Higgs inference experiments

Cyril Becot, Johann Brehmer, Kyle Cranmer, Lukas Heinrich,
Gilles Louppe, and Juan Pavez

22nd January 2018

Contents

1. Introduction

We apply different likelihood-free techniques including `carl` [1, 2] to estimate likelihood ratios between dimension-six operator hypotheses in WBF Higgs production in the four-lepton mode. Even at parton level and leading order, this process has a 15-dimensional phase space, making the likelihood intractable. But under some idealized assumptions about the detector response, we can access the exact likelihood ratio for the problem, allowing us to evaluate the performance of the different approaches. This physics problem and the data set is described in more detail in Section 2.

Our different inference strategies are described in Section 3, and the results can be found in Section 4.

The appendices provide a detailed look at the point-by-point strategies both for the full problem as well as for a simpler two-dimensional inference problem.

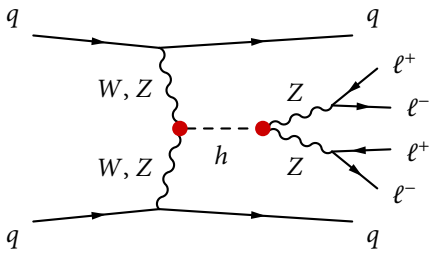

Figure 1: Feynman diagram for WBF Higgs production in the 4ℓ mode. The red dots show the Higgs-gauge interactions affected by the dimension-six operators of our analysis.

2. Problem statement

2.1. WBF Higgs to four leptons

We analyze WBF / VBF Higgs production with a Higgs decay into four leptons:

$$qq \to qq\, h \to qq\, ZZ \to qq\, \ell^+\ell^-\, \ell^+\ell^- \tag{1}$$

with $\ell = e, \mu$, as shown in Figure 1.

2.2. Parameter space

The dominant signatures of heavy new physics are expected to be described by two dimension-six operators. In the Hagiwara-Ishihara-Szalapski-Zeppenfeld conventions [3], they read

$$\mathcal{L} = \mathcal{L}_{\text{SM}} + \frac{f_W}{\Lambda^2} \underbrace{\frac{\text{i}g}{2} \left(D^\mu \phi\right)^\dagger \sigma^a D^\nu \phi \; W^a_{\mu\nu}}_{\mathcal{O}_W} - \frac{f_{WW}}{\Lambda^2} \underbrace{\frac{g^2}{4} \left(\phi^\dagger \phi\right) W^a_{\mu\nu} \, W^{\mu\nu\, a}}_{\mathcal{O}_{WW}} \tag{2}$$

with real Wilson coefficients f_W and f_{WW} and unknown new physics scale Λ.

We parameterise these coefficients in a dimensionless form as

$$\boldsymbol{\theta} = \frac{v^2}{\Lambda^2} \begin{pmatrix} f_W \\ f_{WW} \end{pmatrix}, \tag{3}$$

where $v = 246$ GeV is the SM Higgs vev. The SM therefore corresponds to $\boldsymbol{\theta}_{SM} = \mathbf{0}$, and parameters should always be in the range $-1 < |\theta_i| < 1$.

2.3. Event generation

We generate event samples in exactly the same way as described in Refs. [4, 5], using `MadGraph 5` [6] and its add-on `MadMax` [7–9]. This combination allows us to generate weighted event samples $\{\mathbf{x}, p(\mathbf{x}|\boldsymbol{\theta})\}$ including the full likelihood $p(\mathbf{x}|\boldsymbol{\theta})$ for arbitrary $\boldsymbol{\theta}$.

For this particularly clean channel, shown in Figure 1, the backgrounds are not the limiting factor, so we omit them for our toy study: a calculation with `MadGraph 5` shows that in the relevant phase-space region the cross section of the dominant irreducible $ZZ^* jj$ background is more than one order of magnitude smaller than the SM Higgs signal. For this truth-level analysis we include neither a parton shower nor detector effects: we assume that the four-momenta of the leptons and quarks can be measured precisely, the latter in the form of jets. For the leptons, we also assume that the charge and flavour is always measured exactly.

We generate a sample of $5.55 \cdot 10^6$ weighted events, which we randomly split into a main sample of $5.00 \cdot 10^6$ events for training and intermediate testing plus a smaller sample of $0.55 \cdot 10^6$ events for the final evaluation.

In these original samples, the weights are not are not always the same, with individual events in the training sample carrying an individual probability to be drawn of up to 10^{-3}. This is, of course, far from ideal ideal, but unfortunately a consequence of using `MadMax`.

2.4. Observables

We train classifiers and estimate the likelihood ratio $r(\mathbf{x})$ based on different sets of kinematic features.

1. First, we consider a simple **two-dimensional feature space** consisting of
   - the transverse momentum of the hardest jet, $p_{T,j1}$, as well as
   - the azimuthal angle between the two jets, $\Delta\phi_{jj}$.

   These observables are known to be sensitive to the operators of interest. Broadly speaking, the jet p_T distribution provides a probe of $\mathcal{O}_W$, while $\Delta\phi_{jj}$ is more sensitive to $\mathcal{O}_{WW}$ [4, 5].
2. We extend this to a **medium set**, consisting of
   - the energy, transverse momentum, pseudorapidity η, and azimuthal angle ϕ of each of the two jets, ordered by p_T;
   - the energy, transverse momentum, pseudorapidity η, and azimuthal angle ϕ of the four-lepton system (which reconstructs the Higgs);
   - the invariant mass of the dijet system, m_{jj};
   - the separation in pseudorapidity between the two jets, $\Delta\eta_{jj}$; and
   - the separation in azimuthal angle between the two jets, $\Delta\phi_{jj}$.

This set of 15 observables completely characterises the production of an on-shell Higgs in weak boson fusion at leading order (which has a five-dimensional phase space). We aim to answer the question whether training `carl` on these 15 observables is enough to estimate the true likelihood ratio based on the fully differential kinematics.

3. In a next step, we use the **full kinematics** of the leading-order process including the Higgs decay. We parameterise the four-momenta of the four leptons and the two jets with their energy, transverse momentum, pseudorapidity η, and azimuthal angle ϕ. In this way we include 24 variables, 9 more than the 15 dimensions of the phase space at leading order.[1]
4. Finally, we add some **derived variables** to this full set and check whether this improves the classifier performance. In addition to the 24 variables of the previous set, we include the following quantities:
   - the invariant mass of the dijet system, m_{jj};
   - the separation in pseudorapidity between the two jets, $\Delta\eta_{jj}$;
   - the separation in azimuthal angle between the two jets, $\Delta\phi_{jj}$;
   - the invariant masses of the two reconstructed Z bosons, m_{Z1} and m_{Z2};
   - the energy, transverse momentum, pseudorapidity, and azimuthal angle of the four-lepton system; and finally
   - the energy, transverse momentum, pseudorapidity, and azimuthal angle of the reconstructed Z bosons.

   All in all, this makes 42 observables. We do not (yet) explicitly take into account the decay angles that characterize the $h \to 4\ell$ decay chain since they carry less information on $\mathcal{O}_W$ and $\mathcal{O}_{WW}$ than production-side observables [4, 5].

2.5. Exact likelihood ratio

These samples $\{\mathbf{x}, p(\mathbf{x}|\boldsymbol{\theta})\}$ allow for a straightforward calculation of the truth-level likelihood ratio. When the full set of kinematic variables is taken into account, it is given by

$$r_{\text{full}}(\mathbf{x}) = \frac{p(\mathbf{x}|\boldsymbol{\theta}_0)}{p(\mathbf{x}|\boldsymbol{\theta}_1)} \,. \tag{4}$$

We compare the medium set of observables, the full kinematics, and the full kinematics including derived observables to this exact ratio.

For the reduced two-dimensional feature space discussed in the appendix, we first have to integrate (sum) over the unobserved directions in phase space:

$$r_{\text{2d}}(\mathbf{v}_0) = \frac{\sum\limits_{\mathbf{x}|\mathbf{v}(\mathbf{x})\approx\mathbf{v}_0} p(\mathbf{x}|\boldsymbol{\theta}_0)}{\sum\limits_{\mathbf{x}'|\mathbf{v}(\mathbf{x})\approx\mathbf{v}_0} p(\mathbf{x}'|\boldsymbol{\theta}_1)} \,. \tag{5}$$

[1] $(2+6)\cdot 4_{\text{four-momenta}} - 4_{E,\,p\text{ conservation}} - 8_{\text{on-shell conditions}} - 4_{\text{beam directions}} - 1_{\text{overall }\phi} = 15$

Here $\mathbf{v} \in \mathbf{R}^2$ are the two-dimensional observables, $\mathbf{x}$ denote 15-dimensional phase-space vectors, and $\mathbf{v}(\mathbf{x}) \approx \mathbf{v}_0$ means that the two-dimensional observables calculated from $\mathbf{x}$ are (approximately) equal to $\mathbf{v}_0$. This is realised by binning the two-dimensional feature space and summing over all events $\mathbf{x}$ in the same bin. We use 20 equidistant bins for $\Delta\phi_{jj}$ times 25 bins for $p_{T,j1}$. The jet momentum bins have a varying size: 10 GeVup to a p_T of 100 GeV, 20 GeVup to 200 GeV, 40 GeVup to 400 GeV, 80 GeVup to 1200 GeV, and an overflow bin for $p_T > 1200$ GeV. Note that for simplicity we will later use $\mathbf{x}$ to label the two observables of this process.

3. Inference strategies

3.1. Algorithms

A first class of algorithms approaches the inference problem point by point in parameter space, learning the likelihood ratio $r(\mathbf{x}; \boldsymbol{\theta}_0, \boldsymbol{\theta}_1) \equiv p(\mathbf{x}|\boldsymbol{\theta}_0)/p(\mathbf{x}|\boldsymbol{\theta}_1)$ between a fixed pair of hypotheses. Only in the end the different analysed pairs $(\boldsymbol{\theta}_0, \boldsymbol{\theta}_1)$ are compared and interpolated.

Instead of estimating the likelihood ratio separately for different pairs $(\boldsymbol{\theta}_0, \boldsymbol{\theta}_1)$, we can also train one parameterized classifier or regressor $\hat{r}(\mathbf{x}; \boldsymbol{\theta}_0, \boldsymbol{\theta}_1)$. This allows us to go beyond the likelihood-free setting and include the information from the score when it is available, or to use physics knowledge about the $\boldsymbol{\theta}$ dependence.

...

We consider five different inference setups. For the **likelihood-free setting**, we use the normal `carl` approach. We first define a "raw" version: for the mapping $\hat{r}(\hat{s})$ we simply assume a perfectly trained classifier, $\hat{s}(\mathbf{x}) = p(\mathbf{x}|\boldsymbol{\theta}_0)/(p(\mathbf{x}|\boldsymbol{\theta}_0) + p(\mathbf{x}|\boldsymbol{\theta}_1))$. Second, we use a calibrated version, where we generate a new calibration histogram for each of the 1016 values of $\boldsymbol{\theta}_0$ used in the evaluation. For the calibration we always use the same (weighted) events, which is not possible in the most general likelihood-free setup (but may be in situations like the EFT described here).

For the setup in which the **score is tractable**, we define two parameterized estimators: one is just trained by regressing on the score data. The second uses a combination of `carl`'s cross-entropy loss and the mean squared error of the score regression problem. A parameter α weights these two terms in the loss function, aiming for an approximately equal size of the two terms. Again, we consider a raw and a calibrated version of this setup.

Finally, we define two regressors for the case in which the **likelihood is tractable**: standard regression on $\log r$, both for the point-by-point and the parameterized setup. For the parameterized setup, we also use a combination of regression on $\log r$ and on the score. The regression models are not calibrated.

3.2. Calibration methods

We consider and compare different calibration strategies:

Histogram: ...

Isotonic: ...

Parametric: We also consider two parametric calibration procedures. First, there is sigmoid calibration (Platt's method)... As an alternative we consider the strictly monotonic parametric function

$$s_{\text{calibrated}}(s_{\text{raw}}; \alpha) = \frac{(1-\alpha)s_{\text{raw}}}{(1-\alpha)s_{\text{raw}} + \alpha(1-s_{\text{raw}})} \tag{6}$$

with parameter $0 < \alpha < 1$ as a calibration curve. The value of this parameter is determined by fitting Equation (6) to the raw and calibrated values of $\hat{s}$ of a different calibration method (for now, we are using histograms).

Kernel density estimation (KDE): …

Gradient-adaptive KDE: …

3.3. Training samples

We consider four different training samples. In the **baseline** setup, the training sample consists of 1000 pairs $(\boldsymbol{\theta}_0, \boldsymbol{\theta}_1)$.

$$\boldsymbol{\theta}_1 = (0.393, 0.492) \tag{7}$$

is fixed, while the values of $\boldsymbol{\theta}_0$ are chosen randomly (with a flat distribution over $[-1, 1]^2$]). For each of these pairs, we draw 5 000 events according to $\boldsymbol{\theta}_0$ and 5 000 events according to $\boldsymbol{\theta}_1$. This amounts to a total of 10^7 events.

In our alternative "**random $\boldsymbol{\theta}$**" training sample draw independent values of $\boldsymbol{\theta}_0$ for each event, from a flat prior in $[-1, 1]^2$. $\boldsymbol{\theta}_1$ is fixed as before. The sample consists of 9998227 events, 4998047 of which were drawn according to $\boldsymbol{\theta}_1$.

Finally, for some scenarios we analyse a training sample where only the 15 basis points of the morphing procedure are used as values of $\boldsymbol{\theta}_0$. Again a total of 10^7 events are used.

In all cases, our evaluation sample consists of 100 000 events drawn according to the SM. We calculate the exact likelihood ratio for these events for a total of 1016 values of $\boldsymbol{\theta}_0$, 1000 of which are the same as those used in the first training sample. Again we fix $\boldsymbol{\theta}_1$ as in Equation (7).

3.3.1. Estimator design

In the **baseline** model, our estimators consist of a fully connected neural network. In this way, the network can learn (almost) any dependence on $\boldsymbol{\theta}$ it wants.

In addition we consider a "**physics-aware model**" that uses the knowledge about the compositional structure of the likelihood function (as in the morphing setup):

$$\hat{r}(\mathbf{x}; \boldsymbol{\theta}) = \sum_i w_i(\boldsymbol{\theta})\, \hat{r}_i(\mathbf{x}) \tag{8}$$

where we have explicitly left out the dependence on $\boldsymbol{\theta}_1$, which we always keep fixed. The sum goes over the 15 basis samples used in the morphing setup, and the functions $w_i(\boldsymbol{\theta})$ are analytically known. Following this structure, our network consists of 15 independent estimators $\hat{r}_i(\mathbf{x})$, weighted with the $w_i(\boldsymbol{\theta})$.

For all architectures, We use between one and three hidden layers of 100 units with tanh activation functions for the baseline model, and two hidden layers of 50 units with tanh activation functions.

We implement all setups in `keras` with a `TensorFlow` backend inside a `sklearn` wrapper and train for 20 epochs with early stopping.

3.3.2. Metrics

...

3.3.3. Uncertainty diagnostics

We are experimenting with diagnostic tools that might provide some measure of the uncertainty of the estimators $\hat{r}(\mathbf{x})$. There are different ideas:

1. **Ensemble variance**: repeating the full calculation with different random seeds, one can extract an uncertainty measure from the ensemble variance. This would only capture statistical uncertainties, not a systematic bias. Obviously this increases the computational cost.
2. **Denominator variation**: the relative log likelihood contours (compared to the best-fit point) should not depend on the choice of the denominator hypothesis $\boldsymbol{\theta}_1$ [1]. We can therefore use quantities like

$$\begin{aligned}\Delta \log \hat{r}(\mathbf{x}; \boldsymbol{\theta}_0, \boldsymbol{\theta}_1) = \Big| &(\log \hat{r}(\mathbf{x}; \boldsymbol{\theta}_0, \boldsymbol{\theta}_1) - \max_{\boldsymbol{\theta}} \log \hat{r}(\mathbf{x}; \boldsymbol{\theta}, \boldsymbol{\theta}_1)) \\ &- (\log \hat{r}'(\mathbf{x}; \boldsymbol{\theta}_0, \boldsymbol{\theta}_1') - \max_{\boldsymbol{\theta}} \log \hat{r}'(\mathbf{x}; \boldsymbol{\theta}_0, \boldsymbol{\theta}_1'))\Big| . \quad (9)\end{aligned}$$

   as a measure of systematic uncertainties. Probably such a number only presents a lower bound for the true uncertainty. Again, this increases the computational cost.
3. **Toy experiments**: we perform a series of mock experiments with different numbers of events and calculate the variance in the distribution of $\log \hat{r}\mathbf{x}$ as a function of the number of events. We fit the variance as a function of the event number with

$$\mathrm{var}[\log \hat{r}(\mathbf{x}; \boldsymbol{\theta}_0, \boldsymbol{\theta}_1)](n) = a/n + b\,. \quad (10)$$

   Then $\sqrt{a/n}$ represents the statistical uncertainty from a limited number of events, while

$$\Delta \log \hat{r}(\mathbf{x}; \boldsymbol{\theta}_0, \boldsymbol{\theta}_1) = \sqrt{b} \quad (11)$$

   might provide a measure of a residual systematic uncertainty in the expectation values with large n. There is no guarantee that this strategy works.
4. **Identity check**: for $\boldsymbol{\theta}_0 = \boldsymbol{\theta}_1$ the log likelihood ratio should identically vanish. A non-zero value of

$$\Delta \log \hat{r}(\mathbf{x}) = \log \hat{r}(\mathbf{x}; \boldsymbol{\theta}_0 = \boldsymbol{\theta}_1, \boldsymbol{\theta}_1) \quad (12)$$

   might therefore hint at a systematic uncertainty. Again, there is no guarantee that this number says anything about the behaviour at other values of $\boldsymbol{\theta}_0$. Also, any constant bias in $\log \hat{r}(\mathbf{x})$ would cancel out in the likelihood contours, which consider the difference in the log likelihood with respect to the best-fit point.

5. **Score expectation value**: we know that $E[\mathbf{t}(\mathbf{x}|\boldsymbol{\theta}_0)|\boldsymbol{\theta}_0] = 0$. Perhaps there is a way to translate non-zero expectation values to an uncertainty measure on $\log r(\mathbf{x})$?

4. Results

4.1. Overview

Tables 1 to 4 compare the performance of all different setups tested.

4.2. Baseline results

We show the results of the parameterized baseline experiments in Figures 2 to 7. We find that calibration is important, and that the combinatino of `carl` with score information works well. The tested diagnostic tool for uncertainties, unfortunately, does not seem to be particularly useful: it wrongly assisngs sizeable errors to the ground truth.

The plots shown are based on fully connected neural networks with three hidden layers of 100 units. We are currently experimenting with more shallow or deeper networks.

4.3. Calibration strategies

In Table 5 we compare the performance of different calibration strategies. The corresponding calibration curves are shown in Figures 8 and 9.

4.4. Score density estimation

Instead of learning the classifier output $s(\mathbf{x})$, we can learn the score $\mathbf{t}(\mathbf{x})$, which is the sufficient statistics in the local model, and perform density estimation in either $\boldsymbol{\theta}$ space or in one-dimensional $\Delta\boldsymbol{\theta} \cdot \boldsymbol{\theta}$ space. In Table **??** and Figure **??** we compare different strategies.

4.5. Physics-aware setup

The physics-aware setup is based on the analytic form of the morphing weights $w_i(\boldsymbol{\theta})$. These factors crucially depend on the basis points in the morphing procedure, as we show in Figures 11 and 12. Here we stick to the choice shown in Figure 12, which reduces the weights and the cancellations between them compared to other basis choices. For now we only analyse the training sample consisting only of these basis values of $\boldsymbol{\theta}_0$.

...

Algorithm	Setup	$\mathrm{MSE}_{\mathbf{x}}\left[\log r(\mathbf{x};\boldsymbol{\theta}_0,\boldsymbol{\theta}_1)\right]$		$\mathrm{MSE}_{\boldsymbol{\theta}}\left[E[\log r(\mathbf{x};\boldsymbol{\theta},\boldsymbol{\theta}_{\mathrm{den}})]\right]$	
		$\boldsymbol{\theta}_0 = \boldsymbol{\theta}_T$	$\boldsymbol{\theta}_0 = \boldsymbol{\theta}_{NT}$	$\boldsymbol{\theta}_{\mathrm{den}} = \boldsymbol{\theta}_1$	$\boldsymbol{\theta}_{\mathrm{den}} = \hat{\boldsymbol{\theta}}_{\mathrm{MLE}}$
PbP carl (raw)	PbP (1)	0.206	0.229	50.54	5.98
	PbP (2)	0.177	0.203	48.99	4.75
	PbP (3)	0.201	0.272	49.31	5.97
	PbP learning $\log r(\mathbf{x})$ (1)	0.189	0.193	49.15	**4.14**
	PbP learning $\log r(\mathbf{x})$ (2)	0.269	0.196	48.51	4.59
	PbP learning $\log r(\mathbf{x})$ (3)	**0.114**	**0.157**	**47.81**	4.80
PbP carl (cal.)	PbP (1)	0.094	0.121	3.91	**3.82**
	PbP (2)	0.139	0.239	3.91	3.87
	PbP (3)	0.099	0.132	3.95	3.90
	PbP learning $\log r(\mathbf{x})$ (1)	0.097	0.194	3.93	3.91
	PbP learning $\log r(\mathbf{x})$ (2)	**0.087**	0.155	**3.88**	3.85
	PbP learning $\log r(\mathbf{x})$ (3)	0.113	**0.100**	3.95	4.36
Param carl (raw)	Baseline (1)	0.040	0.074	2.78	0.72
	Baseline (2)	0.083	0.116	29.68	5.49
	Baseline (3)	0.037	**0.051**	4.80	2.64
	Random $\boldsymbol{\theta}$ (1)	0.047	0.066	1.41	1.53
	Random $\boldsymbol{\theta}$ (2)	**0.037**	0.054	0.85	0.70
	Random $\boldsymbol{\theta}$ (3)	0.040	0.055	**0.26**	**0.20**
	Physics-aware (1)				
	Physics-aware (2)				
	Physics-aware (3)				
Param carl (cal.)	Baseline (1)	0.033	0.050	0.24	0.41
	Baseline (2)	**0.027**	**0.038**	0.10	0.09
	Baseline (3)	0.047	0.062	0.58	0.61
	Random $\boldsymbol{\theta}$ (1)	0.034	0.048	0.09	0.30
	Random $\boldsymbol{\theta}$ (2)	0.041	0.055	**0.03**	**0.04**
	Random $\boldsymbol{\theta}$ (3)	0.046	0.064	0.64	0.50
	Physics-aware (1)				
	Physics-aware (2)				
	Physics-aware (3)				

Table 1: Comparison of different inference strategies, part 1. In this table we show likelihood-free strategies based on a point-by-point (PbP) or parameterized setup. The numbers in brackets denote the number of hidden layers in the neural networks. We show two types metrics: the mean squared error of $\log r$ for a fixed pair of hypotheses over different phase-space points and the mean squared error of the expectation value of $\log r$ for different pairs of hypotheses. For the first case, the benchmark point $\boldsymbol{\theta}_T$ was part of all training samples (except for the random $\boldsymbol{\theta}$ sample), while the benchmark point $\boldsymbol{\theta}_{NT}$ is unknown to the parameterized models (it was still part of the point-by-point training). In the latter case, the expectation value sums over $\mathbf{x}$ following the SM distribution.

Algorithm	Setup	$\mathrm{MSE}_{\mathbf{x}}\left[\log r(\mathbf{x};\boldsymbol{\theta}_0,\boldsymbol{\theta}_1)\right]$		$\mathrm{MSE}_{\boldsymbol{\theta}}\left[E[\log r(\mathbf{x};\boldsymbol{\theta},\boldsymbol{\theta}_{\mathrm{den}})]\right]$	
		$\boldsymbol{\theta}_0 = \boldsymbol{\theta}_T$	$\boldsymbol{\theta}_0 = \boldsymbol{\theta}_{NT}$	$\boldsymbol{\theta}_{\mathrm{den}} = \boldsymbol{\theta}_1$	$\boldsymbol{\theta}_{\mathrm{den}} = \hat{\boldsymbol{\theta}}_{\mathrm{MLE}}$
Param score (raw)	Baseline (1)	**0.060**	0.067	0.23	0.25
	Baseline (2)	0.060	**0.064**	**0.16**	**0.06**
	Baseline (3)	0.104	0.110	15.30	0.09
	Random $\boldsymbol{\theta}$ (1)	0.099	0.135	3.51	3.57
	Random $\boldsymbol{\theta}$ (2)	0.082	0.113	3.76	3.19
	Random $\boldsymbol{\theta}$ (3)	0.135	0.177	14.09	3.20
	Physics-aware (1)				
	Physics-aware (2)				
	Physics-aware (3)				
Param score (cal.)	Baseline (1)	0.100	0.105	0.96	0.90
	Baseline (2)	**0.076**	**0.094**	**0.43**	**0.43**
	Baseline (3)				
	Random $\boldsymbol{\theta}$ (1)	0.088	0.172	2.01	2.15
	Random $\boldsymbol{\theta}$ (2)	0.115	0.533	2.36	2.36
	Random $\boldsymbol{\theta}$ (3)				
	Physics-aware (1)				
	Physics-aware (2)				
	Physics-aware (3)				
Param carl + score (raw)	Baseline (1)	0.037	0.049	2.11	0.19
	Baseline (2)	**0.014**	**0.016**	**0.66**	0.16
	Baseline (3)	0.041	0.047	10.99	**0.15**
	Random $\boldsymbol{\theta}$ (1)	0.050	0.082	1.53	3.62
	Random $\boldsymbol{\theta}$ (2)	0.038	0.063	2.08	3.50
	Random $\boldsymbol{\theta}$ (3)	0.053	0.074	3.01	4.20
	Physics-aware (1)				
	Physics-aware (2)				
	Physics-aware (3)				
Param carl + score (cal.)	Baseline (1)	0.028	0.041	0.07	**0.07**
	Baseline (2)	**0.012**	**0.014**	**0.04**	0.09
	Baseline (3)	0.028	0.023	0.56	0.46
	Random $\boldsymbol{\theta}$ (1)	0.037	0.058	0.49	0.97
	Random $\boldsymbol{\theta}$ (2)	0.024	0.035	0.32	0.47
	Random $\boldsymbol{\theta}$ (3)	0.034	0.047	0.63	0.98
	Physics-aware (1)				
	Physics-aware (2)				
	Physics-aware (3)				

Table 2: Comparison of different inference strategies, part 2. In this table we show parameterized architectures that utilize the score information. The numbers in brackets denote the number of hidden layers in the neural networks. We show two types metrics: the mean squared error of $\log r$ for a fixed pair of hypotheses over different phase-space points and the mean squared error of the expectation value of $\log r$ for different pairs of hypotheses. For the first case, the benchmark point $\boldsymbol{\theta}_T$ was part of all training samples (except for the random $\boldsymbol{\theta}$ sample), while the benchmark point $\boldsymbol{\theta}_{NT}$ is unknown to the parameterized models (it was still part of the point-by-point training). In the latter case, the expectation value sums over $\mathbf{x}$ following the SM distribution.

