Classifiers versus Normalizing Flows for uncertainty-aware estimation of a signal ratio

Sluijter, Benjamin

Classifiers versus Normalizing Flows for uncertainty-aware estimation of a signal ratio.

THESIS

submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE
in
PHYSICS

Author :	Benjamin Sluijter
Student ID :	s2354276
Supervisor :	Dorothea Samtleben
External supervisor :	Benjamin Nachman

Berkeley (CA), United States of America, October 2024

Classifiers versus Normalizing Flows for uncertainty-aware estimation of a signal ratio.

Benjamin Sluijter

Huygens-Kamerlingh Onnes Laboratory, Leiden University
P.O. Box 9500, 2300 RA Leiden, The Netherlands

October 2024

Abstract

In recent years, many advancements have been made in the field of Simulation-Based Inference (SBI), due to the lack of tractable likelihoods in modern physics experiments. In the High-Energy Physics (HEP) literature, a popular choice for doing SBI is by using binary classifiers, which can be used to obtain likelihood ratios by means of the likelihood-ratio trick. In the Astrophysics literature, on the other hand, more research is done on Normalizing Flows, which directly model the likelihoods. In this thesis, we compare the two methods, assessing their performance on a general HEP problem: inference of a signal ratio in the presence of a nuisance parameter. We perform this comparison on both a toy Gaussian example and a realistic Higgs decay example and do not find a clear winner over the two cases. We do find interesting qualitative differences, especially for poorly performing models, suggesting that it may be beneficial to implement both methods rather than selecting just one.

Acknowledgements

First and foremost, I would like to thank Benjamin Nachman for welcoming me into his Machine Learning for Fundamental Physics group at LBNL and opening up this incredible opportunity for me. His guidance in this research project has been invaluable.
I am immensely grateful to Sascha Diefenbacher for my daily supervision and for patiently answering all my questions in the middle of her own work.
Thanks to everyone else at the lab, including Alkaid Cheng, Jordan Dudley, Jonas Glombitza, Aishik Gosh, Radha Mastandrea, Vinicius Mikuni, Dennis Noll, Inbar Savoray, and Gurpreet Singh for all the valuable discussions and for helping me on practical matters.
Finally, I would like to thank Dorothea Samtleben for her supervision from Leiden, including our meetings at very inconvenient hours.

Contents

Chapter 1

Introduction

Today's experiments in the physical sciences, such as those at the Large Hadron Collider, are more complicated than ever. The theory behind these experiments does not provide analytical relations between observables and theory parameters. In other words, one cannot access a function $p(D|\theta)$ describing the likelihood of theory parameters θ given observations D. These experiments make use of simulators, which allow one to input theory parameters θ and stochastically output possible observations D, but, crucially, not the other way around. Performing inference in these situations calls for so-called likelihood-free inference, also known as simulation-based inference.

The conventional approach to this problem is to project the data to some summary statistics. A likelihood function in the resulting lower dimensional data can then be estimated for example with histograms. However, recent years have seen a lot of research in the use of machine learning models and specifically Neural Networks, as these potentially use more of the information in the high dimensional data that these experiments entail.

One type of models that have been proposed for analysis in the LHC is that of classifiers, which can be used to model the ratio of the likelihood $p(D|\theta)$ to some reference likelihood $p(D|\theta_0)$ by means of the likelihood ratio trick [1–3]. For parameter inference of θ, this ratio is just as useful as the likelihood itself.

Another type of models, that are used to directly model the likelihood itself, are Normalizing Flows [4, 5]. These have been used in the literature for example in [6], in a Bayesian setting where they are combined with priors to calculate posteriors of GW parameters. Although Normalizing flows and classifiers both have their extra features (Flows can be used for fast sampling from the distribution and Classifiers can be used to do

reweighting), both can theoretically be used for the same simulation-based inference problems [7, 8]. However, there is no concrete comparison between the two, where they are evaluated on the same problem.
In this work, we compare a conditional normalizing flow and a classifier model on the common High-Energy Physics problem of inferring a signal ratio in the presence of a nuisance parameter.

Chapter 2

Theory

2.1 Uncertainty aware inference of the signal ratio

The parameter of interest is the signal-to-background ratio, from here on referred to as μ, defined as:

$$\mu = \frac{p(sig|\mu)}{p(bkg|\mu)} \tag{2.1}$$

Since we know that, per definition, $p(sig|\mu) + p(bkg|\mu) = 1$, we can rewrite to:

$$p(sig|\mu) = \frac{\mu}{\mu + 1} \tag{2.2}$$

$$p(bkg|\mu) = \frac{1}{\mu + 1} \tag{2.3}$$

We want to infer this μ parameter, based on some observed data $D = \{x_i\}$, consisting of both signal and background events. The distributions of the two may depend on some *nuisance parameters* z and are denoted as $p(x_i|sig, z)$ for signal and $p(x_i|bkg, z)$ for background. Hence, the total probability of observing x_i, given signal ratio μ and nuisance parameter z is given by:

$$p(x_i|\mu, z) = \frac{\mu}{\mu + 1} p(x_i|sig, z) + \frac{1}{\mu + 1} p(x_i|bkg, z) \tag{2.4}$$

Under the assumption that our data is i.i.d., the likelihood of finding the observed data $\{x_i\}$ is

$$p(\{x_i\}|\mu, z) = \prod_i [\frac{\mu}{\mu+1} p(x_i|sig, z) + \frac{1}{\mu+1} p(x_i|bkg, z)] \tag{2.5}$$

The idea of likelihood-free inference in this context is that we do not have access to these likelihoods, but that we do have access to a signal and a background simulator, both with tunable z parameter, allowing us to sample $x \sim p(x|sig, z)$ and $x \sim p(x|bkg, z)$. Data sampled from these simulators can be used to train machine learning models to approximate the likelihood or the likelihood ratio, which will be explained in the next sections.

2.2 Method of normalizing flows

One type of machine learning model that can be used is that of normalizing flows [5]. A normalizing flow consists of a simple base probability distribution $\pi(u)$ and an invertible, differentiable function f that transforms variable u into variable x. This leaves one with a new probability distribution over x, $p_f(x)$, that is given by the change of variable rule for probability densities [4]:

$$p_f(x) = \pi(f^{-1}(x))|det(\frac{\partial f^{-1}}{\partial x})| \tag{2.6}$$

When taking the function f to be a neural network, one can optimize the parameters of this net with gradient descent, such that $p_f(x)$ approximates a desired probability function $p(x)$. By taking a neural network that depends on a context parameter, one can also model conditional probabilities, which is done for example in [6]. In our case, we have the nuisance parameter z as context parameter and we then train one normalizing flow to approximate $p(x|sig, z)$ and another flow to approximate $p(x|bkg, z)$. These are then combined to obtain the approximation of the total likelihood $p(\{x_i\}|\mu, z)$ with eq. 2.5.

2.3 Method of binary classifiers

2.3.1 Likelihood ratio

For this method, instead of finding the likelihood of (μ, z) given some data $\{x_i\}$, we try to find the likelihood ratio to reference values (μ_0, z_0). This is

obtained from eq. 2.5 as:

$$\frac{p(\{x_i\}|\mu,z)}{p(\{x_i\}|\mu_0,z_0)} = \prod_i [\frac{\mu}{\mu+1}\frac{p(x_i|sig,z)}{p(x_i|\mu_0,z_0)} + \frac{1}{\mu+1}\frac{p(x_i|bkg,z)}{p(x_i|\mu_0,z_0)}] \tag{2.7}$$

This allows us to write the factors in eq. 2.1 as (taking $\mu_0 = 1$):

$$\frac{p(x|\mu,z)}{p(x|\mu_0=1,z_0)} = \frac{\mu}{\mu+1}[\frac{2\cdot p(x|sig,z)}{p(x|sig,z_0)+p(x|bkg,z_0)}] + \frac{1}{\mu+1}[\frac{2\cdot p(x|bkg,z)}{p(x|sig,z_0)+p(x|bkg,z_0)}] \tag{2.8}$$

Now, the two terms in brackets can be estimated with the help of machine learning by means of the "likelihood ratio trick".

2.3.2 Likelihood ratio trick

When a binary classifier with input vector $\mathbf{x}$ is "Bayes optimally trained" with respect to a binary cross-entropy loss, the output will be:

$$f(\mathbf{x}) = \frac{p(\mathbf{x}|+)}{p(\mathbf{x}|+)+p(\mathbf{x}|-)} \tag{2.9}$$

with + indicating the positive class and - the negative class. Hence, by a simple transformation, we obtain the likelihood ratio:

$$\frac{p(\mathbf{x}|+)}{p(\mathbf{x}|-)} = \frac{f(\mathbf{x})}{1-f(\mathbf{x})} \tag{2.10}$$

So all that is left is choosing the positive and negative class in the right way.

