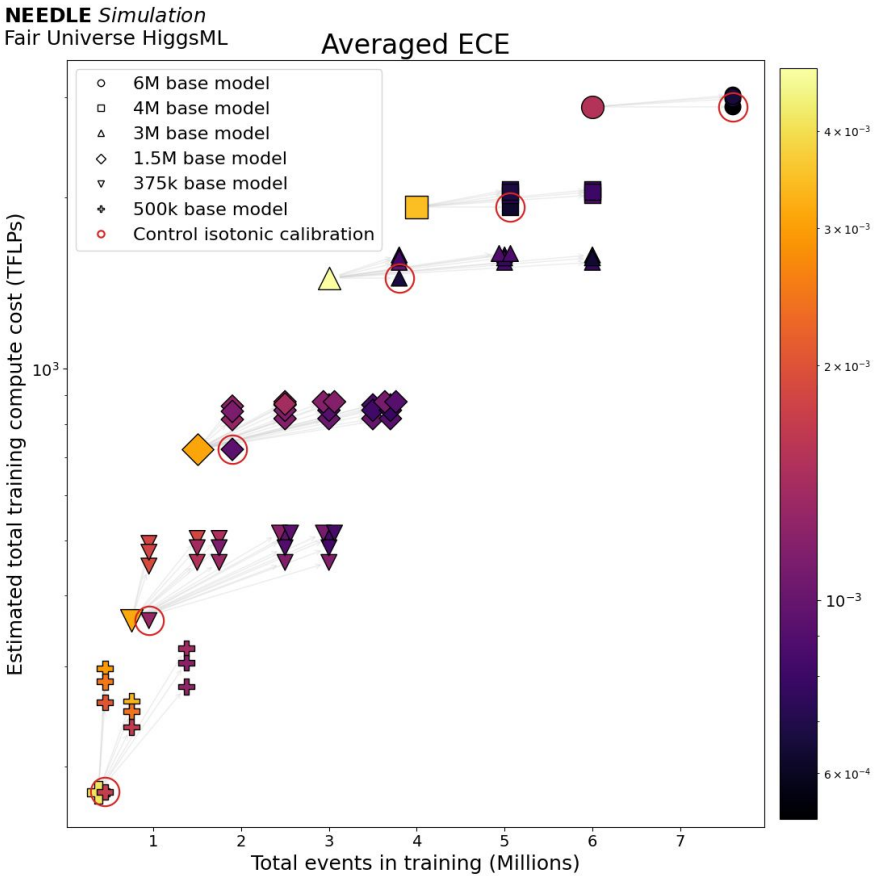
Results: base models versus calibrated/refined variants (reweighting EMD aggregates, minority→majority class)

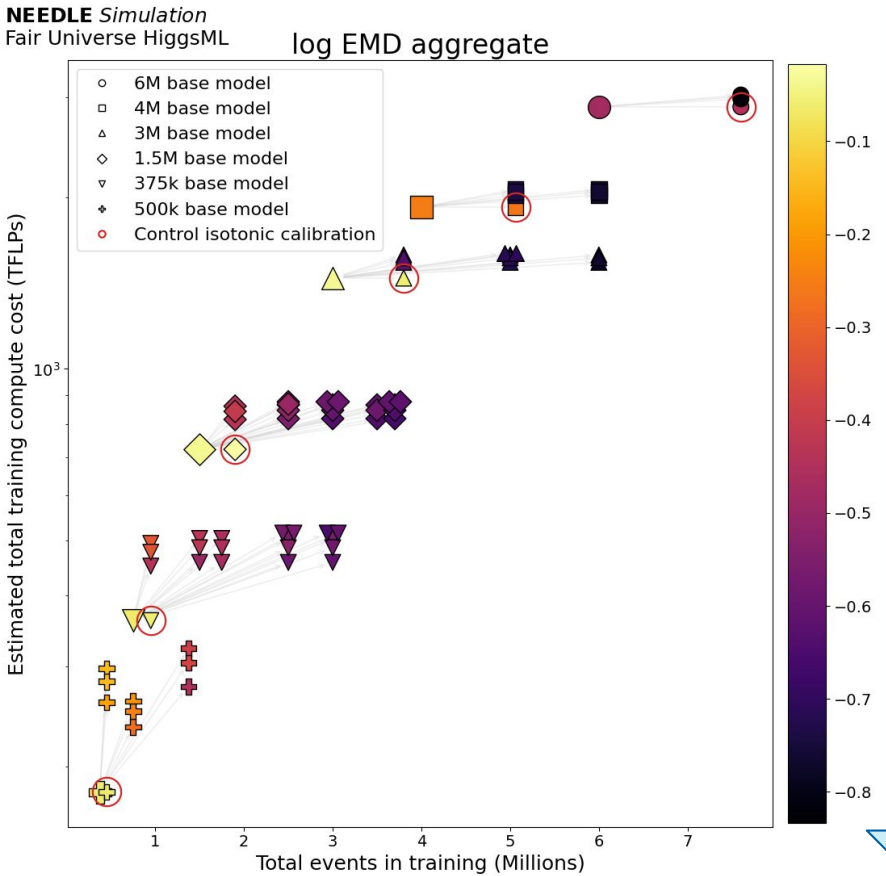


Results: base models versus calibrated/refined variants (reweighting EMD aggregates, majority→minority class)



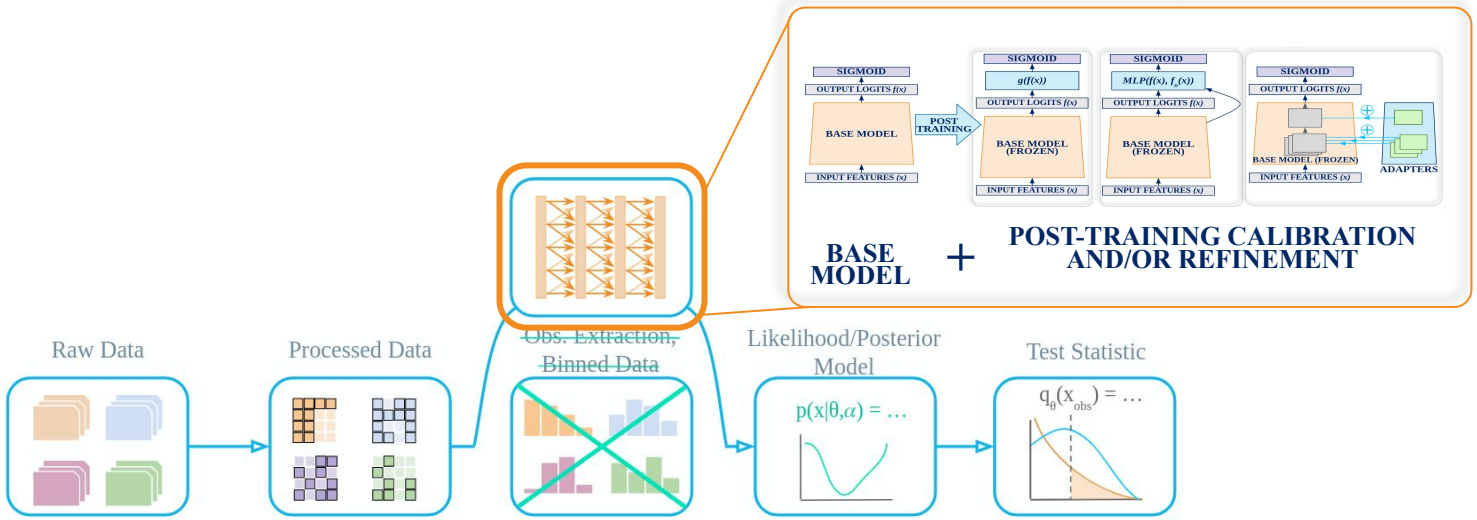
Results: base models versus calibrated/refined variants
A better view of the comp. costs and training events





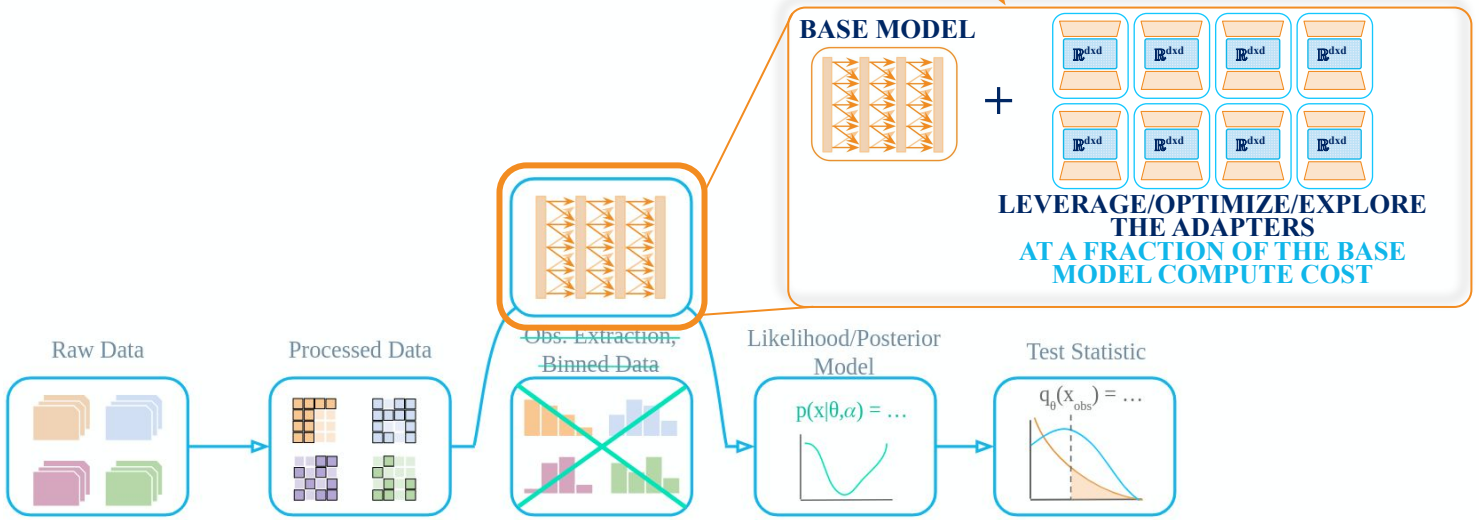
Conclusions

- **Obtaining high-quality, reliable probability ratios can require more than just removing the calibration error**
  - Post-hoc methods cannot reduce grouping loss, unlike methods which have access to the base model
  - One way to determine the effect of grouping loss is to compare model improvement based on post-hoc and refinement methods