Algorithm	Setup	$\mathrm{MSE}_{\mathbf{x}}\left[\log r(\mathbf{x};\boldsymbol{\theta}_0,\boldsymbol{\theta}_1)\right]$		$\mathrm{MSE}_{\boldsymbol{\theta}}\left[E[\log r(\mathbf{x};\boldsymbol{\theta},\boldsymbol{\theta}_{\mathrm{den}})]\right]$	
		$\boldsymbol{\theta}_0 = \boldsymbol{\theta}_T$	$\boldsymbol{\theta}_0 = \boldsymbol{\theta}_{NT}$	$\boldsymbol{\theta}_{\mathrm{den}} = \boldsymbol{\theta}_1$	$\boldsymbol{\theta}_{\mathrm{den}} = \hat{\boldsymbol{\theta}}_{\mathrm{MLE}}$
PbP regression (raw)	PbP (1)	0.011	0.015	3.79	3.92
	PbP (2)	0.005	**0.005**	3.78	**3.83**
	PbP (3)	**0.004**	0.006	**3.60**	4.02
Param regression (raw)	Baseline (1)	0.015	0.022	0.12	0.12
	Baseline (2)	**0.004**	**0.005**	0.16	0.14
	Baseline (3)	0.007	0.007	0.30	0.29
	Random $\boldsymbol{\theta}$ (1)	0.015	0.021	0.40	0.22
	Random $\boldsymbol{\theta}$ (2)	0.005	0.007	0.20	**0.02**
	Random $\boldsymbol{\theta}$ (3)	0.007	0.008	**0.04**	0.03
	Physics-aware (1)				
	Physics-aware (2)				
	Physics-aware (3)				
Param regr. + score (raw)	Baseline (1)				
	Baseline (2)				
	Baseline (3)				
	Random $\boldsymbol{\theta}$ (1)				
	Random $\boldsymbol{\theta}$ (2)				
	Random $\boldsymbol{\theta}$ (3)				
	Physics-aware (1)				
	Physics-aware (2)				
	Physics-aware (3)				

Table 3: Comparison of different inference strategies, part 3. In this table we show point-by-point (PbP) and parameterized architectures that require a tractable likelihood. The numbers in brackets denote the number of hidden layers in the neural networks. We show two types metrics: the mean squared error of $\log r$ for a fixed pair of hypotheses over different phase-space points and the mean squared error of the expectation value of $\log r$ for different pairs of hypotheses. For the first case, the benchmark point $\boldsymbol{\theta}_T$ was part of all training samples (except for the random $\boldsymbol{\theta}$ sample), while the benchmark point $\boldsymbol{\theta}_{NT}$ is unknown to the parameterized models (it was still part of the point-by-point training). In the latter case, the expectation value sums over $\mathbf{x}$ following the SM distribution.

Algorithm	Setup	$\mathrm{MSE}_{\mathbf{x}}\left[\mathbf{t}(\mathbf{x};\boldsymbol{\theta}_0,\boldsymbol{\theta}_1)\right]$	
		$\boldsymbol{\theta}_0 = \boldsymbol{\theta}_T$	$\boldsymbol{\theta}_0 = \boldsymbol{\theta}_{NT}$
Param carl	Baseline (1)	**0.42**	**0.81**
	Baseline (2)	1.12	1.46
	Baseline (3)	4.29	2.85
	Random $\boldsymbol{\theta}$ (1)	0.43	0.85
	Random $\boldsymbol{\theta}$ (2)	1.20	1.45
	Random $\boldsymbol{\theta}$ (3)	3.21	3.31
	Physics-aware (1)		
	Physics-aware (2)		
	Physics-aware (3)		
Param score	Baseline (1)	0.19	0.48
	Baseline (2)	**0.09**	**0.22**
	Baseline (3)	0.10	0.23
	Random $\boldsymbol{\theta}$ (1)	0.40	0.77
	Random $\boldsymbol{\theta}$ (2)	0.37	0.72
	Random $\boldsymbol{\theta}$ (3)	0.40	0.75
	Physics-aware (1)		
	Physics-aware (2)		
	Physics-aware (3)		
Param carl + score	Baseline (1)	0.24	0.55
	Baseline (2)	**0.12**	**0.27**
	Baseline (3)	0.15	0.34
	Random $\boldsymbol{\theta}$ (1)	0.38	0.77
	Random $\boldsymbol{\theta}$ (2)	0.36	0.71
	Random $\boldsymbol{\theta}$ (3)	0.37	0.71
	Physics-aware (1)		
	Physics-aware (2)		
	Physics-aware (3)		
Param regression	Baseline (1)	0.27	0.59
	Baseline (2)	**0.16**	**0.25**
	Baseline (3)	0.30	0.48
	Random $\boldsymbol{\theta}$ (1)	0.28	0.64
	Random $\boldsymbol{\theta}$ (2)	0.23	0.30
	Random $\boldsymbol{\theta}$ (3)	0.46	0.84
	Physics-aware (1)		
	Physics-aware (2)		
	Physics-aware (3)		
Param regr. + score	Baseline (1)		
	Baseline (2)		
	Baseline (3)		
	Random $\boldsymbol{\theta}$ (1)		
	Random $\boldsymbol{\theta}$ (2)		
	Random $\boldsymbol{\theta}$ (3)		
	Physics-aware (1)		
	Physics-aware (2)		
	Physics-aware (3)		

Table 4: Comparison of different inference strategies, part 4. In this table we compare the estimation of the scores in the different parameterized models. The benchmark point $\boldsymbol{\theta}_T$ was part of all training samples (except for the random $\boldsymbol{\theta}$ sample), while the benchmark point $\boldsymbol{\theta}_{NT}$ is unknown to the parameterized models (it was still part of the point-by-point training). The numbers in brackets denote the number of hidden layers in the neural networks.

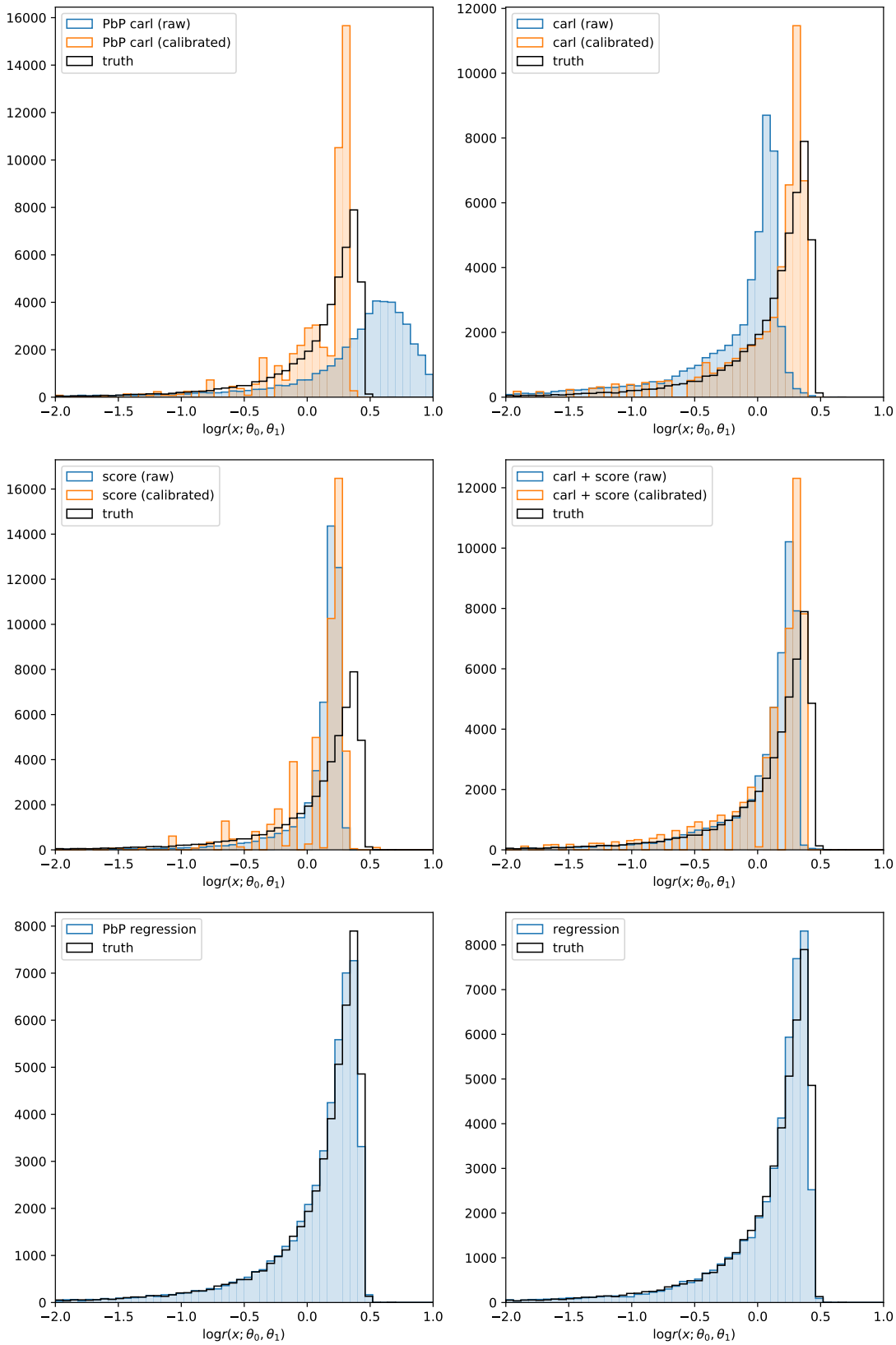

Figure 2: Likelihood ratio histograms for individual phase-space points $\mathbf{x}$, for one particular hypothesis comparison $(\boldsymbol{\theta}_0, \boldsymbol{\theta}_1)$. We show the baseline version of the different parameterized models.

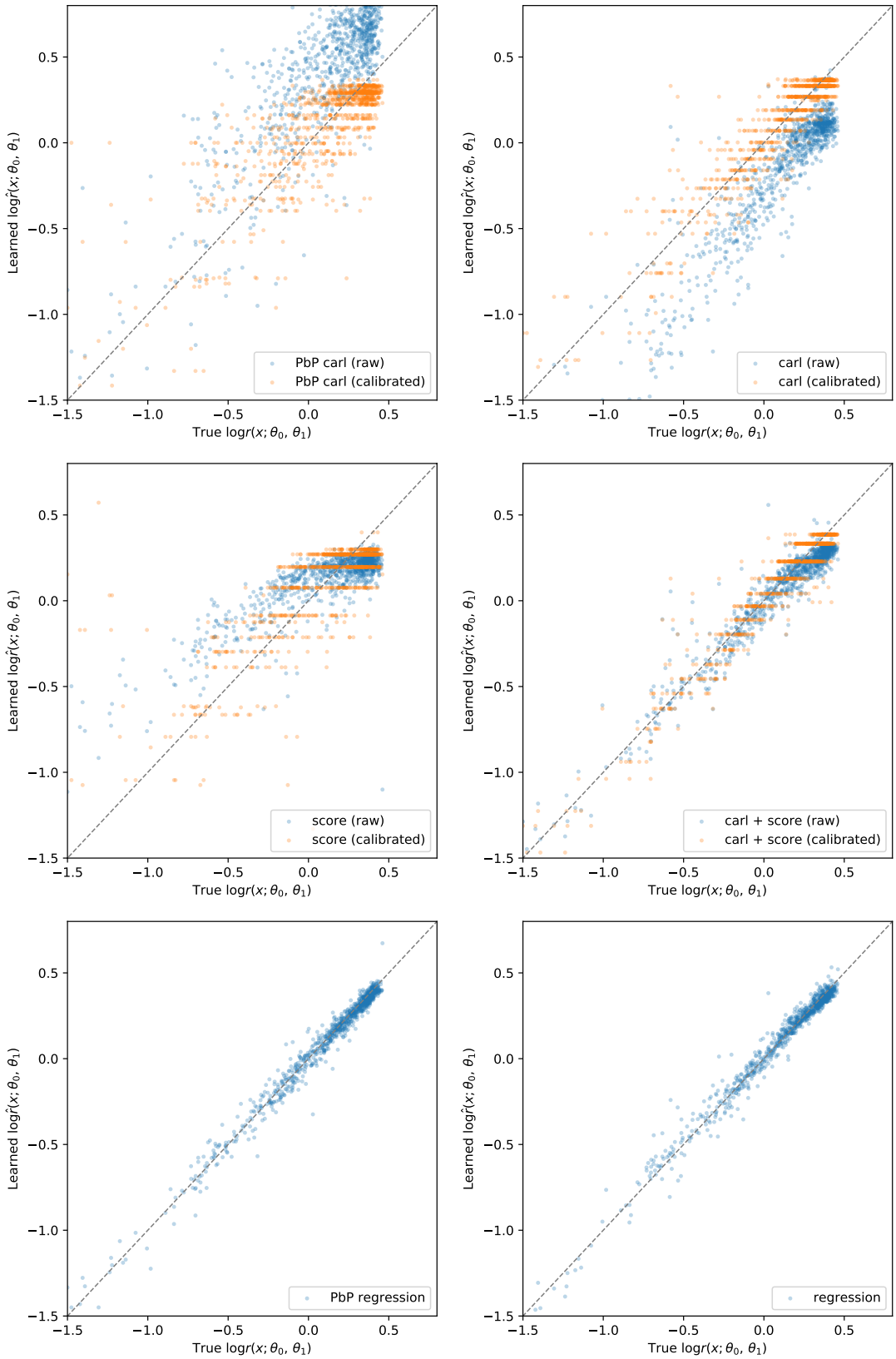

Figure 3: Likelihood ratio scatter plot for individual phase-space points $\mathbf{x}$, for one particular hypothesis comparison $(\boldsymbol{\theta}_0, \boldsymbol{\theta}_1)$. We show the baseline version of the different parameterized models.

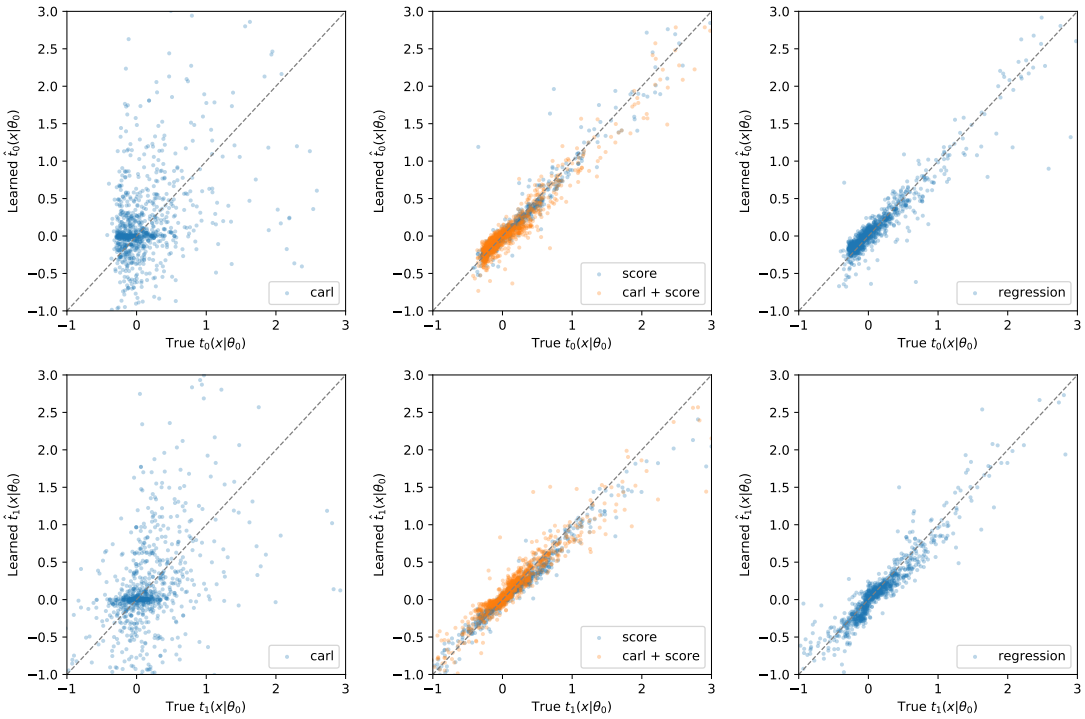

Figure 4: Score scatter plots for individual phase-space points $\mathbf{x}$, for one particular hypothesis comparison $(\boldsymbol{\theta}_0, \boldsymbol{\theta}_1)$. We show the baseline version of the different parameterized models.

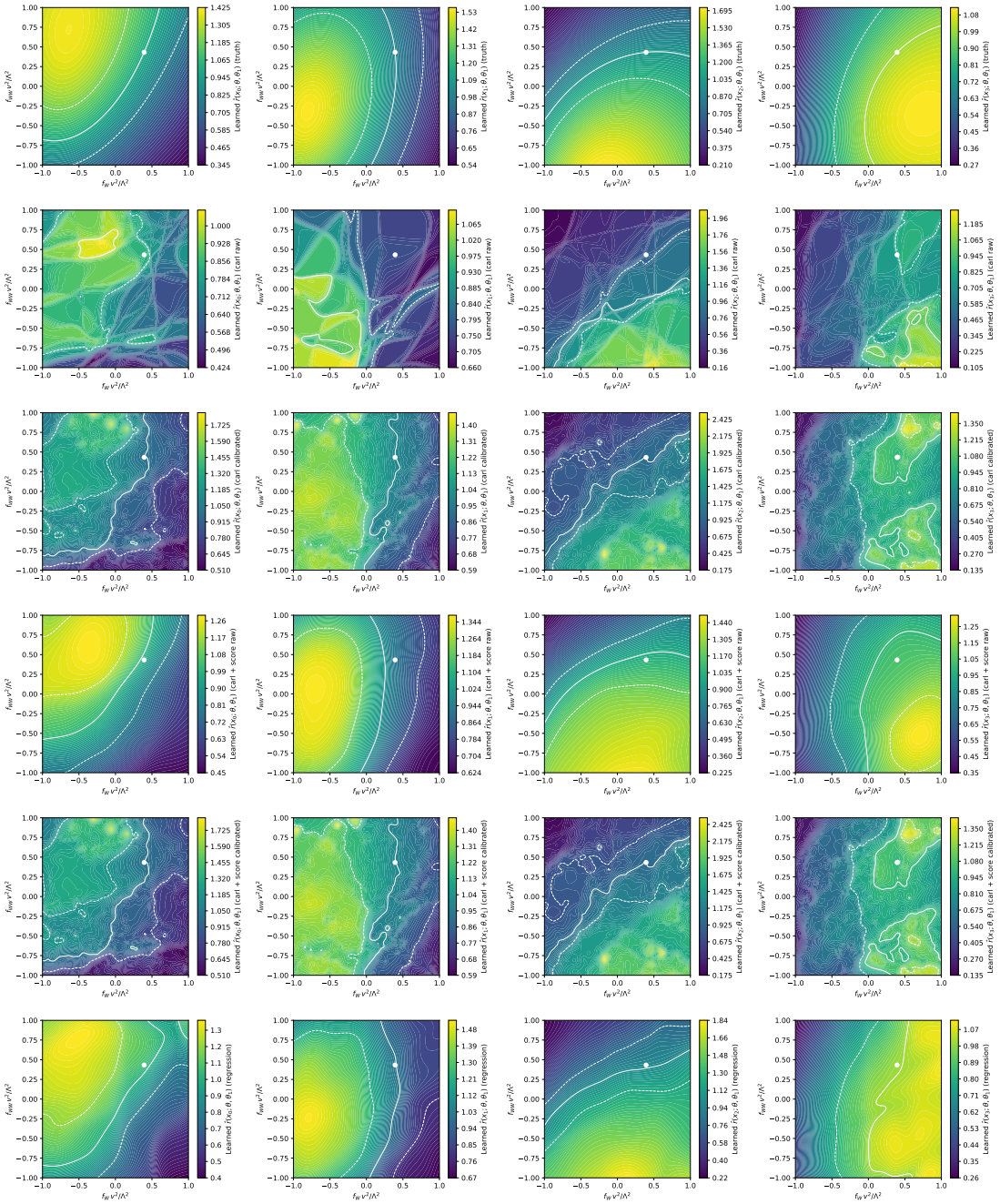

Figure 5: Exact and estimated likelihood ratios as a function of $\boldsymbol{\theta}_0$, for four different individual events $\mathbf{x}$ (columns). We show the baseline version of the different parameterized models (rows).

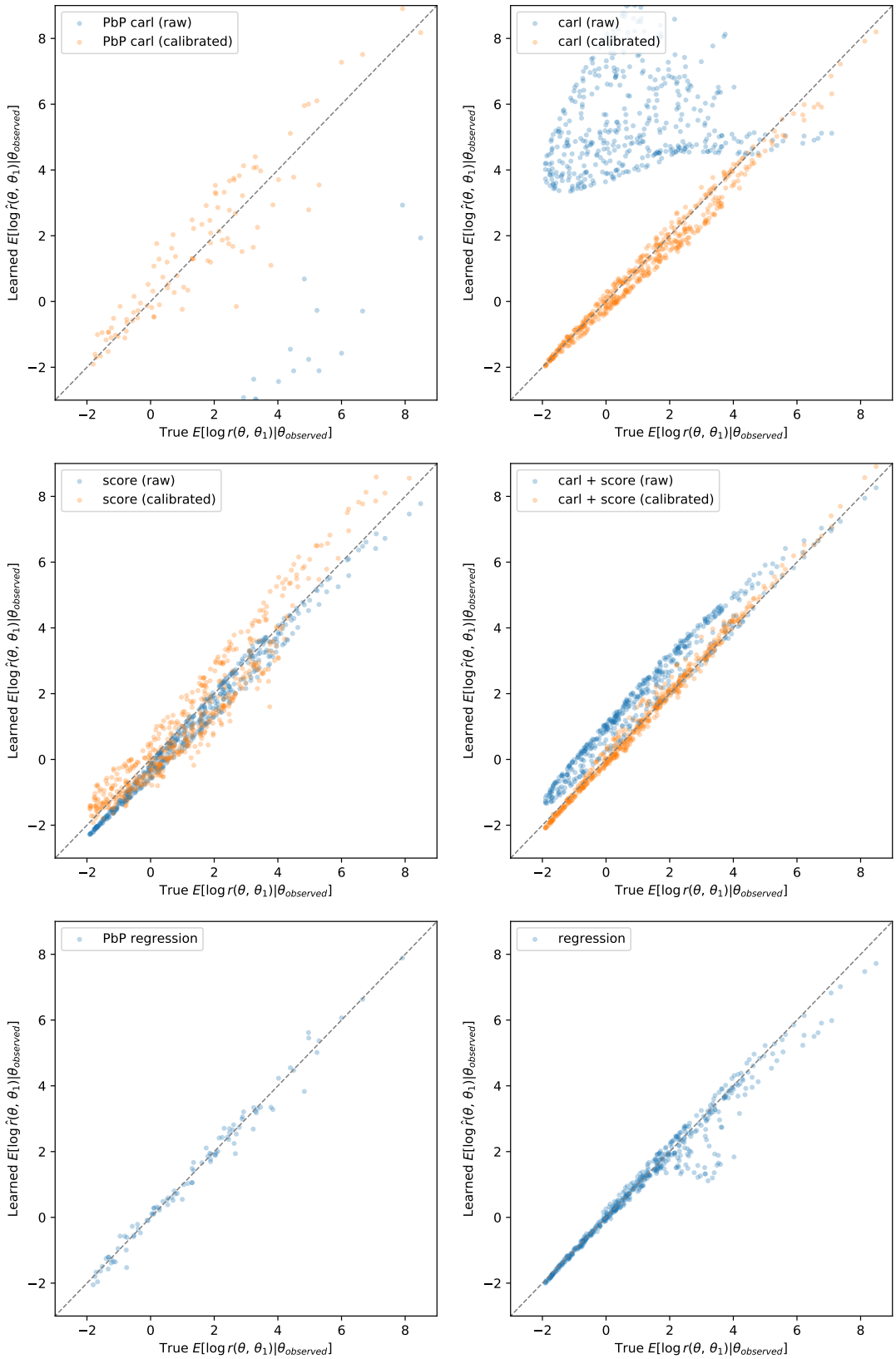

Figure 6: Scatter plot of the expected likelihood ratios for different values of $\boldsymbol{\theta}_0$, taking the expectation value over $\mathbf{x}$. We show the baseline version of the different parameterized models.