2.3.3 Classes

In our case the nuisance parameter z is added to the input vector, giving:

$$\frac{p(\mathbf{x},z|+)}{p(\mathbf{x},z|-)} = \frac{f(\mathbf{x},z)}{1-f(\mathbf{x},z)} \tag{2.11}$$

One classifier model, which we will call f_{sig}, is trained to approximate the first term in brackets in eq. 2.8 and another one, f_{bkg}, to approximate the second term in brackets.
The generative process of the positive class works as follows: First, z is drawn from a distribution that we will denote as $p_z(z)$. This can be any distribution, but in this thesis we used a uniform one. Next, $\mathbf{x}$ is obtained

from the signal simulator in the case of f_{sig} and from the background simulator in the case of f_{bkg}, where the simulator its z value is set to the value that we just obtained. This gives $p(\mathbf{x}, z|+) = p(\mathbf{x}|z, sig)p_z(z)$ for f_{sig} and $p(\mathbf{x}, z|+) = p(\mathbf{x}|z, bkg)p_z(z)$ for f_{bkg}.
For the negative class, there is no difference between f_{sig} and f_{bkg}. First, z is drawn from the same distribution $p_z(z)$. $\mathbf{x}$ is then obtained from either the signal or the background simulator, each with a probability of 50%. The z value of the simulator is set to the fixed reference value z_0. This gives $p(\mathbf{x}, z|-) = [\frac{1}{2}p(\mathbf{x}|z_0, sig) + \frac{1}{2}p(\mathbf{x}|z_0, bkg)]p_z(z)$ for both f_{sig} and f_{bkg}. Dividing $p(\mathbf{x}, z|+)$ by $p(\mathbf{x}, z|-)$, the $p_z(z)$ factors cancel out, and the terms in brackets in eq. 2.8 are obtained from the two different classifiers.

2.4 Confidence Intervals and Wilks theorem

A Confidence Interval (CI) with coverage α is an interval in a parameter space Θ that depends on observation X and on average contains the true parameter θ_{true}, $\alpha \cdot 100\%$ of the time, where the underlying distribution is some $p(x|\theta)$.
In case of access to the function $p(x|\theta)$, one can construct such a CI exactly, which requires the procedure of finding a so-called *confidence belt* (see for example chapter 9.2 of [9]). However, especially for higher dimensional data x, this gets computationally infeasible, which is why it is common to make use of approximations such as Wilks theorem.

When Wilks theorem applies [10], it says that the random variable $-2log\lambda(X)$, with

$$\lambda(X) = \frac{p(X|\theta, \hat{\theta}_1, .., \hat{\theta}_n)}{p(X|\hat{\theta}, \hat{\theta}_1, .., \hat{\theta}_n)} \tag{2.12}$$

, is approximately distributed as a chi-squared distribution χ^2_1 for a sufficient number of observations. Here, θ is the parameter over which the CI will be calculated, and the $\{\theta_i\}$ are all remaining theory parameters.
Hence, for any choice of threshold value t, the probability that the random variable $-2log\lambda$ is observed below threshold t, is given by the integral over the chi-squared function:

$$P(-2log\lambda < t) = \int_0^t \chi^2_1(t')dt' \tag{2.13}$$

Of course, this only holds when λ is calculated from observation x with the true θ, which we do not know. However, if we calculate $-2log\lambda(x)$ for

threshold	coverage
$-2log\lambda < 1$	68.3%
$-2log\lambda < 4$	95.4%
$-2log\lambda < 9$	99.7%

Table 2.1: *Examples of thresholds on $-2log\lambda$ that can be used to construct CIs with Wilks theorem and the resulting CI coverages. These follow from equation 2.13*

a sufficiently large range on Θ and the CI is constructed as the interval on Θ for which $-2log\lambda(x)$ is below threshold t:

$$CI = \{\theta \in \Theta | -2log\lambda < t\} \tag{2.14}$$

, we know that the probability that the true θ is included is the probability that the true $-2log\lambda$ is below t, which is given by eq. 2.13.
Thus, this gives one a straightforward method to construct a CI by simply choosing some threshold t, where the corresponding coverage can be calculated very easily. Some examples of common choices are given in table 2.1.

Chapter 3

Methods

3.1 Datasets

We performed the comparisons on two different datasets. One is a particle physics dataset similar to the one in [11], where the signal is a Higgs boson decay and the background a Z boson decay. We will refer to this dataset as the *Higgs case*. The downside of the Higgs case is that we do not have access to the true likelihoods. This is why we will first compare the modeltypes on a toy example, consisting of simple Gaussian probability densities, allowing us to compare the results of the models to the true answer. We used two variants of this *Gaussian case*, which we will refer to as the *Gaussian large distance case* and the *Gaussian small distance case*.
All training sets consist of separate signal and background sets. In the test sets, signal and background events are mixed according to the μ value of the test set, as $N_{signal} = int(N_{total}(\frac{\mu}{\mu+1}))$, where $int()$ means rounding off to an integer, N_{signal} is the number of signal events in the test set and N_{total} is the test set size. The amount of background events in the test set is then given by $N_{background} = N_{total} - N_{signal}$.

3.1.1 Gaussian case

For our Gaussian example we generate our own data, very similar to [11]. Signal data is drawn from a 2-dimensional Gaussian with a standard deviation of 1 and a mean given by polar coordinates: $\phi = z$ (with z the nuisance parameter) and $r = 0.5$ or $r = 2$. $r = 0.5$ gives us the small distance case and $r = 2$ the long distance case. Background data is gener-

ated identically, except for the radial polar coordinate of the mean being $r = -0.5$ or $r = -2$ for the small and the long-distance case respectively. Examples are given in figure 3.1. Since we generate the samples ourselves, we have infinite training and test data at our disposal.
For each model we trained, we generated a new training set of 10^6 events;

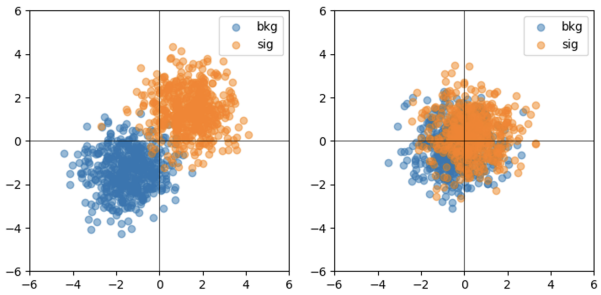

Figure 3.1: *Gaussian datasets. Left: large distance case with $z = \pi/4$. Right: small distance case with $z = \pi/4$.*

$5 \cdot 10^5$ signal and $5 \cdot 10^5$ background events. The z values for the events were drawn from a uniform distribution on the interval $[0, 0.5\pi]$. For each classifier model, an additional training set was generated, with the same amount of events, but a fixed z value of $z_0 = 0.25\pi$ for all events. We will refer to these two training sets respectively as the z training set and the z_0 training set.
Each validation batch is generated from scratch, where the z values are either drawn from the uniform distribution on the interval $[0, 0.5\pi]$ or set to $z_0 = 0.25\pi$.
Finally, for each model we tested, we generated a new set of test sets. Such a set consists of 5 test sets of 9000 events for each of 9 parameter combinations of μ and z, given by a grid of $\mu = [0.1, 0.2, 0.3]$ by $z = [0.15\pi, 0.25\pi, 0.35\pi]$, giving us a total of 45 independent 9000-event test sets.

3.1.2 Higgs case

The data for the physical case was simulated with open-source software Pythia 8.2 [12] and Delphes 3.5.0 [13]. Pythia was used to simulate 13 TeV center-of-mass energy proton-proton collisions. Subsequently, Delphes was used to simulate the collision products being measured by the ATLAS detector.

	signal	background
train	$4.1 \cdot 10^6$	$2.9 \cdot 10^6$
validation	$2.0 \cdot 10^5$	$2.0 \cdot 10^5$
test	5 x $4.5 \cdot 10^3$	5 x $1.2 \cdot 10^6$

Table 3.1: *Datasizes in Higgs dataset*

The signal consists of a Higgs boson decaying to a pair of tau leptons ($H \rightarrow \tau\tau$) events and the background consists of a Z boson decaying to a pair of tau leptons ($Z \rightarrow \tau\tau$). In both processes, one τ particle decays hadronically and one decays leptonically. The energy scale of the hadronically decaying τ, the Tau Energy Scale (TES) [14], forms a systematic uncertainty and is used as the nuisance parameter z in our experiment. Data was simulated with the nominal TES value of 1.0 and the effect of non-nominal values is applied afterward with a skewing function [15], following [11, 16].
There are 30 different features that describe the events. However, for simplicity we used only two: the invariant mass of the hadronic tau and the lepton, *DER_mass_vis*, and the ratio between the transverse momentum of the lepton and that of the hadronic tau, *DER_pt_ratio_lep_had*. The distributions of these two features for z values of 0.9, 1.0, 1.1 are given in 3.2. We picked specifically these two for their large dependence on the nuisance parameter. We used a simulated dataset consisting of a training set, a validation set, and 5 separate test sets, each consisting of a background part and a signal part. The sizes of the sets are given in table 3.1.

