A. Likelihood-free Markov chain Monte Carlo samplers

Algorithm 2 Likelihood-free Metropolis-Hastings

```
Inputs:            Initial parameter $\boldsymbol{\theta}_0$
                   Prior $p(\boldsymbol{\theta})$
                   Transition distribution $q(\boldsymbol{\theta})$
                   Trained ratio estimator $\mathbf{d}(\mathbf{x}, \boldsymbol{\theta})$
                   Observation $\mathbf{x}$
Outputs:           Markov chain $\boldsymbol{\theta}_{0:T}$
Hyperparameters:   Steps $T$
1:  $t \leftarrow 0$
2:  $\boldsymbol{\theta}_t \leftarrow \boldsymbol{\theta}_0$
3:  for $t < T$ do
4:      $\boldsymbol{\theta}' \sim q(\boldsymbol{\theta} \mid \boldsymbol{\theta}_t)$
5:      $\lambda \leftarrow (\log \hat{r}(\mathbf{x} \mid \boldsymbol{\theta}') + \log p(\boldsymbol{\theta}')) - (\log \hat{r}(\mathbf{x} \mid \boldsymbol{\theta}_t) + \log p(\boldsymbol{\theta}_t))$
6:      $\rho \leftarrow \min(\exp(\lambda) \frac{q(\boldsymbol{\theta}_t \mid \boldsymbol{\theta}')}{q(\boldsymbol{\theta}' \mid \boldsymbol{\theta}_t)},\ 1)$
7:      $\boldsymbol{\theta}_{t+1} \leftarrow \begin{cases} \boldsymbol{\theta}' & \text{with probability } \rho \\ \boldsymbol{\theta}_t & \text{with probability } 1-\rho \end{cases}$
8:      $t \leftarrow t + 1$
9:  end for
10: return $\boldsymbol{\theta}_{0:T}$
```

Algorithm 3 Likelihood-free Hamiltonian Monte Carlo

```
Inputs:            Initial parameter $\boldsymbol{\theta}_0$
                   Prior $p(\boldsymbol{\theta})$
                   Momentum distribution $q(\mathbf{m})$
                   Trained ratio estimator $\mathbf{d}(\mathbf{x}, \boldsymbol{\theta})$
                   Observation $\mathbf{x}$
Outputs:           Markov chain $\boldsymbol{\theta}_{0:T}$
Hyperparameters:   Steps $T$.
                   Leapfrog-integration steps $l$ and stepsize $\eta$.
1:  $t \leftarrow 0$
2:  $\boldsymbol{\theta}_t \leftarrow \boldsymbol{\theta}_0$
3:  for $t < T$ do
4:      $\mathbf{m}_t \sim q(\mathbf{m})$
5:      $k \leftarrow 0$
6:      $\mathbf{m}_k \leftarrow \mathbf{m}_t$
7:      $\boldsymbol{\theta}_k \leftarrow \boldsymbol{\theta}_t$
8:      for $k < l$ do
9:          $\mathbf{m}_k \leftarrow \mathbf{m}_k + \frac{\eta}{2} \frac{\nabla_{\boldsymbol{\theta}} \hat{r}(\mathbf{x} \mid \boldsymbol{\theta}_k)}{\hat{r}(\mathbf{x} \mid \boldsymbol{\theta}_k)}$
10:         $\boldsymbol{\theta}_k \leftarrow \boldsymbol{\theta}_k + \eta \mathbf{m}_k$
11:         $\mathbf{m}_k \leftarrow \mathbf{m}_k + \frac{\eta}{2} \frac{\nabla_{\boldsymbol{\theta}} \hat{r}(\mathbf{x} \mid \boldsymbol{\theta}_k)}{\hat{r}(\mathbf{x} \mid \boldsymbol{\theta}_k)}$
12:         $k \leftarrow k + 1$
13:     end for
14:     $\lambda \leftarrow (\log \hat{r}(\mathbf{x} \mid \boldsymbol{\theta}_k) + \log p(\boldsymbol{\theta}_k)) - (\log \hat{r}(\mathbf{x} \mid \boldsymbol{\theta}_t) + \log p(\boldsymbol{\theta}_t)) + K(\mathbf{m}_k) - K(\mathbf{m}_t)$
15:     $\rho \leftarrow \min(\exp(\lambda), 1)$
16:     $\boldsymbol{\theta}_{t+1} \leftarrow \begin{cases} \boldsymbol{\theta}_k & \text{with probability } \rho \\ \boldsymbol{\theta}_t & \text{with probability } 1-\rho \end{cases}$
17:     $t \leftarrow t + 1$
18: end for
19: return $\boldsymbol{\theta}_{0:T}$
```

B. Correctness of Algorithm 1

The core of our contribution rests on the proper estimation of the likelihood-to-evidence ratio. In this section, we show that the minimization of the binary cross-entropy (BCE) loss of a classifier tasked to distinguish between dependent input pairs $(\mathbf{x}, \boldsymbol{\theta}) \sim p(\mathbf{x}, \boldsymbol{\theta})$ and independent input pairs $(\mathbf{x}, \boldsymbol{\theta}) \sim p(\mathbf{x})p(\boldsymbol{\theta})$ results in an optimal classifier.

Using calculus of variations and reproducing the structure of Algorithm 1, we define the loss functional

$$\begin{aligned} L[\mathbf{d}(\mathbf{x}, \boldsymbol{\theta})] &= \int d\boldsymbol{\theta} \int d\mathbf{x} \int d\boldsymbol{\theta}' p(\boldsymbol{\theta}) p(\mathbf{x} \mid \boldsymbol{\theta}) p(\boldsymbol{\theta}') \Big[-\log \mathbf{d}(\mathbf{x}, \boldsymbol{\theta}) - \log(1 - (\mathbf{d}(\mathbf{x}, \boldsymbol{\theta}')) \Big] \\ &= \int d\boldsymbol{\theta} \int d\mathbf{x} \underbrace{p(\boldsymbol{\theta}) p(\mathbf{x} \mid \boldsymbol{\theta}) \Big[-\log \mathbf{d}(\mathbf{x}, \boldsymbol{\theta}) \Big] + p(\boldsymbol{\theta}) p(\mathbf{x}) \Big[-\log(1 - \mathbf{d}(\mathbf{x}, \boldsymbol{\theta})) \Big]}_{F(\mathbf{d})} . \end{aligned} \tag{20}$$