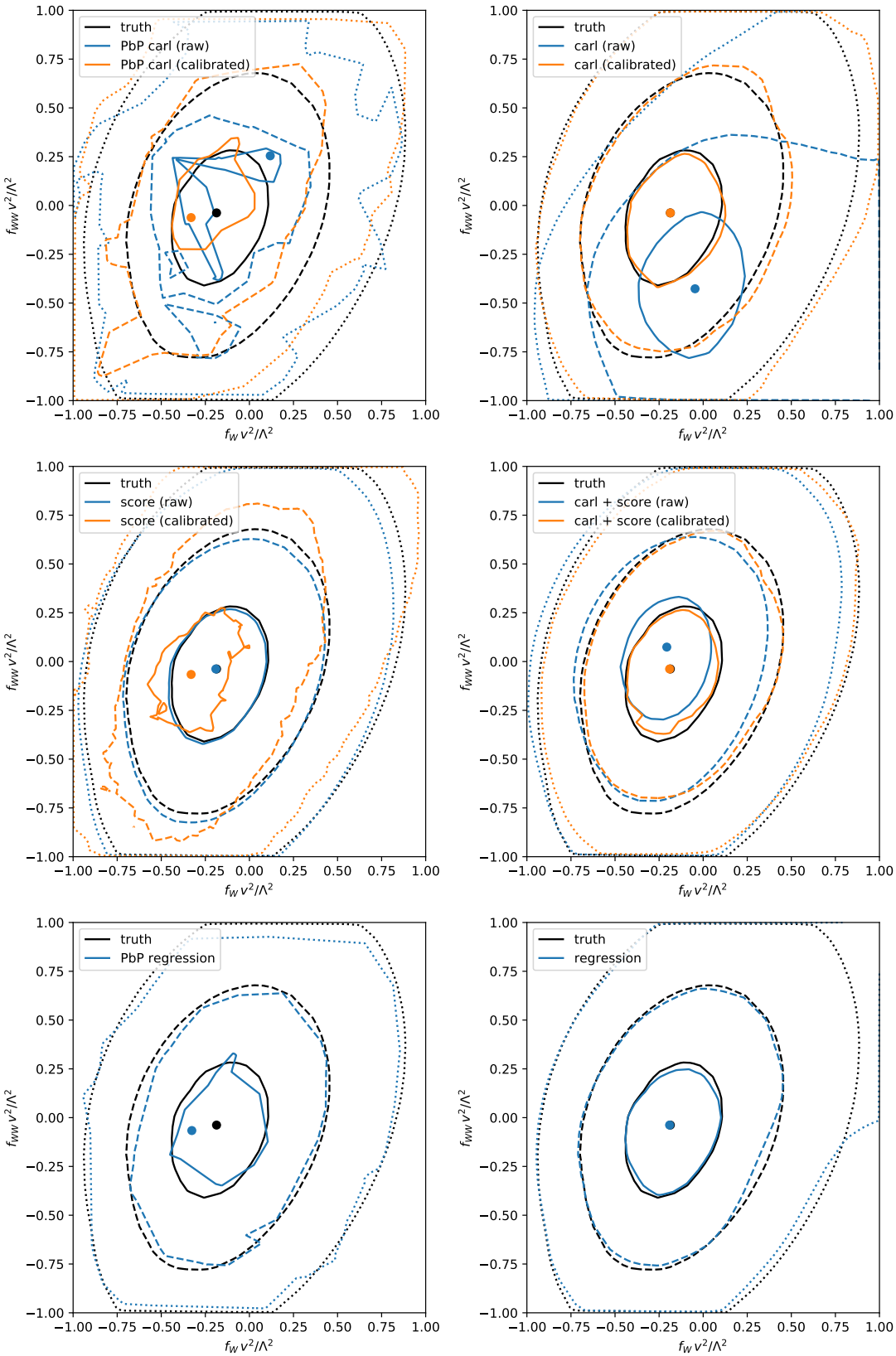

Figure 7: Contours of the expected likelihood ratio at $-2\Delta \log r = 1$ (solid), 4 (dashed), 9 (dotted). We show the baseline version of the different parameterized models. The shaded error bands show an estimate of the uncertainties following Equation (11).

Calibration strategy	Settings	$\mathrm{MSE}_{\mathbf{x}}\left[\log r(\mathbf{x};\boldsymbol{\theta}_0,\boldsymbol{\theta}_1)\right]$	
		PbP regression	PbP carl
No calibration		**0.0026**	**0.260**
Histogram	25 equidistant bins	0.0065	0.034
	50 equidistant bins	0.0038	0.032
	100 equidistant bins	0.0030	0.032
	200 equidistant bins	0.0028	0.032
	25 equidistant bins, linear interpolation	0.0052	0.033
	50 equidistant bins, linear interpolation	0.0034	**0.032**
	100 equidistant bins, linear interpolation	0.0029	0.032
	200 equidistant bins, linear interpolation	**0.0027**	0.032
	25 equidistant bins, spline interpolation	0.0051	0.032
	50 equidistant bins, spline interpolation	0.0034	0.034
	100 equidistant bins, spline interpolation	0.0029	0.032
	200 equidistant bins, spline interpolation	0.0060	0.041
	25 variable bins	1.5307	0.945
	50 variable bins	1.6809	1.186
	100 variable bins	1.7720	1.908
	200 variable bins	2.8246	3.601
Isotonic	no interpolation	**0.0025**	**0.032**
	linear interpolation	0.0025	0.036
Parametric	sigmoid	0.0250	0.046
	Equation (6)	**0.0026**	**0.036**
KDE	fixed bandwidth	**0.0176**	0.043
	adaptive (α = 0.25)	0.0185	0.043
	adaptive (α = 0.5)	0.0196	**0.042**
	adaptive (α = 0.75)	0.0206	0.043
	adaptive (α = 1.)	0.0304	0.048
KDE	grad-adaptive (β = 0.2)	**0.0051**	0.037
	grad-adaptive (β = 0.5)	0.0070	**0.033**
	grad-adaptive (β = 1)	0.0101	0.035
	grad-adaptive (β = 2)	0.0207	0.046
	grad-adaptive (β_{num} = 1.66, β_{den} = 1.13)	0.0109	0.034

Table 5: Comparison of different calibration methods. As test scenario we use two inference strategies, carl and regression, both in a point-by-point setup for one particular benchmark $\boldsymbol{\theta}_0$.

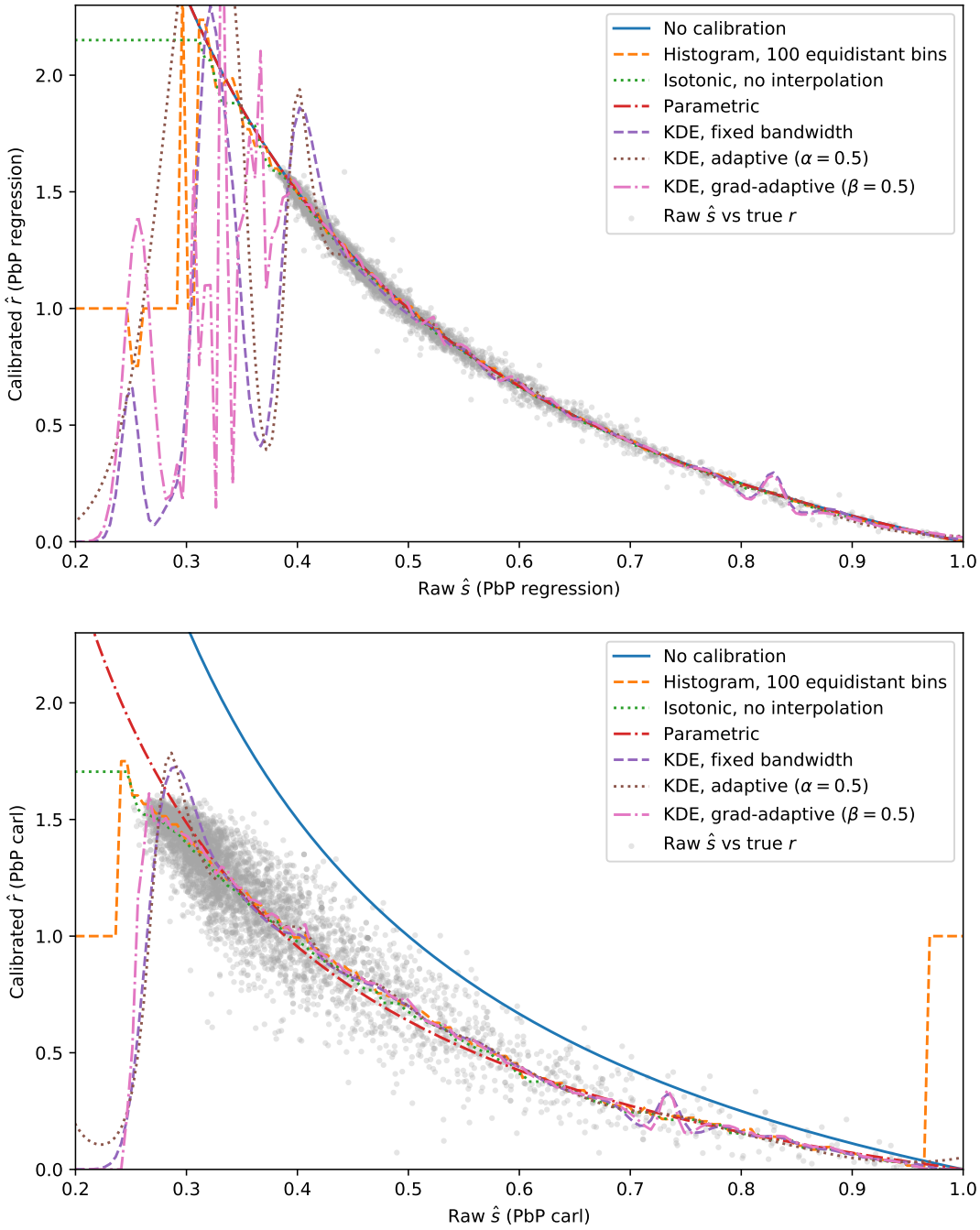

Figure 8: Calibration curves $r_{\text{calibrated}}(s_{\text{raw}})$ for different calibration strategies. The parametric approach is defined in Equation (6). Top: point-by-point regression. Bottom: point-by-point carl.

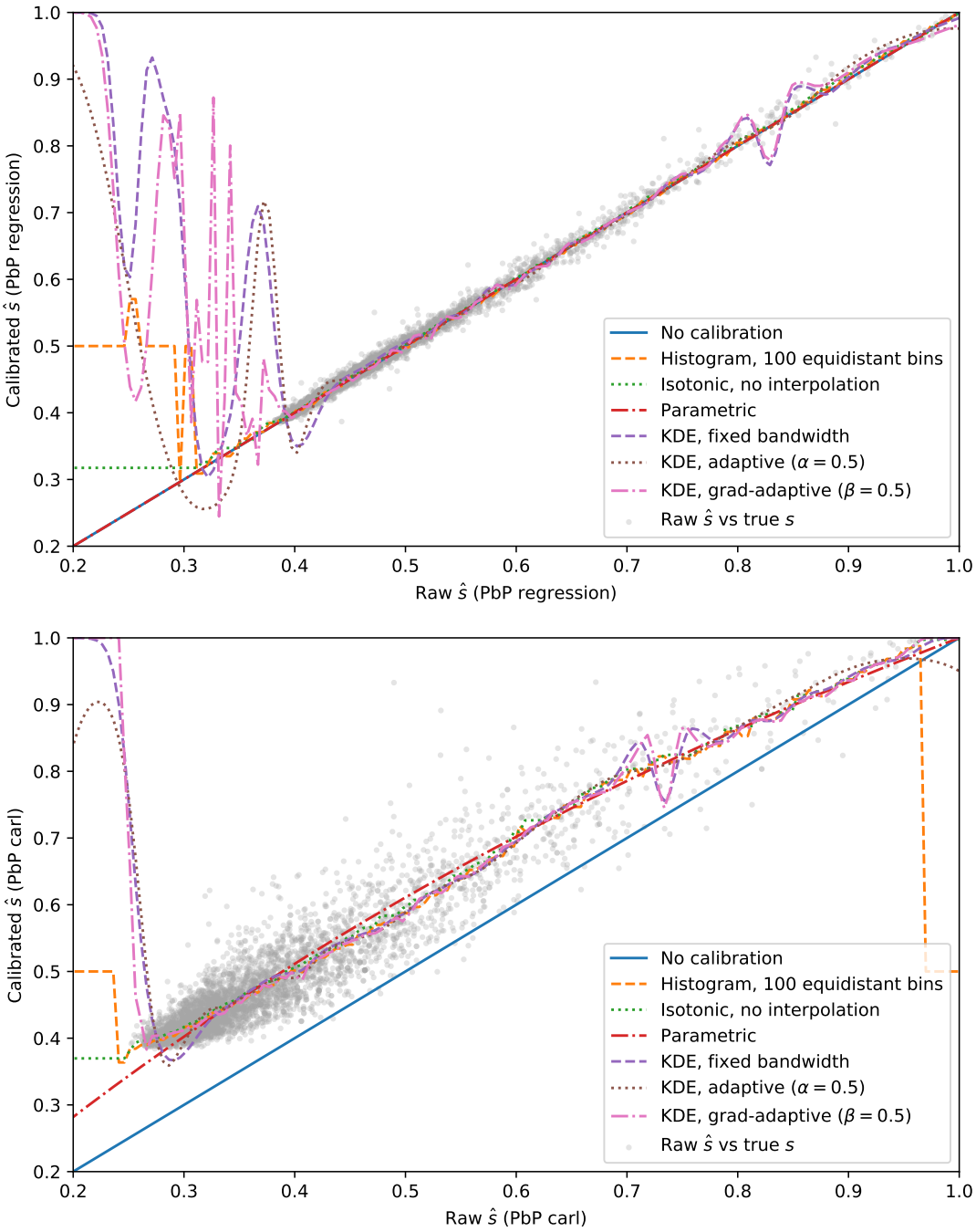

Figure 9: Calibration curves $s_{\text{calibrated}}(s_{\text{raw}})$ for different calibration strategies. The parametric approach is defined in Equation (6). Top: point-by-point regression. Bottom: point-by-point carl.

Density estimation strategy	Settings	$\mathrm{MSE}_{\mathbf{x}}\left[\log r(\mathbf{x};\boldsymbol{\theta}_0,\boldsymbol{\theta}_1)\right]$
t histogram	10^2 equidistant bins	0.229
	20^2 equidistant bins	0.211
	30^2 equidistant bins	0.102
	40^2 equidistant bins	0.108
	42^2 optimized bins	**0.010**
t KDE	fixed bandwidth	0.174
	adaptive (α = 0.25)	0.127
	adaptive (α = 0.5)	0.102
	adaptive (α = 0.75)	0.089
	adaptive (α = 1)	**0.086**
Local model fit		**0.057**
$t \cdot \Delta\theta$ histogram	25 equidistant bins	0.124
	50 equidistant bins	0.041
	100 equidistant bins	0.024
	200 equidistant bins	0.012
	500 equidistant bins	0.009
	25 equidistant bins, linear interpolation	0.164
	50 equidistant bins, linear interpolation	0.045
	100 equidistant bins, linear interpolation	0.025
	200 equidistant bins, linear interpolation	0.010
	500 equidistant bins, linear interpolation	0.008
	25 equidistant bins, spline interpolation	0.388
	50 equidistant bins, spline interpolation	0.066
	100 equidistant bins, spline interpolation	0.032
	200 equidistant bins, spline interpolation	0.010
	500 equidistant bins, spline interpolation	**0.008**
	25 variable bins	0.029
	50 variable bins	0.014
	100 variable bins	0.010
	200 variable bins	0.009
	500 variable bins	0.008
	146 optimized bins	0.008
$t \cdot \Delta\theta$ KDE	fixed bandwidth	0.124
	adaptive (α = 0.25)	0.090
	adaptive (α = 0.5)	0.068
	adaptive (α = 0.75)	0.053
	adaptive (α = 1)	**0.046**
$t \cdot \Delta\theta$ KDE	grad-adaptive (β = 0.2)	**0.035**
	grad-adaptive (β = 0.5)	0.046
	grad-adaptive (β = 1)	0.043
	grad-adaptive (β = 2)	0.046

Table 6: Comparison of different density estimation strategies after estimating the score.

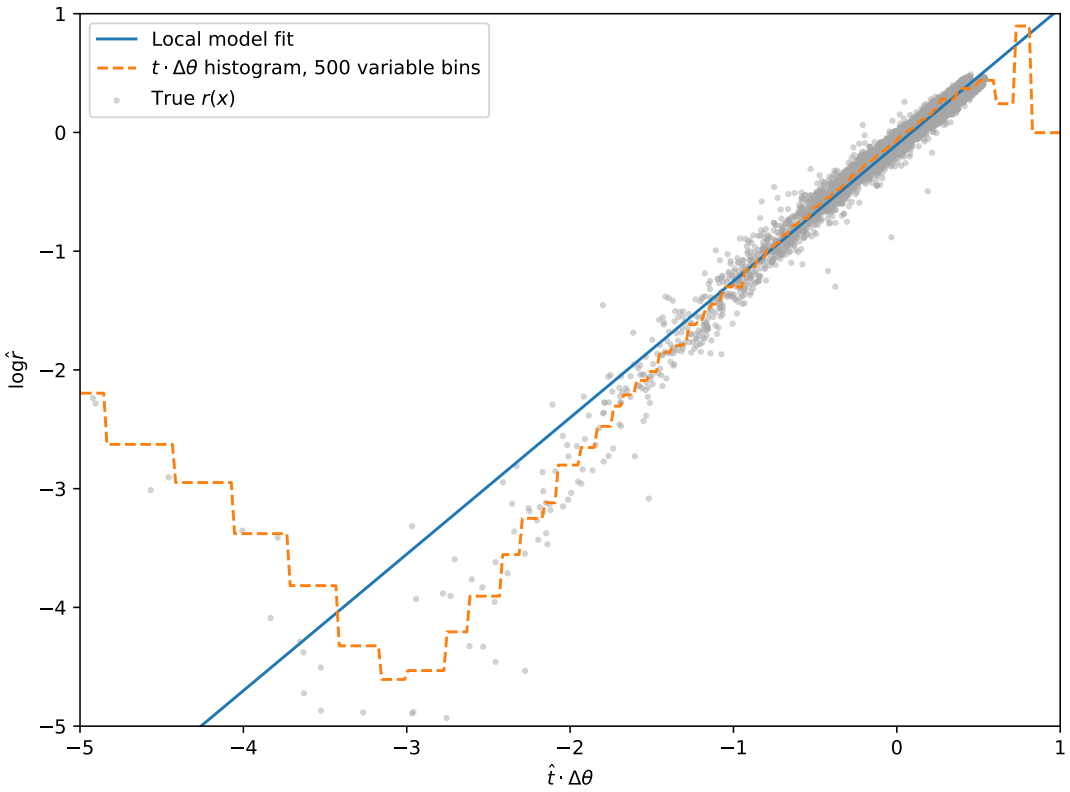

Figure 10: Comparison of different density estimation strategies after estimating the score.

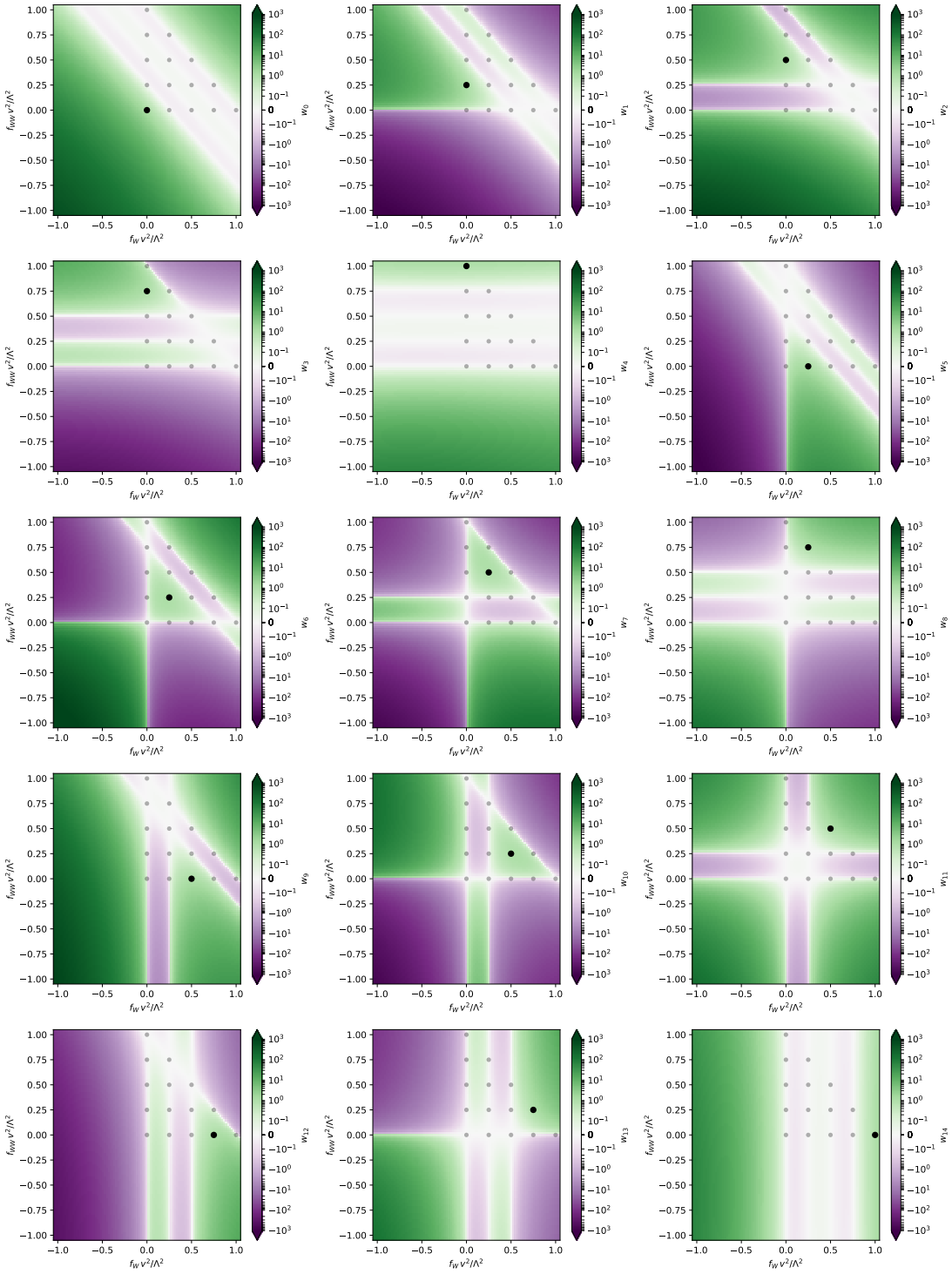

Figure 11: Morphing weights $w_i(\boldsymbol{\theta})$ for basis points chosen only in one quadrant of the parameter space of interest.

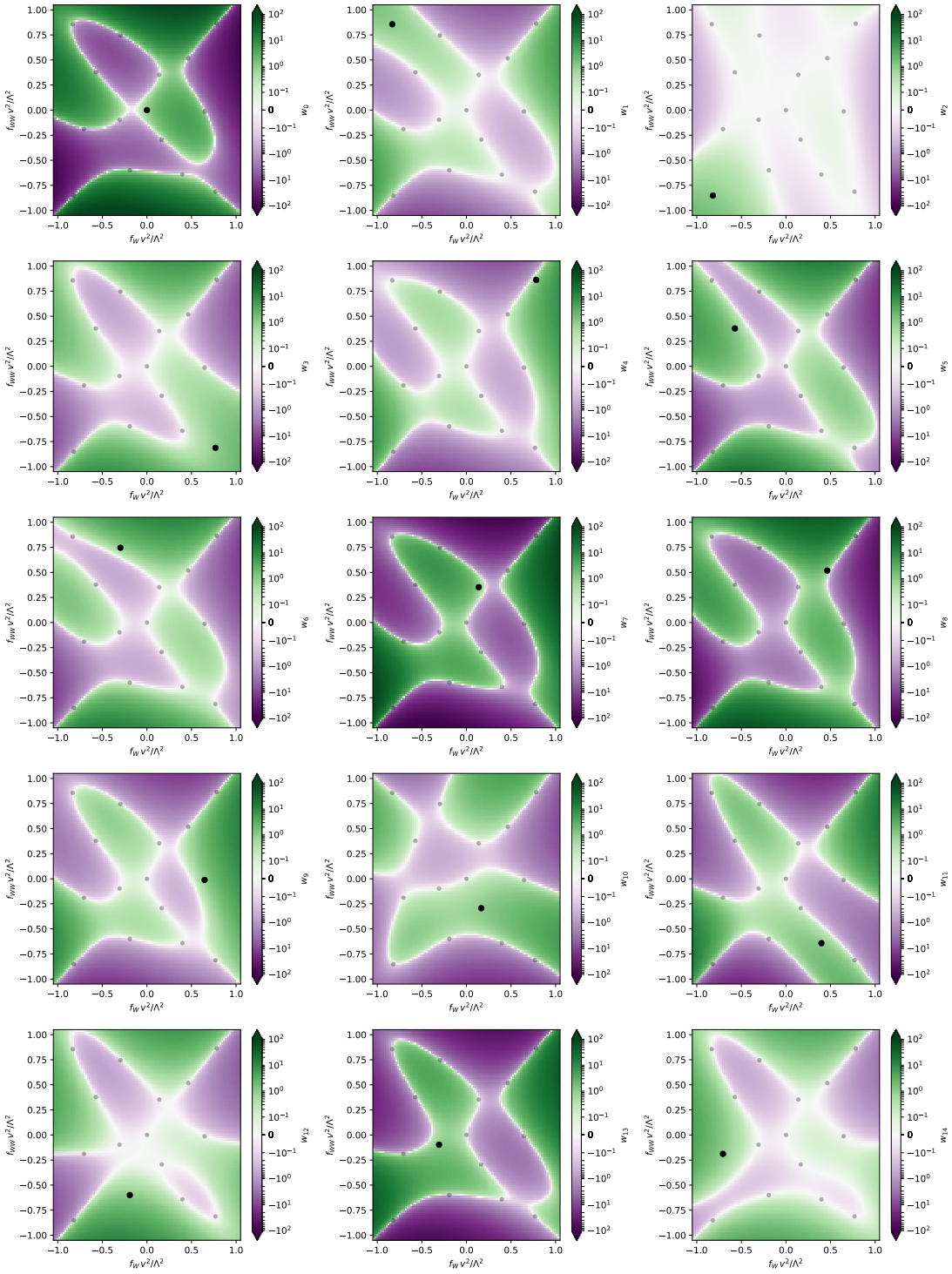

Figure 12: Morphing weights $w_i(\boldsymbol{\theta})$ for basis points distributed over the full relevant parameter space.

5. Conclusions

…

A. Two-dimensional likelihood

Instead of estimating the density over the full 15-dimensional phase-space, we also check some of the approaches on the much simpler problem of estimating the density of the two-dimensional distribution $p(p_{T,j1}, \Delta\phi_{jj}; \boldsymbol{\theta})$.

A.1. Benchmark hypothesis test

For two benchmark parameter points

$$\boldsymbol{\theta}_0 = \begin{pmatrix} -0.2 \\ -0.2 \end{pmatrix}, \quad \boldsymbol{\theta}_1 = \begin{pmatrix} 0.2 \\ 0.2 \end{pmatrix} \tag{13}$$

we first analyse how well `carl` can estimate the likelihood ratio

$$r(\mathbf{x}) = \frac{p(\mathbf{x}|\boldsymbol{\theta}_0)}{p(\mathbf{x}|\boldsymbol{\theta}_1)} \tag{14}$$

depending on the classifier, its hyperparameters, and a number of different settings.