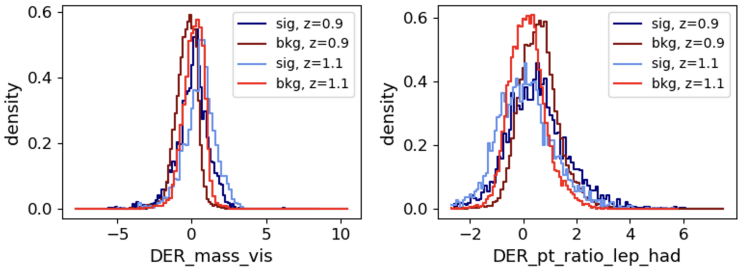

Figure 3.2: *Higgs dataset.*

For each flow/classifier model, we took the entire training and validation sets and sampled z values from a uniform distribution on the interval $[0.8, 1.2]$. All events were then skewed according to their z value. For each classifier model, an additional z_0 training/validation set was constructed

Version of October 2024– Created October 11, 2024 - 08:42

by taking the full training/validation set again and leaving the z values at $z_0 = 1.0$, thus not shifting the events in feature space.
Each model was tested on test sets with 9 parameter combinations of μ and z, given by a grid of $\mu = [0.1, 0.2, 0.3]$ by $z = [0.93, 1.00, 1.07]$. 9 of these test sets were taken from each of the 5 independent test datasets given in figure 3.1, by mixing signal events and background events according to μ and then skewing the events according to z, giving a total of 45 test sets.

3.1.3 Bootstrapsets

To extract more statistics from the 9000 event test sets, we used bootstrapping. Bootstrapping means sampling multiple datasets from the same dataset by sampling with replacement. Specifically, we sampled (with replacement) 100 sets consisting of 1000 events from each 9000 event test set. These 1000 event sets, we will from here on refer to as *bootstrapsets*.
Note that we fixed the N_{signal} and $N_{background}$ numbers in the testsets (see 3.1), such that the N_{signal} and $N_{background}$ in the bootstrapsets are distributed binomially with $p(sig|\mu) = \frac{\mu}{\mu+1}$ and $p(bkg|\mu) = \frac{1}{\mu+1}$.

3.2 Models

3.2.1 Classifier model

We implemented our classifier models with PyTorch [17]. The signal and background models are identical fully connected deep neural networks with 3 input nodes (2 features and the z value), 3 hidden layers of 120 nodes, ReLU activation functions, and one node in the final layer with Sigmoid activation to give a binary outcome. We use the Adam optimizer with learning rates of [1e-6, 3e-6, 1e-5, 3e-5, 1e-4, 3e-4] and a batch size of 512. Each batch must be constructed in a sophisticated way to satisfy the theory laid out in 2.3.3: a 512-instance batch consists of 256 positive class instances and 256 negative class instances. For the positive class, we take instances from the z training set, signal for the signal model, and background for the background model. For the negative class, we sample 256 instances from the z_0 training set, with 50% probability for each instance to be signal or background. Finally, we replace the z values of the 256 negative class instances (all z_0) by the set of z values from the positive class.
In the Gaussian case, we trained for 30k iterations, and in the Higgs case, we trained for 100k iterations.
Every 50 iterations, the validation loss was calculated with the validation

batches constructed in the same way as the training batches, but with a batch size of 20000 instead of 512, and the events being newly generated in the Gaussian cases and coming from the validation set in the Higgs case.
For each of the above-mentioned learning rates, 30 models were trained for each of the Gaussian cases and 40 models were trained for the Higgs case.

3.2.2 Flow model

We implemented our normalizing flows with the nflows package [18]. The signal and background models are identical and consist of 4 layers of masked piecewise rational-quadratic autoregressive transforms [4] with 64 hidden features, 10 bins, a tail bound of 7, linear tails, 2 blocks, using residual blocks and with ReLU activation, followed by a random permutation layer. The base distribution is a normal distribution. We have two input features and one context feature the nuisance parameter z. We use the Adam optimizer with learning rates of [1e-6, 3e-6, 1e-5, 3e-5, 1e-4, 3e-4] and batch size of 512.
In the Gaussian case, we trained for 30k iterations and in the Higgs case, we trained for 100k iterations.
Every 50 iterations, the validation loss was calculated with batches of size 4000, and the events being newly generated in the Gaussian cases and coming from the validation set in the Higgs case.
For each of the above-mentioned learning rates, 30 models were trained for each of the Gaussian cases and 40 models were trained for the Higgs case.

3.3 Analysis

As mentioned in the methods section, for each combination of dataset, model, and learning rate, we trained a number of independent models; 30 in the Gaussian case and 40 in the Higgs case. We trained more models for the Higgs case, as that is the most relevant case.
Naturally, for the "true likelihood" model in the Gaussian case, all 30 models are exactly the same, so only the data is different. We tested each of the models on 45 different test sets, consisting of five independent test sets for each of nine different (μ, z) values). We will now explain how we assessed the performance of each model on each test set, where we will start with the concept of the Negative Log Likelihood scan, referred to as $NLL(\mu, z)$.

3.3.1 $NLL(\mu, z)$

To test a model on some test data $\{x_i\}$ (either bootstrap set or full test set), we calculate its $NLL(\mu, z)$, where NLL stands for Negative Log Likelihood for this data, which is done in the following way.
For every event x_i in the test set, we scan over a grid of (μ, z) values and calculate the negative log of the model's prediction. Giving us (the model's approximation to) $-log[p(x_i|\mu, z)]$ for the flow and $-log[p(x_i|\mu, z)] + log[p(x_i|\mu_0, z_0)]$ for the classifier. For both the Gaussian and the Higgs case we used 600 μ values, evenly spaced between 0 and 0.5. We used 300 z values, evenly spaced between 0.1π and 0.4π for the Gaussian case and between 0.9 and 1.1 for the Higgs case. Note that these are smaller z parameter spaces than the ones we trained the models on. This was done for the reason that the classifiers often attributed erroneously high likelihoods towards the edges of the z parameter space it was trained on (see appendix A).
We then have a 2D array of negative log likelihoods, for each event. By summing these 2D arrays for all the events $\{x_i\}$ we get the negative log-likelihood for the full dataset, since with i.i.d. data: $-log[p(\{x_i\}|\mu, z)] = -\sum_i log[p(x_i|\mu, z)]$. Finally, we subtract the minimum value in the 2D array from the whole array, to obtain what we refer to as $NLL(\mu, z)$.

$$NLL(\mu, z) = -log[p(\{x_i\}|\mu, z)] + C \tag{3.1}$$

Where C is a constant with respect to μ and z, which carries no relevant information. For the classifier, the $log[p(\{x_i\}|\mu_0, z_0)]$ term is absorbed in this C, making $NLL(\mu, z)$ comparable between flow and classifier models. An $NLL(\mu, z)$ in itself is interesting to visualize, which we will do (3.3.4, and it can also be used to calculate CIs, which we will talk about next.

3.3.2 Confidence Intervals (CIs): coverage and mean width

Confidence Intervals give a method of measuring how good the modeled likelihood function is, without access to the true likelihood function. We construct 68.3% CIs and therefore, the closer we find the coverage to 68.3%, the better the modeled likelihood function. This is in the theoretical situation where the true likelihood CIs give an exact coverage of 68.3%. However, due to non-exact CI construction and limited statistics, both of which will be discussed in this section, even for the true likelihood CIs we might observe some deviation from this value. Luckily, in the Gaussian cases, we could calculate the true likelihood coverages as well to compare too. For the Higgs case, we have to use what we learn from the true likelihood

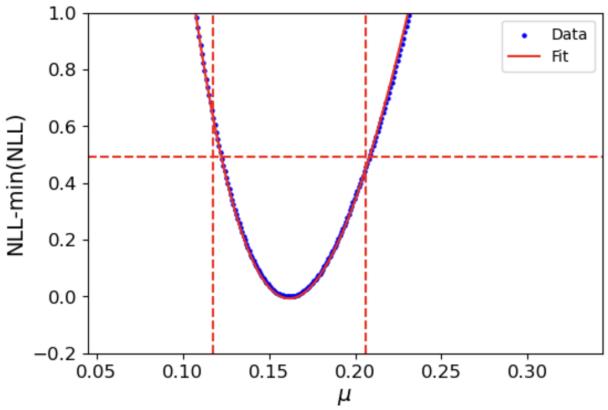

Figure 3.3: *Example of a μ CI construction. In blue are the values of the $NLL(\mu, \hat{z})$. The horizontal dashed line indicates the $-2log\lambda = 1$ line and the vertical lines indicate the boundaries of the constructed CI. Horizontal and vertical line intersections do not always coincide exactly with the fit, due to parabolic assumption.*

coverages and mean widths in the Gaussian case, to draw conclusions.