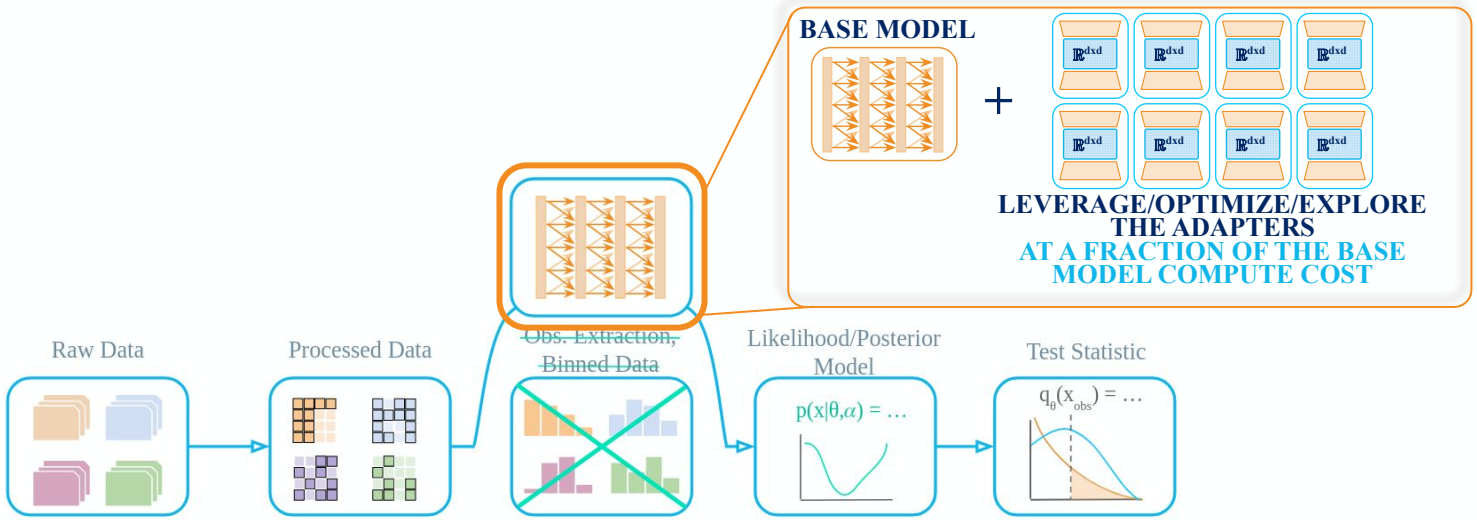
Conclusions

- **Obtaining high-quality, reliable probability ratios can require more than just removing the calibration error**
  - Post-hoc methods cannot reduce grouping loss, unlike methods which have access to the base model
  - One way to determine the effect of grouping loss is to compare model improvement based on post-hoc and refinement methods
- **Adapter-based model modification can also be used to improve a typical NLRE workflow**
  - Observed behaviour also strongly indicates base model training + fine-tuning performs better than one-shot model training



Conclusions

- **Obtaining high-quality, reliable probability ratios can require more than just removing the calibration error**
  - Post-hoc methods cannot reduce grouping loss, unlike methods which have access to the base model
  - One way to determine the effect of grouping loss is to compare model improvement based on post-hoc and refinement methods
- **Adapter-based model modification can also be used to improve a typical NLRE workflow**
  - Observed behaviour also strongly indicates base model training + fine-tuning performs better than one-shot model training
- **Probability ratio estimation also allows us to obtain likelihoods directly via DRDPM**
  - Fast and works even in cases when there's a lack of support between the classes



Stay tuned: many updates and a preprint coming out soon!

Calibration paper preprint

Refining Reliable Likelihood Ratio Estimators: From Calibration To Grouping Loss

Nino Kovačić*, Stephen Jiggins†, Judith Katzy†
*University of Zagreb Faculty of Science,
nkovacic.phy@pmf.hr
†Deutsches Elektronnen-Synchrotron, DESY

needle-sbi CHEP 2026 in-proceedings

NEEDLE: A Columnar Workflow Orchestrator for Large-Scale Neural Simulation-Based Inference in HEP

Kylian Schmidt[1]*, *Felix* Kahlhöfer[1], *Judith* Katzy[2], *Levi* Evans[2], *Nicolò* Trevisani[1], *Nino* Kovacic[3], *Stephen* Jiggins[2], and *Ulrich* Husemann[2]
[1]Karlsruhe Institute of Technology (KIT)
[2]Deutsches Elektronen-Synchrotron (DESY)
[3]University of Zagreb

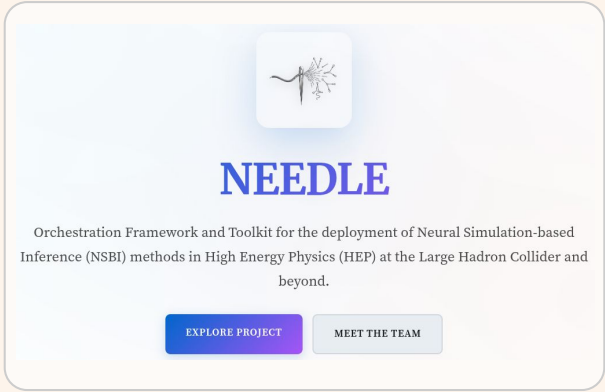
Updates to needle-sbi public release in the pipeline:

- b2luigi and python API for needle-sbi
- NLRE Calibration and validation package
- DRPDM package

Check out our GitHub and Website!

GitHub

Website



Thank you for your attention!

Acknowledgement:
This research was supported by the European Union – NextGenerationEU through the National Recovery and Resilience Plan 2021–2026 Institutional grants of University of Zagreb Faculty of Science
(Project PMF-GOOUT-2026-03 “Calibrated Density Ratio Estimation Approach to Diffusion Probabilistic Models as a Fast Neural Simulation-based Inferencing Method for Large Hadron Collider Physics” as well as
RP3 “quark-gluon plasma research, hardware development at CERN and bioelectronics”)

Funded by the European Union
NextGenerationEU

References

[1] For a general overview of NSBI see Cranmer et al. *The frontier of simulation-based inference*, **ArXiv link**

[2] Cranmer et al. *Approximating Likelihood Ratios with Calibrated Discriminative Classifiers*, **ArXiv link**

[3] Luigi Analysis Workflow, **ArXiv link**

[4] batch 2 luigi, **docs link**

[5] Perez-Lebel et al. *Beyond calibration: estimating the grouping loss of modern neural networks*, **ArXiv link**

[6] Platt *Probabilistic Outputs for Support Vector Machines and Comparisons to Regularized Likelihood Methods*, **ResearchGate link**

[7] Zadrozny and Elkan *Transforming Classifier Scores into Accurate Multiclass Probability Estimates*, **ResearchGate link**, **ACM Digital Library link**

[8] Rahimi et al. P*ost-hoc Calibration of Neural Networks by g-Layers*, **ArXiv link**

[9] Hu et al. LoRA: Low-Rank Adaptation of Large Language Models, **ArXiv link**

[10] Benato et al. *FAIR Universe HiggsML Uncertainty Dataset and Competition*, **Codabench link**, **ArXiv link**

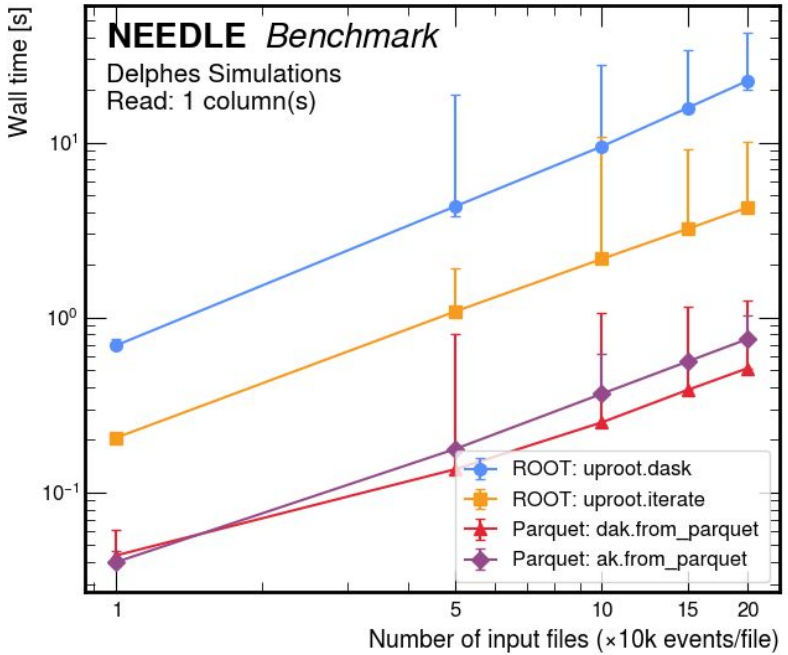
[11] For a general overview of classifier performance and calibration metrics see, e.g., Lane *A comprehensive review of classifier probability calibration metrics,* **ArXiv link**

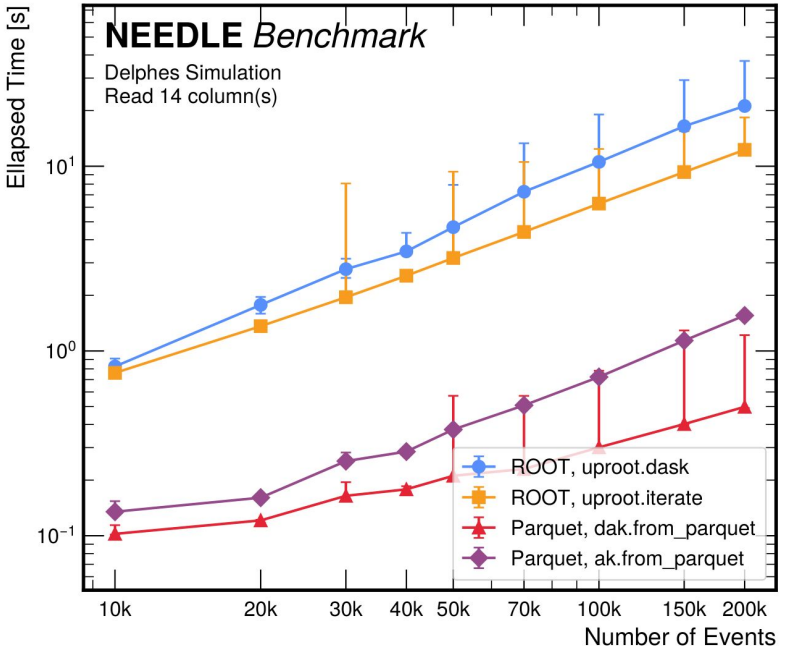
Backup

needle-sbi: data loading speed benchmark

needle-sbi data loading speed benchmarks:

- Reading from ROOT files and from parquet files
- With two variants for parquet: one with and one without parsing in the array shape before data loading (ak-vs-dak*)