For each of the four feature sets outlined above (2D, medium set, full kinematics, full kinematics plus derived variables), we tune the parameters in two steps. First, we perform a randomized scan over the hyperparameters, maximizing the classification power between event samples sampled from $p(\mathbf{x}|\boldsymbol{\theta}_0)$ and $p(\mathbf{x}|\boldsymbol{\theta}_1)$. In a second step we finetune these parameters on the mean squared error between the `carl` estimate $\log \hat{r}(\mathbf{x})$ and the true value $\log r(\mathbf{x})$.

The first classifier we consider is a **random forest**, more precisely extremely randomized trees, in the `sklearn.ensemble.ExtraTreesClassifier` implementation. The following parameters are tuned:

- `n_estimators`, the number of trees in the forest;
- `max_features`, the number of features considered in the search for the best split;
- `max_depth`, the maximum depth of the trees;
- `min_samples_split`, the minimum fraction of events at a node required for a split; and
- `min_samples_leaf`, the minimum fraction of events in a leaf.

A **multi-layer perceptron** implemented as a `sklearn.neural_network.MLPClassifier` is our second classifier. We optimize the following parameters:

- `hidden_layer_sizes`, describing the number of hidden layers and the number of neurons in each layer;
- `activation`, the activation function; and
- `alpha`, an L^2 penalty term.

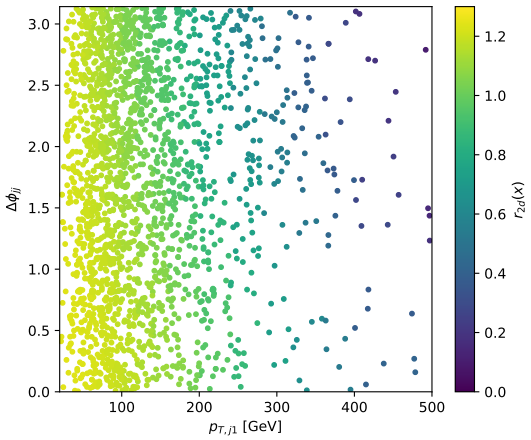

Figure 13: Truth likelihood ratio $r(\mathbf{x})$ as defined in Equation 14 for the 2D case. The effect of the binning in the calculation of $r_{2d}(\mathbf{x})$ is clearly visible.

For both classifiers, the kinematic features are first rescaled to a normal distribution with a `sklearn.preprocessing.StandardScaler`.

First we only consider two kinematic features, the leading jet momentum and the azimuthal angle between the two jets. Figure 13 shows how the truth likelihood ratio between the two points defined in Equation 13 depends on these observables.

We first tune hyperparameters with a randomized scan on the classification problem between unweighted event samples drawn from $p(\mathbf{x}|\boldsymbol{\theta}_0)$ and $p(\mathbf{x}|\boldsymbol{\theta}_1)$. Using `sklearn.model_selection.RandomizedSearchCV` we optimize on the ROC AUC. We give the optimal parameters in Table 7. Key to a good performance for the random forest is a maximal depth around 8. Only considering one feature at each splitting or requiring a large number of events for a splitting or at each leaf lead to a worse performance. Above a certain threshold, the number of trees does not have a huge effect. For the neural network, two hidden layers with 20 neurons each and tanh or ReLU activation functions perform best, while the value of the regulator α is not very important.

In Figures 14 and 15 we show how well `carl` can estimate the true likelihood ratio $r(\mathbf{x})$ with the tuned classifiers given in the right column of Table 7.

Next, we vary all parameters one by one and see how this affects the mean squared error of $\log \hat{r}(\mathbf{x})$. The results are shown in Figures 16 to 18.

The parameters that are best for the classification problem also work best for the estimation of the likelihood ratio with `carl`. The performance of the random forest crucially depends on the maximal depth, while the neural network works well with pretty much any setup, as long as each layer has enough neurons. To reduce training time, we therefore define a final setup of the neural network with a reduced number of 8 neurons in each hidden layer. For the random forest, we stick to the parameters in the right column of Table 7.

Classifier	Parameter	Range	Good	Best
Random forest	`n_estimators`	$50 \ldots 200$	$50 \ldots 200$	100
	`max_features`	$1, 2$	2	2
	`max_depth`	$1 \ldots 20, \infty$	$6 \ldots 8$	8
	`min_samples_split`	$0 \ldots 1$	$10^{-4} \ldots 10^{-3}$	10^{-3}
	`min_samples_leaf`	$0 \ldots 0.5$	0	0
Neural network	number of hidden layers	$1, 2, 3$	2	2
	neurons at each layer	$2 \ldots 20$ per layer	$(20, 20)$	$(20, 20)$
	activation function	tanh, ReLU, logistic	tanh, ReLU	tanh
	α	$0 \ldots 100$	$0 \ldots 1$	10^{-3}

Table 7: Hyperparameter scan on the classification problem between $\boldsymbol{\theta}_0$ and $\boldsymbol{\theta}_1$ for the 2D case. For both classifier and each parameter we show the considered range, the range of values with good results (i. e. ROC AUC close to the optimum), and the (rounded) optimal value.

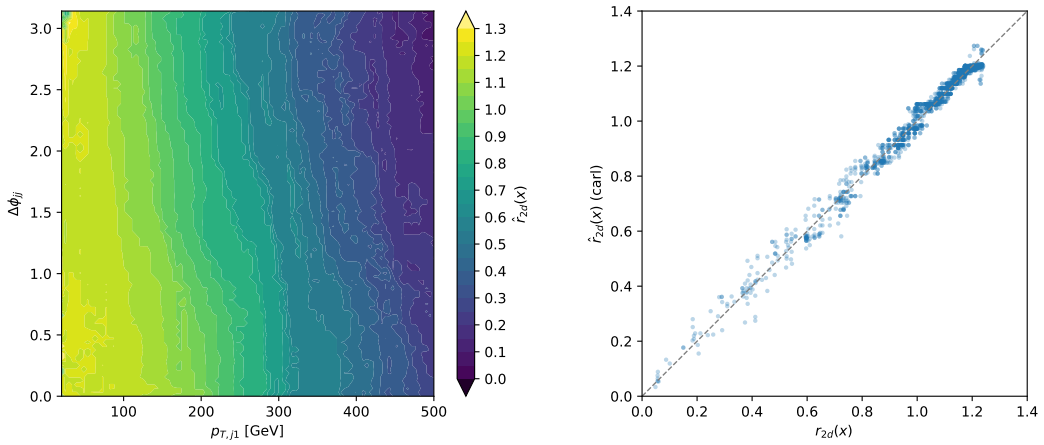

Figure 14: Likelihood ratio estimation with the tuned random forest for the 2D case. Left: Distribution of `carl`'s estimate $\hat{r}(\mathbf{x})$ as a function of the observables. Right: scatter plot between the true $r(\mathbf{x})$ and the estimate $\hat{r}(\mathbf{x})$

.

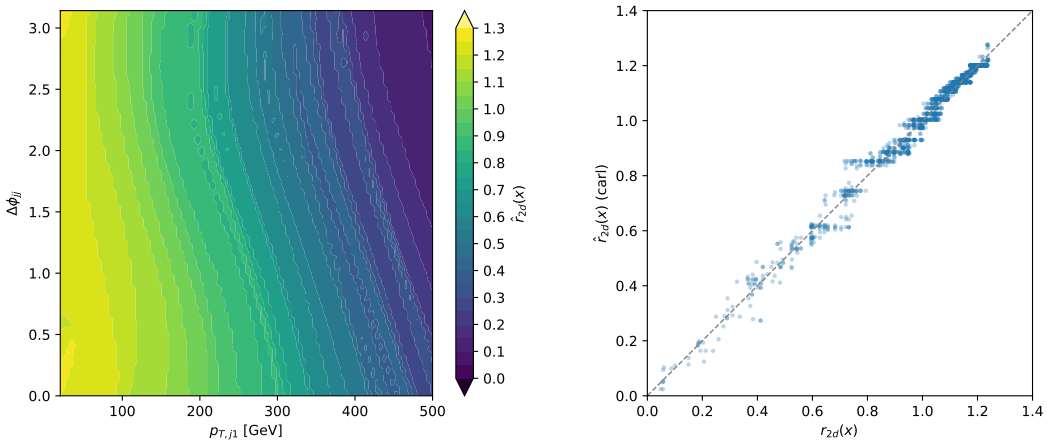

Figure 15: Likelihood ratio estimation with the tuned neural network for the 2D case. Left: Distribution of `carl`'s estimate $\hat{r}(\mathbf{x})$ as a function of the observables. Right: scatter plot between the true $r(\mathbf{x})$ and the estimate $\hat{r}(\mathbf{x})$

.

Figure 18 shows the effect of the size of the training samples as well as the binning of `carl`'s calibration histograms. Regardless of the problematic distribution of weights in the input samples, the performance becomes much better with larger training samples. We pick a size of 200 000 events as a balance between computation time and performance. The calibration histograms should have at least 20 bins, and variable bin widths do not seem to work well at all.

As demonstrated in Figure 19, training random forests with weighted samples does not improve the performance, even if the full training sample of 5 million events is used. However, we did not try to tune the random forest parameters with weighted input.

We summarise `carl`'s performance before and after tuning and compare the two classifiers in Figure 20. We find that the neural network performs slightly better, and is much less dependent on the choice of parameters. The output of `carl` with this final neural network setup is shown in Figure 21. As a matter of fact, the smooth neural net output might be closer to the real likelihood ratio than our calculated $r(\mathbf{x})$ with its binning artefacts.

A.2. Validation

After tuning the parameters for one specific choice of benchmark points, we now check how well these settings work for the likelihood ratio between two different benchmark points

$$\boldsymbol{\theta}'_0 = \begin{pmatrix} 0.5 \\ -0.5 \end{pmatrix}, \quad \boldsymbol{\theta}'_1 = \begin{pmatrix} -0.5 \\ 0.5 \end{pmatrix}. \tag{15}$$

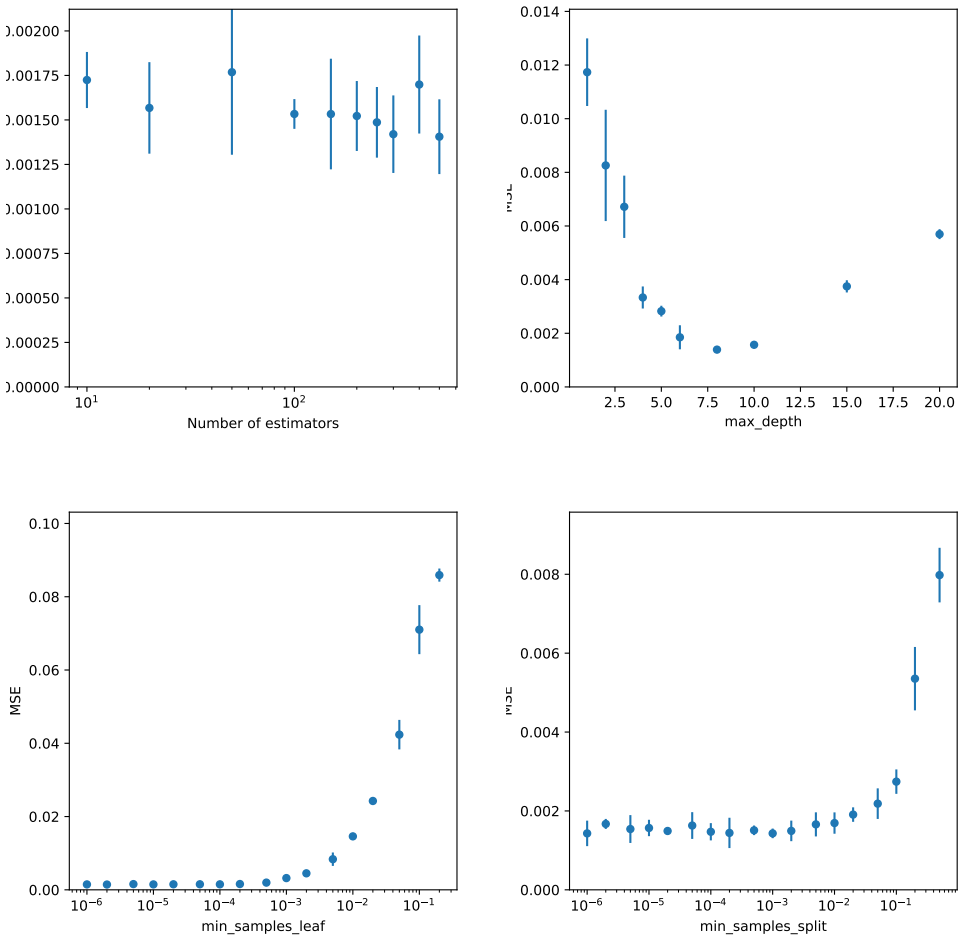

Figure 16: carl performance as a function of the random forest hyperparameters. We show the mean squred error of the estimated log likelihood ratio between two benchmark points for the 2D case. Each data point shows the mean of five calculations, the error bars are 95% confidence intervals.

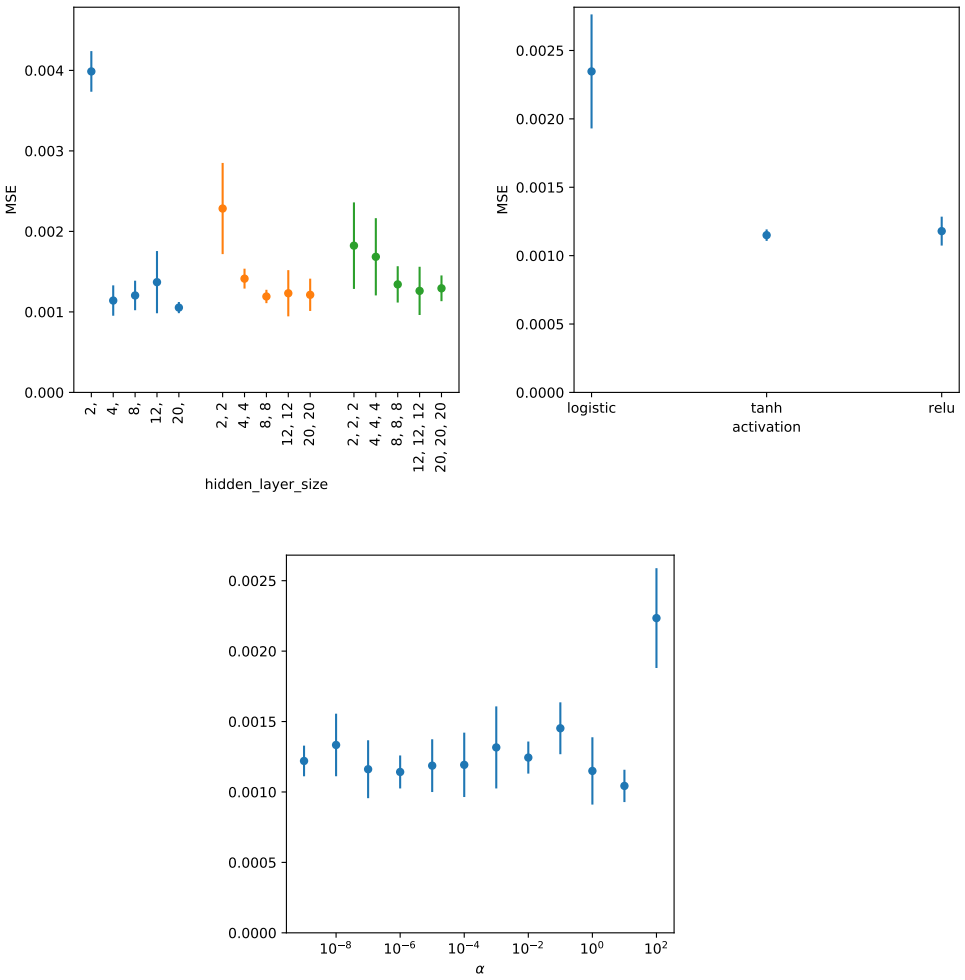

Figure 17: carl performance as a function of the neural network hyperparameters. Mean squred error of the estimated log likelihood ratio between two benchmark points for the 2D case. Each data point shows the mean of five calculations, the error bars are 95% confidence intervals.

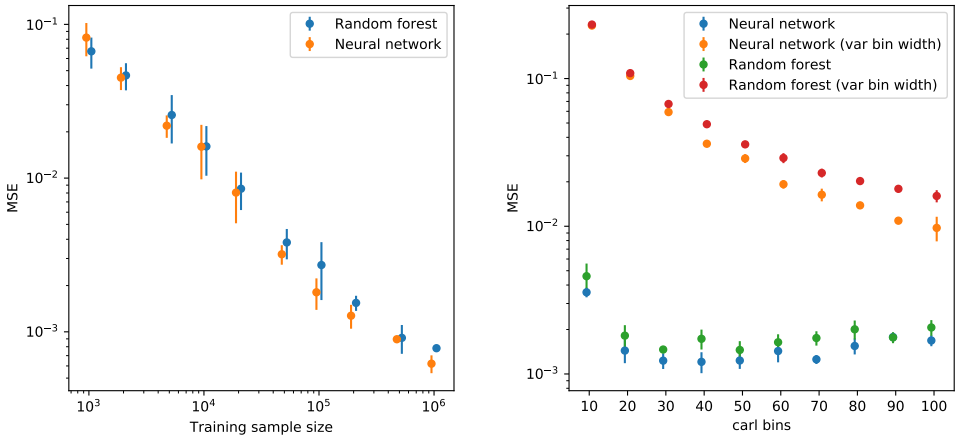

Figure 18: carl performance as a function of the training sample size (left) and the number of bins in the `carl` calibration histograms. We show the mean squred error of the estimated log likelihood ratio between two benchmark points for the 2D case. Each data point shows the mean of five calculations, the error bars are 95% confidence intervals.

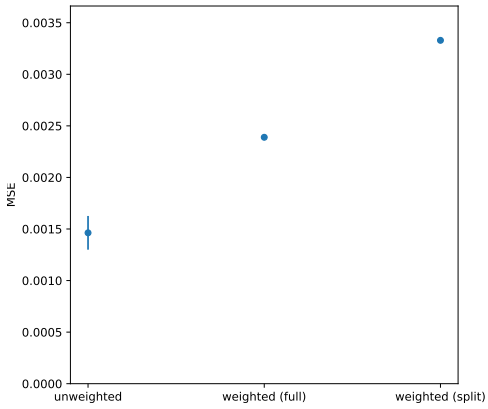

Figure 19: carl performance with weighted training samples. We show the mean squred error of the estimated log likelihood ratio between two benchmark points based on the two-dimensional feature space.

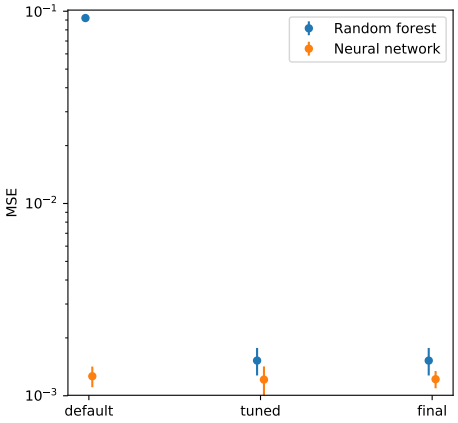

Figure 20: carl performance for the 2D case with `sklearn` default hyperparameters, the best parameters according to the initial hyperparameter scan, and the final parameters as defined in the text. We show the mean squred error of the estimated log likelihood ratio between two benchmark points. Each data point shows the mean of five calculations, the error bars are 95% confidence intervals.

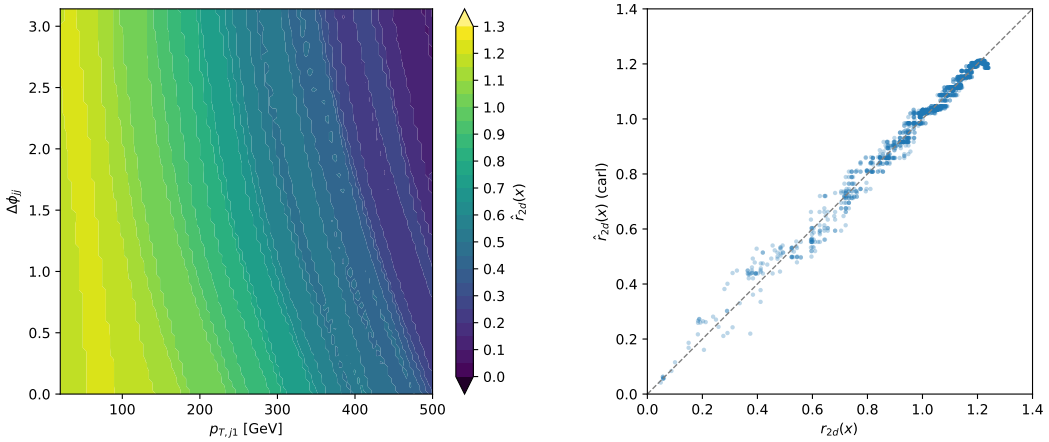

Figure 21: Likelihood ratio estimation with the final neural network for the 2D case. Left: Distribution of `carl`'s estimate $\hat{r}(\mathbf{x})$ as a function of the observables. Right: scatter plot between the true $r(\mathbf{x})$ and the estimate $\hat{r}(\mathbf{x})$

.

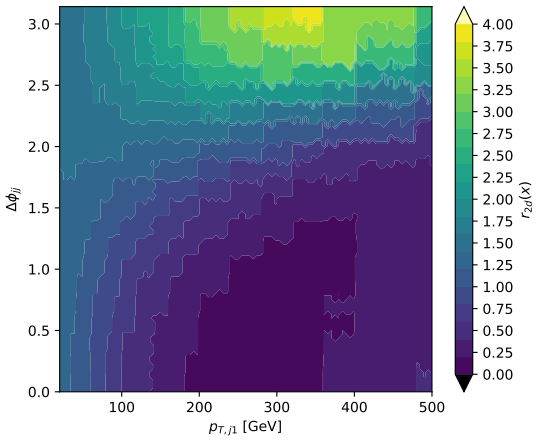

Figure 22: Truth likelihood ratio $r(\mathbf{x})$ between the validation benchmark points for the 2D case as a function of the observables.

Figure 22 shows the truth likelihood ratio between the validation benchmark points of Equation 15 as a function of the two observables.

In Figures 23 to 25 we show `carl`'s estimation based on the final settings tuned in the previous section. How this performance depends on the choice of hyperparameters is illustrated in Figure 26.

All in all, we find that the neural net performs very well without further fine-tuning, i. e. that the same settings optimised on the comparison of $\boldsymbol{\theta}_0$ versus $\boldsymbol{\theta}_1$ also work well for $\boldsymbol{\theta}'_0$ versus $\boldsymbol{\theta}'_1$. The random forest, on the other hand, is not so flexible. The two different hypothesis comparisons require different choices of the maximal depth of the trees, and what works best for one choice of parameter points leads to a poor performance for a different choice.

A.3. Likelihood contours

After optimizing our classifier setup on the distinction between two distinct hypotheses, and validating the results on another two hypotheses, we now turn towards the more relevant problem of estimating $\boldsymbol{\theta}$ based on some measurements. We generate 25 000 toy events sampled from the probability distribution for the SM,

$$\boldsymbol{\theta}_{\text{observed}} = \begin{pmatrix} 0 \\ 0 \end{pmatrix} . \tag{16}$$

For 100 points randomly sampled in $\boldsymbol{\theta} \in [-0.9, 0.9]^2$, we calculate the true expected likelihood ratio to

$$\boldsymbol{\theta}_1 = \begin{pmatrix} -0.23 \\ 0.30 \end{pmatrix} \tag{17}$$

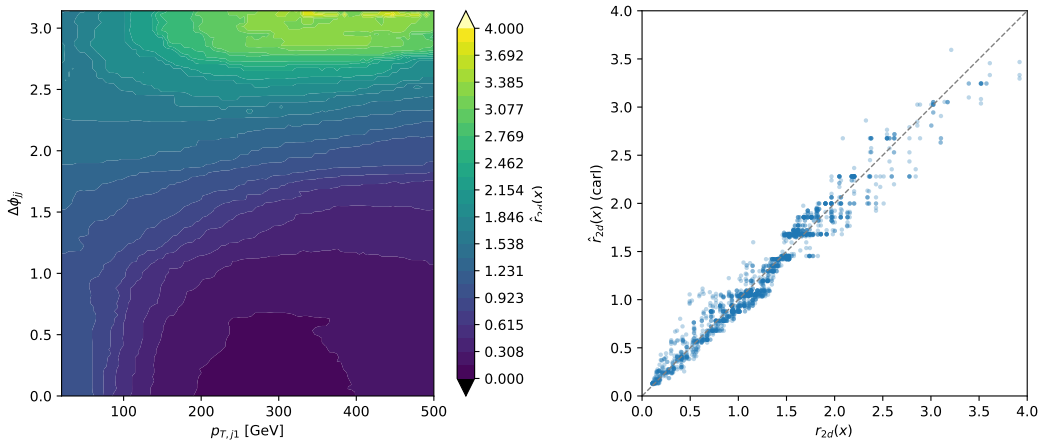

Figure 23: Likelihood ratio estimation with the tuned random forest on separate validation hypotheses for the 2D case. Left: Distribution of `carl`'s estimate $\hat{r}(\mathbf{x})$ as a function of the observables. Right: scatter plot between the true $r(\mathbf{x})$ and the estimate $\hat{r}(\mathbf{x})$

.