Construction of CIs

We constructed Confidence Intervals for both μ and z, using the theory in 2.4. Looking at equation 2.12, for the μ CI case we have $\theta = \mu$ and $\theta_1 = z$ and for the z case, we have $\theta = z$ and $\theta_1 = \mu$. We first take the $NLL(\theta, \theta_1)$ and profile over the θ_1 parameter, giving us $NLL(\theta, \hat{\theta}_1)$. Next, we fit this to a 4th-degree polynomial, in order to get the full profile likelihood instead of just the values on our grid. By subtracting the minimum from the fit, $NLL(\hat{\theta}, \hat{\theta}_1)$, we obtain $-log[\frac{p(\{x_i\}|\theta,\hat{\theta}_1)}{p(\{x_i\}|\hat{\theta},\hat{\theta}_1)}] = -log\lambda(\{x_i\})$, where we use eq. 3.1 and see that the constant cancels. The Confidence Interval is then obtained by taking the interval $-log\lambda < 0.5$, corresponding to a coverage of 68.3%. An example of this procedure with $\theta = \mu$ is given in figure 3.3. Note that the intersections of the CI boundaries and the $-2log\lambda = 1$ line are not exactly on our fit. This is due to the fact that we used the assumption that our fits would be polynomials of 2nd-degree, which is not always exactly true. In appendix B we address the difference this makes, which is that, according to the 4th-degree fits, we constructed CIs of maximally $\sim$75% coverage and minimally around $\sim$65%.

Analysis of CIs

We measured the coverage for each test set by constructing a CI for each of its bootstrap sets and calculating the fraction that contained the true parameter. Using bootstrapping to measure the coverage can lead to a bias in the observed coverage compared to the true coverage, as shown in [19]. This bias might depend on the sizes of the bootstrap set n and that of the test set N. Therefore, we did an experiment where we measured true likelihood coverages as a function of $k = N/n$, shown in section 4.1. For our main experiments, we use $k = 9$ and $n = 1000$, which means with our 9000 event test sets, we have 1000 event bootstrap sets.
The width of the CIs is summarized as the mean width over the CIs of the 100 bootstrapsets. Hence, for each test set, we obtain a coverage and a mean width number for both μ and z.

3.3.3 Overlap

For the Gaussian case, we have the full true likelihoods. This information we exploit by comparing the full likelihood, as approximated by the model, with the true likelihood, by overlapping the two. We do this for 1 dimensional profiled likelihoods $NLL(\mu) = NLL(\mu, \hat{z})$ (μ *overlap*) and $NLL(z) = NLL(\hat{\mu}, z)$ (z *overlap*), but also with the full 2 dimensional likelihood $NLL(\mu, z)$ (*total overlap*). We will now show the procedure for $NLL(\theta)$, where θ is either μ, z or (μ, z).
First, we calculate the likelihood as:

$$p(\{x_i\}|\theta) \cdot A = e^{-NLL(\theta)} \tag{3.2}$$

where A is some constant that we do not care about.
We eliminate the constant by means of normalization:

$$p_{norm}(\theta|\{x_i\}) = \frac{A \cdot p(\{x_i\}|\theta)}{A \cdot \sum_{grid} p(\{x_i\}|\theta)} \tag{3.3}$$

Overlap between the normalized likelihood as calculated by the model, p_{norm}, and the true normalized likelihood, $p_{norm,true}$, is given by:

$$overlap = 1 - \frac{1}{2} \sum_{grid} |p_{norm}(\{x_i\}|\theta) - p_{norm,true}(\{x_i\}|\theta)| \tag{3.4}$$

Where the $\sum_{grid}$ is either the sum over the full grid ($\theta = (\mu, z)$) or the 1D versions of it ($\theta = \mu/z$).
This gives a metric that equals 1 when the likelihood is exactly the same as

the true likelihood (up to a constant factor) and goes to 0 when it is completely wrong.
The overlap number for each test set was obtained by calculating the overlap on each of its 100 bootstrapsets and then averaging over those.

3.3.4 Plotting contours

We visualize our results by plotting the 0.5 and 2.0 contours of the $NLL(\mu, z)$. This allows us to show contours for 9 different test sets in one figure, one for each of the (μ, z) combinations that we used. In all the contours we show in this thesis, we constructed the $NLL(\mu, z)$ for a full 9000 event test set and then divided by 9. This way, one obtains something that can be considered as the expected value of a bootstrap $NLL(\mu, z)$ over the full test set.

Chapter 4

Results

4.1 True likelihood coverages

For this experiment, we calculated coverages for the true likelihoods in the two Gaussian cases, both with and without bootstrapping.
Every non-bootstrapping coverage number was calculated on 2700 independent test sets of 1000 events; 300 for each of the 9 (μ, z) parameter combinations discussed in 3.1.1. This was then done four times to obtain four coverage numbers, over which mean and std were calculated, which are represented as [mean-std, mean+std] bands in figure 4.1.
The bootstrapping coverages were calculated for two different sizes of bootstrap sets, being $n = 1000$ and $n = 5000$. We used test set sizes different from the $N = 9000$ test set size in our main experiments, in order to obtain a range of $k = N/n$ values. The bootstrapping results are given as the lines in figure 4.1. Each bootstrap value was obtained by calculating the coverage on the 100 bootstrap sets of each of 45 test sets (see 3.1.1) and averaging over those.

The non-bootstrapping bands are all found between 66% and 70% with the z bands being smaller than the μ ones. Interestingly, for low k, all the small case coverages and the large case z coverages, are way below their corresponding non-bootstrapping values, at around 52%. For higher k, coverage rapidly increases to around 65%. However, the curves slowly flatten out, making it seem like they need much higher k to reach their non-bootstrapping coverage. The large case μ coverages on the other hand, are already found in the corresponding non-bootstrapping band for $k > 3$. Furthermore, it seems like the coverages only depend on the ratio k and not on the absolute sizes n and N.

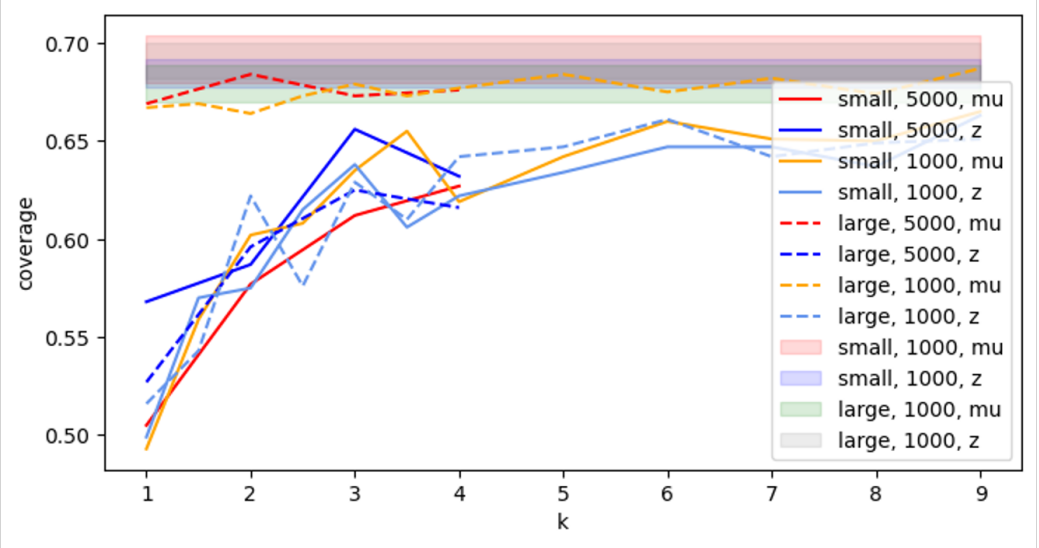

Figure 4.1: *Coverages in the Gaussian case of CIs constructed with true likelihoods, with and without bootstrapping. The bands show the non-bootstrapping results and the lines show the bootstrapping results as a function of k, which is the fraction between the bootstrap set size (numbers in the legend) and the full test set size.*

4.2 Main results

As described in 3.3, for each model we trained, we calculated a number of metrics on a total of 45 test sets. The metrics are μ and z CI coverage, μ and z CI mean width, and in the Gaussian case also μ, z, and total overlap. We averaged these numbers over the 45 test sets to get a final value of these metrics for each model.
The mean and standard deviations over the 30/40 models we trained are shown in figure 4.2 as lines with bands, for all our experiments. For the Gaussian case, we also show the results with the true likelihood. Just like for the classifier and flow models, the true likelihood was assessed on 30 sets of 45 test sets. The fact that the true likelihood results also have bands (standard deviations), is due to the fact that in the Gaussian case, the 30 sets of 45 testsets are all different. The true likelihood results are not shown for the overlaps, since the true likelihood results there are 1 by construction (you get $p_{true} - p_{true}$) in eq. 3.4.