This loss functional is minimized for a function $\mathbf{d}^*(\mathbf{x}, \boldsymbol{\theta})$ such that

$$0 = \left. \frac{\delta F}{\delta \mathbf{d}} \right|_{\mathbf{d}^*} = p(\boldsymbol{\theta}) p(\mathbf{x} \mid \boldsymbol{\theta}) \Big[-\frac{1}{\mathbf{d}^*(\mathbf{x}, \boldsymbol{\theta})} \Big] + p(\boldsymbol{\theta}) p(\mathbf{x}) \Big[\frac{1}{1 - \mathbf{d}^*(\mathbf{x}, \boldsymbol{\theta})} \Big]. \tag{21}$$

As long as $p(\boldsymbol{\theta}) > 0$, this is equivalent to

$$p(\mathbf{x} \mid \boldsymbol{\theta}) \frac{1}{\mathbf{d}^*(\mathbf{x}, \boldsymbol{\theta})} = p(\mathbf{x}) \frac{1}{1 - \mathbf{d}^*(\mathbf{x}, \boldsymbol{\theta})}, \tag{22}$$

and finally

$$\mathbf{d}^*(\mathbf{x}, \boldsymbol{\theta}) = \frac{p(\mathbf{x} \mid \boldsymbol{\theta})}{p(\mathbf{x} \mid \boldsymbol{\theta}) + p(\mathbf{x})}. \tag{23}$$

□

C. Recommended strategy for applications

This section discusses several recommended strategies to successfully apply our technique to (scientific) applications. We show several code listings, with a focus on a Pytorch (Paszke et al., 2017) implementation. A reference implementation can be found at https://github.com/montefiore-ai/hypothesis. As shown in Figure 9, we directly output the log ratio before applying the sigmoidal projection to improve numerical stability when sampling from the posterior using MCMC. The output of the decision function $\mathbf{d}(\mathbf{x}, \boldsymbol{\theta})$ is also given. This architecture forms the basis for accurate posterior inference. Figure 10 shows the base ratio estimator implementation. For completeness, the variable name `inputs` relates to the model parameters $\boldsymbol{\theta}$ while `outputs` relates to observations $\mathbf{x}$. This particular naming scheme is chosen to depict their relation with respect to the simulation model.

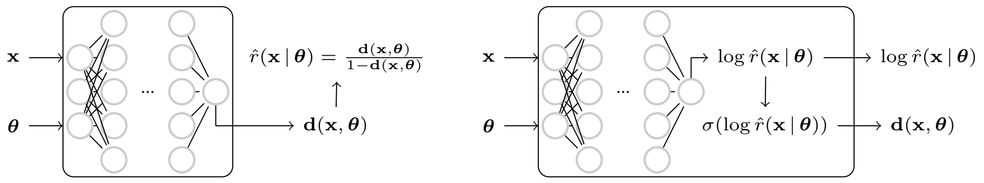

$\mathbf{x}$
$\boldsymbol{\theta}$
$\hat{r}(\mathbf{x} \mid \boldsymbol{\theta}) = \frac{\mathbf{d}(\mathbf{x},\boldsymbol{\theta})}{1-\mathbf{d}(\mathbf{x},\boldsymbol{\theta})}$
$\mathbf{d}(\mathbf{x}, \boldsymbol{\theta})$
$\mathbf{x}$
$\boldsymbol{\theta}$
$\log \hat{r}(\mathbf{x} \mid \boldsymbol{\theta})$
$\sigma(\log \hat{r}(\mathbf{x} \mid \boldsymbol{\theta}))$
$\log \hat{r}(\mathbf{x} \mid \boldsymbol{\theta})$
$\mathbf{d}(\mathbf{x}, \boldsymbol{\theta})$

Figure 9. Two approaches to extract the approximate ratio $\hat{r}(\mathbf{x} \mid \boldsymbol{\theta})$ from a parameterized classifier. *(Left):* The vanilla architecture which is susceptible to numerical errors and loss of information as $\hat{r}(\mathbf{x} \mid \boldsymbol{\theta})$ is computed by transforming the sigmoidal projection σ. This issue arises if the classifier is (almost) able to perfectly discriminate between samples from $p(\mathbf{x} \mid \boldsymbol{\theta})$ and $p(\mathbf{x} \mid \boldsymbol{\theta}_{\text{ref}})$. *(Right):* The modified architecture directly outputs $\log \hat{r}(\mathbf{x} \mid \boldsymbol{\theta})$ before applying the sigmoidal projection.

```
class RatioEstimator(torch.nn.Module):
    def __init__(self):
        super(RatioEstimator, self).__init__()
        self.network = (...)  # Define your neural network

    def forward(self, inputs, outputs):
         # Process the inputs (model parameters)
         (...)

         # Process the outputs (observations)
         (...)

         log_ratio = self.network(inputs, outputs)
         classifier_output = log_ratio.sigmoid()

         return classifier_output, log_ratio
```

Figure 10. Base ratio estimator.

C.1. Dataset generation and training

Algorithm 1 actively samples from the prior and the simulation model in the optimization loop. This is not efficient in practice. A dataset consisting of samples from the joint can be generated offline before training the ratio estimators. Note that no class labels are assigned to specific samples of the dataset. In our training algorithm, the independence of $\mathbf{x}$ and $\boldsymbol{\theta}$ can be guaranteed by sampling two batches from the dataset, and simply switch the $\boldsymbol{\theta}$ tensors in the computation of each individual loss. Alternatively, the mini-batch containing $\boldsymbol{\theta}$ or $\mathbf{x}$ can be randomly shuffled. As a result, the implementation of the optimization loop does not depend on the prior. This produces a mathematically equivalent procedure to Algorithm 1. Figure 11 shows a `pytorch` implementation of the proposed optimization loop for an even number of batches.