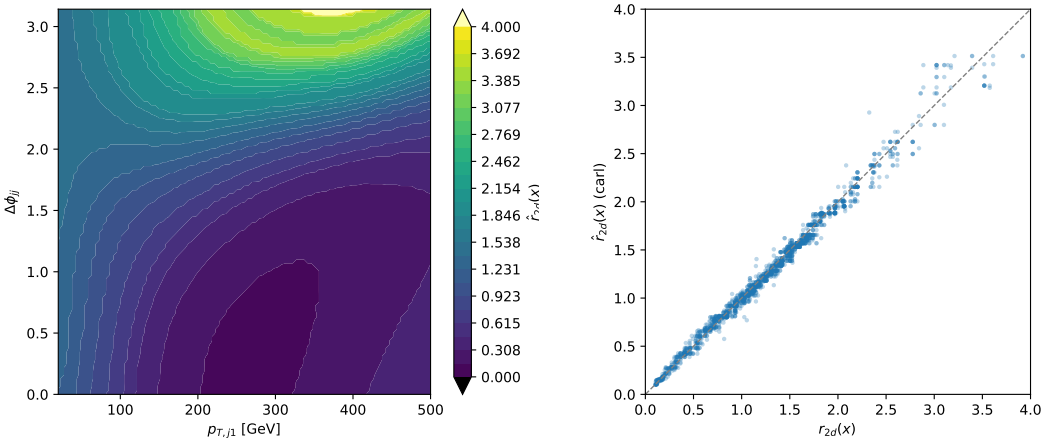

Figure 24: Likelihood ratio estimation with the neural network on separate validation hypotheses for the 2D case. Left: Distribution of `carl`'s estimate $\hat{r}(\mathbf{x})$ as a function of the observables. Right: scatter plot between the true $r(\mathbf{x})$ and the estimate $\hat{r}(\mathbf{x})$

.

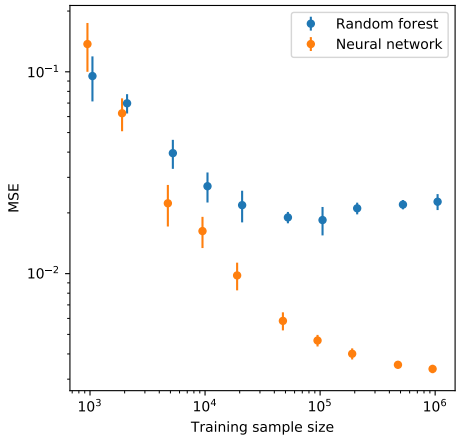

Figure 25: carl performance on the validation hypotheses for different training sample sizes and classifiers). We show the mean squred error of the estimated log likelihood ratio between two validation benchmark points for the 2D case. Each data point shows the mean of five calculations, the error bars are 95% confidence intervals.

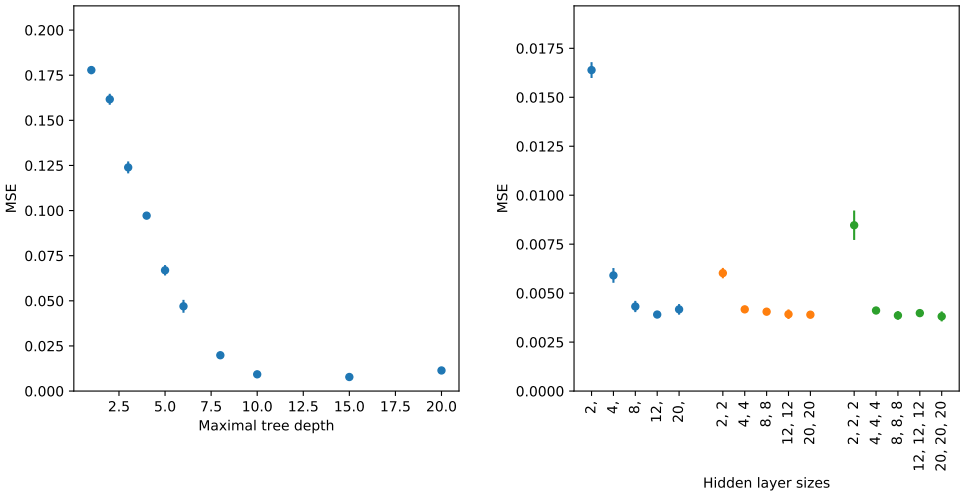

Figure 26: carl performance on the validation hypotheses for different hyperparameters. We show the mean squred error of the estimated log likelihood ratio between two validation benchmark points for the 2D case. Each data point shows the mean of five calculations, the error bars are 95% confidence intervals.

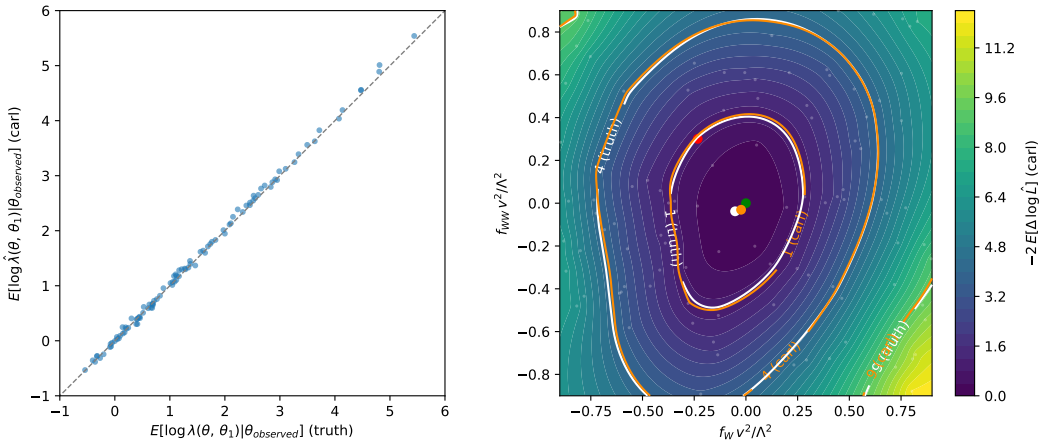

Figure 27: Inference from truth likelihood ratio and `carl`'s estimate for the 2D case. Left: scatter plot showing the difference between the exact expected likelihood ratio for 100 randomly sampled points and $\boldsymbol{\theta}_1$ and `carl`'s estimate. Right: true (white) and approximate (orange) likelihood contours, using a Gaussian Process for interpolation. The white and orange dots show the exact and approximate maximum-likelihood estimators. The green and red dots show $\boldsymbol{\theta}_{\text{observed}}$ and $\boldsymbol{\theta}_1$, respectively. Finally, the small grey dots show the sampled parameter points at which the likelihood ratio was evaluated.

as well as the corresponding `carl` estimate. Finally, we interpolate between these points with a Gaussian Process with Matérn kernel with $\nu = 0.5$.

Following the results of the previous sections, for the two-dimensional feature space we use a neural network with two hidden layers of 8 neurons each, a tanh activation function, and a regulator $\alpha = 0.001$. The training samples consist of 200 000 unweighted samples each, and the `carl` default for the fixed-size binning of the calibration histograms is used.

In Figure 27 we show the results for the two-dimensional feature space. The approximate likelihood map agrees impressively well with the exact one.

B. Detailed tuning of the point-by-point algorithms

One class of algorithms simplifies the inference problem by estimating the likelihood ratio $r(\mathbf{x}; \boldsymbol{\theta}_0, \boldsymbol{\theta}_1)$ for different pairs $(\boldsymbol{\theta}_0, \boldsymbol{\theta}_1)$ separately. We analyse two approaches: calibrated classification (`carl`), and regression on the likelihood ratio.

B.1. Calibrated classifiers

B.1.1. Benchmark hypothesis test

For two benchmark parameter points

$$\boldsymbol{\theta}_0 = \begin{pmatrix} -0.2 \\ -0.2 \end{pmatrix}, \quad \boldsymbol{\theta}_1 = \begin{pmatrix} 0.2 \\ 0.2 \end{pmatrix} \tag{18}$$

we first analyse how well `carl` can estimate the likelihood ratio

$$r(\mathbf{x}) = \frac{p(\mathbf{x}|\boldsymbol{\theta}_0)}{p(\mathbf{x}|\boldsymbol{\theta}_1)} \tag{19}$$

depending on the classifier, its hyperparameters, and a number of different settings.

In Figure 28 we show the distributions of all 42 considered kinematic variables for these two parameter points.

For each of the four feature sets outlined above (2D, medium set, full kinematics, full kinematics plus derived variables), we tune the parameters in two steps. First, we perform a randomized scan over the hyperparameters, maximizing the classification power between event samples sampled from $p(\mathbf{x}|\boldsymbol{\theta}_0)$ and $p(\mathbf{x}|\boldsymbol{\theta}_1)$. In a second step we finetune these parameters on the mean squared error between the `carl` estimate $\log \hat{r}(\mathbf{x})$ and the true value $\log r(\mathbf{x})$.

The first classifier we consider is a **random forest**, more precisely extremely randomized trees, in the `sklearn.ensemble.ExtraTreesClassifier` implementation. The following parameters are tuned:

- `n_estimators`, the number of trees in the forest;
- `max_features`, the number of features considered in the search for the best split;
- `max_depth`, the maximum depth of the trees;
- `min_samples_split`, the minimum fraction of events at a node required for a split; and
- `min_samples_leaf`, the minimum fraction of events in a leaf.

A **multi-layer perceptron** implemented as a `sklearn.neural_network.MLPClassifier` is our second classifier. We optimize the following parameters:

- `hidden_layer_sizes`, describing the number of hidden layers and the number of neurons in each layer;

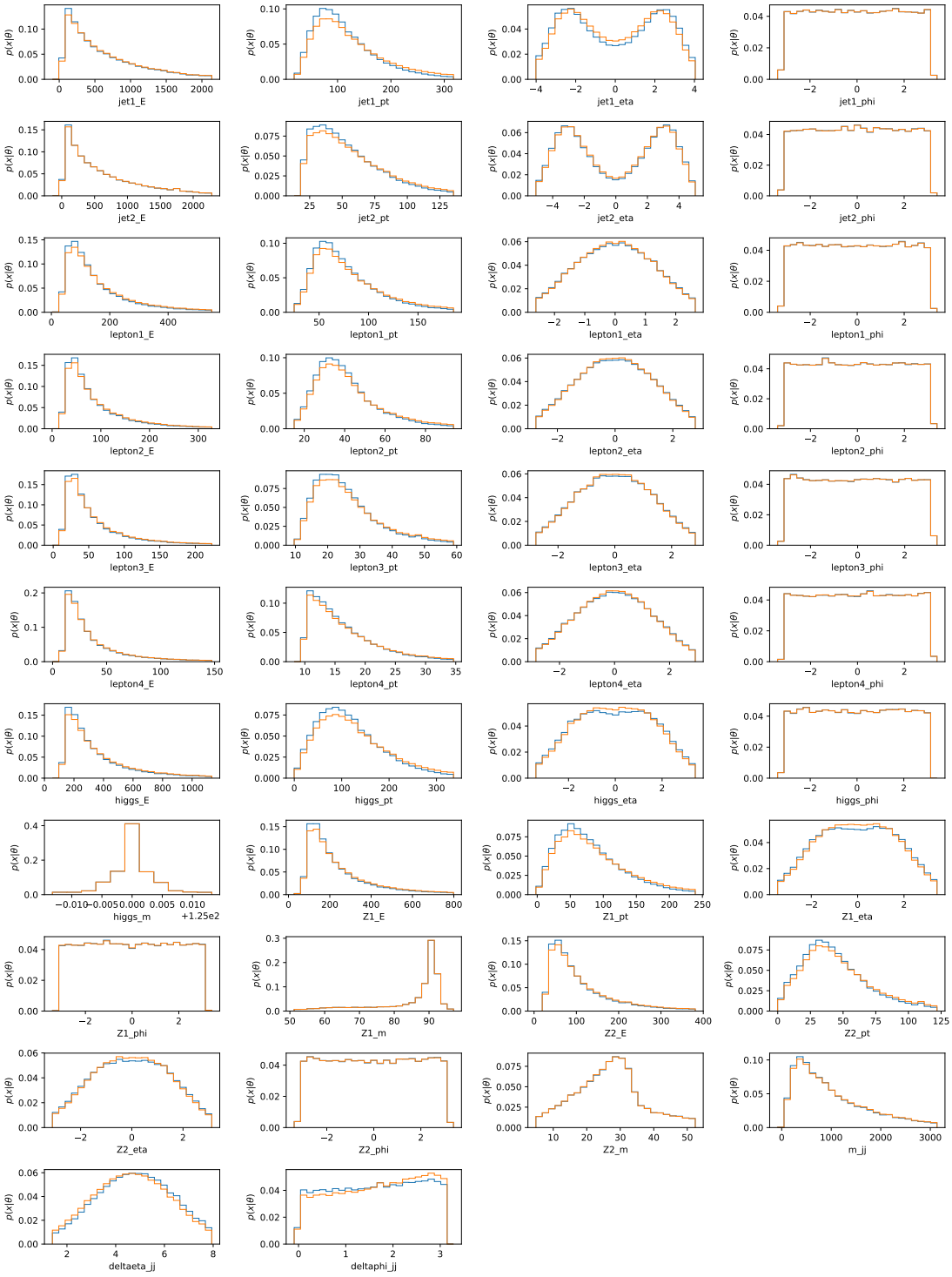

Figure 28: Distributions of all kinematic observables for $\boldsymbol{\theta}_0$ (blue) and $\boldsymbol{\theta}_1$ (orange).

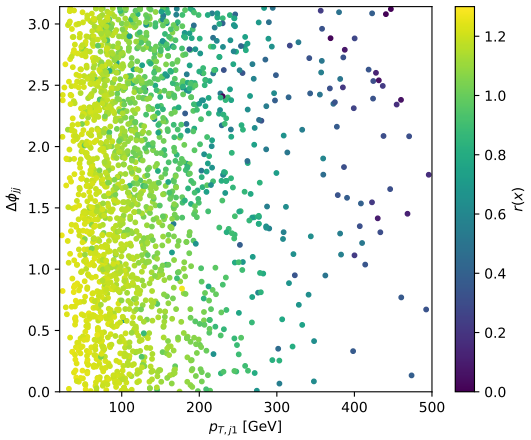

Figure 29: Truth likelihood ratio $r(\mathbf{x})$ as defined in Equation 14 for the fully differential kinematics as a function of two particular observables.

- `activation`, the activation function; and
- `alpha`, an L^2 penalty term.

For both classifiers, the kinematic features are first rescaled to a normal distribution with a `sklearn.preprocessing.StandardScaler`.

Estimators for the medium feature set

We now try to see if we can estimate the fully differential likelihood ratio equally well. This is a much more difficult problem: Figure 29 shows a scatter plot of the truth likelihood ratio between the two points defined in Equation 13 over the jet p_T and $\Delta\phi_{jj}$, showing sizable fluctuations between close points due to the other directions in phase space. First we train `carl` on the medium set of 15 observables defined in subsection 2.4.

We first tune hyperparameters with a randomized scan on the classification problem between unweighted event samples drawn from $p(\mathbf{x}|\boldsymbol{\theta}_0)$ and $p(\mathbf{x}|\boldsymbol{\theta}_1)$. Using `sklearn.model_selection.RandomizedSearchCV` we optimize on the ROC AUC. We give the optimal parameters in Table 9. The optimal random forests are now quite a bit deeper than in the 2d case.

In Figure 30 we show how well `carl` can estimate the true likelihood ratio $r(\mathbf{x})$ with the tuned classifiers given in the right column of Table 8. The performance is much worse than in the 2d case. Figure 31 shows a scatter plot of the kinematic observables versus the exact ratio $r(\mathbf{x})$ and the estimate $\hat{r}(\mathbf{x})$.

Next, we vary all parameters one by one and see how this affects the mean squared error of $\log \hat{r}(\mathbf{x})$. The results are shown in Figures 32. and 33.

	Parameter	Range	Random	Bayes 1	Bayes 2	Benchmark
RF	`n_estimators`	50 . . . 200	100			100
	`max_features`	1 . . . 24	6			6
	`max_depth`	1 . . . 20, ∞	12			12
	`min_samples_split`	0 . . . 1	0			0
	`min_samples_leaf`	0 . . . 0.5	0			0
NN	number of hidden layers	1 . . . 5	2	1	1	2
	neurons at each layer	2 . . . 100 per layer	$(8, 8)$	(5)	(5)	$(8, 8)$
	activation function	tanh, ReLU, logistic	logistic	logistic	logistic	logistic
	α	0 . . . 100	0	10^{-9}	0	0
	initial learning rate	0.0001 . . . 0.01		0.01	0.0067	0.001

Table 8: Hyperparameter scan on the classification problem between $\boldsymbol{\theta}_0$ and $\boldsymbol{\theta}_1$ using the medium feature set as input. For each parameter of the random forrest (RF) and the neural network (NN) we show the considered range and the best settings as determined by a randomized search CV, a Bayesian optimization procedure geared towards exploitation, and a Bayesian optimization procedure geared towards exploration. The final column shows our baseline parameters for the next step of the optimization procedure.

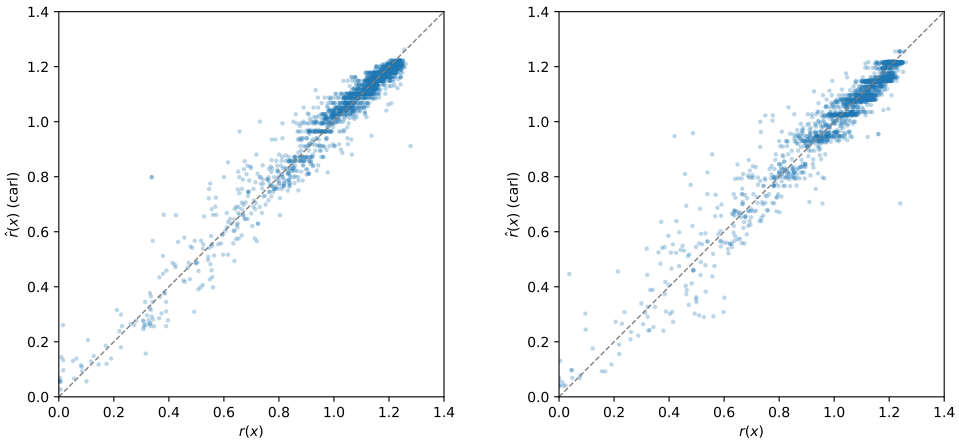

Figure 30: Likelihood ratio estimation with the tuned classifiers (left: random forest, right: neural network) using the medium feature set as input. We show a scatter plot between the true $r(\mathbf{x})$ and the estimate $\hat{r}(\mathbf{x})$.

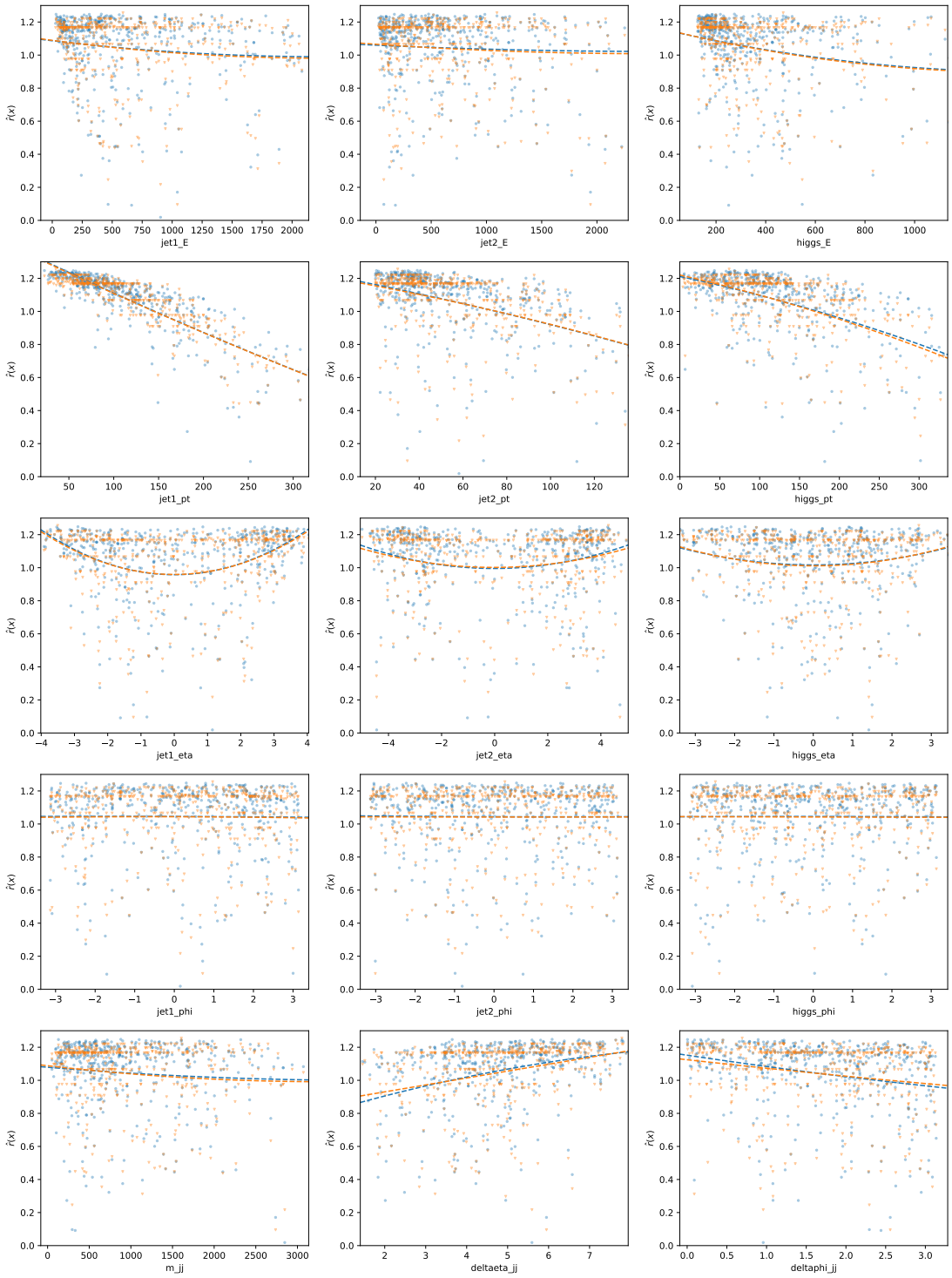

Figure 31: Correlation of $r(\mathbf{x})$ (blue) and `carl` estimate $\hat{r}(\mathbf{x})$ (orange) with various kinematic observables. As classifier we use the tuned random forest with the medium feature set as input. To guide the eye we also show fitted second-order polynomials.

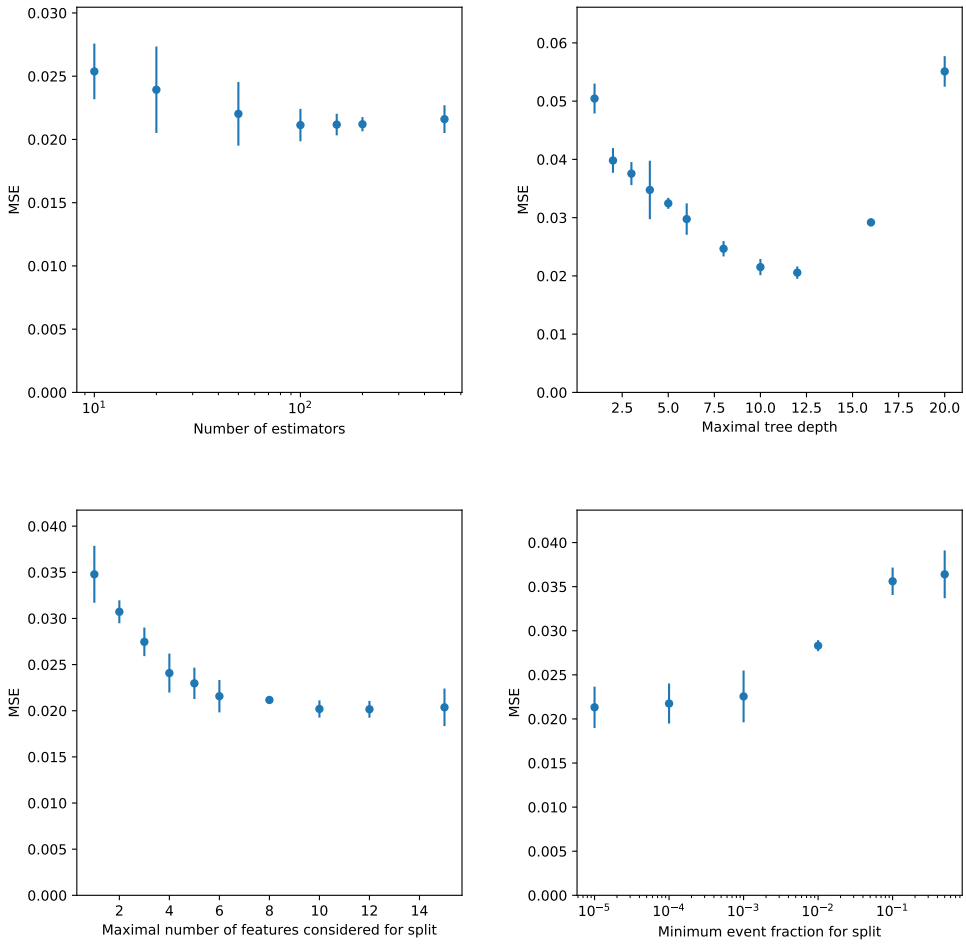

Figure 32: carl performance as a function of the random forest hyperparameters. We show the mean squred error of the estimated log likelihood ratio between two benchmark points using the medium feature set as input. Each data point shows the mean of three calculations, the error bars are 95% confidence intervals.