From these figures, we choose for each of the three cases the ”best” learning rates to take into our further analysis. These are stated in table 4.1.

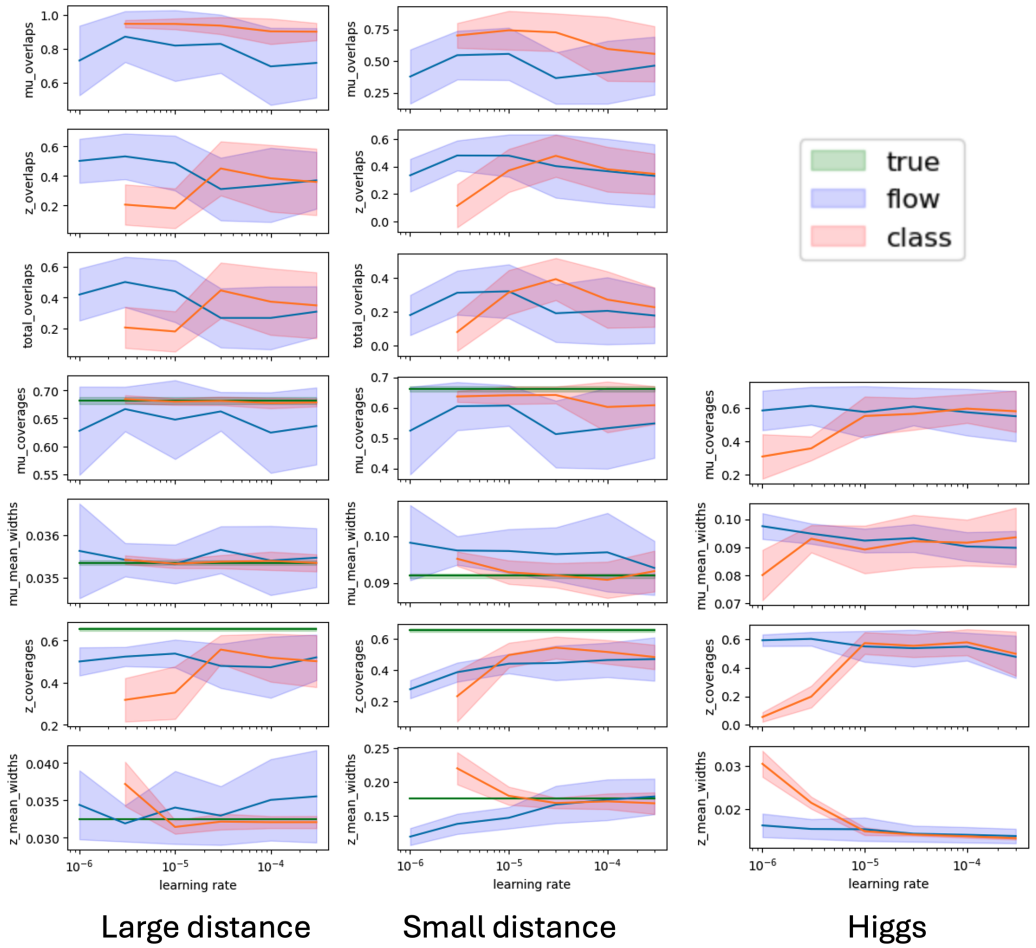

Figure 4.2: *All metrics as a function of learning rate. Lines and bands are means and standard deviations over the models.*

4.2.1 Gaussian large distance case

Histograms of the 30 flow models with learning rate $3 \cdot 10^{-6}$ and the 30 classifier models with learning rate $3 \cdot 10^{-5}$ of the large distance case are shown in figure 4.3. For both classifier and flow, we see that μ overlaps are close to 1, and μ coverages and mean widths are close to the true likelihood results. This is reflected in the contours of these models, some examples for one flow model and one classifier model being shown in figure 4.4. One can see that in terms of μ the model contours are very close to the true contours.
The flow model does have 3 outliers, with low μ overlap/coverage.
The z overlaps and coverages are not as close to the desired values. This

	large distance	small distance	Higgs
Flow	$3 \cdot 10^{-6}$	$1 \cdot 10^{-5}$	$3 \cdot 10^{-6}$
Classifier	$3 \cdot 10^{-5}$	$3 \cdot 10^{-5}$	$1 \cdot 10^{-4}$

Table 4.1: *Learning rates picked for further analysis based on the results shown in figure 4.2.*

can be seen from the model contours, having a small systematic mismatch with the true contours, depending on the true z value. For the flow for example, for $z = 0.15\pi$ we see the model systematically overestimating z, for $z = 0.25\pi$ it underestimates the parameter and for $z = 0.35\pi$ it overestimates it again. To illustrate what, by eye, small differences in contours reflect big differences in overlap/coverage: for the classifier example, the bottom row of contours ($z = 0.15\pi$) correspond to a mean z overlap of 0.89 and mean z coverage of 0.61, while the top row ($z = 0.35\pi$) corresponds to a mean z overlap of 0.36 and a mean z coverage of 0.54. Differences in the size of this mismatch per model are reflected in the z overlap histograms, where the flow and classifier both have performances spread out between 0.2 and 0.8. Interestingly, while the flow seems comparable or even better than the classifier in terms of z overlap, the z coverages are clearly better for the classifier. The classifiers also have higher z mean widths, where the distribution seems to be more centered on the true value than the flow distribution.

	Flow		Classifier	
	large distance	small distance	large distance	small distance
μ overlap	0.95	0.55	0.94	0.69
z overlap	0.47	0.41	0.52	0.47
total overlap	0.47	0.29	0.52	0.41
μ coverage	0.65	0.61	0.66	0.64
z coverage	0.52	0.50	0.60	0.63
μ mean width	0.035	0.096	0.036	0.089
z mean width	0.032	0.15	0.031	0.17

Table 4.2: *Metric values corresponding to the Gaussian case contours in figure 4.4.*

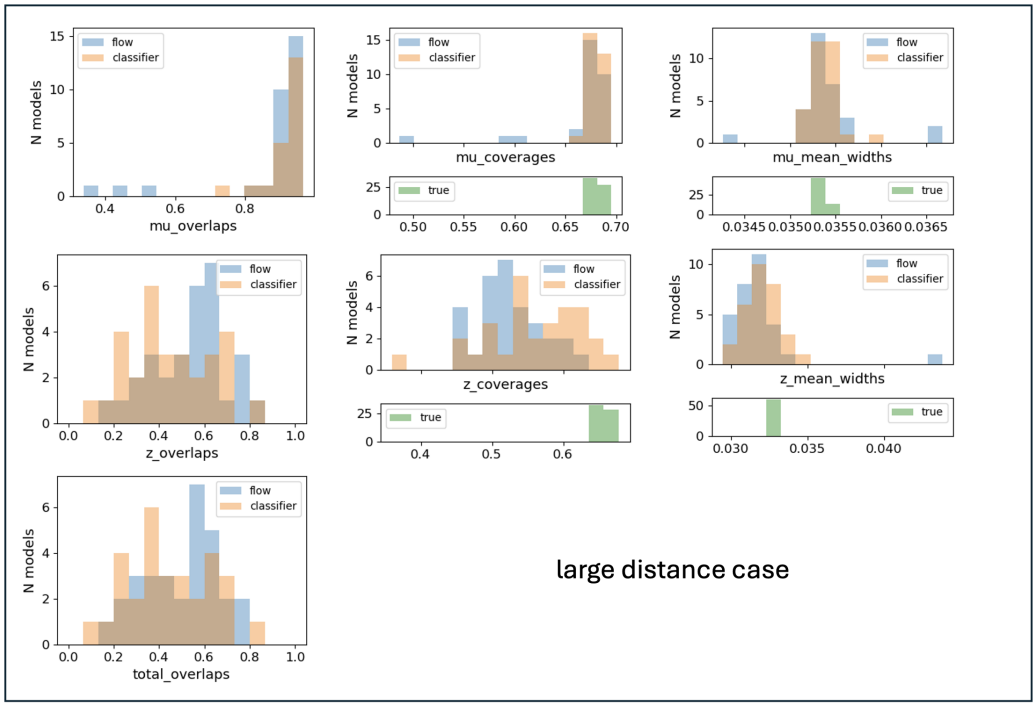

Figure 4.3: *Histograms of all metrics for the 30 flow (with lr=3e-6) and 30 classifier (with lr=3e-5) models that we trained in the Gaussian large distance case. Values for 30 times the true likelihood are also shown in green.*