The optimal discriminator is the one which minimizes the training criterion described in Algorithm 1. Therefore, the accuracy of the approximation can be improved by using techniques such as learning rate scheduling, or by increasing the batch size to reduce the variance of the gradient. From an architectural perspective, we found that the ELU (Clevert et al., 2015) and SELU (Klambauer et al., 2017) activations work well in general. However, RELUs typically required significantly less parameters (weights) to accurately approximate sharp posteriors. We hypothesize that this behavior is attributable to the sparsity induced by RELU activations. We did not perform a study on the required number of simulations to properly approximate $r(\mathbf{x} \mid \boldsymbol{\theta})$. This aspect is left for future work.

```
loader = DataLoader(dataset, batch_size=batch_size)
num_iterations = len(loader) // 2
loader = iter(loader)
ratio_estimator.train()

for batch_index in range(num_iterations):
    # Load the data and move to the device.
    a_inputs, a_outputs = next(loader)
    a_inputs = a_inputs.to(device, non_blocking=True)
    a_outputs = a_outputs.to(device, non_blocking=True)
    b_inputs, b_outputs = next(loader)
    b_inputs = b_inputs.to(device, non_blocking=True)
    b_outputs = b_outputs.to(device, non_blocking=True)

    # Apply a forward pass with the ratio estimator.
    y_dep_a, _ = ratio_estimator(a_inputs, a_outputs)
    y_idep_a, _ = ratio_estimator(a_inputs, b_outputs)
    y_dep_b, _ = ratio_estimator(b_inputs, b_outputs)
    y_idep_b, _ = ratio_estimator(b_inputs, a_outputs)

    # Loss and backward.
    loss_a = criterion(y_dep_a, ones) +
             criterion(y_idep_a, zeros)
    loss_b = criterion(y_dep_b, ones) +
             criterion(y_idep_b, zeros)
    loss = loss_a + loss_b
    optimizer.zero_grad()
    loss.backward()
    optimizer.step()
```

Figure 11. Proposed optimization loop.

C.2. Validation and inference

In general, we recommend to train multiple (if the computational budget allows) ratio estimators. Besides the improvements that ensembling typically brings, the resulting estimators can be used to determine the variance of the approximation. From our empirical evaluations, large variances in the approximation of the likelihood-to-evidence ratio indicate that the capacity of the ratio estimator might be insufficient (see Appendix E). As mentioned in Section 3.2, the accuracy of the approximation can be verified in a more principled way by means of a ROC curve and its AUC. Contrary to the experimental section of the main manuscript, real applications do not have access to the generating parameters $\boldsymbol{\theta}^*$. There are currently two approaches to test the accuracy of the ratio estimator using the ROC diagnostic: (i) test the ratio estimator for all distinct modes of the posterior and (ii) test the ratio estimator for a set of random samples $\boldsymbol{\theta} \sim p(\boldsymbol{\theta})$. While the first approach specifically tests the solution, the latter validates the behavior of the ratio estimator across the prior $p(\boldsymbol{\theta})$. Alternatively, Simulation Based Calibration (Talts et al., 2018) is a frequentist test for a Bayesian computation, but it cannot verify the accuracy of a single approximate posterior. After validating the ratio estimator, MCMC can be used to draw samples from the posterior. If the dimensionality of the problem permits, estimates of the PDF can be obtained directly.

D. Experimental details and additional results

D.1. Overview of hyperparameters and model architectures

Hyperparameter	Tractable problem	Detector calibration	Lensing	Lotka-Volterra	M/G/1
Activation function	SELU	SELU	RELU	RELU	RELU
AMSGRAD	Yes	Yes	Yes	Yes	Yes
Architecture	MLP	MLP	RESNET-18	MLP	MLP
Batch normalization	No	No	Yes	No	No
Batch size	256	256	256	1024	256
Criterion	BCE	BCE	BCE	BCE	BCE
Dropout	No	No	No	No	No
Epochs	250	250	100	1000	1000
Learning rate	0.001	0.0001	0.001	0.00005	0.0001
Learning rate scheduling	No	No	No	Yes	Yes
Optimizer	ADAM	ADAM	ADAM	ADAM	ADAM
Weight decay	0.0	0.0	0.0	0.0	0.0

Table 3. Hyperparameters associated with the training procedure of our ratio estimator.

D.2. Tractable problem

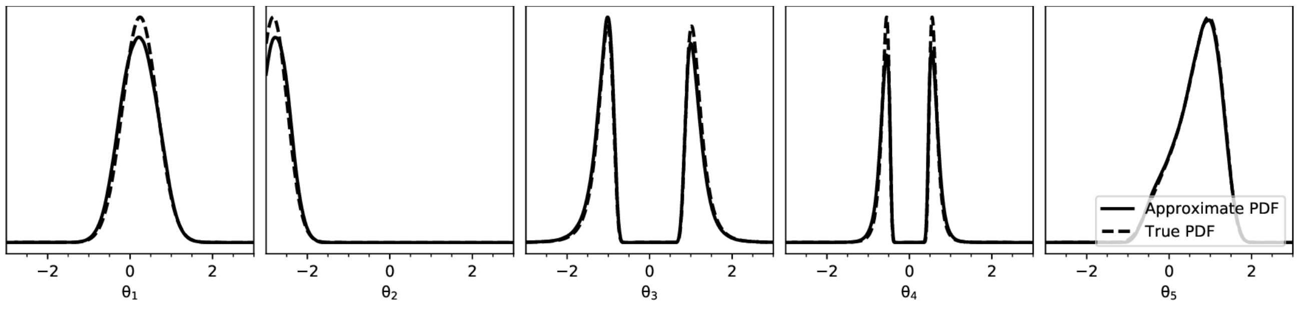
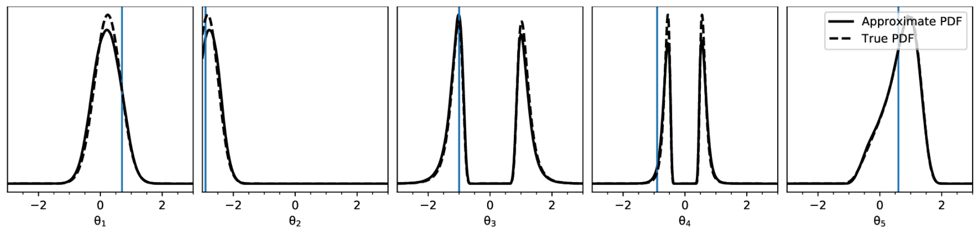


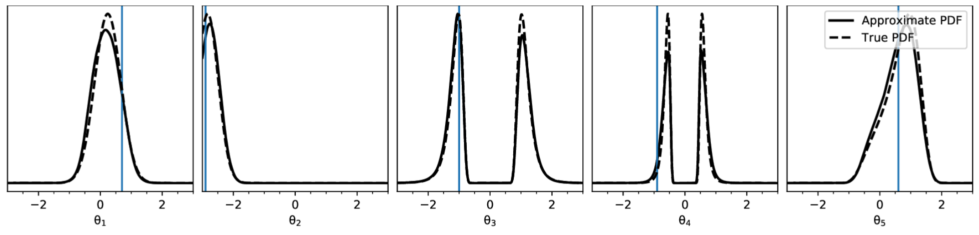
Figure 12. PDF of true posterior marginals $p(\boldsymbol{\theta}_i \mid \mathbf{x})$ and the corresponding approximations extracted from our ratio estimator. This can be computed for arbitrary model parameters $\boldsymbol{\theta}$ and observations $\mathbf{x}$ by simply computing $p(\boldsymbol{\theta})\hat{r}(\mathbf{x} \mid \boldsymbol{\theta})$.