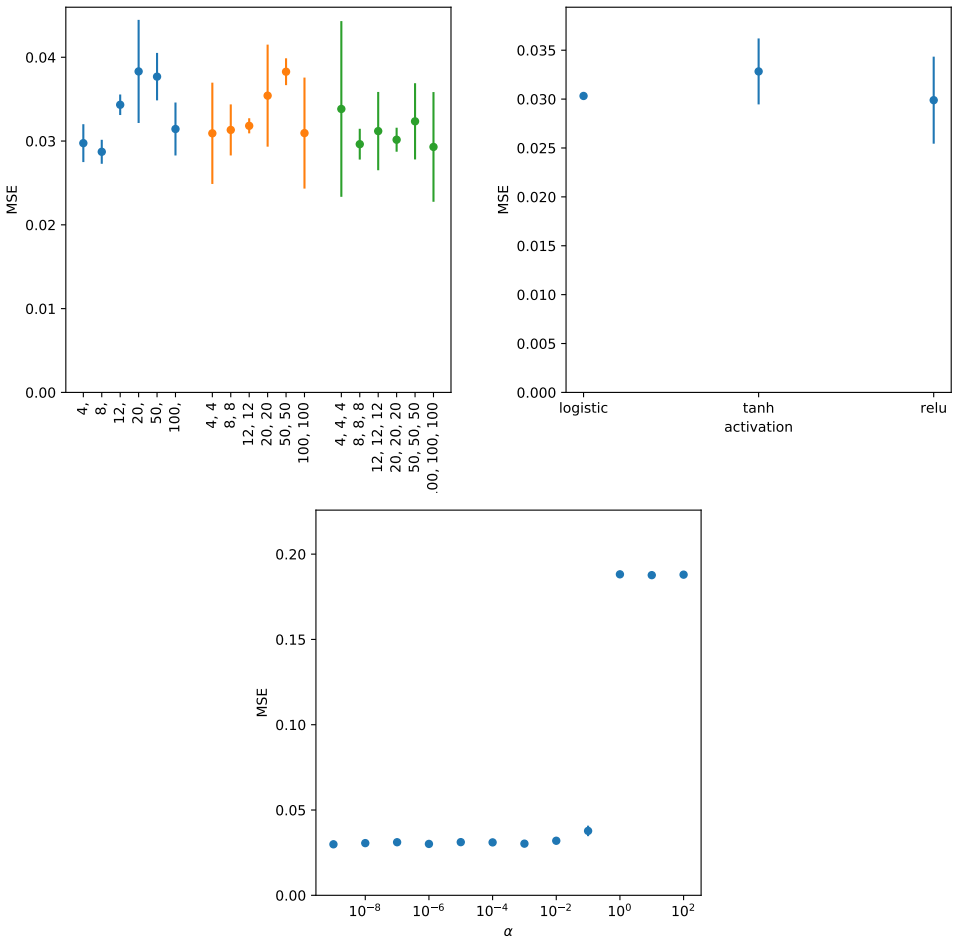

Figure 33: carl performance as a function of the neural network hyperparameters. Mean squred error of the estimated log likelihood ratio between two benchmark points using the medium feature set as input. Each data point shows the mean of three calculations, the error bars are 95% confidence intervals.

	Parameter	Range	Random	Bayes 1	Bayes 2	Benchmark
RF	`n_estimators`	50 ... 200	200			200
	`max_features`	1 ... 24	20			20
	`max_depth`	1 ... 20, ∞	12			12
	`min_samples_split`	0 ... 1	0.001			0.001
	`min_samples_leaf`	0 ... 0.5	0			0
NN	number of hidden layers	1 ... 5	2	2	2	2
	neurons at each layer	2 ... 100 per layer	$(4, 12)$	$(50, 50)$	$(50, 20)$	$(4, 12)$
	activation function	tanh, ReLU, logistic	logistic	logistic	logistic	logistic
	α	0 ... 100	0	0	10^{-9}	0
	initial learning rate	0.0001 ... 0.01		0.0013	0.0018	0.001

Table 9: Hyperparameter scan on the classification problem between $\boldsymbol{\theta}_0$ and $\boldsymbol{\theta}_1$ using the full feature set as input. For each parameter of the random forrest (RF) and the neural network (NN) we show the considered range and the best settings as determined by a randomized search CV, a Bayesian optimization procedure geared towards exploitation, and a Bayesian optimization procedure geared towards exploration. The final column shows our baseline parameters for the next step of the optimization procedure.

For the random forest, the results of the hyperparameter scan lead to a good performance, and there is no obvious reason to change any of the settings in the right column of Table 8. For the neural network, we define a final setup with a ReLU activation function instead of the logistic one, with all other settings in the right column of Table 8 unchanged.

Estimators for the full feature set

Now we use the full kinematics including the Higgs decay patterns, parametrized by the 24 energies, transverse momenta, and angles given in configuration 2 in subsection 2.4.

We first tune hyperparameters with a randomized scan on the classification problem between unweighted event samples drawn from $p(\mathbf{x}|\boldsymbol{\theta}_0)$ and $p(\mathbf{x}|\boldsymbol{\theta}_1)$. Using `sklearn.model_selection.RandomizedSearchCV` we optimize on the ROC AUC. We give the optimal parameters in Table 9. The optimal random forests are now quite a bit deeper than in the 2d case. For the neural networks we find a surprising preference for small hidden layers, and logistic activation functions are now preferred.

In Figure 34 we show how well `carl` can estimate the true likelihood ratio $r(\mathbf{x})$ with the tuned classifiers given in the right column of Table 9. The performance is much worse than in the 2d case.

Next, we vary all parameters one by one and see how this affects the mean squared error of $\log \hat{r}(\mathbf{x})$. The results are shown in Figures 35 to 37. For the random forest, we again find that the parameters tuned on the classification problem already lead to a good performance for the estimation of $r(\mathbf{x})$. For the neural network, switching to the ReLU activation function improves the estimation, so we use it in our final setup.

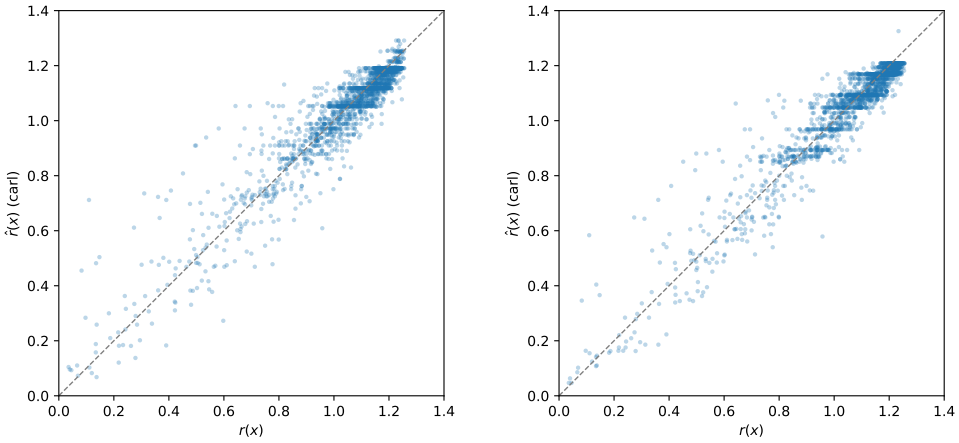

Figure 34: Likelihood ratio estimation with the tuned classifiers (left: random forest, right: neural network) using the full feature set as input. We show a scatter plot between the true $r(\mathbf{x})$ and the estimate $\hat{r}(\mathbf{x})$

.

Figure 37 shows the effect of the size of the training samples. Again, we choose 200 000 events per sample as a compromise between computation time and performance.

Estimators for the full plus derived feature set

Finally, we analyse whether the estimator is improved if we include an additional 18 derived quantities, see subsection 2.4. The results of the hyperparameter scan are given in Table 9.

In Figure 38 we show how well `carl` can estimate the true likelihood ratio $r(\mathbf{x})$ with the tuned classifiers given in the right column of Table 10. The performance is slightly better than without the derived variables.

Again, we vary the parameters one by one and check if we can improve the performance, see Figures 39 to 41. The results mirror those in the previous section. The only improvement we can find is switching to the ReLU activation function for the neural network in our final setup, all other settings remain as in the right column of Table 10.

Calibration

We now stick to a random forest trained on the medium feature set with hyperparameters as given in the last column of Table 8 and have a closer look at `carl`'s calibration procedure.

First, Figure 42 shows the output of an uncalibrated classifier and compares it to the calibrated classifier. Clearly, calibration improves the performance.

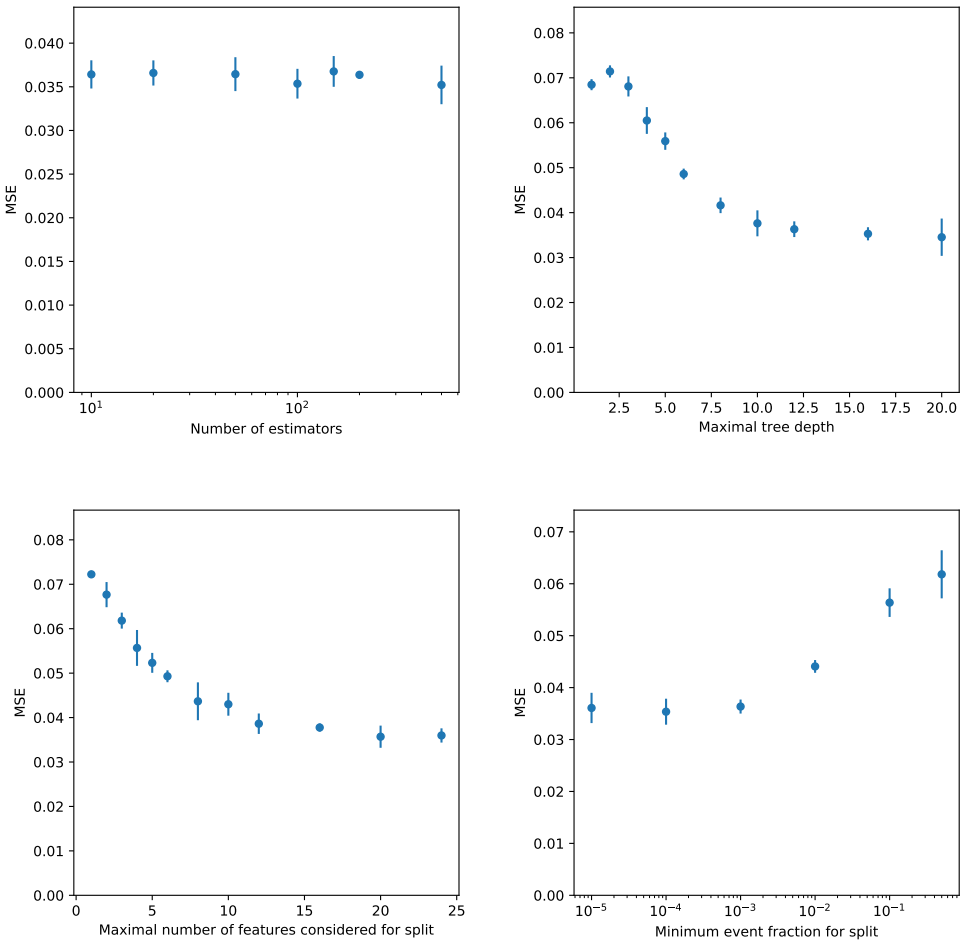

Figure 35: carl performance as a function of the random forest hyperparameters. We show the mean squred error of the estimated log likelihood ratio between two benchmark points using the full feature set as input. Each data point shows the mean of three calculations, the error bars are 95% confidence intervals.

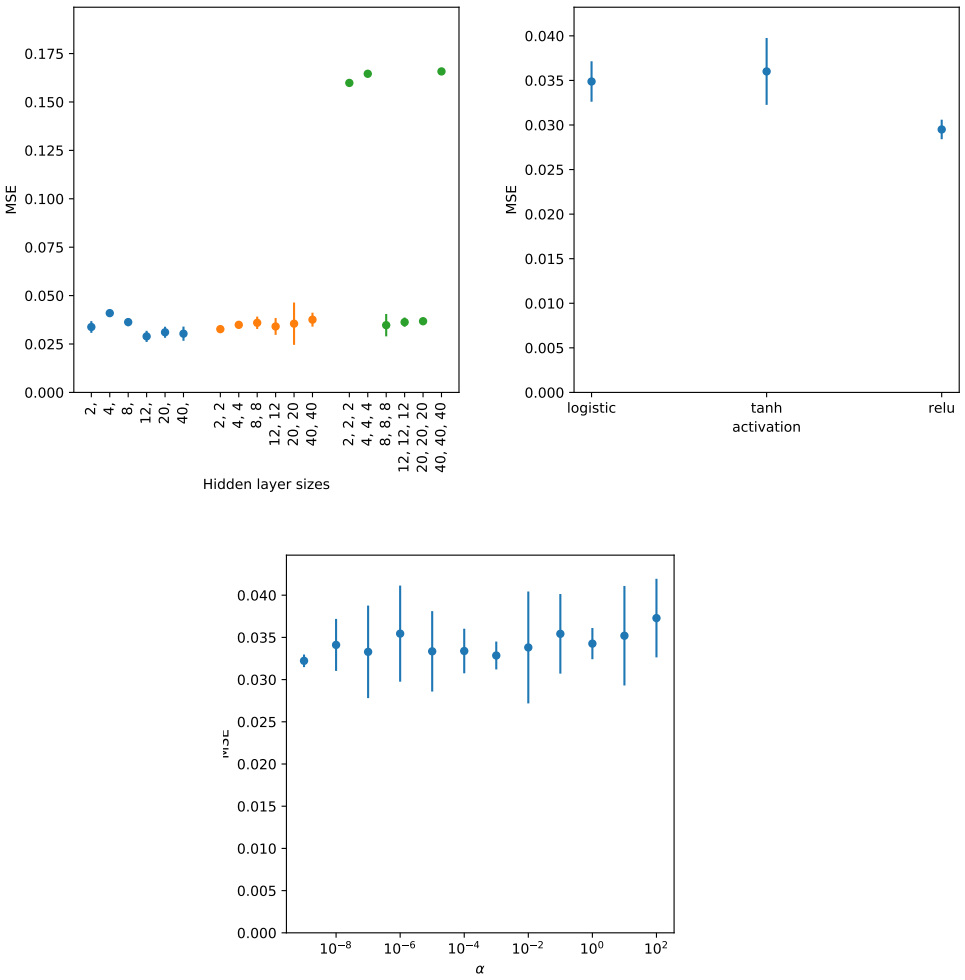

Figure 36: carl performance as a function of the neural network hyperparameters. We show the mean squred error of the estimated log likelihood ratio between two benchmark points using the full feature set as input. Each data point shows the mean of three calculations, the error bars are 95% confidence intervals.

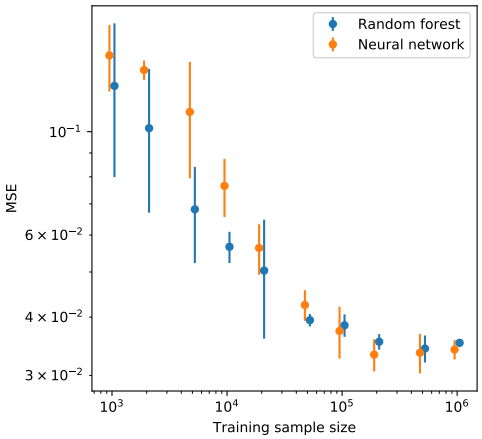

Figure 37: carl performance as a function of the training sample size. We show the mean squred error of the estimated log likelihood ratio between two benchmark points using the full feature set as input. Each data point shows the mean of three calculations, the error bars are 95% confidence intervals.

	Parameter	Range	Random	Bayes 1	Bayes 2	Benchmark
RF	`n_estimators`	50 . . . 200	200			200
	`max_features`	1 . . . 42	42			42
	`max_depth`	1 . . . 20, ∞	10			10
	`min_samples_split`	0 . . . 1	0.001			0.001
	`min_samples_leaf`	0 . . . 0.5	0			0
NN	number of hidden layers	1 . . . 5	2	2	2	2
	neurons at each layer	2 . . . 100 per layer	(4, 12)	(5, 5)	(100, 100)	(4, 12)
	activation function	tanh, ReLU, logistic	logistic	logistic	logistic	logistic
	α	0 . . . 100	0	10^{-8}	0	0
	initial learning rate	0.0001 . . . 0.01		0.00022	0.00064	0.001

Table 10: Hyperparameter scan on the classification problem between $\boldsymbol{\theta}_0$ and $\boldsymbol{\theta}_1$ using the full plus derived feature set as input. For each parameter of the random forrest (RF) and the neural network (NN) we show the considered range and the best settings as determined by a randomized search CV, a Bayesian optimization procedure geared towards exploitation, and a Bayesian optimization procedure geared towards exploration. The final column shows our baseline parameters for the next step of the optimization procedure.

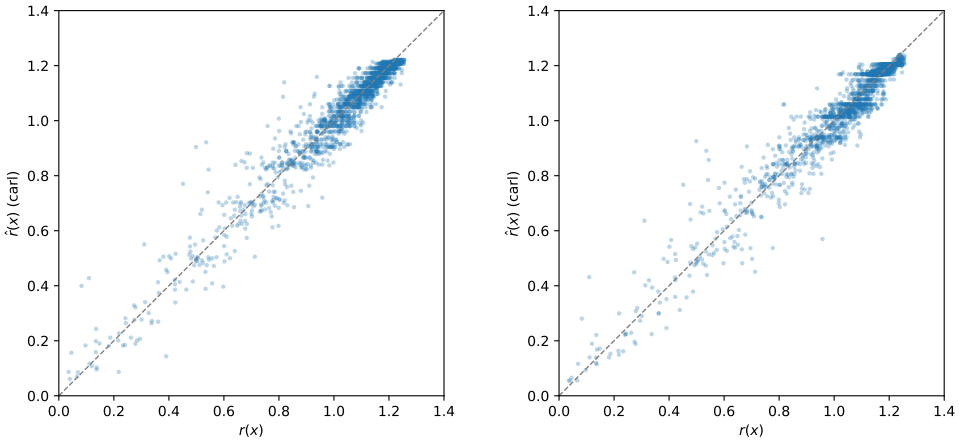

Figure 38: Likelihood ratio estimation with the tuned classifiers (left: random forest, right: neural network) using the full plus derived feature set as input. We show a scatter plot between the true $r(\mathbf{x})$ and the estimate $\hat{r}(\mathbf{x})$.

.

Figure 43 shows how the performance of `carl` depends on the calibration method and its settings. The best results come from the simple histogram calibration (with equidistant bin sizes) or an isotonic calibration.

In Figure 44 and Figure 45 we illuminate the relation between raw and calibrated $s(\mathbf{x})$ for the default histogram-based calibration with fixed bin widths. Figure 46 does the same for isotonic calibration. Both calibration procedures work as expected.

Figures 47 to 50 show the same plots for the histogram calibration procedure with variable bin width. Different bin boundaries for the nominator and denominator histograms lead to problems (see the weird behaviour at $s \gtrsim 0.6$). But even with a common binning, the variable-width calibration setup performs worse than fixed-width histograms.

Comparison

We summarize our different attempts to estimate $r(\mathbf{x})$ for the high-dimensional $\mathbf{x}$ in Figure 51. No setup performs significantly better than a random forest trained on the medium set of 15 production observables.

B.1.2. Likelihood contours

After optimizing our classifier setup on the distinction between two distinct hypotheses, and validating the results on another two hypotheses, we now turn towards the more relevant problem

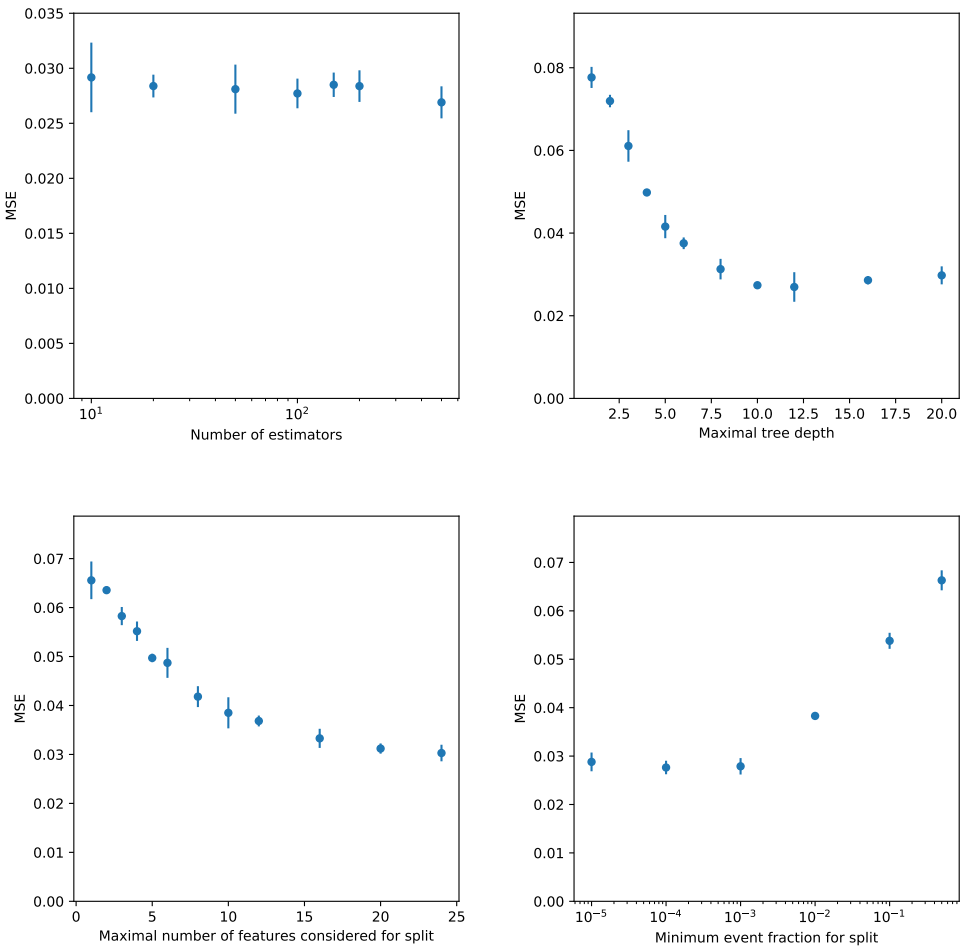

Figure 39: carl performance as a function of the random forest hyperparameters. We show the mean squred error of the estimated log likelihood ratio between two benchmark points using the full plus derived feature set as input. Each data point shows the mean of three calculations, the error bars are 95% confidence intervals.

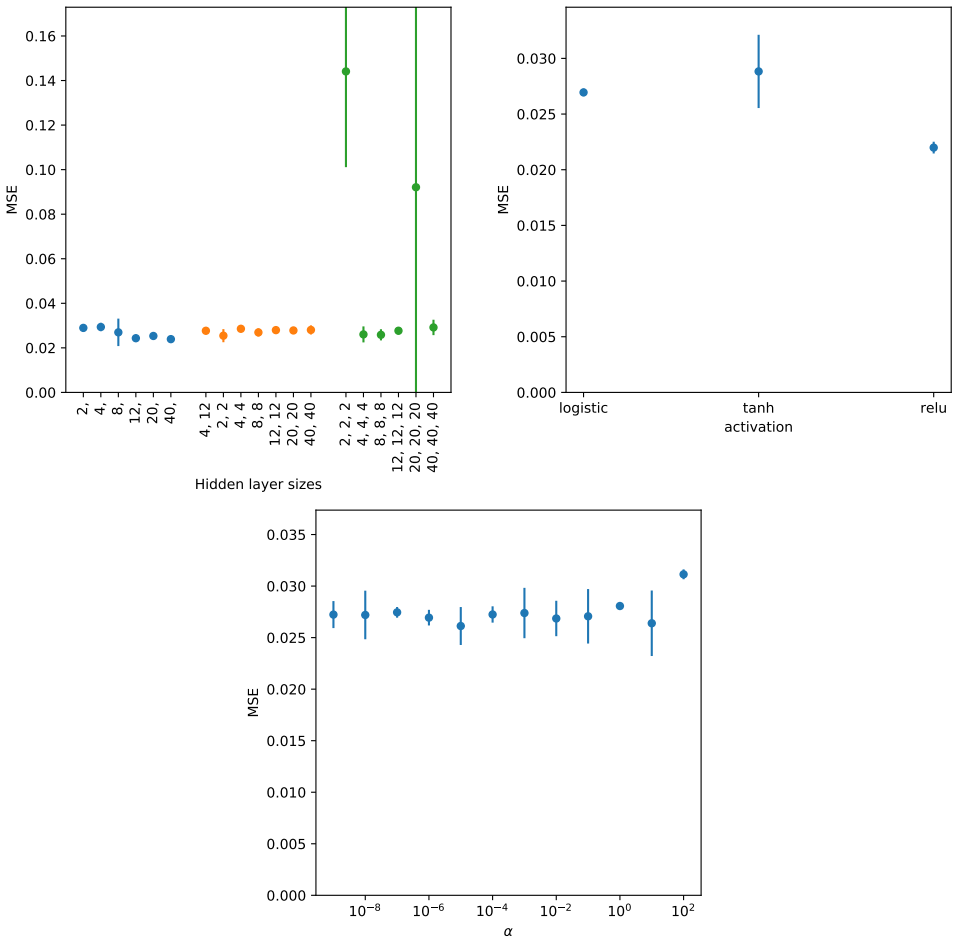

Figure 40: carl performance as a function of the neural network hyperparameters. We show the mean squred error of the estimated log likelihood ratio between two benchmark points using the full plus derived feature set as input. Each data point shows the mean of three calculations, the error bars are 95% confidence intervals.

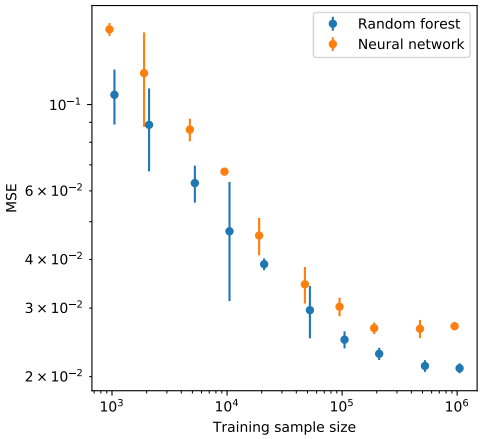

Figure 41: carl performance as a function of the training sample size. We show the mean squred error of the estimated log likelihood ratio between two benchmark points using the full plus derived feature set as input. Each data point shows the mean of three calculations, the error bars are 95% confidence intervals.

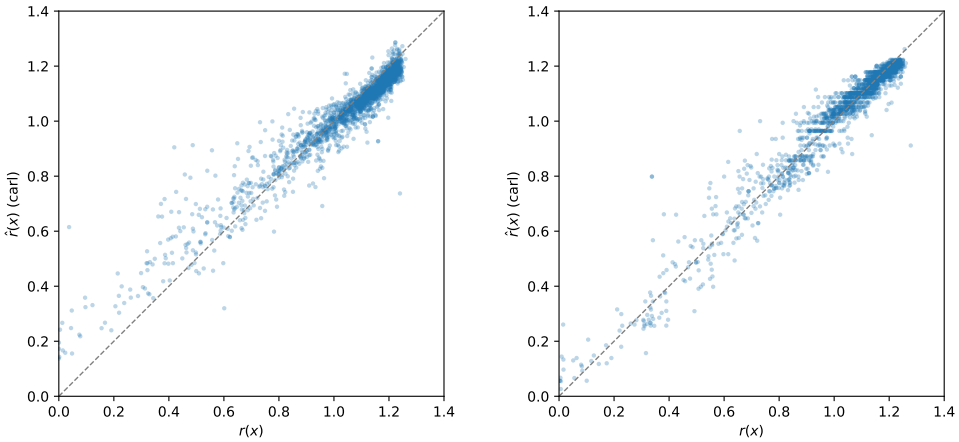

Figure 42: Likelihood ratio estimation with uncalibrated (left) vs calibrated (right) random forests. We show a scatter plot between the true $r(\mathbf{x})$ and the estimate $\hat{r}(\mathbf{x})$.