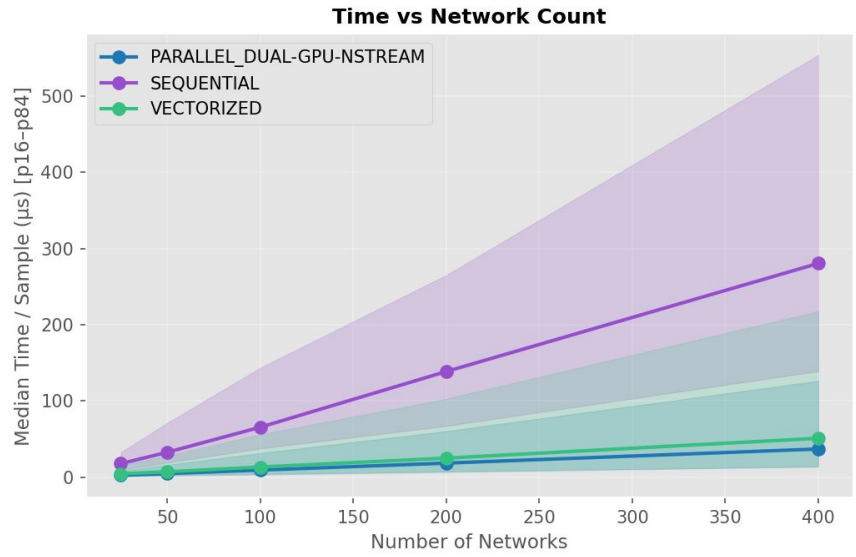


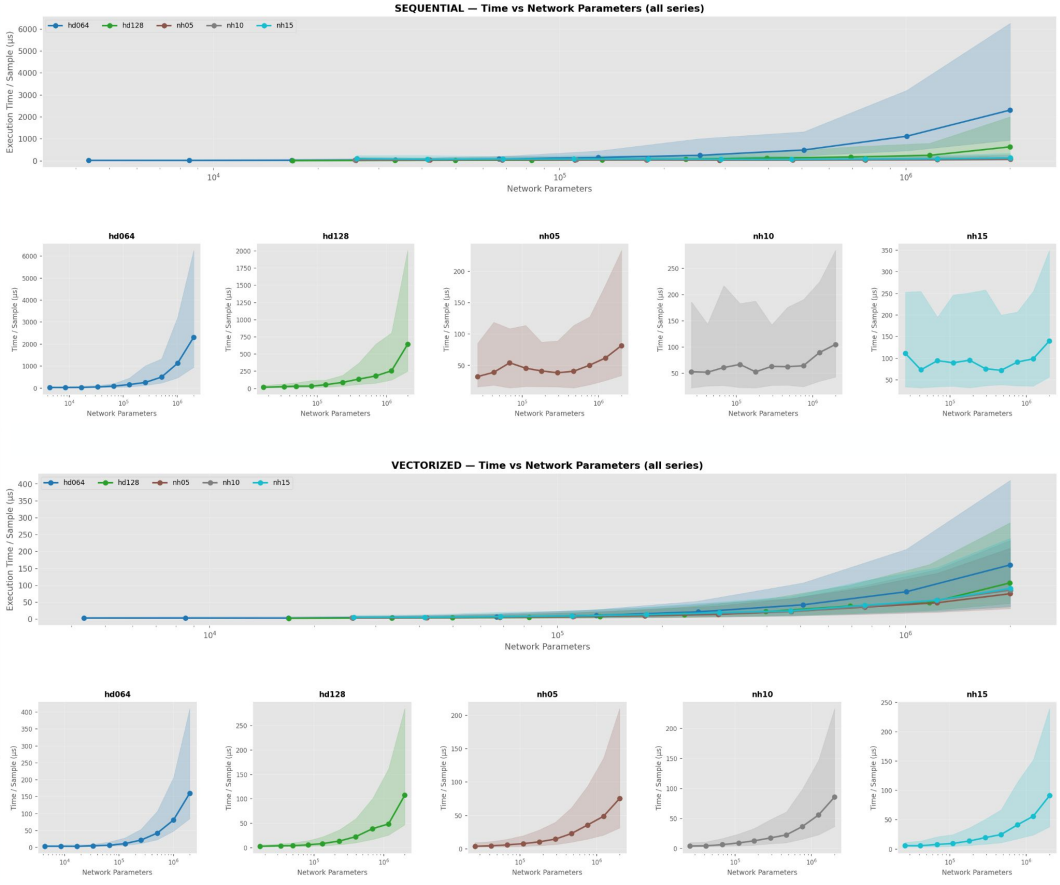
*ak = awkward arrays, dak = dask awkward

needle-sbi: parallel and vectorized model evaluation

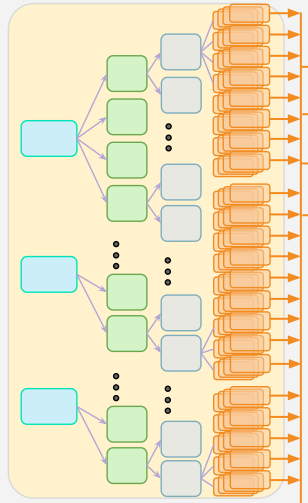
needle-sbi NN evaluation benchmarks:

- Benchmarks performed bare metal on Tesla-V100-SXM2 (32GB)





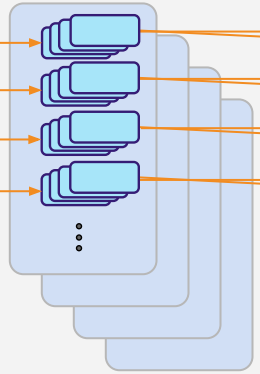
DAG scheme* for NN training and calibration downstream tasks

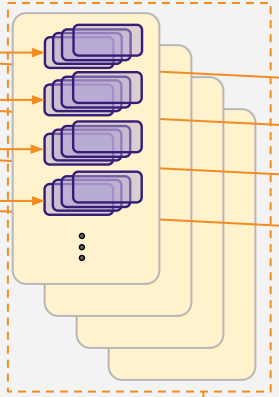


NN training, split over ensembles and folds

- Node = (ensemble id, fold id)

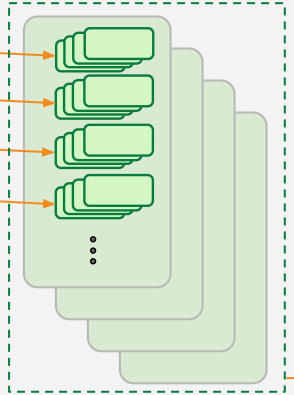
NN validation, per calib./ref. setup, per node

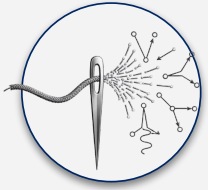




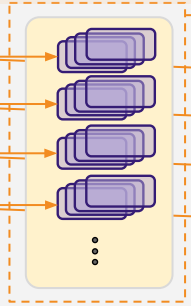
NN calibration/refinement, per node

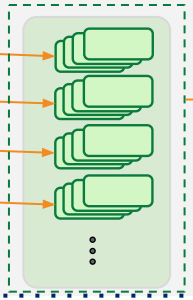
Asimov fits, per calib./ref. setup, per node





NN validation, per node





Validation aggregation

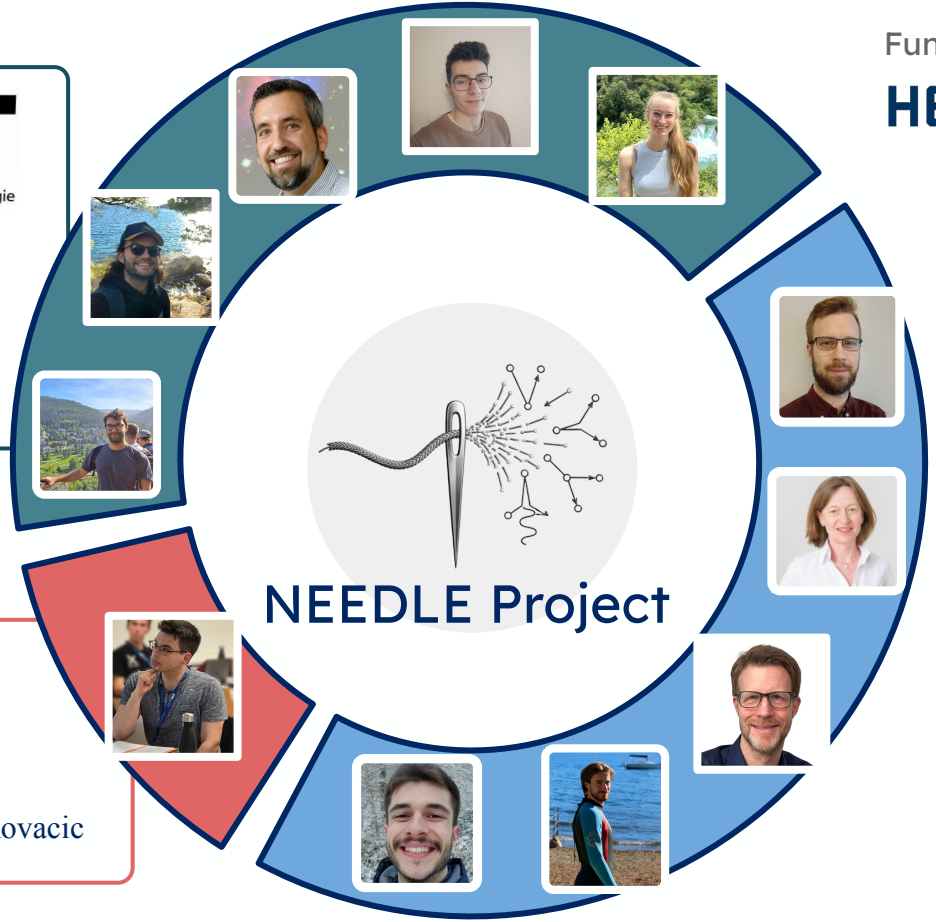
Asimov fits, per node

Aggregation over all results

*Technically what is shown in a sketch of a flow diagram, the DAG structure is inverted compared to this