D.2.1. Regularization and posterior approximation

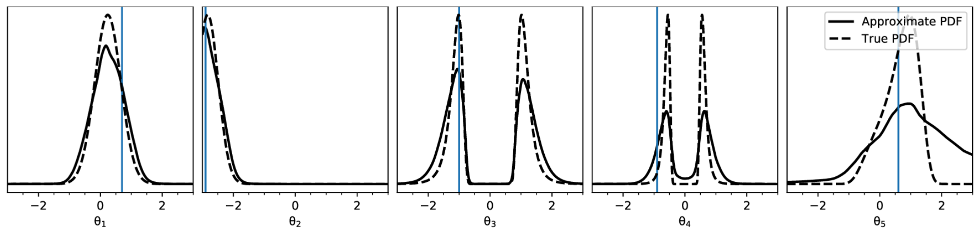
This section studies the effects of regularization on the posterior approximation. We empirically find that the degree of regularization is proportional to an increase in variance of the approximation with respect to the true posterior. This translates into a proportionally larger (test) loss. Similar behavior can be observed in ratio estimators with insufficient capacity (Appendix E). Figure 13 demonstrates the effect of regularization on the approximate posterior with respect to the true posterior. The training and test losses are shown in Figure 14.



(a) Weight decay = 0.0

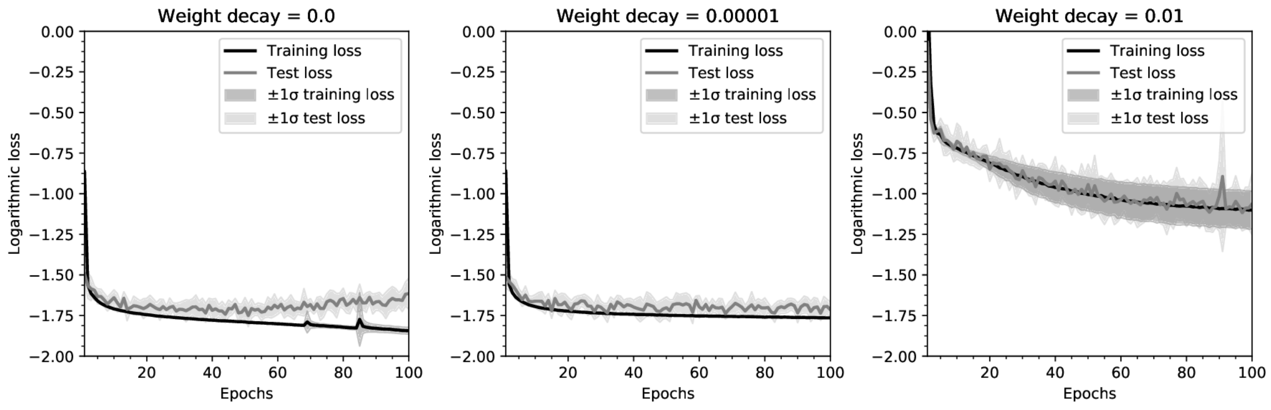


(b) Weight decay = 0.00001



(c) Weight decay = 0.01

Figure 13. The degree of regularization corresponds to a proportionally larger variance of the approximation compared to the truth. A slight bias in the training dataset might explain the consistently larger peak in the third posterior marginal.



(a) Weight decay = 0.0 (b) Weight decay = 0.00001 (c) Weight decay = 0.01

Figure 14. Loss plots of the ratio estimators in Figure 13. We empirically find that a larger loss corresponds with a larger variance of the approximation with respect to the truth. Similar behavior is observed for ratio estimators with insufficient capacity, as studied in Appendix E.

D.3. Detector calibration

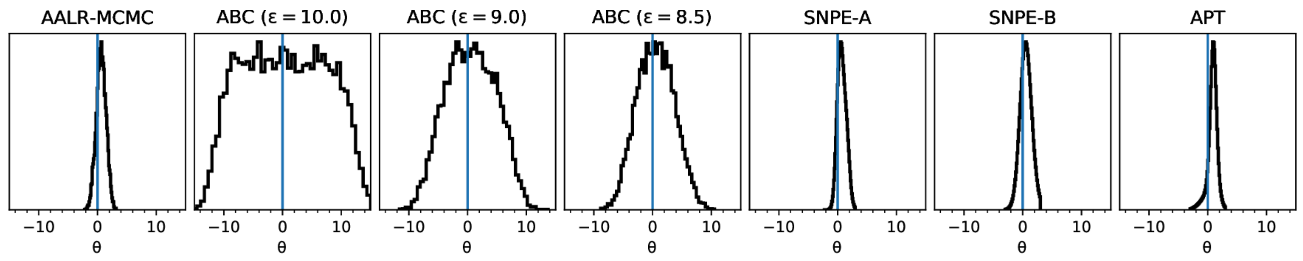


Figure 15. Approximate posteriors for the detector calibration benchmark. The posteriors are subsampled from several experimental runs.