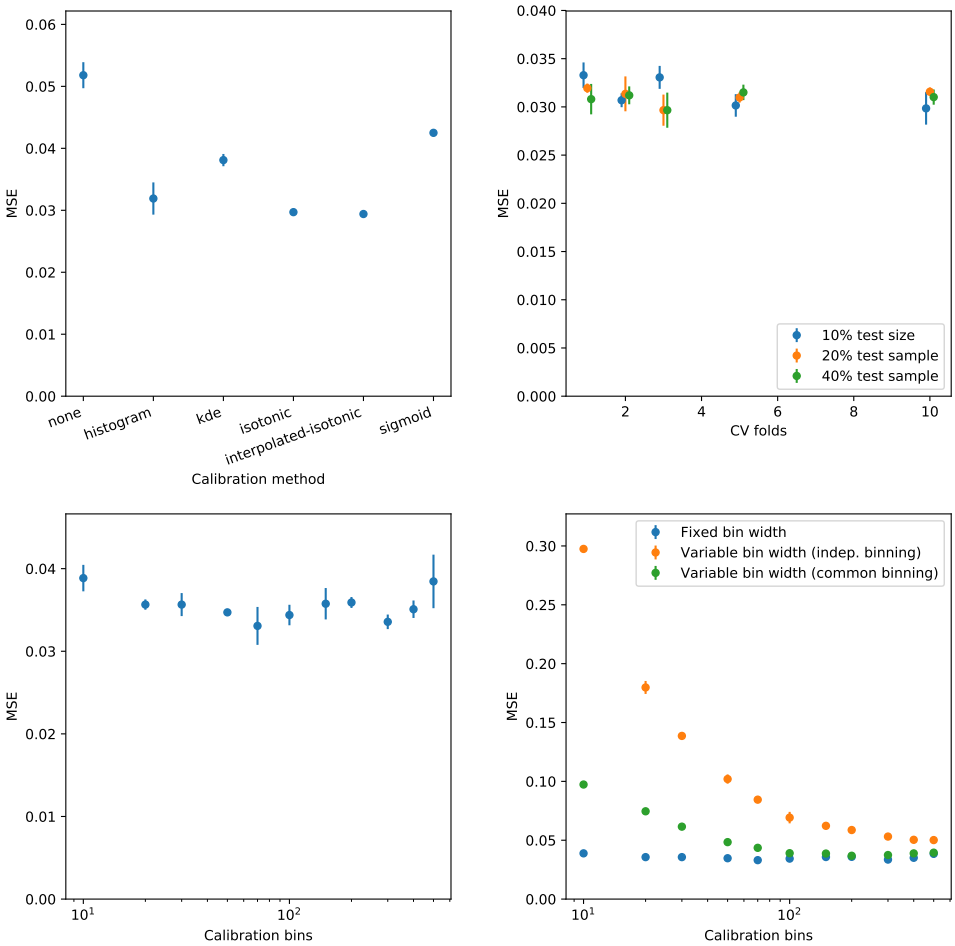

Figure 43: Performance of `carl` as a function of the calibration method and its parameters. Top left: different calibration algorithms. Top right: different cross-validation setups for the histogram validation. Bottom left: different number of bins for the histogram calibration. Bottom right: Same, also showing the results for variable bin widths. We show the mean squred error of the estimated log likelihood ratio between two benchmark points using the medium feature set as input. Each data point shows the mean of three calculations, the error bars are 95% confidence intervals.

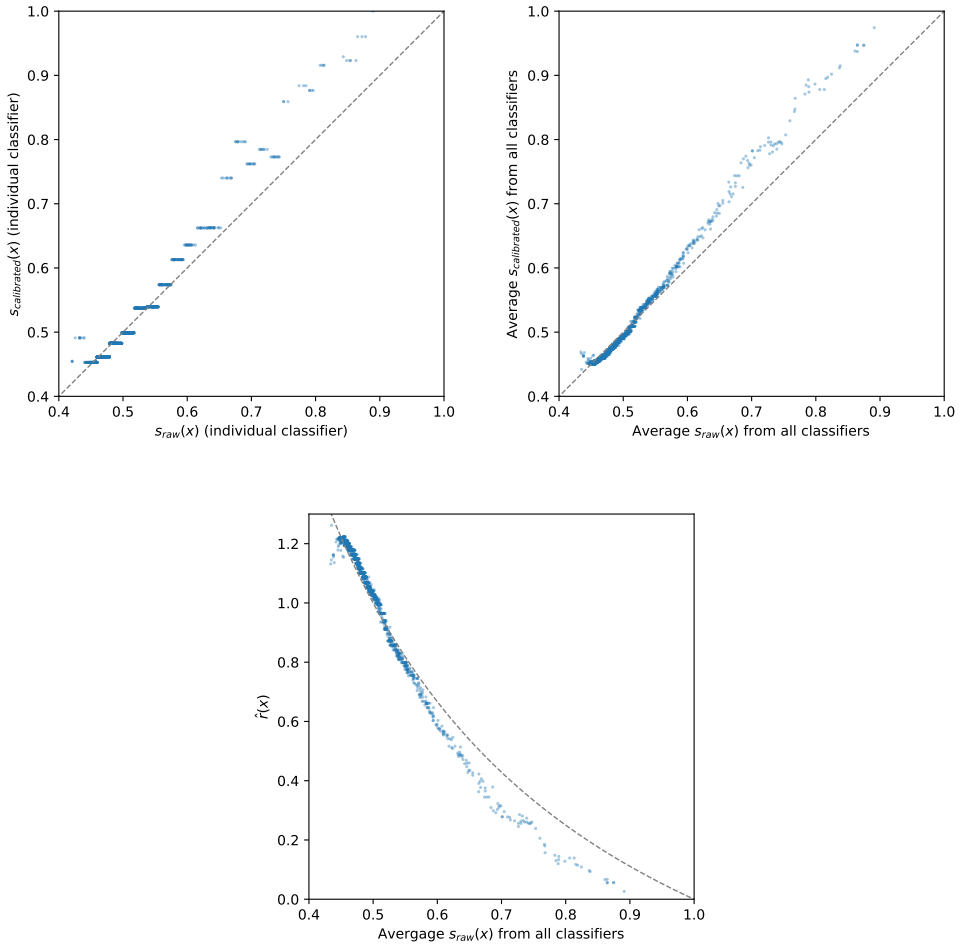

Figure 44: Details of the histogram calibration procedure. Top left: correlation between raw $s(\mathbf{x})$ from one random forest and the corresponding calibrated values. Top right: correlation between average raw $s(\mathbf{x})$ from all random forests in the cross-validator and the average calibrated $s(\mathbf{x})$. Bottom: correlation between average raw $s(\mathbf{x})$ from all random forests in the cross-validator and the final output $\hat{r}(\mathbf{x})$.

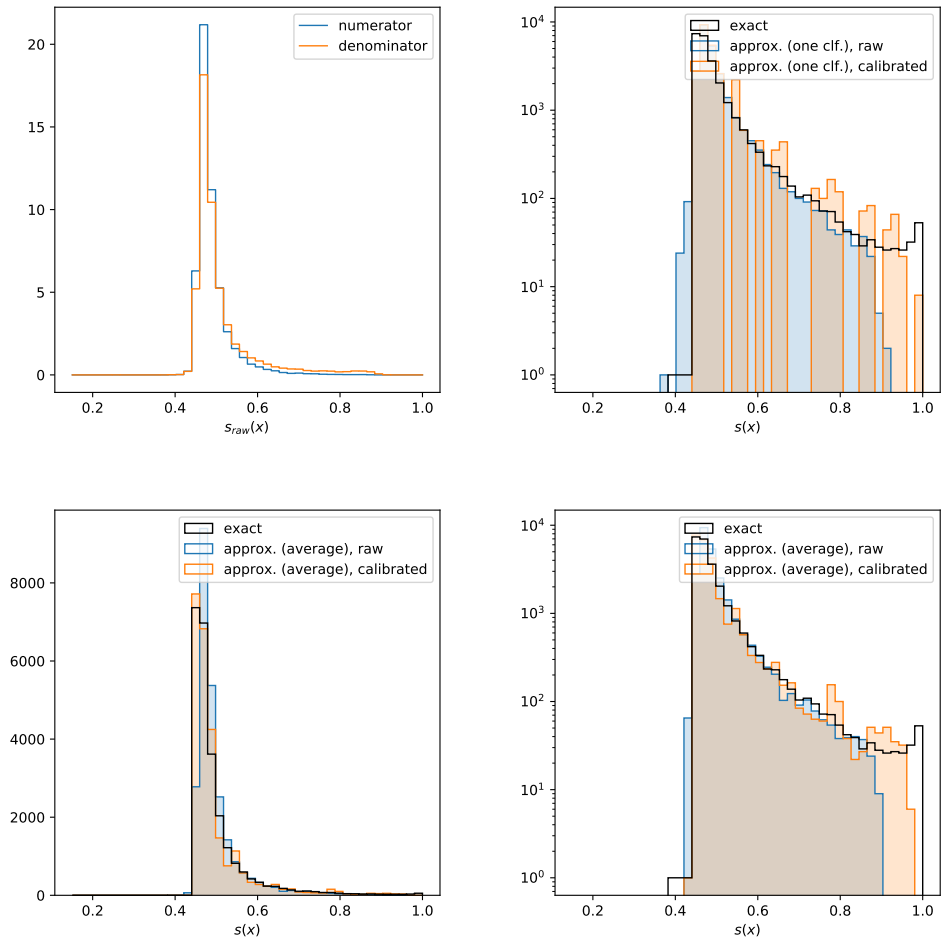

Figure 45: Calibration histograms. Top left: actual calibration histogram for one random forest. Top right: distribution of the raw and calibrated values of $s(\mathbf{x})$ for one random forest. Bottom left: distribution of the average raw and average calibrated values of $s(\mathbf{x})$ for the ensemble of all random forests. Bottom right: same, but with a logarithmic y axis.

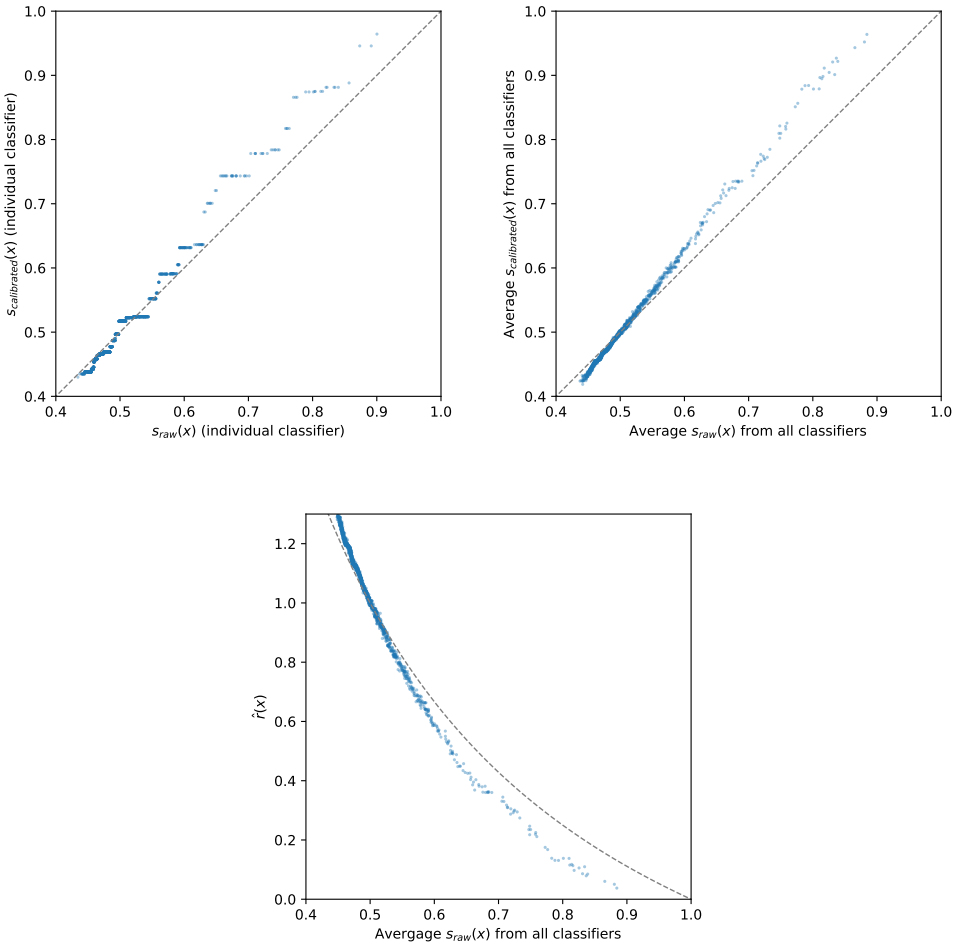

Figure 46: Details of the isotonic calibration procedure. Top left: correlation between raw $s(\mathbf{x})$ from one random forest and the corresponding calibrated values. Top right: correlation between average raw $s(\mathbf{x})$ from all random forests in the cross-validator and the average calibrated $s(\mathbf{x})$. Bottom: correlation between average raw $s(\mathbf{x})$ from all random forests in the cross-validator and the final output $\hat{r}(\mathbf{x})$.

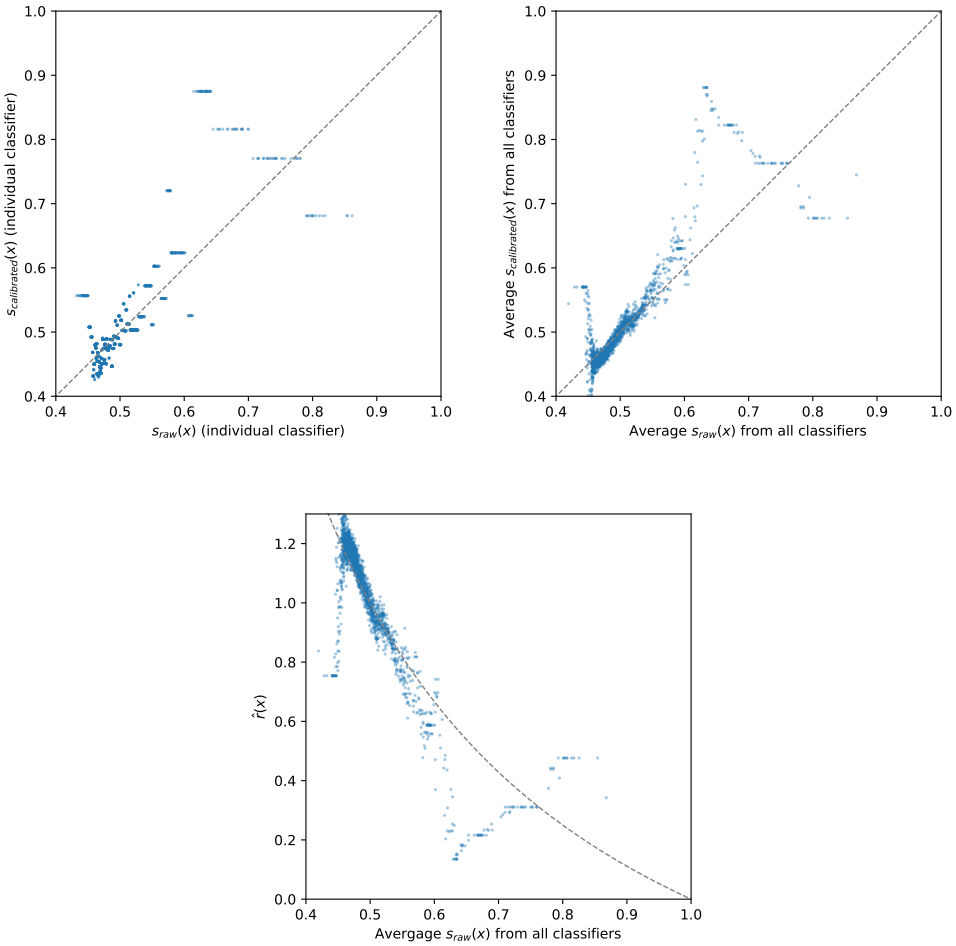

Figure 47: Details of the histogram calibration procedure with variable bin width, with different binnings for nominator and denominator. Top left: correlation between raw $s(\mathbf{x})$ from one random forest and the corresponding calibrated values. Top right: correlation between average raw $s(\mathbf{x})$ from all random forests in the cross-validator and the average calibrated $s(\mathbf{x})$. Bottom: correlation between average raw $s(\mathbf{x})$ from all random forests in the cross-validator and the final output $\hat{r}(\mathbf{x})$.

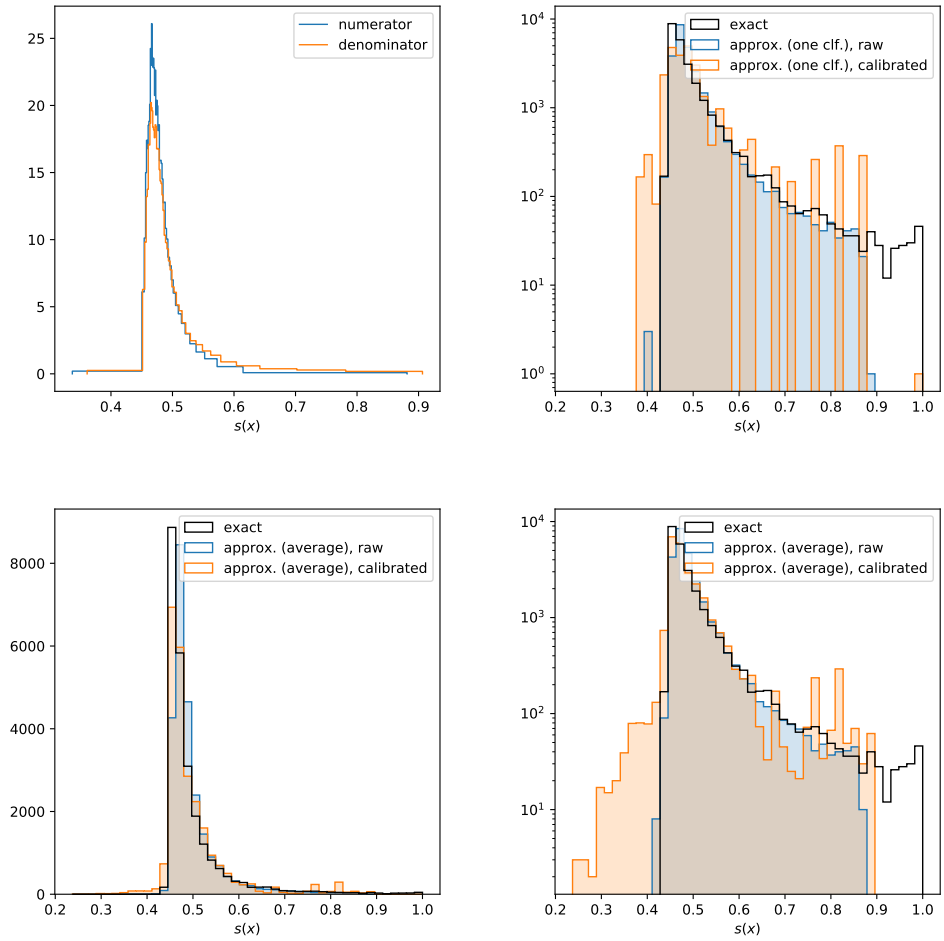

Figure 48: Calibration histograms with variable bin width, with different binnings for nominator and denominator. Top left: actual calibration histogram for one random forest. Top right: distribution of the raw and calibrated values of $s(\mathbf{x})$ for one random forest. Bottom left: distribution of the average raw and average calibrated values of $s(\mathbf{x})$ for the ensemble of all random forests. Bottom right: same, but with a logarithmic y axis.

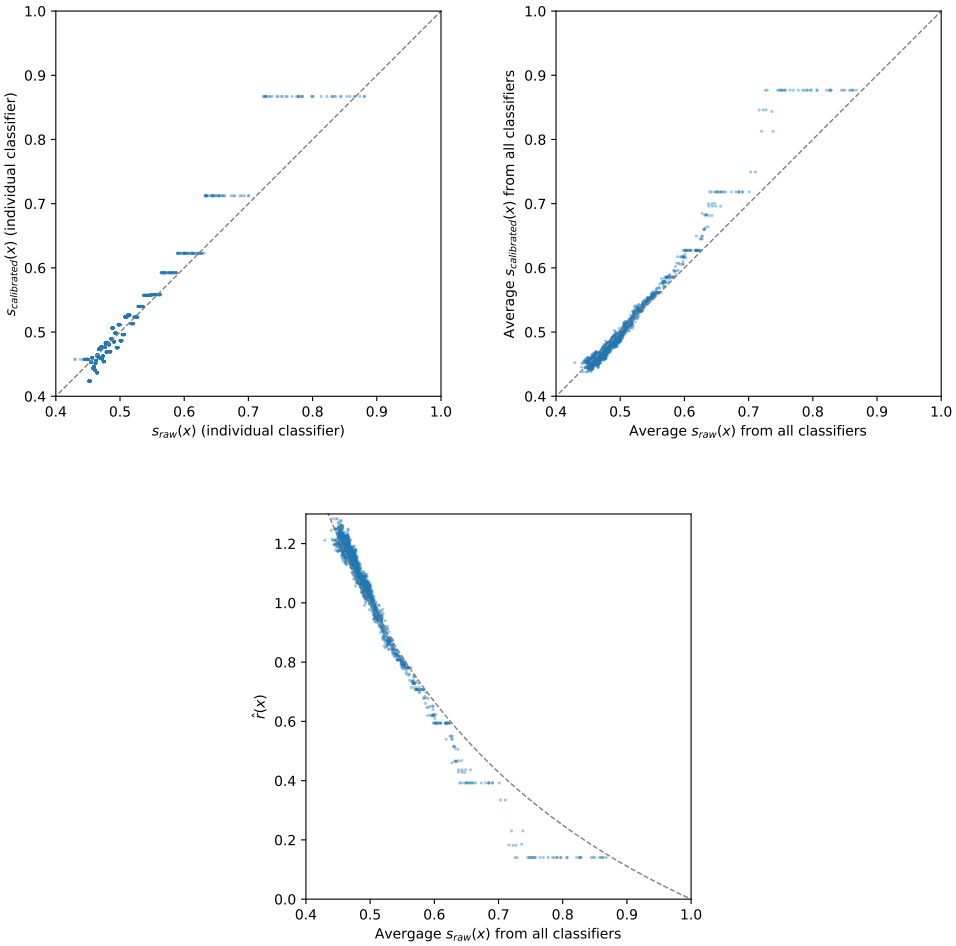

Figure 49: Details of the histogram calibration procedure with variable bin width, with common binning for nominator and denominator. Top left: correlation between raw $s(\mathbf{x})$ from one random forest and the corresponding calibrated values. Top right: correlation between average raw $s(\mathbf{x})$ from all random forests in the cross-validator and the average calibrated $s(\mathbf{x})$. Bottom: correlation between average raw $s(\mathbf{x})$ from all random forests in the cross-validator and the final output $\hat{r}(\mathbf{x})$.

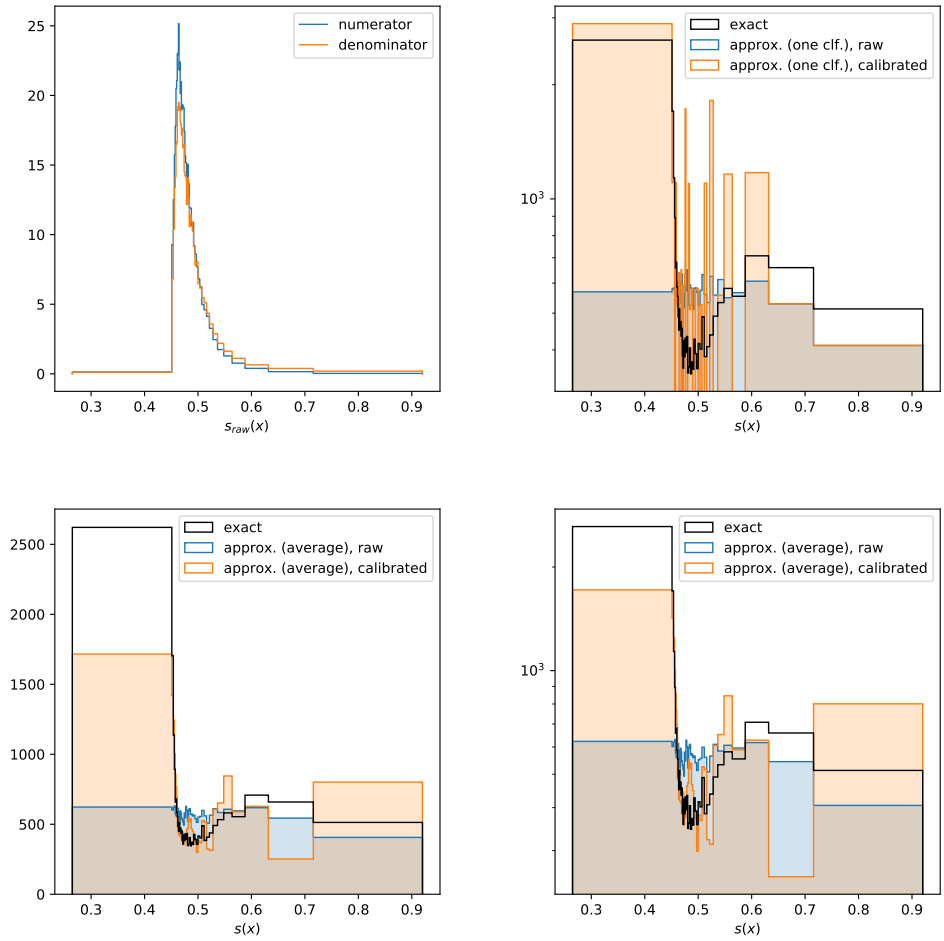

Figure 50: Calibration histograms with variable bin width, with common binning for nominator and denominator. Top left: actual calibration histogram for one random forest. Top right: distribution of the raw and calibrated values of $s(\mathbf{x})$ for one random forest. Bottom left: distribution of the average raw and average calibrated values of $s(\mathbf{x})$ for the ensemble of all random forests. Bottom right: same, but with a logarithmic y axis.