Funded by

HELMHOLTZ AI



CMS

KIT Karlsruher Institut für Technologie

Lena Rathmann
Niklas Reus
Felix Kahlhöfer
Kylian Schmidt
Nicolò Trevisani

ALICE

Nino Kovacic

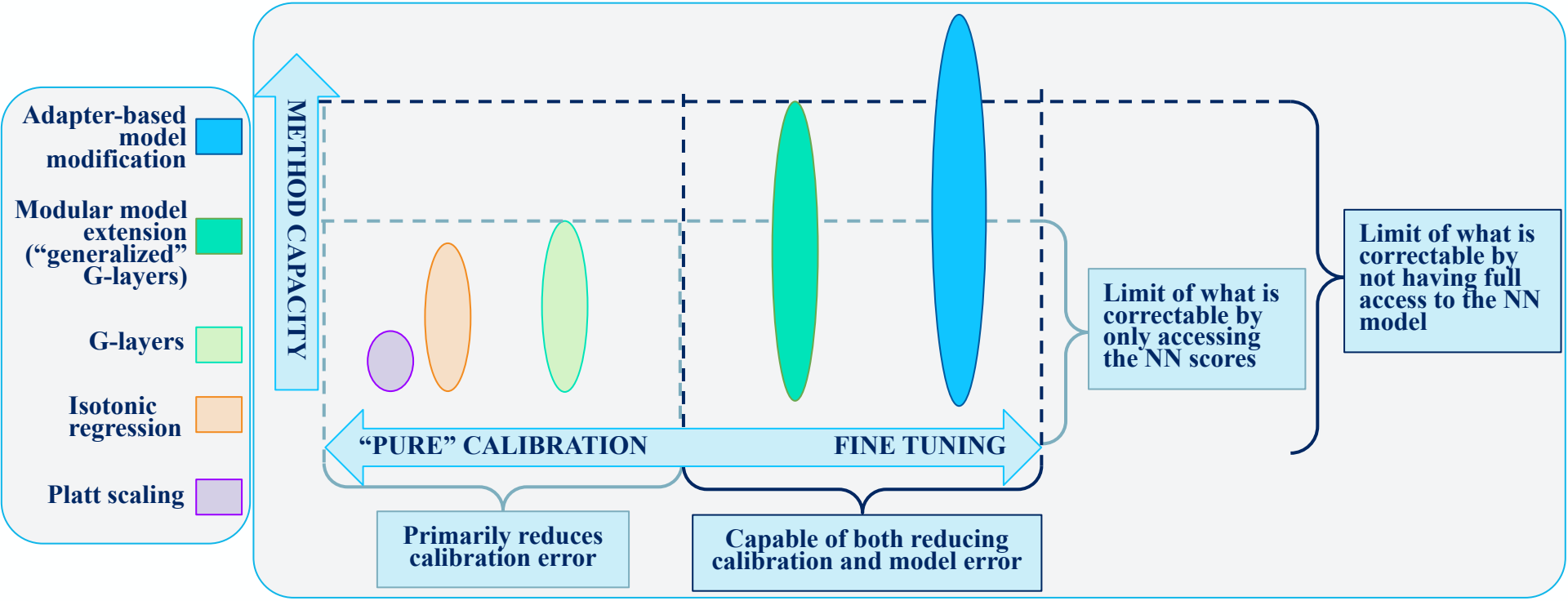
ATLAS EXPERIMENT

DESY.

Stephen Jiggins
Judith Katzy
Ulrich Husemann
Levi Evans
Stefan Katsarov

The “Buffet” of Calibration (Model Improvement) Methods

Sketch of different methods’ hypothetical performance improvement ceilings and base model modification:



Grouping loss, example with Brier score

$$\mathbb{E}\big[(S-Y)^2\big] = \mathbb{E}\big[\big((S-S^\star)+(S^\star-Y)\big)^2\big] = \mathbb{E}\big[(S-S^\star)^2\big] + \mathbb{E}\big[(S^\star-Y)^2\big] + 2\,\mathbb{E}\big[(S-S^\star)(S^\star-Y)\big]$$

$$\mathbb{E}\big[(S^\star-Y)^2\big] = \mathbb{E}\big[\mathrm{Var}(Y\mid X)\big] = \mathbb{E}\big[S^\star(1-S^\star)\big]$$

$$\mathbb{E}\big[(S-S^\star)(S^\star-Y)\big] = \mathbb{E}\Big[(S-S^\star)\,\underbrace{\mathbb{E}\big[S^\star - Y \mid X\big]}_{=0}\Big] = 0$$

$$= \mathbb{E}\big[(S-S^\star)^2\big] + \mathbb{E}\big[S^\star(1-S^\star)\big]$$

$$\mathbb{E}\big[(S-S^\star)^2\big] = \mathbb{E}\big[(S-\bar{S})^2\big] + \mathbb{E}\big[(\bar{S}-S^\star)^2\big] + 2\,\mathbb{E}\big[(S-\bar{S})(\bar{S}-S^\star)\big]$$

$$\mathbb{E}\big[(S-\bar{S})(\bar{S}-S^\star)\big] = \mathbb{E}\Big[\mathbb{E}\big[(S-\bar{S})(\bar{S}-S^\star)\mid S\big]\Big]$$
$$= \mathbb{E}\Big[(S-\bar{S})\,\mathbb{E}\big[\bar{S}-S^\star \mid S\big]\Big]$$
$$= \mathbb{E}\Big[(S-\bar{S})\big(\bar{S}-\mathbb{E}[S^\star\mid S]\big)\Big]$$
$$= \mathbb{E}\big[(S-\bar{S})(\bar{S}-\bar{S})\big] = 0$$

$$\Longrightarrow \underbrace{\mathbb{E}\big[(S-Y)^2\big]}_{\text{Brier}} = \underbrace{\mathbb{E}\big[S^\star(1-S^\star)\big]}_{\text{irreducible,IL}} + \underbrace{\mathbb{E}\big[(S^\star-\bar{S})^2\big]}_{\text{grouping,GL}} + \underbrace{\mathbb{E}\big[(\bar{S}-S)^2\big]}_{\text{calibration,CL}}$$

Where S is the classifier under scrutiny
$S^\star \equiv \mathbb{E}\big[Y \mid X\big]$ **is the Bayes-optimal classifier,**
$\bar{S} \equiv \mathbb{E}\big[Y \mid S\big]$ **is the ideally calibrated classifier**

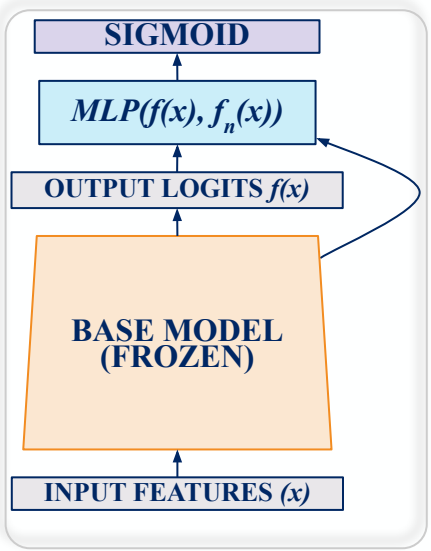
$$\underbrace{Y}_{\text{label}} \longrightarrow \underbrace{\mathbb{E}[Y|X] = S^\star}_{\text{best possible}} \longrightarrow \underbrace{\mathbb{E}[Y|S] = \bar{S}}_{\text{best from the score}} \longrightarrow \underbrace{S}_{\text{what the network outputs}}$$

$$\bar{S} = \mathbb{E}\big[S^\star \mid S\big] \Longrightarrow \mathrm{GL_B} = \mathbb{E}\big[\mathrm{Var}\big(S^\star \mid S\big)\big]$$

A bit more on “generalized” g-layers

- **Upsides:**
  - Cheaper to train than LoRAs, but also has the ability to reduce grouping error
  - Due to their implementation they can “more sharply” adjust the base model behavior
  - Not in competition with LoRA, but rather serves as complementary method:
    - When a large, but smooth correction is needed, LoRAs shine
    - When a sharp refinement is needed, generalized g-layers are better (and more data efficient)

- **Downsides:**
  - Prone to overfitting, needs careful control of parameters
  - If the damage to the model is done “early on”, i.e. is the errors stem from the attention blocks, then this method can’t correct much



Asimov fit setup in bit more detail

Model and likelihood:

$$\nu_c(x) = c\,\nu_1(x) + \nu_0(x)$$

$$\frac{\nu_c(x)}{\nu_0(x)} = 1 + c\,r(x), \qquad r(x) = \frac{\nu_1(x)}{\nu_0(x)} = \frac{s(x)}{1-s(x)} = e^{f(x)}$$

$$\ln L(c) = c_{\text{true}} \sum_{i \in \text{class1}} w_i \ln\left(1 + c\,r_i\right) + \sum_{i \in \text{class0}} w_i \ln\left(1 + c\,r_i\right) - c\,W_1 + \text{const}$$

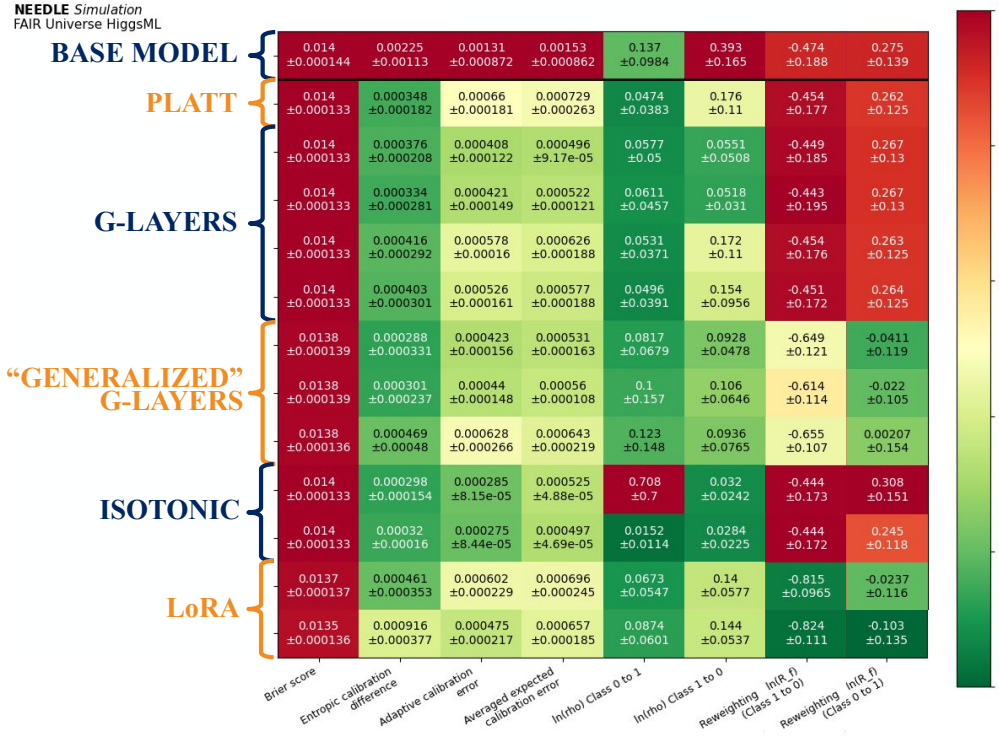
Saturation into a “counting experiment", simplified explanation

$$\frac{\partial}{\partial c} \ln\left(1 + c\,r\right) = \frac{r}{1 + c\,r} \xrightarrow{c\,r \gg 1} \frac{1}{c}$$