4.2.2 Gaussian small distance case

For the small distance case, we see that the contours (examples shown in figure 4.4) are much wider than for the large distance case. Also, we now observe an interesting difference with the flow contours being much more squiggly than the classifier contours.
The story with regard to z estimation is pretty similar to the large distance case. There is a clear systematic error in z estimation as a function of true z, size of the error coming in variety, leading to a large spread in z overlaps and z coverages in the histograms in figure 4.5. Furthermore, we again see similar z overlaps for flow and classifiers, while coverages are better for the classifiers and mean widths are higher and closer to the truth.
The major change compared to the large distance case is observed in the μ prediction, for which models are not all as accurate as in the large distance case. For a lot of models, a global offset between the model- and true contours is found in the μ dimension, like in the classifier example in figure 4.4. For fewer, the difference is more z dependent, like in the flow example, where μ is overestimated at low z and underestimated at high

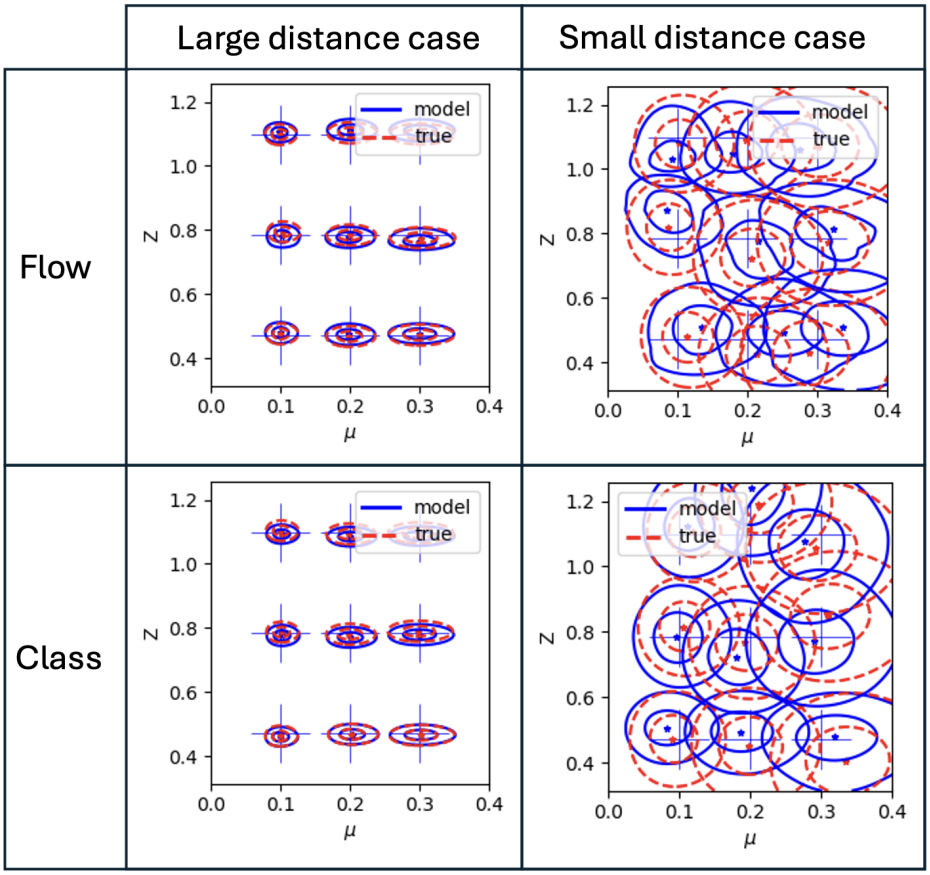

Figure 4.4: *Examples of contours in the Gaussian case. Metric averages over the 9 test sets in each figure are given in 4.2.*

z. These differences are observed in the histograms where we now see μ overlap values spread out between 0.2 and 0.8 for the flow and 0.4 and 0.8 for the classifier. A large part of the μ coverages are found below the lowest true value of 0.65. For the μ mean widths, the most noticeable is the systematically too high values for the flow.

4.2.3 Higgs case

In the Higgs case, we do not have access to the true contours, to assess directly in what way the model contours are wrong. However, trends can be observed when looking at the models with higher coverages versus those with lower coverages for both of the models. An example for both model types with high coverages (on the high end of the coverage distributions observed in figure 4.7) is shown as examples 1 in figure 4.6. These are rel-

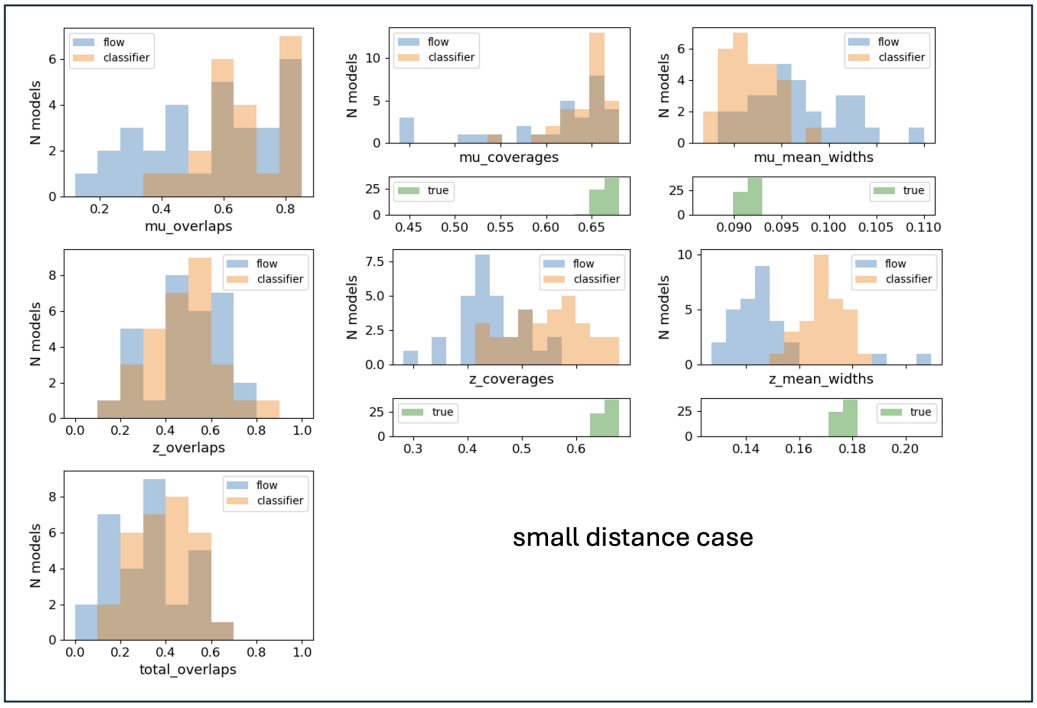

Figure 4.5: *Histograms of all metrics for the 30 flow (with lr=1e-5) and 30 classifier (with lr=3e-5) models that we trained in the Gaussian small distance case. Values for 30 times the true likelihood are also shown in green.*

atively similar. Examples with lower coverage however, shown as examples 2 in the same figure, show larger differences. The classifiers typically have a global offset from the true parameters, such as the lower μ and z in this example. The flows on the other hand typically have this diagonal pattern, with the over- or underestimation of μ relying heavily on the true z parameter.
For the flow, it seems like there is a clearer distinction in 'good' models, looking like example 1 vs 'bad' models, looking like example 2, where for the classifier, there seems to be more of a spectrum with smaller and bigger global offsets. This can be seen nicely from the μ coverage histogram in figure 4.7. Furthermore, from these histograms, we see a significant difference again in the mean width distributions. Interestingly, z mean widths are now, opposite to the Gaussian case, smaller for the classifiers and bigger for the flows.