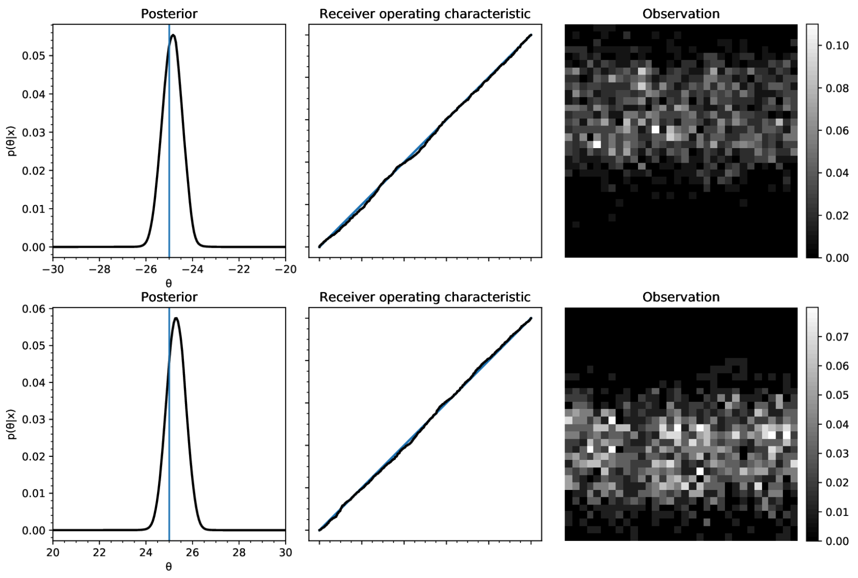
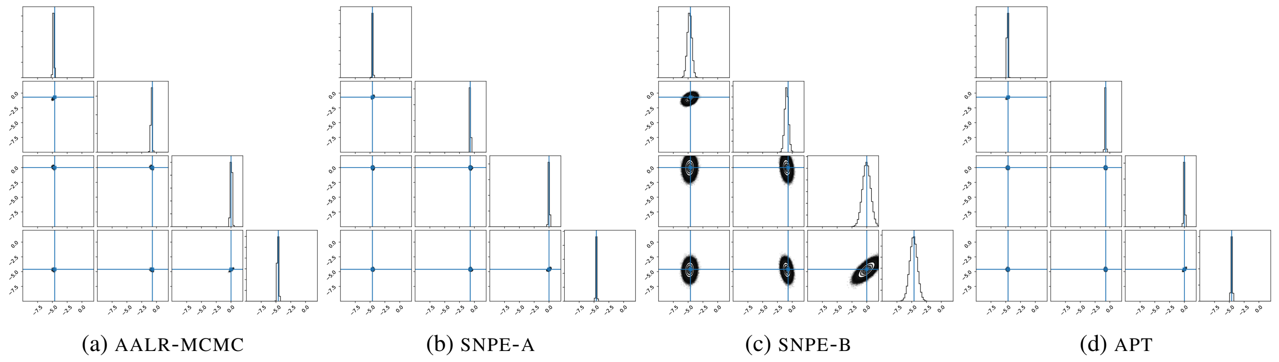


Figure 16. (Left:) Posteriors are obtained using the same ratio estimator. *(Middle):* Diagonal ROC diagnostic, demonstrating the ability of the proposed method to model posteriors for arbitrary observations. *(Right):* Observations $\mathbf{x}_o$.

D.4. Lotka-Volterra



(a) AALR-MCMC (b) SNPE-A (c) SNPE-B (d) APT

Figure 17. Posterior approximations for the Lokta-Volterra problem. AALR-MCMC, SNPE-A and APT are in agreement, while the SNPE-B approximation is significantly broader.

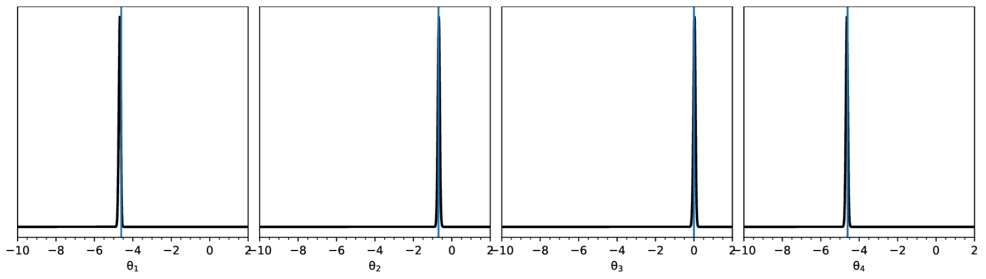
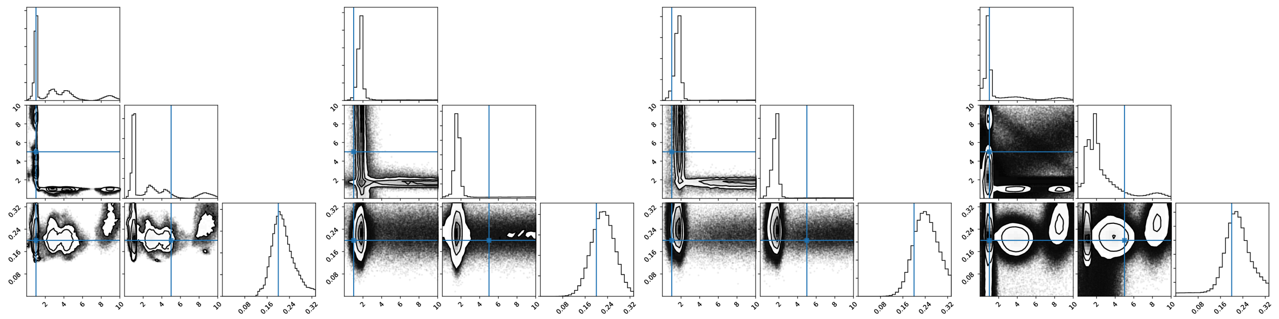


Figure 18. Posterior marginals of our approximation for the Lotka-Volterra problem.

D.5. M/G/1 queuing model



(a) AALR-MCMC (b) SNPE-A (c) SNPE-B (d) APT

Figure 19. Posteriors from the M/G/1 benchmark. The experiments are repeated 10 times and the approximate posteriors are subsampled from those runs. Despite the high variance of APT, it shares the most structure with AALR-MCMC.

D.6. Scientific use case: strong gravitational lensing

The simulation model consists of 4 main components. The first involves the telescope optics. We model the PSF (point spread function) as a Gaussian with standard deviation 0.5 in a 3×3 pixel kernel. The CCD sensor is set to an exposure time of 1000 seconds, background sky level = 0.1 and CCD noise is added. The mass distribution of the foreground galaxy is modeled as an elliptical isothermal (Kormann et al., 1994) at redshift $z = 0.5$ with axis ratio = 0.99, a random orientation-angle and an Einstein radius sampled from the prior. We do not model galaxy foreground light for the marginalization problem. For the Bayesian model selection problem, we model the foreground light of the lensing galaxy as an elliptical sersic with a random orientation angle and a sersic index sampled from $\mathcal{U}(.5, 1.5)$. For every source galaxy, we only model the light profile and their relative positions with respect to the lens. Source galaxies have an assumed redshift of $z = 2$. We assume the Plack15 cosmology. Table 4 describes the parameters and respective distributions we sampled from to generate a light profile for a single source galaxy.