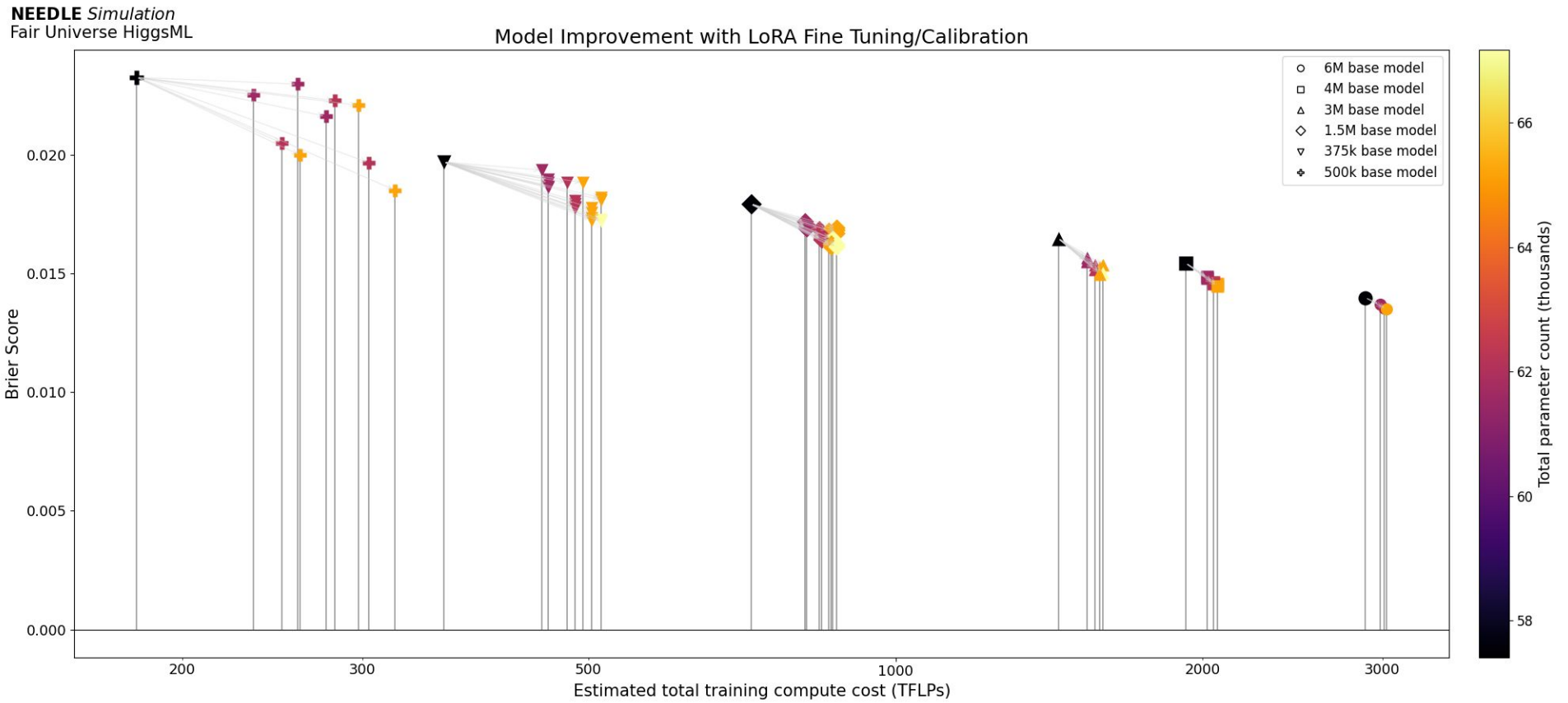
Connection to Fisher information

$$I(c_{\text{true}}) = \int \nu_0 \frac{r^2}{1 + c_{\text{true}}\,r} \xrightarrow{c_{\text{true}} \to 0} \int \nu_0\, r^2$$

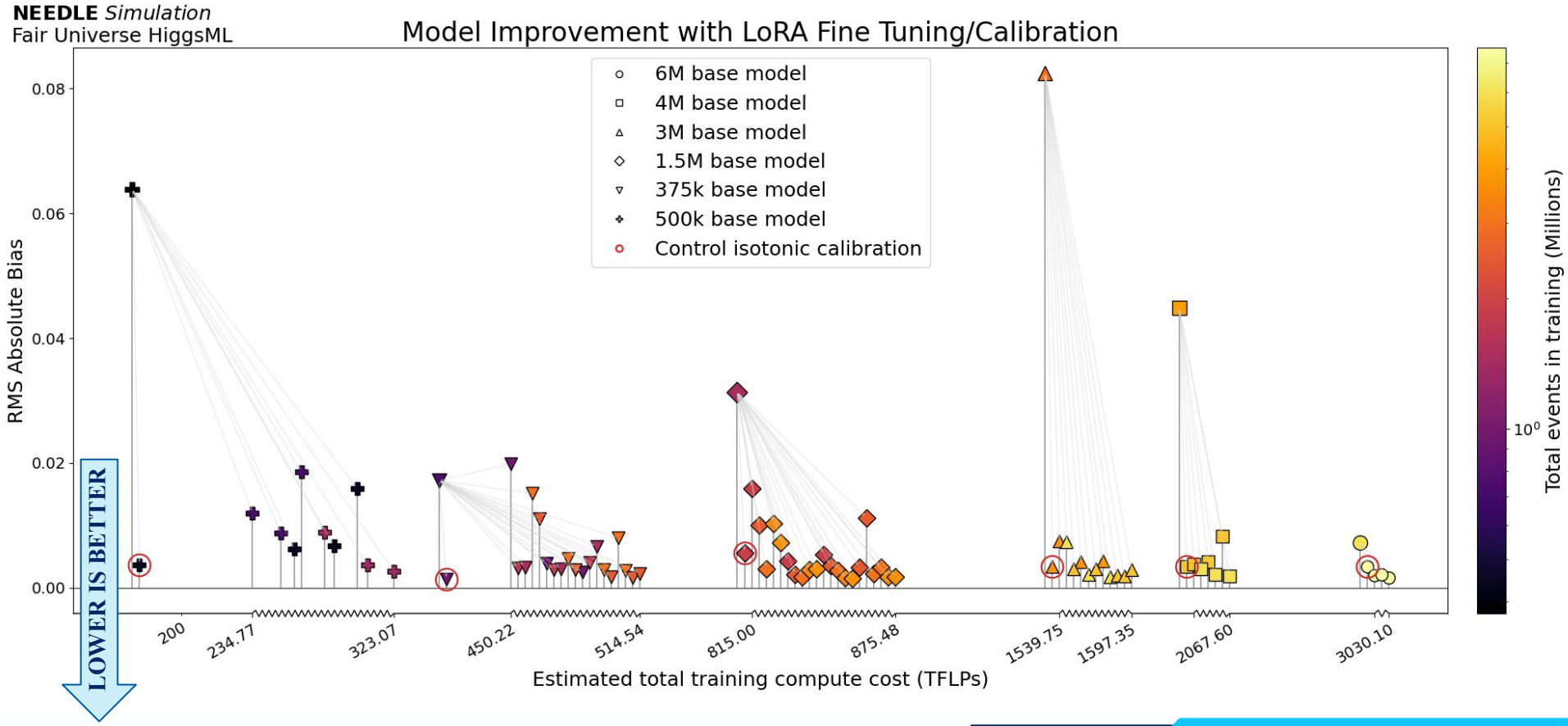
Bonus: diagnosing your model with post-calibration performance checks (calibration/refinement as a form of validation!)



Results: base models versus calibrated/refined variants (Brier score), true x-axis, with total model parameter counts



Results: base models versus calibrated/refined variants (asimov fit aggregate)



More on NLRE: Density Ratio Approach to Diffusion Probabilistic Models

Density Ratio Approach to Diffusion Probabilistic Models: bridging the gap

Where NLRE falls short:

- If there is a clear separation boundary between the two classes (in some phase-space sub-region)*
- You have to perform tricks to get to the likelihood ("to get rid of the ratio")

But, NLRE has very good properties:

- Overall good training+inference speed
- "Simple" training task

*This "lack of support" issue is very well known in NLRE. We're simplifying the discussion on this point (also in relation to NLE) in the interest of time

Density Ratio Approach to Diffusion Probabilistic Models: bridging the gap

Where NLRE falls short:

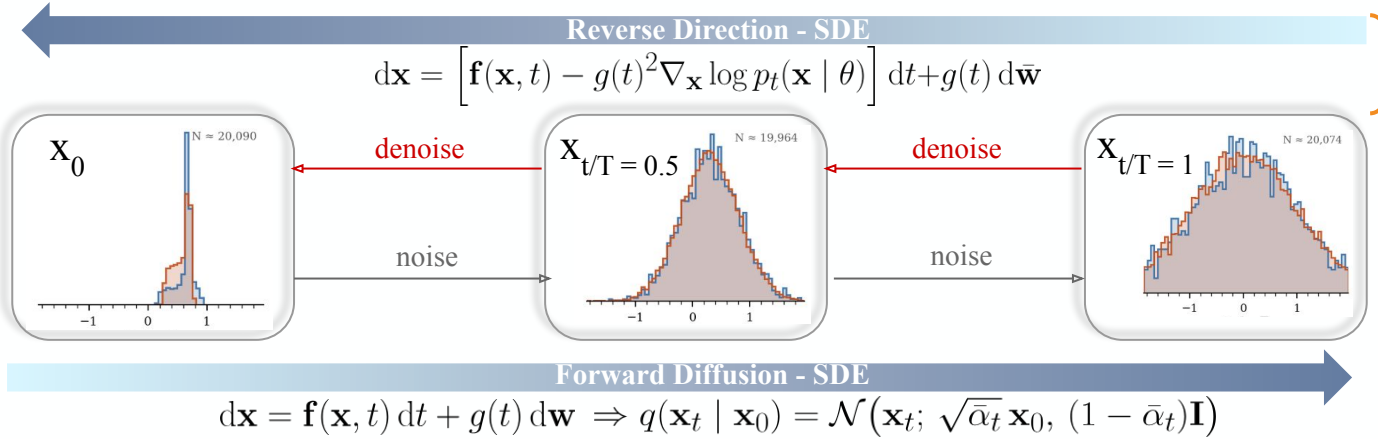
- If there is a clear separation boundary between the two classes (in some phase-space sub-region)
- You have to perform tricks to get to the likelihood ("to get rid of the ratio")

But, NLRE has very good properties:

- Overall good training+inference speed
- "Simple" training task

Where generative NLE approaches fall short:

- Either large total training+inference compute cost*



*Here we give a diffusion probabilistic model as an example. The compute cost/speed argument is very nuanced, and we simplify in the interest of time.