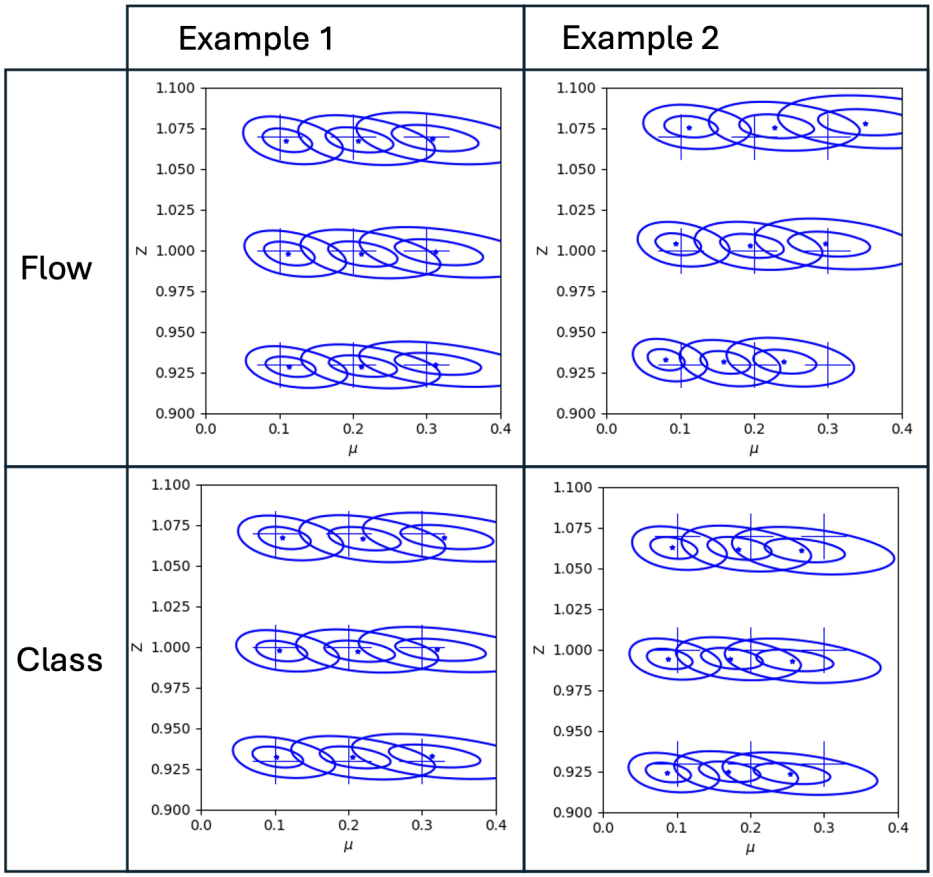

Figure 4.6: *Examples of contours, examples 1 show better performing models and examples 2 worse. Metric averages over the 9 test sets in each figure are given in table 4.3.*

	Flow		Classifier	
	Example 1	Example 2	Example 1	Example 2
μ coverage	0.68	0.56	0.67	0.60
z coverage	0.66	0.59	0.68	0.48
μ mean width	0.095	0.087	0.098	0.085
z mean width	0.014	0.014	0.014	0.13

Table 4.3: *Metric values corresponding to the Higgs case contours in figure 4.6.*

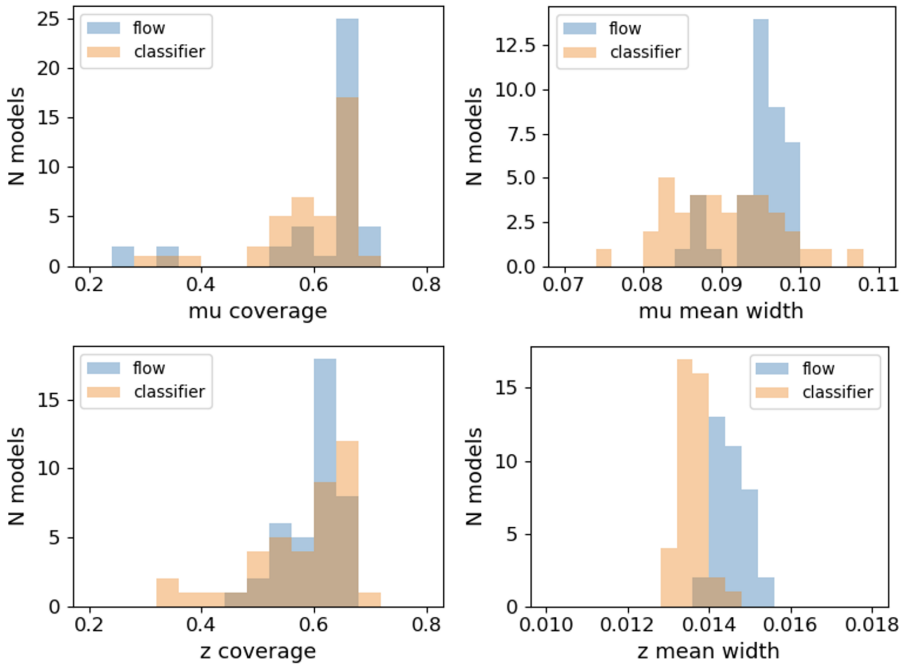

Figure 4.7: *Histograms of coverages and mean widths for the 40 flow (with lr=3e-6) and 40 classifier (with lr=1e-4) models that we trained in the Higgs case.*

Chapter 5

Conclusion

We have tested classifier models and normalizing flows models on two different Gaussian toy cases and one physical "Higgs" case, where for the Gaussian toy cases, we also compared to the true likelihood results. We found both model types able to find the true (μ, z) values in all three cases. However, the quality of the modeled likelihoods strongly differed per case and model.
As is often the case with Machine Learning, there seems to be no clear overall winner between the model types. In the Gaussian case, the classifiers seemed to perform better, and in the Higgs case, the flow models seemed slightly better. The question also arises how fair the comparison is in the first place. We saw how much the influence of the learning rate is on the performances and that is only one of the many tunable hyperparameters. Only little optimization was done for these hyperparameters, due to the fact that there is such a big difference in model performance between identically trained models. Therefore, one has to train a large ensemble of models (like the 30/40 we did here) before you can decide confidently in what direction to adjust a HP. Doing this for multiple HPs at the same time was not feasible for this project.
Furthermore, it is open to question whether the 68.3% Confidence Intervals by themselves are sufficient to judge the performance of these models. This, as the classifiers significantly outperform the flows in Gaussian case z coverage, but not in z overlaps. A confidence interval (CI), of course, only captures a fraction of the information contained in the full underlying likelihood. We cannot rule out the possibility that the flow might have performed better with, for example, 95% CIs. It is also noteworthy that coverage measurements like ours are scarce in the literature related to this topic, where most other papers solely compare the found contours to true

contours.
However, for anyone who is considering using coverage measurement with bootstrapping, our Gaussian case true likelihood results are a very useful addition to [19]. We do not only confirm the bias found there, but we show here that this bias decreases with increasing k factor. Furthermore, the similar trend we found for different bootstrap set sizes, for CIs on two very different parameters, and for both the small and large distance cases, possibly suggests some upper bound on this bias. However, this relation should be investigated on more and completely different datasets in order to confirm this.
The most interesting findings are our more qualitative results. We saw clear differences in the contours between the flow and the classifier models, both in the small distance case and the Higgs case. Now, the particle physicist's biggest nightmare is multiple methods confidently giving them the same wrong answer. Our Higgs case shows a beautiful example of two different methods looking similar when they are performing well, but completely unalike when they are performing badly. Hence, we have shown here that applying both these model types to the same problem and obtaining consistent results can greatly strengthen ones conclusions.

Bibliography

[1] Kyle Cranmer, Juan Pavez, and Gilles Louppe. Approximating likelihood ratios with calibrated discriminative classifiers. 6 2015.

[2] Johann Brehmer, Kyle Cranmer, Gilles Louppe, and Juan Pavez. Constraining effective field theories with machine learning. *Physical Review Letters*, 121, 2018.

[3] Shahzar Rizvi, Mariel Pettee, and Benjamin Nachman. Learning likelihood ratios with neural network classifiers. *Journal of High Energy Physics*, 2024, 2024.

[4] Conor Durkan, Artur Bekasov, Iain Murray, and George Papamakarios. Neural spline flows. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

[5] George Papamakarios, Eric Nalisnick, Danilo Jimenez Rezende, Shakir Mohamed, and Balaji Lakshminarayanan. Normalizing flows for probabilistic modeling and inference, 2021.

[6] Stephen R. Green and Jonathan Gair. Complete parameter inference for gw150914 using deep learning. *Machine Learning: Science and Technology*, 2, 2021.

[7] Johann Brehmer, Felix Kling, Irina Espejo, and Kyle Cranmer. Madminer: Machine learning-based inference for particle physics. *Computing and Software for Big Science*, 4, 2020.

[8] Johann Brehmer and Kyle Cranmer. *Simulation-based inference methods for particle physics*. 2022.