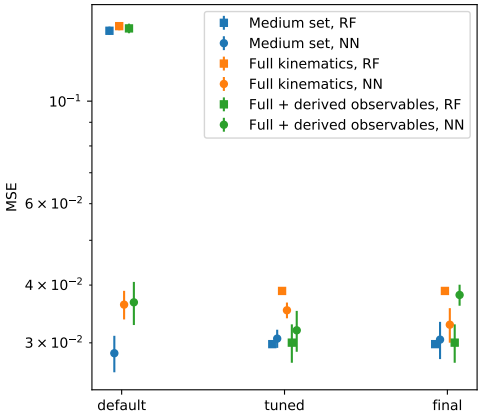

Figure 51: carl performance for `sklearn` default hyperparameters, the best parameters according to the initial hyperparameter scan, and the final parameters as defined in the text, using three different feature sets as inputs. We show the mean squred error of the estimated log likelihood ratio between two benchmark points based on the fully differential feature space. Each data point shows the mean of three calculations, the error bars are 95% confidence intervals.

of estimating $\boldsymbol{\theta}$ based on some measurements. We generate 25 000 toy events sampled from the probability distribution for the SM,

$$\boldsymbol{\theta}_{\text{observed}} = \begin{pmatrix} 0 \\ 0 \end{pmatrix} . \tag{20}$$

For 100 points randomly sampled in $\boldsymbol{\theta} \in [-0.9, 0.9]^2$, we calculate the true expected likelihood ratio to

$$\boldsymbol{\theta}_1 = \begin{pmatrix} -0.23 \\ 0.30 \end{pmatrix} \tag{21}$$

as well as the corresponding `carl` estimate. Finally, we interpolate between these points with a Gaussian Process with Matérn kernel with $\nu = 0.5$.

Based on the tuning described above, we train `carl` on the medium set of 15 production-side observables. We use a random forest with 100 estimators, a maximal tree depth of 10, which considers 6 randomly chosen features for each splitting. The training samples consist of 200 000 unweighted samples each, and the `carl` default for the fixed-size binning of the calibration histograms is used.

The results are shown in Figure 52. There are visible differences between exact and approximate likelihood, but overall the agreement is still pretty good.

We have also checked that a deep and wide neural network does not lead to a better performance in this inference problem.

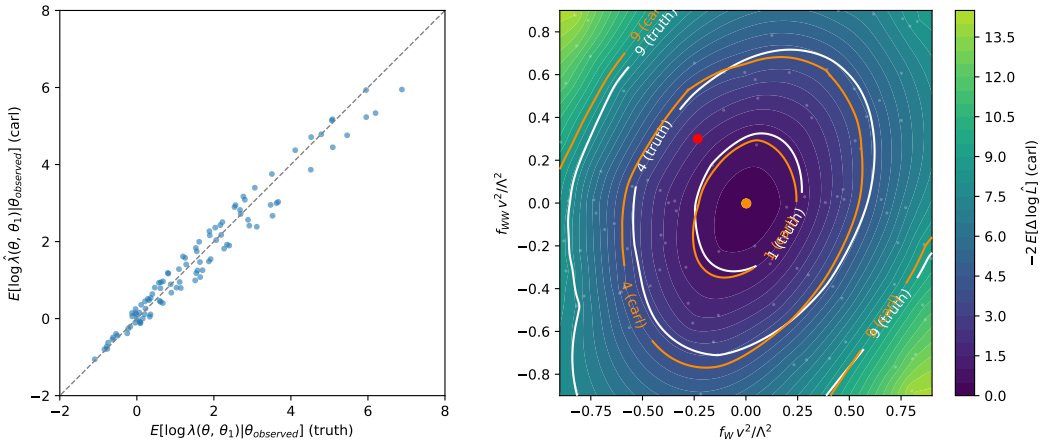

Figure 52: Inference from truth likelihood ratio and `carl`'s estimate for the fully differential case. Left: scatter plot showing the difference between the exact expected likelihood ratio for 100 randomly sampled points and $\boldsymbol{\theta}_1$ and `carl`'s estimate. Right: true (white) and approximate (orange) likelihood contours, using a Gaussian Process for interpolation. The white and orange dots show the exact and approximate maximum-likelihood estimators. The green and red dots show $\boldsymbol{\theta}_{\text{observed}}$ and $\boldsymbol{\theta}_1$, respectively. Finally, the small grey dots show the sampled parameter points at which the likelihood ratio was evaluated.

B.2. Regression

Next, we check how this performance of calibrated classifiers compares to regressor directly trained on the ratio $r(\mathbf{x})$ — which is possible on our truth-level sample, but not in a likelihood-free setup. Note that in this setup the regressors are not calibrated. Since the calibrated classifiers work flawlessly for the simple 2D case, we now stick to the fully differential case.

B.2.1. Benchmark hypothesis test

Again, we first tune the settings on the likelihood ratio between the two parameter points

$$\boldsymbol{\theta}_0 = \begin{pmatrix} -0.2 \\ -0.2 \end{pmatrix}, \quad \boldsymbol{\theta}_1 = \begin{pmatrix} 0.2 \\ 0.2 \end{pmatrix} \tag{22}$$

For now, we only use neural networks (brief experiments with random forests and Gaussian processes lead to different computational problems and errors).

What to learn

In this regression setup, one has to be careful which quantity should be learned by the regressors, or with the definition of the loss function. The naive choice of minimizing the squared error of the likelihood ratio $r(\mathbf{x}) = p(\mathbf{x}|\boldsymbol{\theta}_0)/p(\mathbf{x}|\boldsymbol{\theta}_1)$ does not work very well. First, this loss function changes significantly under the exchange $\boldsymbol{\theta}_0 \leftrightarrow \boldsymbol{\theta}_1$. Second, individual phase-space points with $p(\mathbf{x}|\boldsymbol{\theta}_1) \ll 1$ will have a very large weight and dominate the loss function, potentially decreasing the performance of the regressor for the majority of the phase-space points with $r \approx 1$. Instead, we find that it is advantageous to optimize on the squared error of $\log r(\mathbf{x})$. This quantity is invariant under $\boldsymbol{\theta}_0 \leftrightarrow \boldsymbol{\theta}_1$, and the squared-error loss function is not dominated by few extreme phase-space points.

We demonstrate this Figure 53. Training $r(\mathbf{x})$ directly can work for the right phase-space points (top left panel), but can dramatically fail for other ratios (top right). Training $\log r(\mathbf{x})$ does not depend on the choice of numerator and works reliably in all analysed ratios (bottom panels).

Estimators for the medium feature set

For the regression setup, we simply tune the hyperparameters with a Bayesian optimization procedure on the regression problem to estimate the likelihood ratio $r(\mathbf{x})$ between $p(\mathbf{x}|\boldsymbol{\theta}_0)$ and $p(\mathbf{x}|\boldsymbol{\theta}_1)$. Using `skopt.BayesSearchCV`, we optimize on the mean squared error. We give the optimal parameters in Table 12. Unlike for the calibrated classifiers, the neural networks are now significantly more complex.

In Figure 54 we show how well `carl` can estimate the true likelihood ratio $r(\mathbf{x})$ with the tuned regressor given in the right column of Table 11. We find impressive agreement.

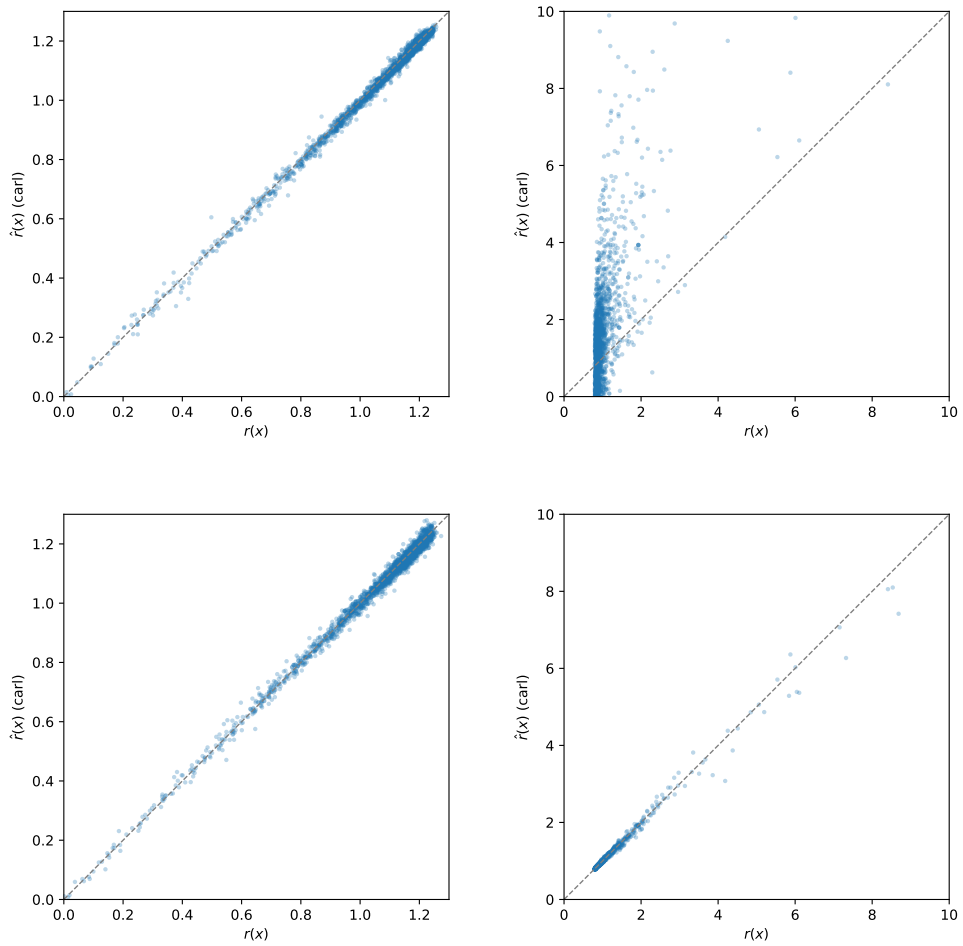

Figure 53: Likelihood ratio estimation with regression, learning $r(\mathbf{x}) = p(\mathbf{x}|\boldsymbol{\theta}_0)/p(\mathbf{x}|\boldsymbol{\theta}_1)$ (top left), $r'(\mathbf{x}) = 1/r(\mathbf{x})$ (top right), $\log r(\mathbf{x})$ (bottom left), and $\log r'(\mathbf{x})$ (bottom right). In each case we use the medium feature set as input and train a neural network regressor with four hidden layers of 100 neurons each. We show a scatter plot between the true $r(\mathbf{x})$ and the estimate $\hat{r}(\mathbf{x})$

.

	Parameter	Range	Bayes 1	Bayes 2	Benchmark
NN	number of hidden layers	1 . . . 5	3	5	3
	neurons in last hidden layer	5 . . . 100	5	20	5
	neurons in other hidden layers	5 . . . 100	100	50	100
	activation function	tanh, ReLU, logistic	tanh	ReLU	tanh
	α	0 . . . 100	0	0.02	0
	initial learning rate	0.0001 . . . 0.01	0.0002	0.0003	0.0002

Table 11: Hyperparameter scan on the regression problem of $r(\mathbf{x})$ between $\boldsymbol{\theta}_0$ and $\boldsymbol{\theta}_1$ using the medium feature set as input. For each parameter of the random forrest (RF) and the neural network (NN) we show the considered range, the best settings as determined by a Bayesian optimization procedure geared towards exploitation, and the best settings determined by a Bayesian optimization procedure geared towards exploration. The final column shows our baseline parameters for the next step of the optimization procedure.

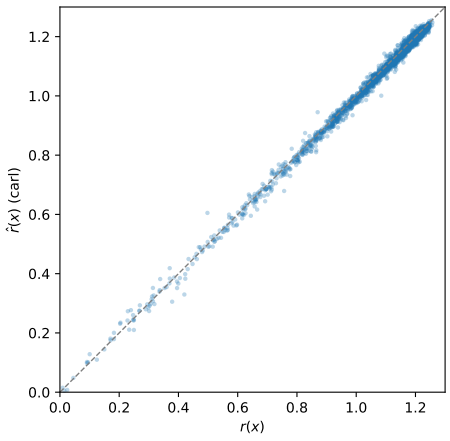

Figure 54: Likelihood ratio estimation with regression using the medium feature set as input. We show a scatter plot between the true $r(\mathbf{x})$ and the estimate $\hat{r}(\mathbf{x})$

.

	Parameter	Range	Bayes 1	Bayes 2	Benchmark
NN	number of hidden layers	1 . . . 5	5	5	5
	neurons in last hidden layer	5 . . . 100	10	10	10
	neurons in other hidden layers	5 . . . 100	50	50	50
	activation function	tanh, ReLU, logistic	tanh	tanh	tanh
	α	0 . . . 100	0	0	0
	initial learning rate	0.0001 . . . 0.01	0.0002	0.0003	0.0002

Table 12: Hyperparameter scan on the regression problem of $r(\mathbf{x})$ between $\boldsymbol{\theta}_0$ and $\boldsymbol{\theta}_1$ using the full feature set as input. For each parameter of the random forrest (RF) and the neural network (NN) we show the considered range and the best settings as determined by a randomized search CV, a Bayesian optimization procedure geared towards exploitation, and a Bayesian optimization procedure geared towards exploration. The final column shows our baseline parameters for the next step of the optimization procedure.

Estimators for the full feature set

Now we use the full kinematics including the Higgs decay patterns, parametrized by the 24 energies, transverse momenta, and angles given in configuration 3 in subsection 2.4.

We first tune hyperparameters with a randomized scan on the classification problem between unweighted event samples drawn from $p(\mathbf{x}|\boldsymbol{\theta}_0)$ and $p(\mathbf{x}|\boldsymbol{\theta}_1)$. Using `sklearn.model_selection.RandomizedSearchCV` we optimize on the ROC AUC. We give the optimal parameters in Table 12. Again, we find a preference for around four hidden layers with 100 neurons each.

In Figure 55 we show how well `carl` can estimate the true likelihood ratio $r(\mathbf{x})$ with the tuned classifiers given in the right column of Table 12.

Estimators for the full plus derived feature set

Finally, we analyse whether the estimator is improved if we include an additional 18 derived quantities, see subsection 2.4. The results of the hyperparameter scan are given in Table 12.

We first tune hyperparameters with a randomized scan on the classification problem between unweighted event samples drawn from $p(\mathbf{x}|\boldsymbol{\theta}_0)$ and $p(\mathbf{x}|\boldsymbol{\theta}_1)$. Using `sklearn.model_selection.RandomizedSearchCV` we optimize on the ROC AUC. We give the optimal parameters in Table 13. Again, we find a preference for around four hidden layers with 100 neurons each.

In Figure 56 we show how well `carl` can estimate the true likelihood ratio $r(\mathbf{x})$ with the tuned classifiers given in the right column of Table 13.

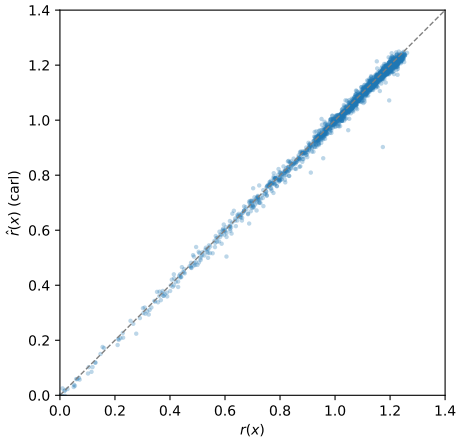

Figure 55: Likelihood ratio estimation with the tuned regressor using the full feature set as input. We show a scatter plot between the true $r(\mathbf{x})$ and the estimate $\hat{r}(\mathbf{x})$

.

	Parameter	Range	Bayes 1	Bayes 2	Benchmark
NN	number of hidden layers	1 . . . 5	3	3	3
	neurons in last hidden layer	5 . . . 100	50	100	50
	neurons in other hidden layers	5 . . . 100	100	100	100
	activation function	tanh, ReLU, logistic	tanh	logistic	tanh
	α	0 . . . 100	0	0.00004	0
	initial learning rate	0.0001 . . . 0.01	0.0003	0.0006	0.0003

Table 13: Hyperparameter scan on the regression problem of $r(\mathbf{x})$ between $\boldsymbol{\theta}_0$ and $\boldsymbol{\theta}_1$ using the full plus derived feature set as input. For each parameter of the random forrest (RF) and the neural network (NN) we show the considered range and the best settings as determined by a randomized search CV, a Bayesian optimization procedure geared towards exploitation, and a Bayesian optimization procedure geared towards exploration. The final column shows our baseline parameters for the next step of the optimization procedure.

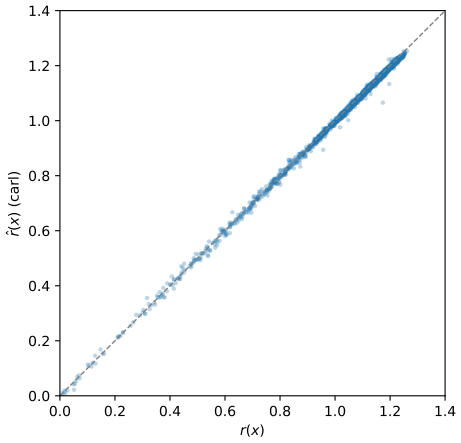

Figure 56: Likelihood ratio estimation with the tuned regressor using the full feature plus derived set as input. We show a scatter plot between the true $r(\mathbf{x})$ and the estimate $\hat{r}(\mathbf{x})$

.

Comparison

We summarize our different attempts to estimate $r(\mathbf{x})$ for the high-dimensional $\mathbf{x}$ with different regressors in Figure 57. The different input sets make a negligible difference, so for the sake of computation time we from now on stick to the simple medium set.

B.2.2. Likelihood contours

After optimizing our regressor setup on the likelihood ratio between two distinct hypotheses, we now turn towards the more relevant problem of estimating $\boldsymbol{\theta}$ based on some measurements. We generate 25 000 toy events sampled from the probability distribution for the SM,

$$\boldsymbol{\theta}_{\text{observed}} = \begin{pmatrix} 0 \\ 0 \end{pmatrix} . \tag{23}$$

For 100 points randomly sampled in $\boldsymbol{\theta} \in [-0.9, 0.9]^2$, we calculate the true expected likelihood ratio to

$$\boldsymbol{\theta}_1 = \begin{pmatrix} -0.23 \\ 0.30 \end{pmatrix} \tag{24}$$

as well as the corresponding `carl` estimate. Finally, we interpolate between these points with a Gaussian Process with Matérn kernel with $\nu = 0.5$.

We train `carl` on the medium feature set. We use a neural network regressor with two hidden layers of 100 neurons each and another hidden layer with 5 neurons, the tanh activation function,

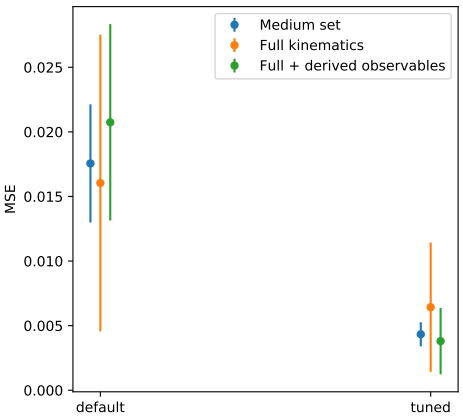

Figure 57: Performance of `carl` with regression for `sklearn` default hyperparameters, the best parameters according to the initial hyperparameter scan, and the final parameters as defined in the text, using three different feature sets as inputs. We show the mean squred error of the estimated log likelihood ratio between two benchmark points based on the fully differential feature space. Each data point shows the mean of three calculations, the error bars are 95% confidence intervals.

$\alpha = 0$, and an initial learning rate of 0.0002. The training samples consist of 200 000 unweighted samples each.

The results are shown in Figure 58. We see that `carl` with regression captures the likelihood function qualitatively well and slightly better than the calibrated classifiers.

B.3. ROC curves

In Figure 59 we show some ROC curves corresponding to the classification problem of Appendix B. We compare the TPRs and FPRs of six different scores:

- the raw output $\hat{s}(\mathbf{x})$ of one random forest, using the medium feature set as input and the tuned settings described in Appendix B;
- the same output, calibrated with the histogram method;
- the average of the outputs of five such calibrated random forests;
- the corresponding `carl` estimate $\hat{r}(\mathbf{x})$;
- the estimate $\hat{r}(\mathbf{x})$ based on a regressor as described in subsubsection B.2.1; and
- the true $r(\mathbf{x})$, which defines the optimal classifier according to the Neyman-Pearson lemma.

The resulting ROC curves are pretty much indistinguishable. This shows that the classification problem between two hypotheses is solved nearly optimally by our learning setup, and that this is much easier than the regression / density estimation problem of learning $r(\mathbf{x})$.

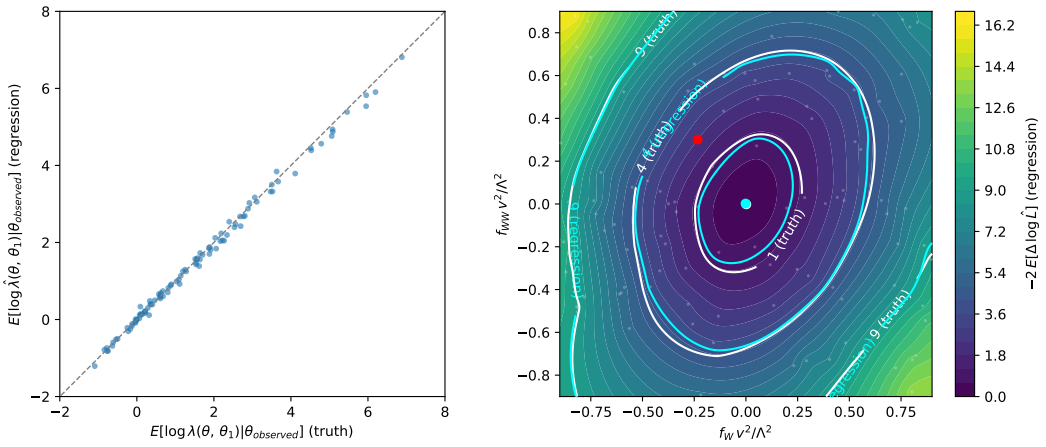

Figure 58: Inference from truth likelihood ratio and `carl`'s estimate for the fully differential case with regression. Left: scatter plot showing the difference between the exact expected likelihood ratio for 100 randomly sampled points and $\boldsymbol{\theta}_1$ and `carl`'s estimate. Right: true (white) and approximate (cyan) likelihood contours, using a Gaussian Process for interpolation. The white and cyan dots show the exact and approximate maximum-likelihood estimators. The green and red dots show $\boldsymbol{\theta}_{\text{observed}}$ and $\boldsymbol{\theta}_1$, respectively. Finally, the small grey dots show the sampled parameter points at which the likelihood ratio was evaluated.

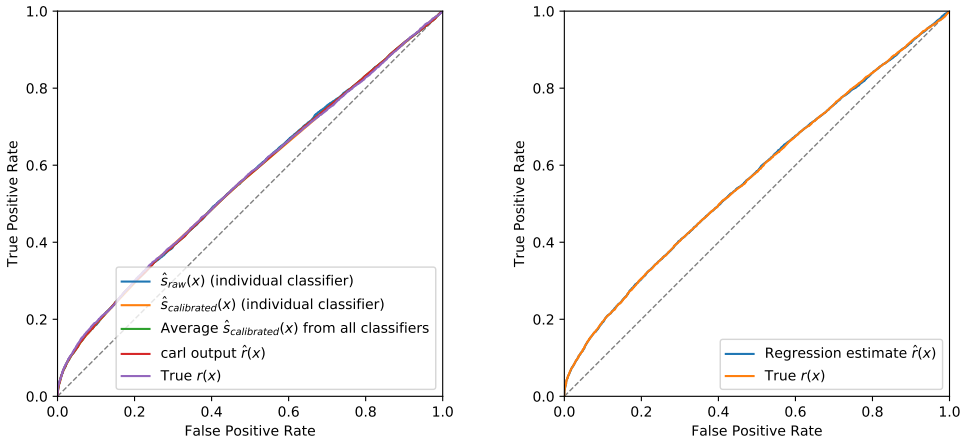

Figure 59: ROC curves for the classification between event samples based on $\boldsymbol{\theta}_0$ and $\boldsymbol{\theta}_1$ as defined in Equation 13. Left: calibrated classifiers (with random forests as described in Appendix B). Right: regression (with a neural network as described in subsubsection B.2.1.

References

[1] K. Cranmer, J. Pavez, and G. Louppe: 'Approximating Likelihood Ratios with Calibrated Discriminative Classifiers' , 2015. arXiv:1506.02169.

[2] G. Louppe, K. Cranmer, and J. Pavez: 'carl: a likelihood-free inference toolbox'. J. Open Source Softw. , 2016.

[3] K. Hagiwara, S. Ishihara, R. Szalapski, and D. Zeppenfeld: 'Low-energy effects of new interactions in the electroweak boson sector'. Phys. Rev. D48, p. 2182, 1993.

[4] J. Brehmer, K. Cranmer, F. Kling, and T. Plehn: 'Better Higgs boson measurements through information geometry'. Phys. Rev. D95 (7), p. 073002, 2017. arXiv:1612.05261.

[5] J. Brehmer: 'New Ideas for Effective Higgs Measurements'. Ph.D. thesis, Heidelberg U., 2017.

[6] J. Alwall, R. Frederix, S. Frixione, et al.: 'The automated computation of tree-level and next-to-leading order differential cross sections, and their matching to parton shower simulations'. JHEP 07, p. 079, 2014. arXiv:1405.0301.

[7] K. Cranmer and T. Plehn: 'Maximum significance at the LHC and Higgs decays to muons'. Eur. Phys. J. C51, p. 415, 2007. arXiv:hep-ph/0605268.

[8] T. Plehn, P. Schichtel, and D. Wiegand: 'Where boosted significances come from'. Phys. Rev. D89 (5), p. 054002, 2014. arXiv:1311.2591.

[9] F. Kling, T. Plehn, and P. Schichtel: 'Maximizing the significance in Higgs boson pair analyses'. Phys. Rev. D95 (3), p. 035026, 2017. arXiv:1607.07441.