Density Ratio Approach to Diffusion Probabilistic Models: bridging the gap

Where NLRE falls short:

- If there is a clear separation boundary between the two classes (in some phase-space sub-region)
- You have to perform tricks to get to the likelihood ("to get rid of the ratio")

But, NLRE has very good properties:

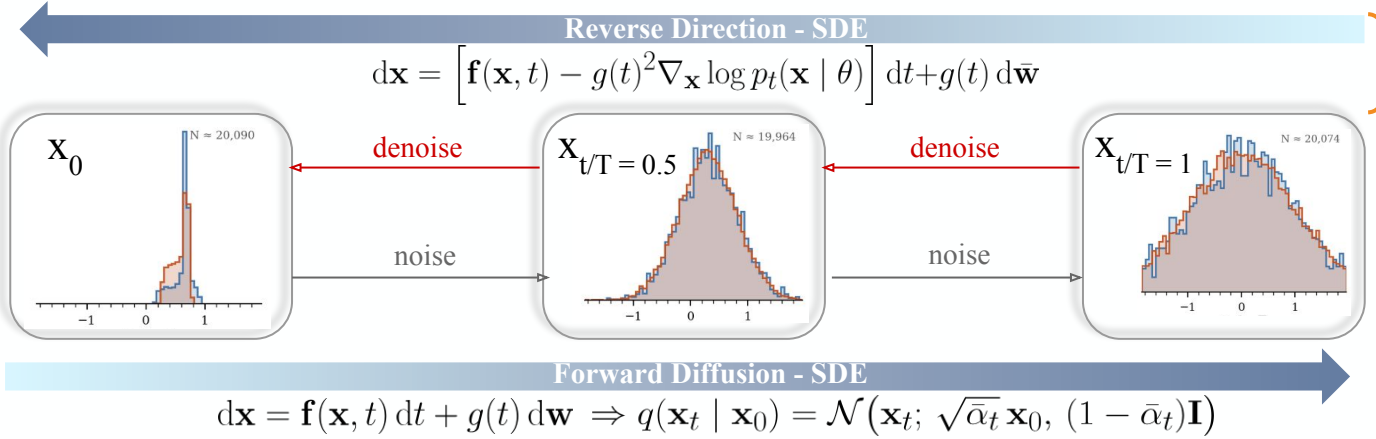
- Overall good training+inference speed
- "Simple" training task

Where generative NLE approaches fall short:

- Either large total training+inference compute cost*
  or
- Architecture limitations

But, generative NLE:

- Has no problems if an inter-sample separation boundary exists
- Gives you the likelihood outright



*Here we give a diffusion probabilistic model as an example. The compute cost/speed argument is very nuanced, and we simplify in the interest of time.

Density Ratio Approach to Diffusion Probabilistic Models: bridging the gap

Where NLRE falls short:

- If there is a clear separation boundary between the two classes (in some phase-space sub-region)
- You have to perform tricks to get to the likelihood ("to get rid of the ratio")

But, NLRE has very good properties:

- Overall good training+inference speed
- "Simple" training task

Where generative NLE approaches fall short:

- Either large total training+inference compute cost*
  or
- Architecture limitations

But, generative NLE:

- Has no problems if an inter-sample separation boundary exists
- Gives you the likelihood outright

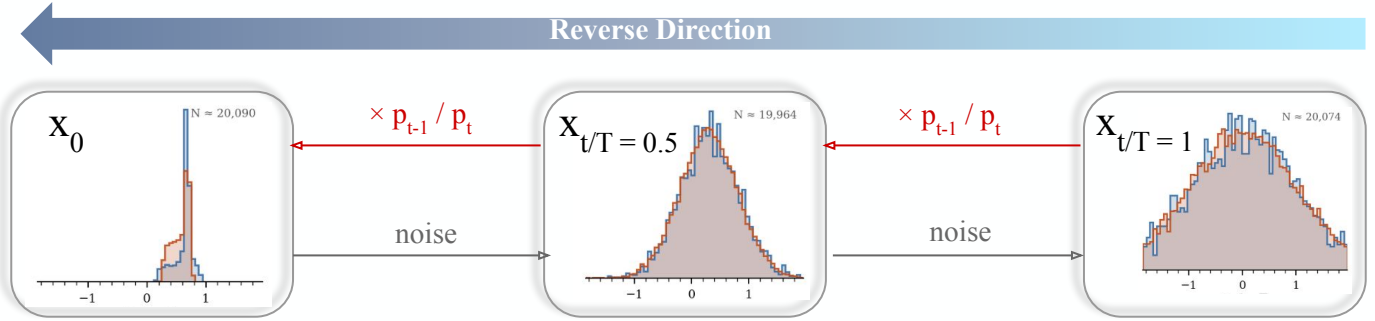
What DRDPM (Density Ratio Approach to Diffusion Probabilistic Models) achieves?

- Takes "best of both worlds": uses probability ratios to directly compute the likelihood
- Primarily focused on speed

- DRDPM still needs its probability ratios to be of very high quality!
- Thus, the same calibration/refinement arguments as for NLRE apply

*Here we give a diffusion probabilistic model as an example. The compute cost/speed argument is very nuanced, and we simplify in the interest of time.

DRDPM: How it works



$$\mathrm{d}\mathbf{x} = \mathbf{f}(\mathbf{x}, t)\,\mathrm{d}t + g(t)\,\mathrm{d}\mathbf{w} \;\Rightarrow\; q(\mathbf{x}_t \mid \mathbf{x}_0) = \mathcal{N}\left(\mathbf{x}_t;\ \sqrt{\bar{\alpha}_t}\,\mathbf{x}_0,\ (1 - \bar{\alpha}_t)\mathbf{I}\right)$$

DRDPM classifier NN gives you:

Class conditional probability

$$p(t|x, \theta) = \mathrm{softmax}(z(x, t))$$

A softmax over diffusion noise levels:
"*given x, say which time step t it came from*"

In inference, only 1 forward pass is needed

What the DRDPM likelihood needs:

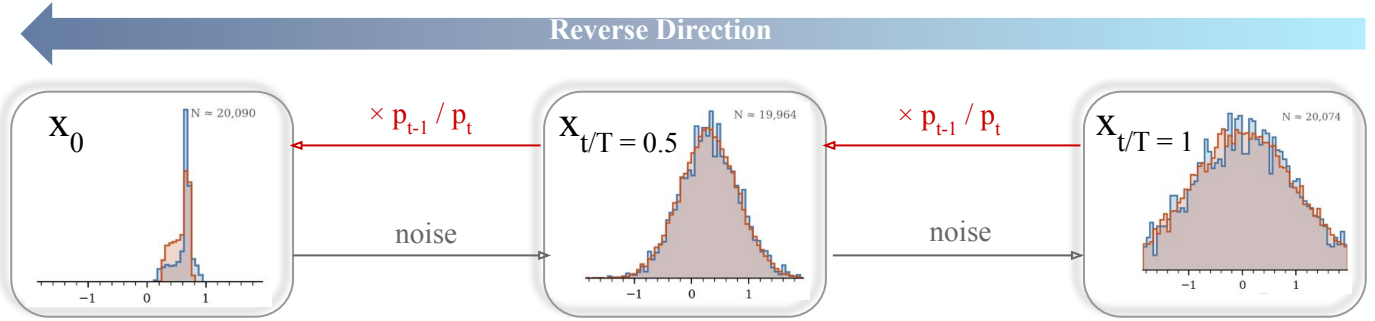
Only the difference of two logits

$$\log\left(p(x_t|\theta)\right) \approx \log\left(p(t|x_t, \theta)\right) - \log\left(p(T|x_t, \theta)\right)$$

No Laplacian, no Jacobian trace, no hutchinson estimator calculation is required

0 backward passes in inference

DRDPM: How it works



$$\mathrm{d}\mathbf{x} = \mathbf{f}(\mathbf{x}, t)\,\mathrm{d}t + g(t)\,\mathrm{d}\mathbf{w} \;\Rightarrow\; q(\mathbf{x}_t \mid \mathbf{x}_0) = \mathcal{N}\left(\mathbf{x}_t;\ \sqrt{\bar{\alpha}_t}\,\mathbf{x}_0,\ (1 - \bar{\alpha}_t)\mathbf{I}\right)$$

DRDPM classifier NN gives you:

Class conditional probability

$$p(t|x, \theta) = \mathrm{softmax}(z(x, t))$$

A softmax over diffusion noise levels:
"*given x, say which time step t it came from*"

In inference, only 1 forward pass is needed

What the DRDPM likelihood needs:

Only the difference of two logits

$$\log\left(p(x_t|\theta)\right) \approx \log\left(p(t|x_t, \theta)\right) - \log\left(p(T|x_t, \theta)\right)$$

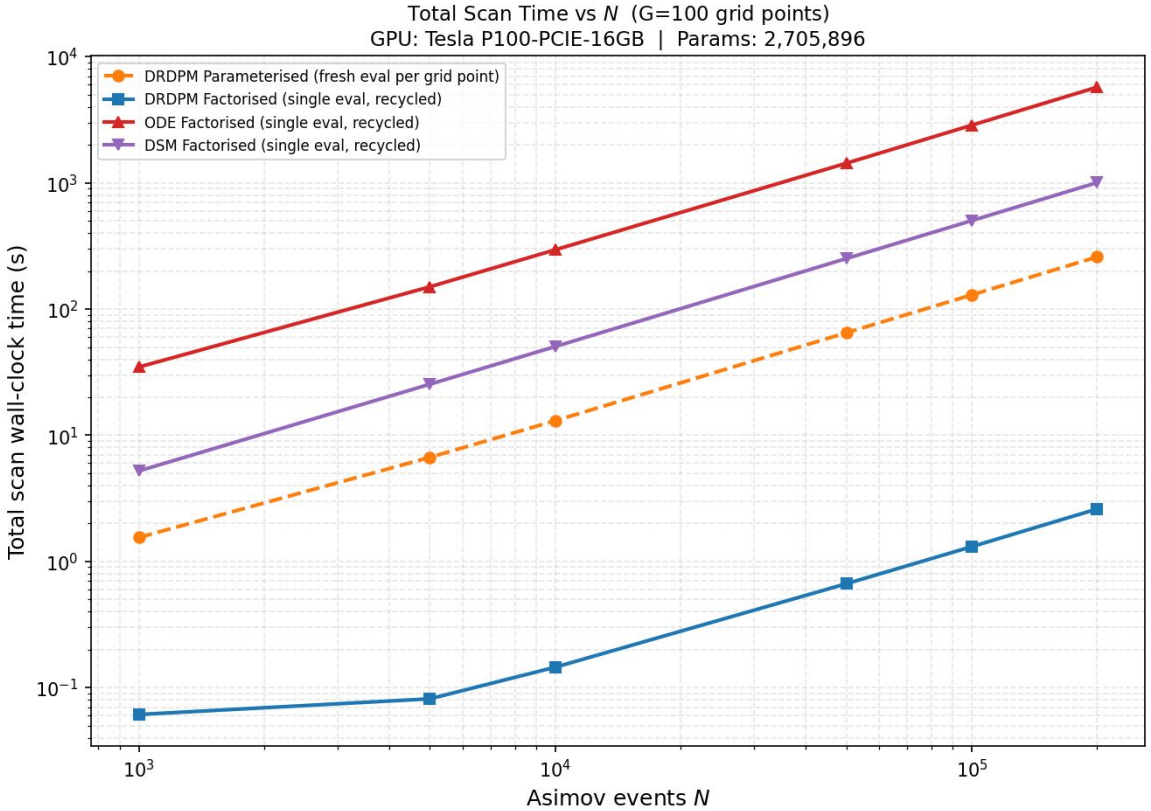
No Laplacian, no Jacobian trace, no hutchinson estimator calculation is required