Parameter	Distribution
Axis ratio	$\mathcal{U}(0.1, 0.9)$
Effective radius	$\mathcal{U}(0.1, 0.4)$
Intensity (flux)	$\mathcal{U}(0.1, 0.5)$
Location x	$\mathcal{U}(-1.0, 1.0)$
Location y	$\mathcal{U}(-1.0, 1.0)$
Axis orientation	$\mathcal{U}(0, 360)$
Sersic index	$\mathcal{U}(0.5, 3.0)$

Table 4. A complete description of the parameters describing the light profile is described in the `autolens` documentation.

E. Capacity of the ratio estimator and its effect on the estimated posterior

We investigate the approximation error in relation to the capacity of a ratio estimator. We consider a simple simulation model which accepts a model parameter $\boldsymbol{\theta} \triangleq (x, y, r)$ and produces an image with a resolution of 64×64 containing a circle at position x, y with radius r. Given the deterministic nature of the simulation model, we expect the posterior to be tight surrounding the generating parameters. The radius parameter r is multimodal due to squaring operation, which implies there should be a peak at $-r$ as well. We consider a uniform prior in the range $[-1, 1]$ for the parameters x and y. The radius has a uniform prior between $[-.5, .5]$. Random samples from the simulation model under $p(\boldsymbol{\theta})$ are shown in Figure 20. We evaluate the following architectures: (i) a fully connected architecture with 3 hidden layers and 128 units each (assumed low capacity), (ii) LENET (Lecun et al., 1998) (assumed mid-range capacity), (iii) and RESNET-18 (He et al., 2016) (assumed high-capacity). All models are trained according to the procedure described in Appendix C. We train the ratio estimators using a batch-size of 256 samples and the ADAM (Kingma & Ba, 2014) optimizer. As in other experiments, the neural networks use the SELU (Klambauer et al., 2017) activation function. The networks are trained for 250 epochs. No regularization or data normalization techniques are applied, with the exception of batch normalization (Ioffe & Szegedy, 2015) for the RESNET-18 architecture. For every architecture, we train 5 models. Figure 21 shows the mean loss curves and their standard deviations. We did not explore other hyperparameters or invest in additional training iterations. The loss plots indicate that the ratio estimator based on RESNET-18 should perform best.

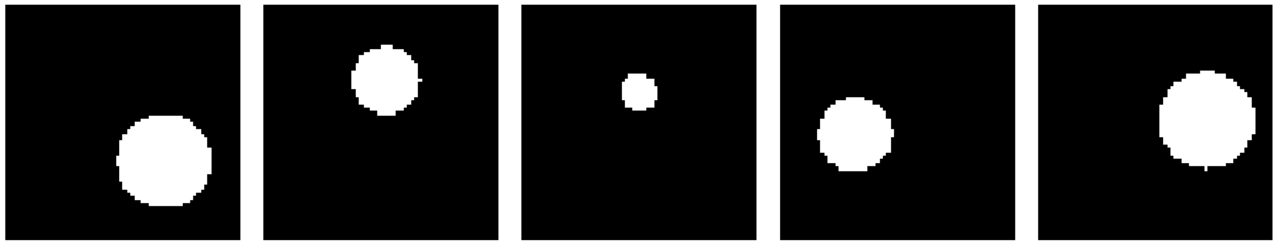


Figure 20. Random subsamples from the marginal model $p(\mathbf{x})$ for the circle problem.

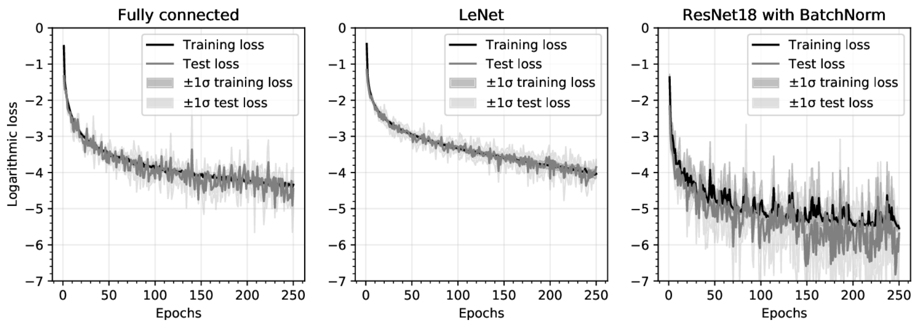
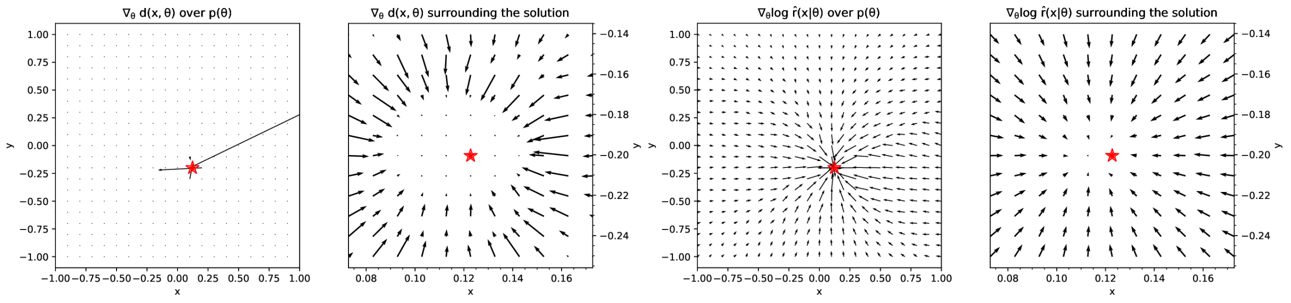


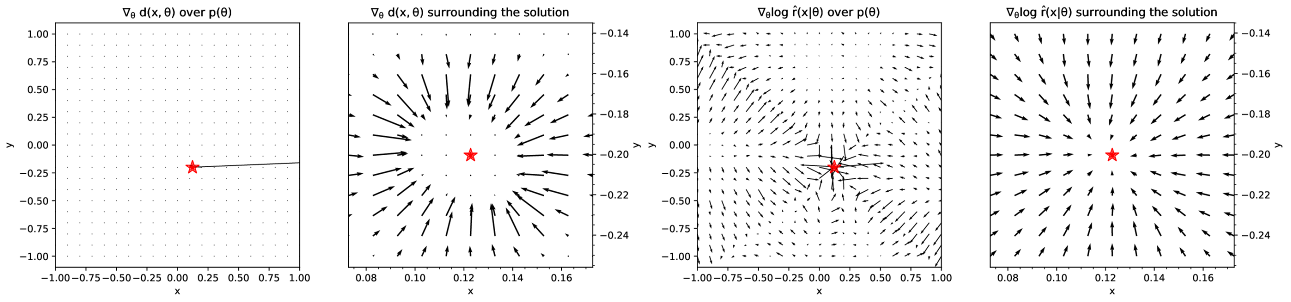
Figure 21. Loss plots of ratio estimators with different architectures. Mean training and test loss is reported, including the respective standard deviations. The plots indicate that the RESNET-18 ratio estimator should preform best. *(Left)*: The fully connected architecture. *(Middle)*: The LENET architecture. *(Right)*: The RESNET-18 architecture.