[9] Glen Cowan. *Statistical Data Analysis*. Clarendon Press, Oxford, 1998.

[10] S. S. Wilks. The large-sample distribution of the likelihood ratio for testing composite hypotheses. *The Annals of Mathematical Statistics*, 9, 1938.

[11] Aishik Ghosh, Benjamin Nachman, and Daniel Whiteson. Uncertainty-aware machine learning for high energy physics. *Physical Review D*, 104, 2021.

[12] Torbjörn Sjöstrand, Stefan Ask, Jesper R. Christiansen, Richard Corke, Nishita Desai, Philip Ilten, Stephen Mrenna, Stefan Prestel, Christine O. Rasmussen, and Peter Z. Skands. An introduction to pythia 8.2. *Computer Physics Communications*, 191, 2015.

[13] J. De Favereau, C. Delaere, P. Demin, A. Giammanco, V. Lemaître, A. Mertens, and M. Selvaggi. Delphes 3: A modular framework for fast simulation of a generic collider experiment. *Journal of High Energy Physics*, 2014, 2014.

[14] Victor Estrade, Cécile Germain, Isabelle Guyon, David Rousseau, and Upsud Inp. Adversarial learning to eliminate systematic errors : a case study in high energy physics. *Deep Learning for Physical Sciences Workshop (NIPS)*, 2017.

[15] victor estrade. victor-estrade/datawarehouse: First release, December 2018.

[16] Victor Estrade, Cécile Germain, Isabelle Guyon, and David Rousseau. Systematics aware learning: A case study in high energy physics. In *ESANN 2018 - Proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning*, 2018.

[17] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[18] Conor Durkan, Artur Bekasov, Iain Murray, and George Papamakarios. nflows: normalizing flows in PyTorch, November 2020.

[19] Robert Schall. The empirical coverage of confidence intervals: Point estimates and confidence intervals for confidence levels. *Biometrical Journal*, 54, 2012.

Appendix A

z parameter space edges

For a lot of the classifier models we saw issues arising when we scanned the likelihood over the full z parameter space on which the model was trained. Somehow, the classifiers always overestimate the likelihood towards the edge of the seen z parameter space. The nearer the true parameter is to the edge the more likely this effect is to play a role. An example of one of the small Gaussian case classifier models is shown in figure A.1, where the NLL(μ, z) was scanned over the full training space. Also, to clearly illustrate the effect, we chose true z closer to the edges of the z space with $[0.05\pi, 0.25\pi, 0.45\pi]$.

As can be clearly seen, for high true z the contours are "pushed away" to the edge of the z space. For this specific model, it only happens for the high z values, but for others, it happens for the low z (Gaussian example is symmetric in z). We saw this in the Higgs case as well. For the Flow, we did not run into such problems. Training over a bigger parameter space/scanning a over smaller parameter space solves the problem for the classifier and should be possible in most practical applications, which is what we did in this thesis as well.

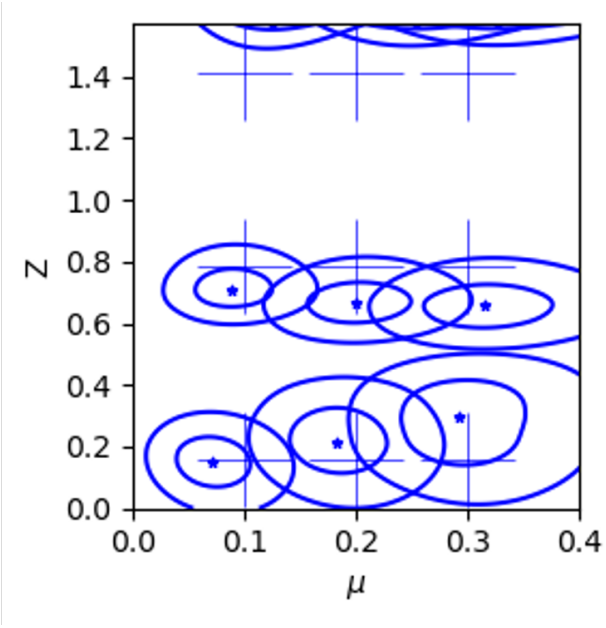

Figure A.1: *Example contours of a classifier model that is evaluated on its full training parameter space. This can lead to an overestimation of the likelihood towards the edges on the z space, which is observed here for the contours with true* z=0.45π.

Appendix B

Parabolic fit assumption

As mentioned in 3.3.2, we constructed the CIs in a way that basically assumed our 4th-degree polynomial fits, were very close to 2nd-degree shaped. We calculated the error that this gave, compared to calculating the CIs without this assumption: for the CIs of 9 test sets (1 for each true (μ, z)) we calculated the $-2log\lambda$ values of our 4th-degree polynomial fit at the bounds of the CI (calculated with the 2nd-degree assumption). In other words, in figure 3.3, the NLL value at the intersections between the fit and the vertical lines, minus the NLL value at the minimum of the fit. We then calculate the corresponding coverage values by means of eq. 2.13. For CIs constructed without the 2nd-degree assumption, this experiment would give 68.3%, by construction. What we get with our 2nd-degree assumption is given in figure B.1. For the Gaussian case, we used the true likelihood CIs and for the Higgs case, we used two different classifiers and two different flows, all with 60%+ mean average coverages, which all gave similar results.
It is seen that asymmetry of the μ NLLs (see the one in figure 3.3)) give distinct peaks at a lower- and higher than 68.3% coverage. The symmetric but non-fully parabolic z NLLs give one peak centered around 68%. Note that, since towards one bound of the CI, coverage is higher, and lower towards the other bound, the measured coverage can be anywhere in between. This is what we see in the true likelihood, non-bootstrap coverages (so no bootstrap bias) in 4.1, which do not get higher than 70%. Hence, especially since we do not draw any conclusions based on differences in coverage above 60%, we decided that the parabolic assumption should not affect this thesis in a significant way.

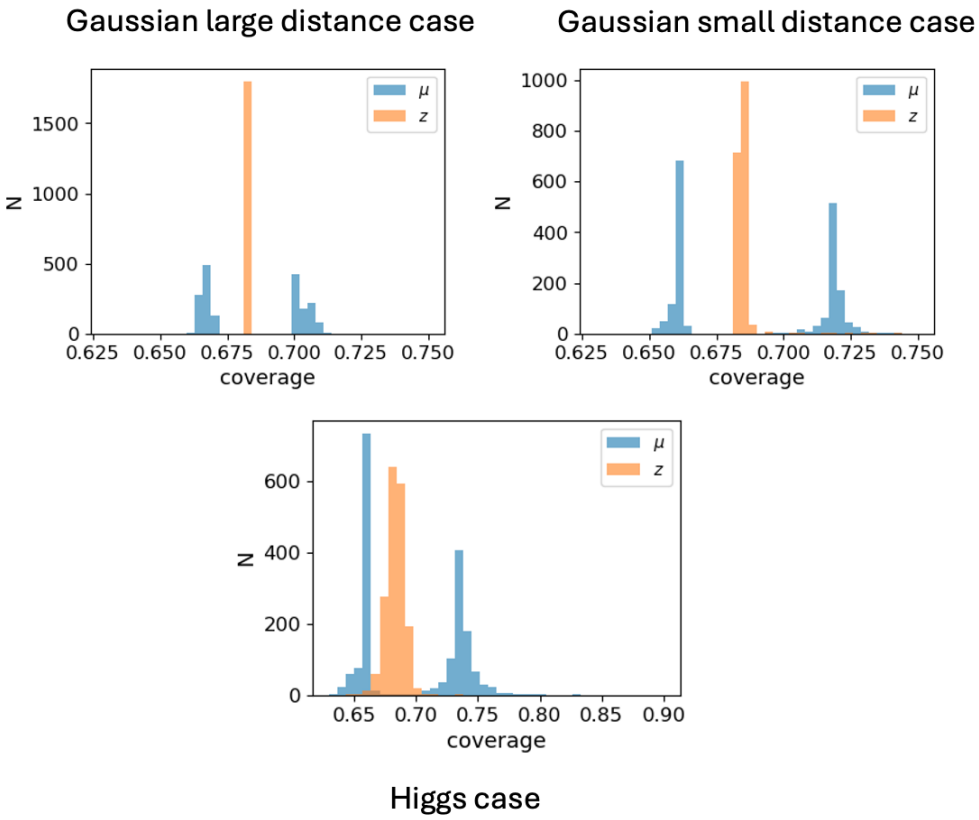

Figure B.1: *Coverages as calculated with eq. 2.13 from the $-2log\lambda$ (calculated with 4th-degree polynomial fit) values at the edges of our CIs, indicating possible deviations to the coverage, due to the assumption of the fit being of 2nd-degree.*