0 backward passes in inference

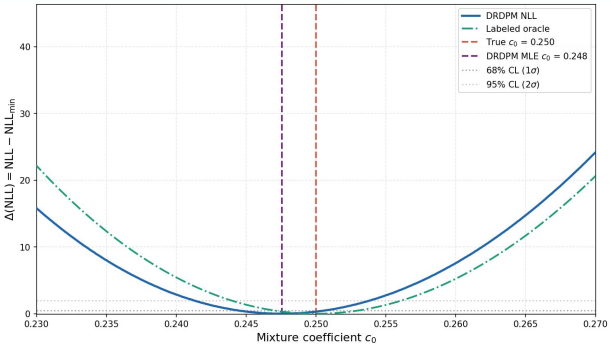
Cost of a likelihood scan

~~T diffusion steps ×~~ **1** × ~~~K solver evaluations~~ **1** × ~~(forward+backward) passes~~ **1** × **N** events × **G** scan points in θ

DRDPM: Inference speed

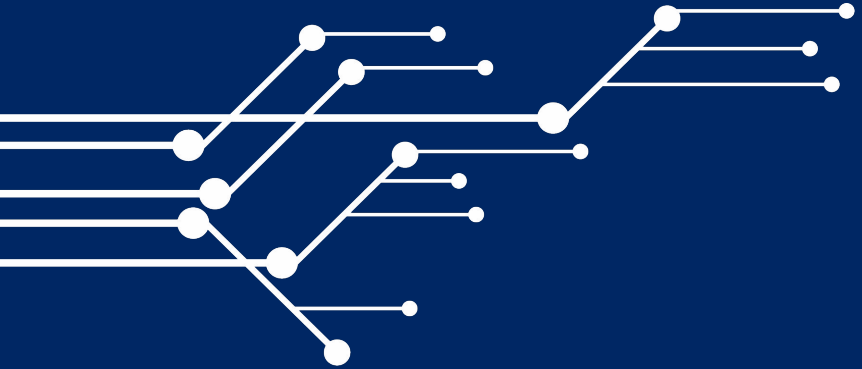


And it works on asimov signal injection fits! (see backup)



Cost of a likelihood scan

~~T diffusion steps~~ × **1** × ~~K solver evaluations~~ **1** × ~~(forward+backward) passes~~ **1** × **N** events × **G** scan points in θ

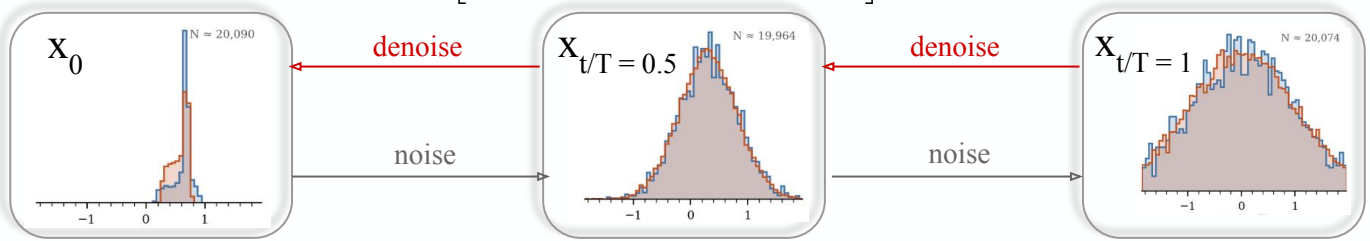


Backup: DRDPM

Diffusion probabilistic models, inference cost

Reverse Direction - SDE

$$\mathrm{d}\mathbf{x} = \left[\mathbf{f}(\mathbf{x}, t) - g(t)^2 \nabla_{\mathbf{x}} \log p_t(\mathbf{x} \mid \theta)\right] \mathrm{d}t + g(t)\, \mathrm{d}\bar{\mathbf{w}}$$



Forward Diffusion - SDE

$$\mathrm{d}\mathbf{x} = \mathbf{f}(\mathbf{x}, t)\, \mathrm{d}t + g(t)\, \mathrm{d}\mathbf{w} \;\Rightarrow\; q(\mathbf{x}_t \mid \mathbf{x}_0) = \mathcal{N}\left(\mathbf{x}_t;\ \sqrt{\bar{\alpha}_t}\, \mathbf{x}_0,\ (1 - \bar{\alpha}_t)\mathbf{I}\right)$$

What the network gives you

A Gradient

$$\nabla_{\mathbf{x}} \log p_t(\mathbf{x} \mid \theta) = -\frac{\boldsymbol{\epsilon}_\phi(\mathbf{x}_t, t)}{\sqrt{1 - \bar{\alpha}_t}}$$

Denoising score matching learns the **first** derivative of the log-density, a vector with *d* components.

1 forward pass

What the likelihood needs

Divergence →differentiate the network

$$\nabla^2_{\mathbf{x}} \log p_t(\mathbf{x} \mid \theta) \;\approx\; -\frac{1}{\sqrt{1 - \bar{\alpha}_t}}\, \mathbf{u}^\top \left(\frac{\partial \boldsymbol{\epsilon}_\phi(\mathbf{x}, t)}{\partial \mathbf{x}}\right) \mathbf{u}$$

∇^2 is the **Laplacian** obtained by back-propagating through the score.

+ 1 backward pass

What needs to be solved

An ODE trajectory

$$\log p_0(\mathbf{x}_0) \;=\; \log p_T(\mathbf{x}_T) - \tfrac{1}{2} \int_0^T \beta(t) \left[\, d + \nabla^2_{\mathbf{x}} \log p_t(\mathbf{x}_t \mid \theta) \,\right] \mathrm{d}t$$

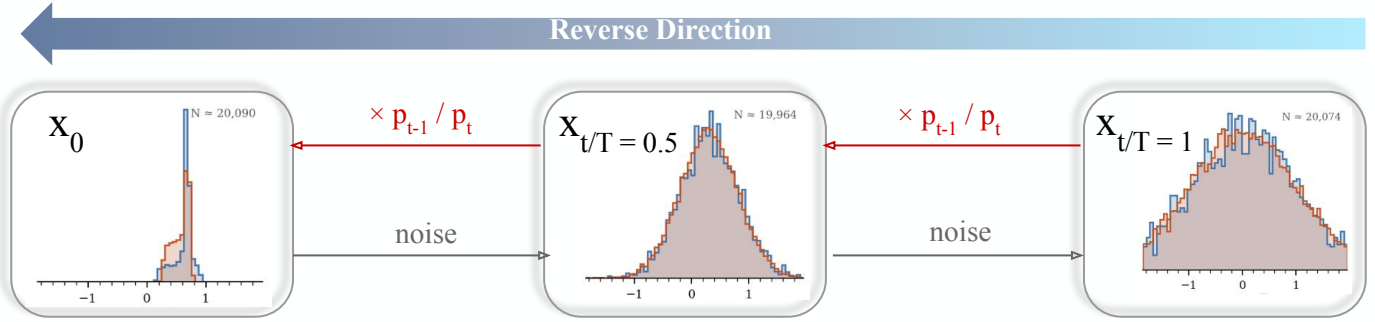
An adaptive solver takes **O(T) evaluations**, and runs **K steps** in solver.

× every solver step

Cost of one likelihood scan

T diffusion steps × **K** solver evaluations × (forward+backward) pass × **N** events × **G** scan points in $\boldsymbol{\theta}$

DRDPM, inference cost



$$\mathrm{d}\mathbf{x} = \mathbf{f}(\mathbf{x}, t)\,\mathrm{d}t + g(t)\,\mathrm{d}\mathbf{w} \;\Rightarrow\; q(\mathbf{x}_t \mid \mathbf{x}_0) = \mathcal{N}\left(\mathbf{x}_t;\ \sqrt{\bar{\alpha}_t}\,\mathbf{x}_0,\ (1 - \bar{\alpha}_t)\mathbf{I}\right)$$

DRDPM classifier NN gives you:

Class conditional probability

$$p(t|x, \theta) = \mathrm{softmax}(z(x, t))$$

A softmax over diffusion noise levels:
"*given x, say which time step t it came from*"

In inference, only 1 forward pass is needed

What the DRDPM likelihood needs:

Only the difference of two logits

$$\log\left(p(x_t|\theta)\right) \approx \log\left(p(t|x_t, \theta)\right) - \log\left(p(T|x_t, \theta)\right)$$

No Laplacian, no Jacobian trace, no hutchinson estimator calculation is required

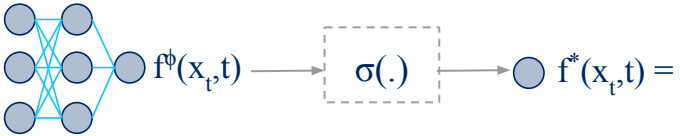
0 backward passes in inference

Cost of a likelihood scan

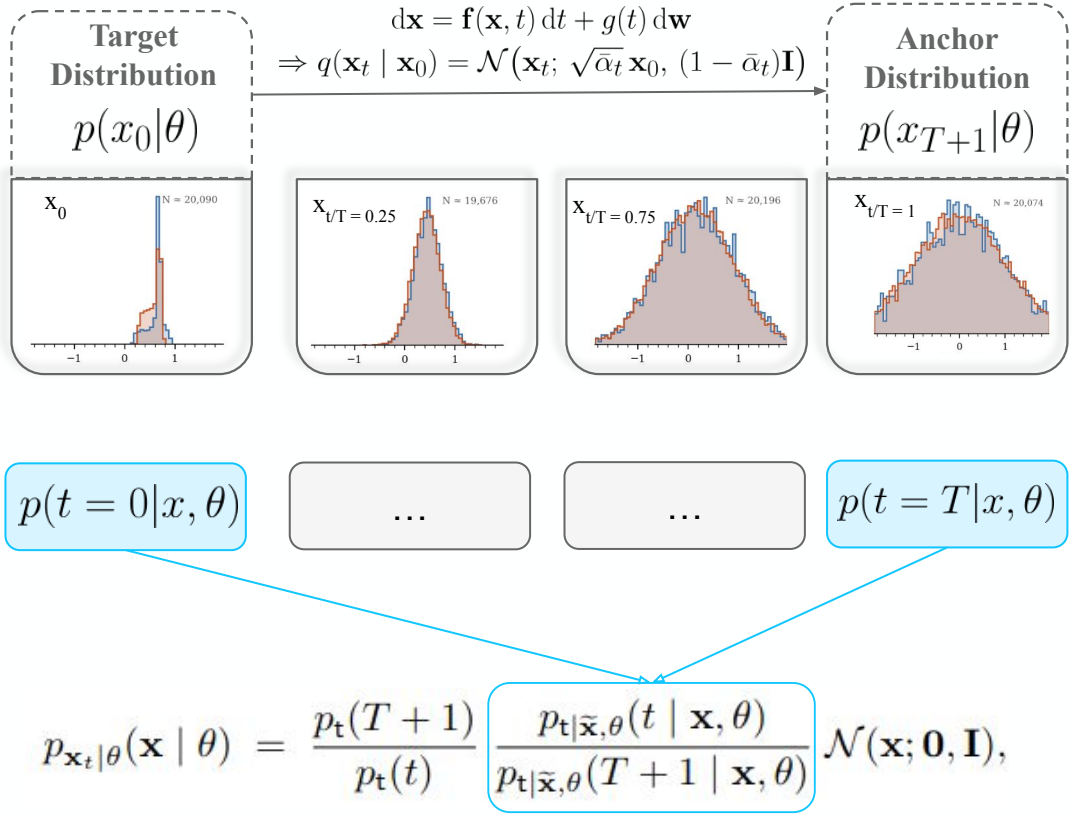
~~**T** diffusion steps ×~~ **1** × ~~~K solver evaluations~~ **1** × ~~(forward+backward) passes~~ **1** × **N** events × **G** scan points in θ