As noted above, the deterministic nature of the simulation model should generate posteriors which are sharp. We compute $\nabla_{\boldsymbol{\theta}}\, \mathbf{d}(\mathbf{x}, \boldsymbol{\theta})$ and $\nabla_{\boldsymbol{\theta}} \log \hat{r}(\mathbf{x} \,|\, \boldsymbol{\theta})$ to investigate how the gradients behave in $p(\boldsymbol{\theta})$. We expect the posteriors to be unimodal for the x and y parameters. As a result, the gradient field should converge to the generating parameter $\boldsymbol{\theta}^*$. Figure 22 shows the gradient fields across the different architectures. All use the same observation. The left-hand side of the figure shows $\nabla_{\boldsymbol{\theta}}\, \mathbf{d}(\mathbf{x}, \boldsymbol{\theta})$, and the right-hand side $\nabla_{\boldsymbol{\theta}} \log \hat{r}(\mathbf{x} \,|\, \boldsymbol{\theta})$. The saturation of the sigmoid operation is clearly visible as the gradients tend to 0. This is not the case for $\log \hat{r}(\mathbf{x} \,|\, \boldsymbol{\theta})$, demonstrating the effectiveness of the improvements put forward in Section 3.1. This behavior is preferable for Hamiltonian Monte Carlo, which relies on $\nabla_{\boldsymbol{\theta}} \log p(\mathbf{x} \,|\, \boldsymbol{\theta})$ to generate proposal states. As expected, the gradients show that the sharpest posterior was obtained by the RESNET estimator. However, the estimation of radius r seems to be problematic. The variance of the PDF among the ratio estimators indicates that ratio estimators are not sufficient to accurately approximate the radius. While the general structure is present in RESNET-18, the posterior for r is not sharp. A strategy to resolve this would be to increase the capacity of the neural network by adding more parameters (weights), or by modifying the architecture to exploit some structure in the data (e.g., a LSTM for time-series). Alternatively, other activation functions could be explored. Experiments indicate that ELU (Clevert et al., 2015) and SELU (Klambauer et al., 2017) activation functions are good initial choices. For sharp posteriors, we found that RELU activations worked best, as demonstrated in Figure 24 and Figure 23 (which uses SELU activations). Interestingly, even though the capacity is insufficient to capture all parameters, it seems that the solution is always included (the ratio estimator is not able to minimize the loss by excluding observations drawn from the generating parameter). This is a desirable property as true model parameters are not excluded, which could be beneficial in a Bayesian filtering setting.

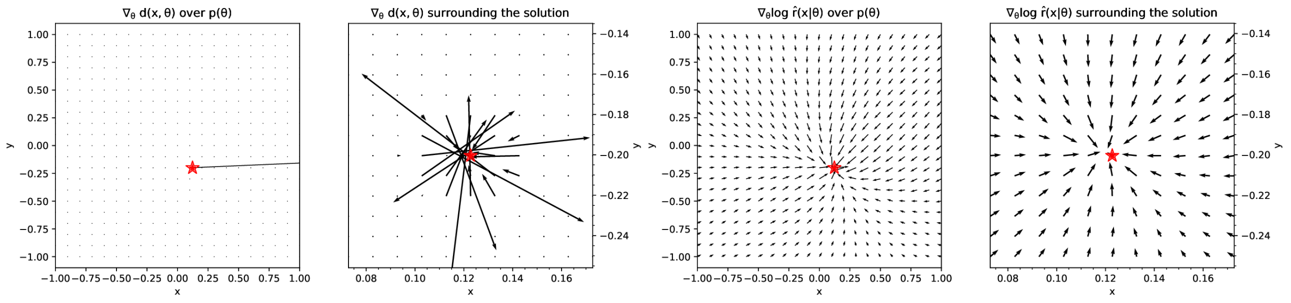
To conclude, the amortization of our ratio estimator requires sufficient representational power to accurately approximate $r(\mathbf{x} \mid \boldsymbol{\theta})$. The complexity of the task at hand determines whether the ratio estimator is able to exploit some structure in observations $\mathbf{x}$ and model parameters $\boldsymbol{\theta}$, thereby potentially reducing the necessary amount of parameters.