DRDPM: more details on the model used and method basics

- **Forward Diffusion Process:**
  - Acts as an interpolation strategy between the target distribution $p(x_0 \mid \theta)$ and a **Gaussian anchor** $p(x_T \mid \theta)$
- **Neural Surrogate:**
  - The neural surrogate learns a **time step** multi-classifier:



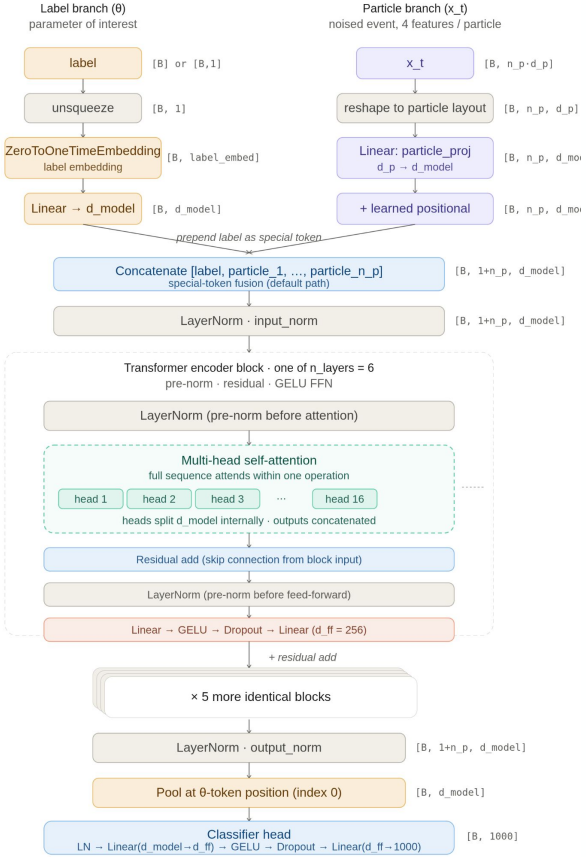
- **Likelihood:**
  - The likelihood can be extracted analytically from the Gaussian probability path by the **density ratio** of the end logits **t=0 & t=T**



DRDPM: more details on the model used and method basics

- **Forward Diffusion Process:**
  - Acts as an interpolation strategy between the target distribution $p(x_0 \mid \theta)$ and a **Gaussian anchor** $p(x_T \mid \theta)$

- **Neural Surrogate:**
  - The neural surrogate learns a **time step** multi-classifier:

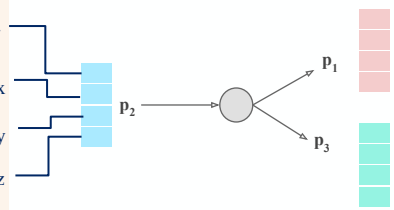


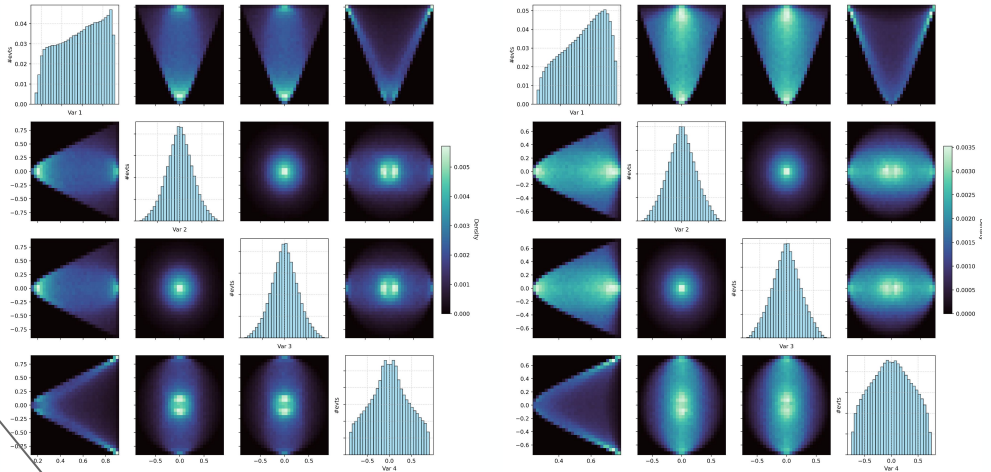
- **Likelihood:**
  - The likelihood can be extracted analytically from the Gaussian probability path by the **density ratio** of the end logits **t=0 & t=T**

DRDPM: 3-body decay dataset details

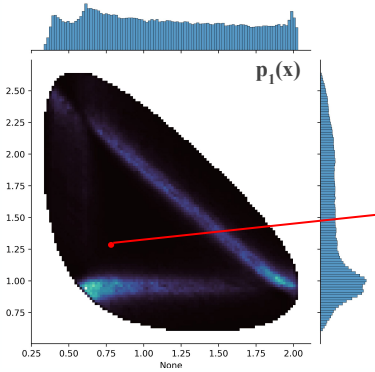
- **Monte Carlo -** 12-Dim. Generator Space
  - Generate 3-particles with 4-vector information:

Energy: E
Momentum *x*-axis: p_x
Momentum *y*-axis: p_y
Momentum z-axis: p_z



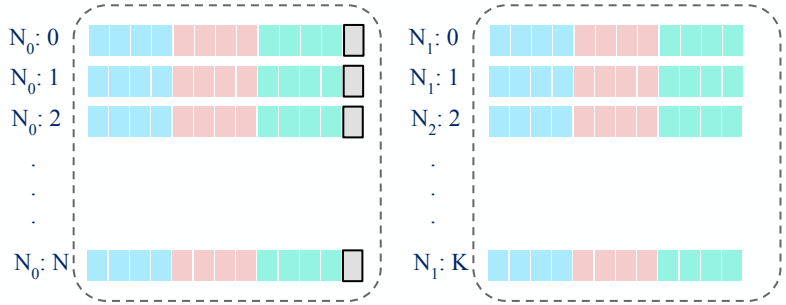


  - Record matrix elements of each sample point in the (s_{12}, s_{23}) space:



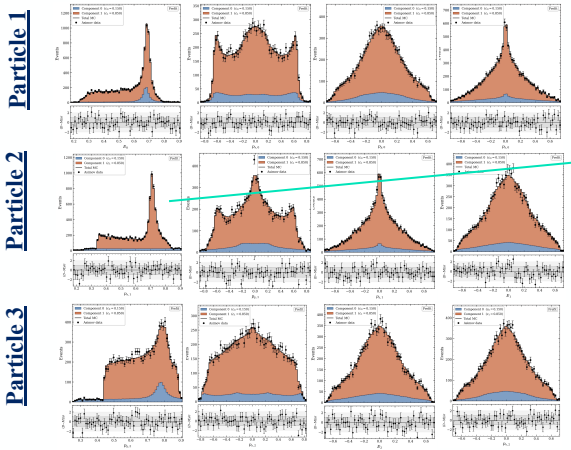
- Evaluate the matrix element
- Calculate the phase-space element
- Normalise by the solid angle of emission in 3D space
- Normalise by the width of the decay

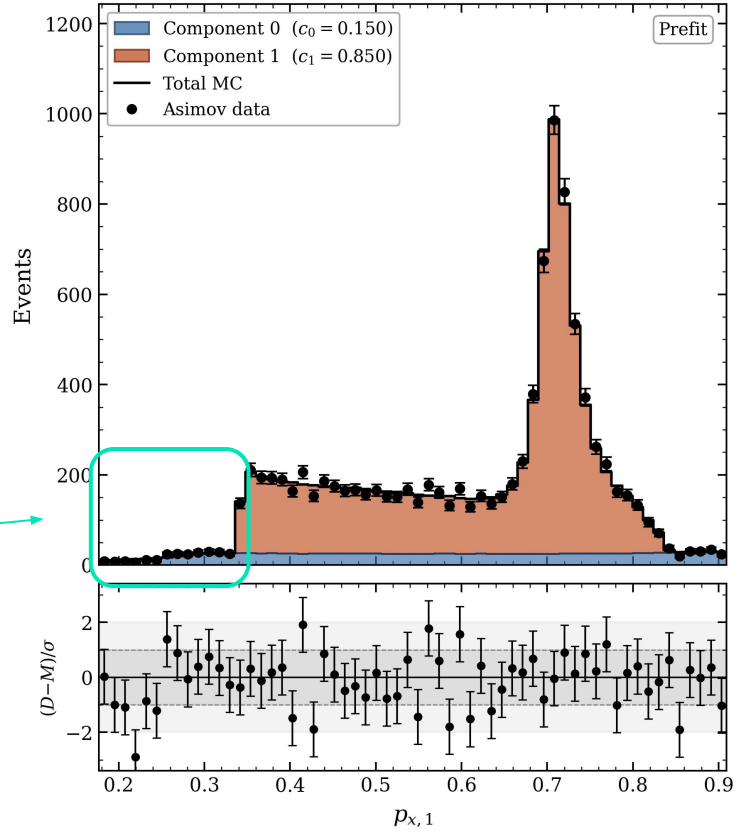
$$p_{12d}(x \mid \theta) = \frac{|\mathcal{M}(x;\theta)|^2 \, d\Phi_3(\theta)}{\Gamma(\theta)\, V_{\text{angles}}}$$



DRDPM: asimov fit results and setup

- **Monte Carlo Asimov**
  - Construct an entirely MC based ‘observed’ dataset with the task of extracting the mixture coefficient ‘***c’:***
    - **Black points:** MC based asimov ‘observed’ data
    - **Histograms:** MC Expectation





DRDPM: asimov fit results and setup

- **Monte Carlo Asimov**
  - Construct an entirely MC based ‘observed’ dataset with the task of extracting the mixture coefficient ‘***c’:***
    - **Black points:** MC based asimov ‘observed’ data
    - **Histograms:** MC Expectation
- **POI Scan:**
  - Run a scan over the mixture coefficient range c ϵ [0.05, 0.45]

$$p(x|c) = c \cdot p(x|\theta_0) + (1 - c) \cdot p(x|\theta_1)$$