(a) Fully connected ratio estimator

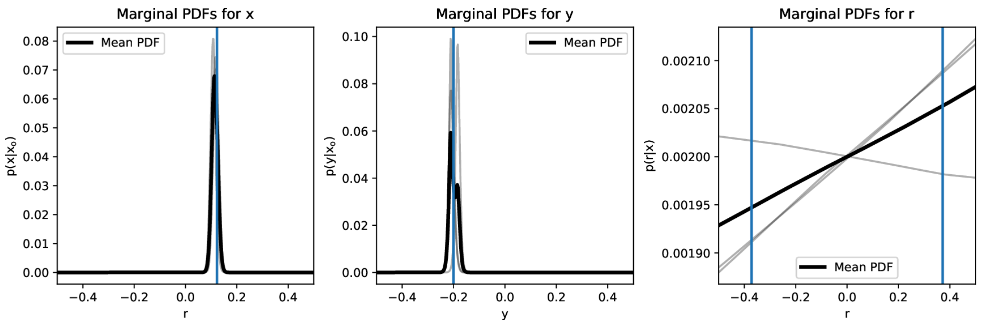


(b) LENET ratio estimator

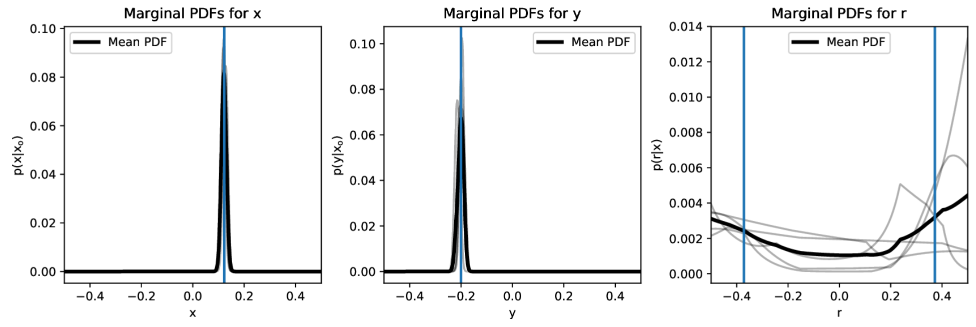


(c) RESNET-18 ratio estimator

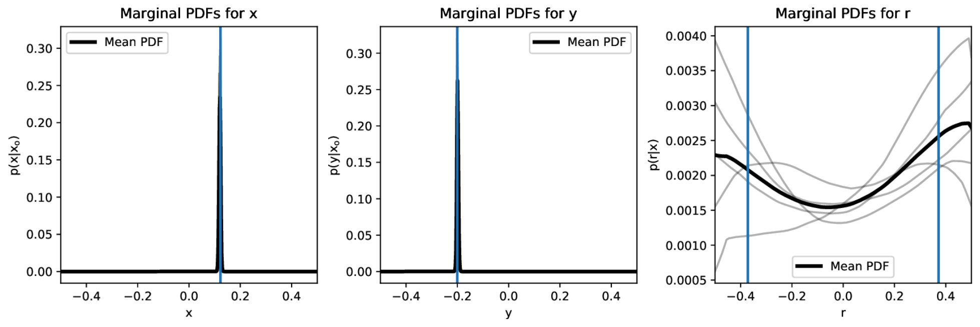
Figure 22. Vector (gradient) fields for the x and y parameters. The red star indicates the true solution. The parameter r remains fixed during the computation of these fields. The left-hand side shows the fields when backpropagating through the classifier output $\mathbf{d}(\mathbf{x}, \boldsymbol{\theta})$. The effect of the sigmoidal operation is clear, as the gradient saturates when the classifier $\mathbf{d}(\mathbf{x}, \boldsymbol{\theta})$ is (almost) able to perfectly discriminate samples. This supports the innovations presented in Section 3.1, as $\nabla_{\boldsymbol{\theta}} \log \hat{r}(\mathbf{x} \mid \boldsymbol{\theta})$ does not show this behavior.



(a) Fully connected ratio estimator

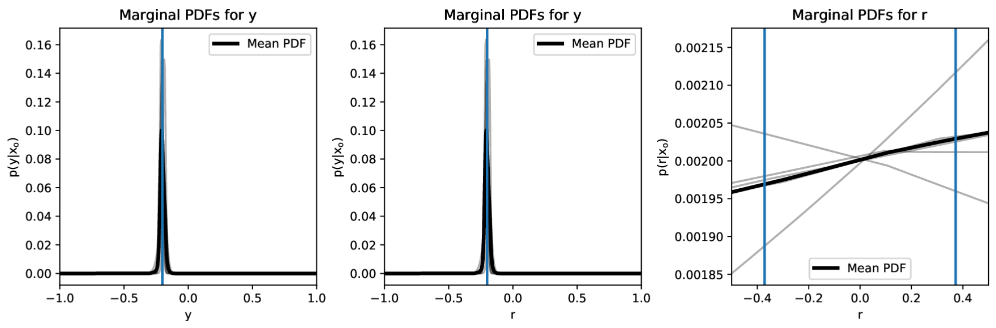


(b) LENET ratio estimator

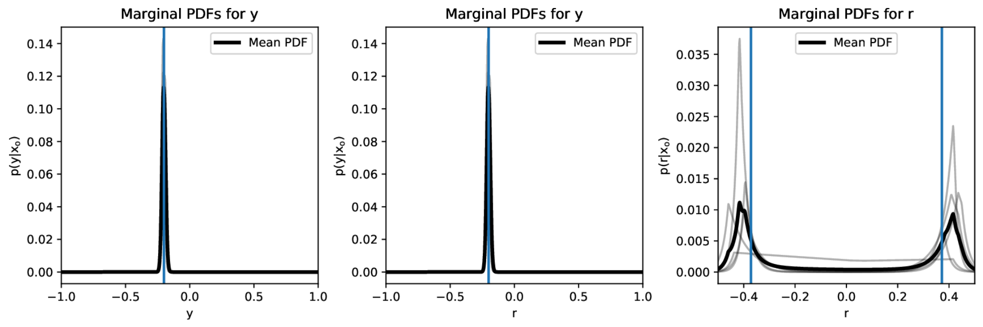


(c) RESNET-18 ratio estimator

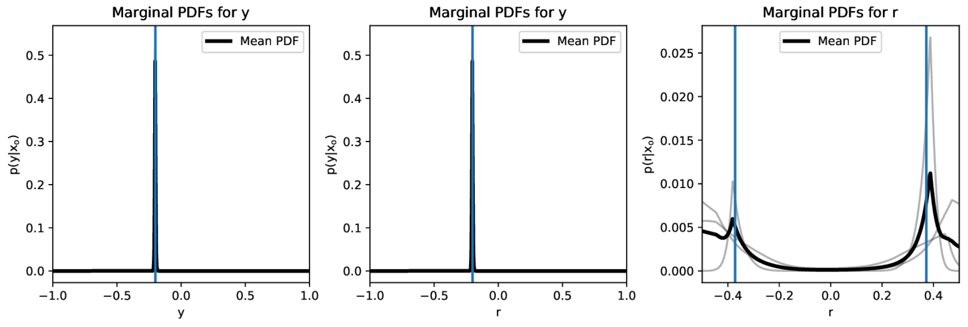
Figure 23. Marginals of the posteriors for the every ratio estimator architecture. The bold dark line shows the mean PDF, while a gray line shows the PDF of a single ratio estimator. The variance for the parameter r shows that the ratio estimators were not able to properly estimate the ratio $r(\mathbf{x} \mid \boldsymbol{\theta})$. This indicates an issue with the capacity, as the other parameters are properly estimated. Otherwise, the training hyperparameters could be at fault.



(a) Fully connected ratio estimator



(b) LENET ratio estimator



(c) RESNET-18 ratio estimator

Figure 24. Marginals of the posteriors for every ratio estimator architecture using RELU activations. The bold dark line shows the mean PDF, while a gray line shows the PDF of a single ratio estimator.